Automatic Recognition of Concurrent and Coupled Human Motion Sequences

zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

der Fakultät für Informatik
des Karlsruher Instituts für Technologie

genehmigte

Dissertation

von

Dirk Gehrig

aus Karlsruhe

Tag der mündlichen Prüfung:	9. Juni 2015
Erste Gutachterin:	Prof. Dr.-Ing. Tanja Schultz
Zweiter Gutachter:	Prof. Dr.-Ing. Rüdiger Dillmann

Abstract

Over the last decades computers have become part of our daily life. However, often the interaction with computers is still unnatural. In recent years, a lot of research has been done on developing advanced human-machine interfaces. For interaction in a certain situations those interfaces need to have knowledge about the current events in their environment, i. e. they need to recognize what humans are doing. Automatic recognition systems are necessary to recognize human actions. An important aspect of human actions are motions.

The observation and recognition of people's daily activities and motions prohibit invasive procedures or body-worn sensors. For unobtrusive motion capture systems optic sensors can be used. Optic sensors such as video cameras have the advantage of enabling additional recognition tasks such as person or object recognition. Although the final system needs to be unobtrusive, the development of most system components, e. g. the pre-processing besides the feature extraction, the model training and recognition algorithms as well as the context models, can be done with various sensors. This allows simultaneous development of an unobtrusive motion capture system as well as the motion modeling and recognition parts.

Typical application areas for human motion recognition are *human-robot interaction*, *entertainment*, *surveillance*, *sports and medicine*, and *video retrieval*, to name only a few. Probably the most challenging task is the recognition of human motions for human-robot interaction, where motions to be recognized range from simple gestures to daily life activities. When humans interact with other humans they simply know what others are doing or what their gestures mean. Humans are mostly unaware of the complex processes in their brain which enable them to recognize simple motions or even complex activities. To empower a robot with such skills, a series of challenges has to be tackled. In this thesis, we focus on those human-robot interactions, more specifically on interaction with humanoid robots. Our application domain is ARMAR III [ARA$^+$06], a humanoid robot that was developed in Collaborative Research Center 588 (CRC 588) "Humanoid Robots - Learning and Cooperating Multimodal Robots". ARMAR requires an online human motion recognition system that runs with the limited sensor information of a humanoid robot. We use the motion recognition, for example, to feed ARMARs intention recognition system.

In this thesis we designed, developed and implemented such a motion recognition system. For this purpose, we systematically develop methods and algorithms for all parts of the recognition system, i. e. Feature Extraction, Motion Segmentation and Labeling, Motion Primitive and Context Modeling as well as Decoding. We implement and evaluate several state-of-the-art techniques.

For this purpose, we collect several datasets to compare our proposed methods with the state-of-the-art in human motion recognition. The main contributions of this thesis are as follows:

Structured functional motion decomposition: The basis of a motion recognition system is a method to generate models of human motions. Many motion recognition systems use *motion primitives*, i. e. meaningful motion units, for their recognition , yet a methodology for structured decomposition of motions into meaningful motion units is missing. We present an approach for functional human motion decomposition which is based on findings in biology and sports sciences. This approach achieves motion segments that facilitate the modeling and recognition of a large variety of human motions.

Flexible and scalable motion recognition: Based on the primitives from the functional motion decomposition, we develop a baseline recognition system for human motions. Afterwards we improve the system to recognize individual body parts. The improvement includes an innovative modeling and search technique for motion primitives, an appropriate context model and a methodology for error measurement. Together these components allow to create a flexible and scalable motion recognition which allows to tackle the recognition of more complex activities than other state-of-the-art systems. The error rate improves by up to 15% relative on a complex dataset compared to the baseline system.

Motion Recognition for a Humanoid Robot: To prove the functionality of our algorithms in real-life scenarios, we use the vision sensors of ARMAR for motion recognition experiments. We develop a recognition system for a humanoid robot that recognizes human daily activities online, i. e. in real-time. The recognition system is integrated into ARMAR's intention recognition system. Most of the methods developed in this thesis are used in online demonstration systems. This proves our concepts to have immediate practical impact and the potential to be applied in human motion recognition.

Contents

1

Introduction

1.1 Motivation

Over the last decades computers became part of our daily life. However, often the interaction with computers is still unnatural. In recent years, a lot of research has been done on developing advanced human-machine interfaces. To enable humans to easily use a computer, those interfaces are tailored towards natural communication channels, such as speech and gestures. The focus of research has been automatic speech recognition and handwriting recognition. Faster hardware and advances in sensors, e. g. webcams, increased the opportunities for automatic motion recognition.

Typical application areas for human motion recognition are *human-robot interaction*, *entertainment*, *surveillance*, *sports and medicine*, and *video retrieval*, to name only a few. When recognizing motions in video retrieval, sports or medicine, the recognition is mostly performed offline, i. e. the recognition is done after the motion has been finished. In contrast, the recognition in areas like human-robot interaction, entertainment or surveillance, has to be done online for a fast reaction time, i. e. the recognition process is started while performing the activity. The focus in surveillance is in recognizing unusual situations whereas in entertainment the goal is to recognize specific simple motions. Probably the most challenging task is the recognition of human motions for human-robot interaction, where motions to be recognized range from simple gestures to daily life activities. An additional challenge is a recognition on a humanoid robot with limited sensors and limited resources for processing the data.

In this thesis, we focus on human-robot interaction more specifically on interaction with humanoid robots. In the field of robotics exists an increasing need for knowledge about human motions, as a humanoid robot has to be empowered with knowledge about sequences of motions [Sch07]. Our application domain is ARMAR III [ARA^{+}06], a humanoid robot that was developed in the Collaborative Research Center 588 (CRC 588) "Humanoid Robots - Learning and Cooperating Multimodal Robots". CRC 588 developed a humanoid robot that looks and acts like humans. In order to act human-like, ARMAR needs to be able to recognize what is going on in its environment. Therefore, we developed an online human motion recognition system that runs with the limited sensor information on a humanoid robot. We used the motion recognition for example to feed ARMARs intention recognition system.

When humans interact with other humans they simply know what others are doing or what their gestures mean. Humans are mostly unaware of the complex processes in their brain which enable them to recognize simple motions or even complex activities. To empower a humanoid robot with such skills, a series of challenges has to be tackled.

1.2 Goal of the Thesis

The goal of this thesis is the development of a motion recognition system for a humanoid robot. The main aspects for the design of the recognition system are:

- unobtrusive capturing of human motions
- continuous recognition of human motions
- online recognition of motions
- integration of context information into the recognition system
- development of motion modeling techniques that scale well to more unconstrained recognition domains

Unobtrusive Capturing: The observation and recognition of people's daily activities and motions prohibit invasive procedures or body-worn sensors. For unobtrusive motion capture systems acoustic or optic sensors can be used. Optic sensors such as video cameras have the advantage of enabling additional recognition tasks on a humanoid robot such as person or object recognition. Although the final system needs to be unobtrusive, the development of most system components, e. g. the pre-processing besides the feature extraction, the model training and recognition algorithms as well as the context models, can be done with various sensors. This allows simultaneous development of an unobtrusive motion capture as well as the motion modeling and recognition parts.

Continuous Recognition: Humans are moving all the time or at least show particular body poses constantly. Therefore, a motion recognition system has to be able to recognize continuous sequences of human motions.

Online Recognition: For an online recognition system such as one for a humanoid robot, it is not sufficient to start the recognition after a person has finished an activity. Rather it is necessary to recognize the motions as early as possible, already while observing the activity.

Integration of Context Information: To fully recognize a human motion the context, e. g. the situation, previous motions and manipulated objects, has to be considered. Therefore, context information has to be integrated into the modeling and recognition process.

Scalability: Humans are capable of performing a wide range of motions. A major aspect that has to be considered to enable good scalability of the system is the ability to perform multiple independent motions in parallel. The system needs to flexibly handle the dynamics of body parts that perform coupled or concurrent motions.
To achieve the above goals, we implemented and evaluated several state-of-the-art techniques. For this purpose, we collected several datasets to compare our proposed methods with the state-of-the-art in human motion recognition. The main contributions of this thesis are as follows:

1. **Structured functional motion decomposition:** The basis of a motion recognition system is a method to generate models of human motions. Many motion recognition systems use *motion primitives*, i. e. meaningful motion units, for their recognition but a methodology for structured decomposition of motions into meaningful motion units is missing. To achieve motion segments that facilitate the modeling and recognition of a large variety of human motions, we developed an approach for human motion decomposition which is based on findings in biology and sports sciences. The structured functional decomposition is described in Chapter 4.
2. **Flexible and scalable motion recognition:** Based on the primitives from the functional motion decomposition, we developed a system that recognizes motions of individual body parts. The improvement includes an innovative modeling and search technique for motion primitives, an appropriate context model and a methodology for error measurement. Together these components allow to create a flexible and scalable motion recognition which allows to tackle more complex activities than other

state-of-the-art systems. Furthermore, we improve the error rate by up to 15% relative on a complex dataset. The flexible and scalable motion recognition is presented in Chapters 6 and 7.

3. **Motion Recognition for a Humanoid Robot:** To prove the functionality of our algorithms in real-life scenarios, we used the vision sensors of ARMAR for motion recognition experiments. We developed a online recognition system for a humanoid robot that recognizes motions of human daily activities in real-time. The recognition system is integrated into ARMAR's intention recognition system. The vision-based recognition system is described in Chapter 8.

1.3 System Overview

Activities can be modeled or described based on the performed motions or based on the changes to the environment. Systems that model activities as changes to the environment typically focus on object affordances. We focus on motion related modeling techniques since they allow to recognize both types of motions, those that change and those that do not change the properties of objects.
The techniques for modeling activities based on the performed motion can be devided into the three categories, structural models, biomechanical models, and models about the internal represention of motions in humans. As we want to recognize motions in an unobtrusive fashion, we concentrate on the two techniques that model motions as they can be observed by looking at people, i. e. the structural models and the biomechanical models. The more general view on *motion sequences*, i. e. sequences of human motions, is a representation using structural models as in [Göh92]. The structural models can be used to model the structure of activities. The structure of activities is part of the motion's context information. Another way of modeling human motions are biomechanical models with kinematic and dynamic features of the movement. We apply this idea for modeling the individual motion primitives.
Figure 1.1 gives an overview of the components necessary to train and use our recognition system. The can-shaped items describe the input and output of the different processing steps which are represented by rectangular boxes. The yellow items are concerned with the training of the system while all red items are concerned with the recognition process. Both processes, the training and the recognition, start with the acquisition of motion data. For training, we define a set of motion primitives and then segment the data into those primitives which gives us labels and segment boundaries for the acquired motion data. We also generate appropriate features from the motion data. The labels and

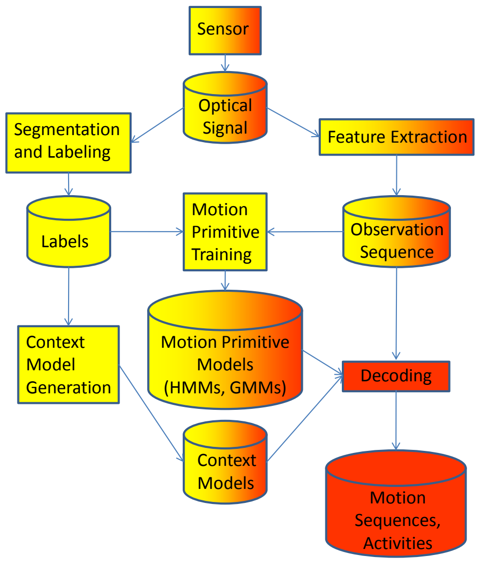
Sensor
Optical Signal
Segmentation and Labeling
Feature Extraction
Labels
Motion Primitive Training
Observation Sequence
Context Model Generation
Motion Primitive Models (HMMs, GMMs)
Decoding
Context Models
Motion Sequences, Activities

Figure 1.1: The diagram shows the components of our motion recognition system.

the preprocessed data are used to train motion primitive models for individual motion primitives and to generate models for the motion primitives' contexts. For recognition the motion primitive models and the context models are used to recognize sequences of human motions or activities. The result is a textual representation of the performed motion primitives or the activity. To build such a system, we need to solve the following tasks:

1. Define a motion primitive alphabet
2. Model motion primitives
3. Extract and select appropriate features
4. Model the motion context
5. Online recognize motion sequences

1.4 Structure of the Thesis

In the following we give a short overview on how this thesis is structured:

Chapter 2 In chapter 2 we give a detailed introduction to human motion modeling and recognition techniques as they are used in this thesis.

Chapter 3 In chapter 3 we describe the datasets collected for this thesis' experiments.

Chapter 4 In chapter 4 we explain the state-of-the-art of human motion recognition and give an overview on the different techniques. We introduce the basic principles of our sequential human motion recognition system. We give an introduction on defining and training motion primitive. We also give an overview on motion contexts including some basic context modeling techniques.

Chapter 5 In chapter 5 we describe and evaluate our sequential human motion recognition system. We first develop a marker-based baseline recognition system for continuous motion recognition. We then extend the system by developing a new object dependent context model.

Chapter 6 In chapter 6 we describe our approach for modeling and recognizing concurrent and coupled human motion sequences. Based on the methods and results of chapters 4 and 5 we develop methods for the concurrent and coupled human motion modeling and recognition. We extend the motion primitive definition and segmentation as well as the training of motion primitives from chapter 4. We develop a new N-Gram context model for concurrent and coupled recognition and describe the recognition process in detail.

Chapter 7 In chapter 7 we evaluate the proposed algorithms for modeling and recognizing concurrent and coupled motions and compare the results to the sequential system from chapter 5.

Chapter 8 In chapter 8 we transfer our knowledge from the marker-based systems to marker-less video-based recognition. The final system in this chapter is a video-based person-independent human motion recognition system for various daily activities.

Chapter 9 Chapter 9 summarizes the developments and results of this thesis, compares these to the proposed goals of the thesis and gives an outlook on future work.

2

Human Motion Recognition

2.1 Goals and Features

In the past decade a lot of work had been done in the area of activity recognition, but the challenging problem of fine-grained recognition of human daily activities had only been addressed rarely. In this thesis we tackle these challenges and solve some of them. The goal is to enable a humanoid robot to recognize fine-grained motions of daily activities.

In our research we focus on motions as they appear in a household environment. Even in this rather limited environment the number of motion sequences that humans perform is basically unlimited. Therefore, the recognition system needs to scale well to a large set of motions.

Thus, it is not feasible to create a separate model for each motion sequence due to the lack of enough training data. It is necessary to find *motion primitives* (small motion units) that are the elements of all activities. We describe each activity in terms of a sequence of motion primitives.

Since we want to recognize those motions with a humanoid robot, the system is limited to the sensors of the humanoid robot for capturing the motions. Therefore, the system needs to enable a recognition process based on limited information about the human motions. Additionally, those motions have to be recognized online, i. e. they need to be recognized with a small latency, so that the robot can react on the human's behavior in time. As motions may happen at all times, we need a continuous recognition of the human's motions.

2.2 Automatic Recognition System for Human Motion

In this section we give an overview on modeling techniques for human motion recognition. Human motion recognition is investigated in different research fields such as computer vision, robotics, and human-computer interaction. Many good overview papers have been written on motion and activity recognition such as [AR11, AC97, MHKS11, DW11, KKUG07, MA07, WH99, PSH97].

Following the taxonomy proposed by Aggarwal and Ryoo [AR11] human motion recognition techniques can be categorized into single-layered approaches and hierarchical approaches. Single-layered approaches are approaches that represent and recognize human activities directly based on sequences of images (or other sensory input). Due to their nature, single-layered approaches are suitable for the recognition of motion primitives with sequential characteristics. In contrast, hierarchical approaches represent high-level human activities by describing them in terms of simpler activities or motion primitives. Recognition systems composed of multiple layers are constructed, making them suitable for the analysis of complex activities.

Single-layered approaches can be split further into space-time approaches and sequential approaches. The space-time approaches use a 3D XYT space-time volume, i. e. they stack 2D images over time to form a 3D volume. Those volumes are analyzed to recognize human motions. Space-time approaches are suitable for recognizing periodic motions and gestures, but they often have difficulties handling speed and motion variations inherently. In contrast, sequential approaches use features to recognize human motions. They consider a sequence of observations (i. e. feature vectors) as an input that is used to recognize a human motion. They first convert the input (e. g. a video) into feature vectors by extracting features. Once the feature vectors have been extracted, they are used to measure how likely the feature vectors are produced by a person performing an activity.

Hierarchical approaches use the single-layered approaches to recognize primitives that are combined into more complex activities. They use either statistical, syntactic or description-based methods to model higher level activities based on motions or low-level activities.

To enable the recognition of a large variety of motions most of the above approaches use primitives (e. g. motion primitives) as the elements of the recognition system. A very common approach using primitives for motion recognition are *Hidden Markov Models* (HMMs). In this thesis we use two approaches. On the one hand, we use a single-layered approach, which is well suited to recognize motion primitives while allowing variability in speed and motion performance. On the other hand, we use an hierarchical syntactic approach to govern higher-level information. In the following we will describe the basics of our recognition system.

2.3 Capturing Human Motion

Human motions need to be captured before they can be recognized. A variety of methods for capturing human motions can be used. These methods can be categorized by the signal that is captured. The most common approaches use optical signals, i. e. motions are captured using video cameras. The capturing of optical signals can be split into *marker-based* methods and *marker-less* methods. Markers are attached to the human body for the marker-based methods. Those markers simplify the capturing of the human motion through having a distinct color or by reflecting a specific waveband of light so that the markers can be easily detected in the captured videos. For details on the marker-based method used in this thesis we refer to Section 4.2. Marker-less methods use videos without attaching markers to the human body. The marker-less methods used in this thesis are presented in Sections 8.1.2 and 8.1.1. The marker-based as well as the marker-less methods extract features, e. g. joint angles. All features belonging to the same time step are stacked to form a *feature vector*. A recognition system can then recognize the observed motions based on a sequence of feature vectors, i. e. an *observation sequence*.

2.4 Overview on HMM-decoder

Our recognition system (see Fig. 2.1) uses an HMM-based decoder, which typically consists of three components. The first component is a set of primitive models, i. e. a set of HMMs, which model sequences of feature vectors. Since the motions are split into motion primitives, a second component (the context model) is typically used to describe the possible combinations of the primitive models during the decoding. The context model can also add information about the environment in which the motion is performed, e. g. available objects. The primitive models and the context model are used to measure how likely a sequence of feature vectors is produced by a person performing an activity. This is done by the third component namely the decoder. We will now describe those techniques in more detail.

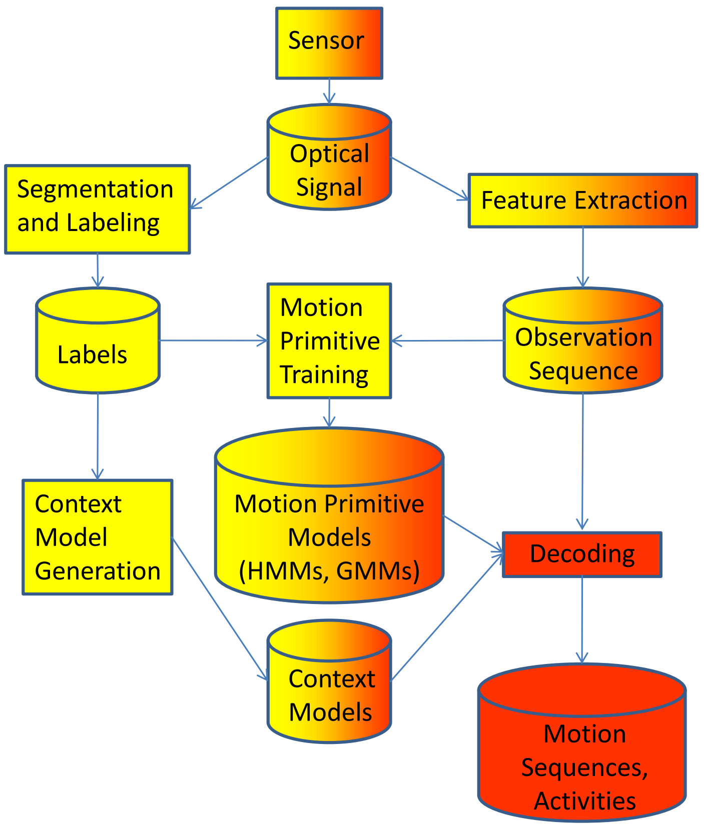
Sensor
Optical Signal
Segmentation and Labeling
Feature Extraction
Labels
Motion Primitive Training
Observation Sequence
Context Model Generation
Motion Primitive Models (HMMs, GMMs)
Decoding
Context Models
Motion Sequences, Activities

Figure 2.1: Overview on a HMM-based recognition system.

2.5 Gaussian Mixture Models

A common approach for modeling the distribution of a set of feature vectors are Gaussian distributions. A Gaussian distribution can be fully described by its *mean* μ and its *covariance matrix* Σ. If a single Gaussian distribution is not enough to represent a set of feature vectors it can be extended to a mixture of Gaussian Models, a *Gaussian Mixture Model* (GMM). A GMM is a weighted sum of Gaussian distributions which can be used as a good approximation for the distribution of a set of feature vectors.

2.6 Hidden Markov Models

Since feature vectors of a motion primitives evolve over time, the GMMs are extended by using HMMs. An HMM is a statistical model that consists of *states* (Circles in Fig. 2.2) representing a temporal sequence. In this thesis we model each motion primitive with one HMM. Each HMM-state has an internal model

(e. g. a GMM) that models the likelihood of observing a feature vector while the temporal process is in this state. The states are hidden and can not be observed directly. Feature vectors (Squares in Fig. 2.2) are called *observations*. To fully describe an HMM λ we need five parameters: the HMM-*states* S_i, *initial state probabilities* π_i, an *observation space*, *emission probabilities* for each state, and *transition probabilities* a_{ij} for each pair of states S_i and S_j. The observation space in this thesis is $\mathbb{R}^D$, whereas the emission probabilities for each state are GMMs over $\mathbb{R}^D$.

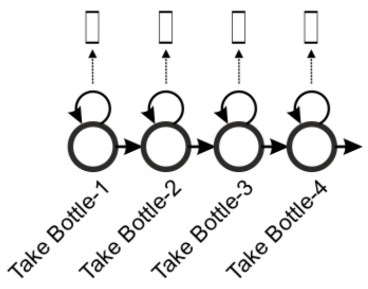

Figure 2.2: Four-state left to right HMM for the motion primitive "Take bottle".

We use left to right HMMs, i. e. a sequence of HMM states whereas each state has two equally likely transitions. Each state has one transition to the successor state and a selfloop (see Fig. 2.2). The actual number of states varies depending on the dataset that is modeled with the HMMs.

2.6.1 HMM Training

Training is performed based on an observation sequence and the corresponding sequence of HMMs. The initial state probability of each HMM is always 1 for the first state and 0 for all others. During HMM training the emission probabilities are optimized based on observation sequences of the training data. In the first step, the Viterbi algorithm is used to assign each feature vector of the observation sequences to an HMM state. Such an assignment is called a *path*. In the second step, each GMM is optimized based on the set of feature vectors assigned to the corresponding state. Both steps are repeated several times.

For calculating the paths for each observation sequence we use the Viterbi algorithm which optimizes equation

$$\hat{Q} = \underset{q_1,\dots,q_T}{argMax} p(q_1 \dots q_T | X, \lambda) \tag{2.1}$$

i. e. it calculates the most likely state sequence $\hat{Q}$ given an observation sequence X of length T and an HMM λ. In order to find the most likely state sequence we define the quantity

$$\delta_t(i) = \max_{q_1,\dots,q_{t-1}} p(q_1 \dots q_t = i, x_1 \dots x_t | \lambda) \tag{2.2}$$

where $\delta_t(i)$ is the best score along a path, at time t, which accounts for the first t observations and ends in state S_i. By induction we have

$$\delta_{t+1}(j) = \max_i (\delta_t(i) a_{ij}) p_{GMM_j}(x_{t+1}) \tag{2.3}$$

The complete procedure for finding the most likely state sequence can be described as follows:

Initialization

$$\delta_1(i) = \pi_i p_{GMM_i}(x_1), 1 \leq i \leq N \tag{2.4}$$
$$\psi_1(i) = 0. \tag{2.5}$$

where N is the number of the number of HMM states.

Recursion

$$\delta_t(j) = \max_{1 \leq i \leq N} (\delta_{t-1}(i) a_{ij}) p_{GMM_i}(x_t), 2 \leq t \leq T, 1 \leq j \leq N \tag{2.6}$$
$$\psi_t(j) = \underset{1 \leq i \leq N}{argMax} (\delta_{t-1}(i) a_{ij}), 2 \leq t \leq T, 1 \leq j \leq N \tag{2.7}$$

Termination The most likely final state for time T ($\hat{q_T}$) can be calculated as

$$\hat{q_T} = \underset{1 \leq i \leq N}{argMax} (\delta_T(i)). \tag{2.8}$$

where i needs to be a final state. The most likely state sequence can then be retrieved by backtracking:

$$\hat{q_t} = \psi_{t+1}(\hat{q_{t+1}}), t = T-1, \dots, 1 \tag{2.9}$$

For a more detailed description on HMMs see [Rab89]. The paths $\hat{Q}$ of all observation sequences can then be used to train the emission probabilities, i. e. the GMMs.

2.7 Context Models

Since we split human motions into motion primitives and model each motion primitive with an HMM, we should also model the structure of the motion sequence, i. e. how the primitives can be combined to form a motion sequence or what motion sequences are more likely than others. This information is called the *context model*, as it describes the context of motion primitives in terms of how likely is a motion primitive in its context e. g. the previously recognized motion primitives. The context can also contain information about the context in terms of the spatial context, e. g. the location a motion is performed in. The context models that are used in this thesis are described in Section 4.4. The context information can significantly reduce the search space and improve the recognition of motion sequences, since it helps the recognition system to find the most likely motion primitives given the context.

2.8 Decoding

For the recognition of a motion based on a set of HMMs we can use the Viterbi algorithm [Rab89]. It calculates a likelihood for an observation sequence for each of the HMMs. The HMM with the highest likelihood specifies which motion has most likely been observed. The Viterbi algorithm works well for short sequences of HMMs. The HMM sequences need to be rolled out for the recognition, i. e. a motion primitive model that is part of multiple sequences is instantiated and processed multiple times. Therefore, the processing time grows exponentially with the length of the HMM sequence, i. e. we need to improve our search technique. We extend the Viterbi algorithm to Beam search, which can be used to search for the most likely state sequence in a sequence of HMMs with arbitrary length. The decoding of the most likely sequence of motion primitives is carried out as a time-synchronous beam search guided by context models, which will be described in more detail in section 2.8.1.

2.8.1 Time-Synchronous Beam Search

Time-synchronous beam search [HAH01] is based on Viterbi search [Rab89]. As with the Viterbi algorithm beam search processes one feature vector at a time and updates the probability of being in a certain HMM state. To recognize a sequence of motion primitives we have to build all possible sequences of motion primitives (HMMs) and search through all of them. It is not possible to use the standard Viterbi search on sequences of HMMs since the number of HMMs to be searched through is growing exponentially. To solve this problem we use beam search which limits the number of HMM states that are traversed using heuristics. For the decoding of human motion sequences we need to optimize the following equation:

$$\hat{M} = \underset{M}{argMax} p(M|X) = \underset{M}{argMax} p(X|M) \cdot p(M) \tag{2.10}$$

We search for the most likely motion primitive sequence $\hat{M}$ over all possible sequences M given the observation sequence X. Applying Bayes rule we get the right hand side of the equation. Therefore, we can search for the motion primitive sequence $\hat{M}$ that optimizes the right hand side. This allows us to handle the likelihood of an observation sequence given a motion primitive sequence $p(X|M)$ separately from the a-priori probability for the motion primitive sequence. While searching within an HMM we mostly have a very small number transitions between states we can traverse to, e. g. as in Fig. 2.2. The main challenge of searching in a set of motion primitive HMMs occurs when traversing from one HMM to another. In the following we will first introduce how to optimize $p(X|M)$ before describing how to integrate $p(M)$, which describes the probability of sequences of motion primitives.

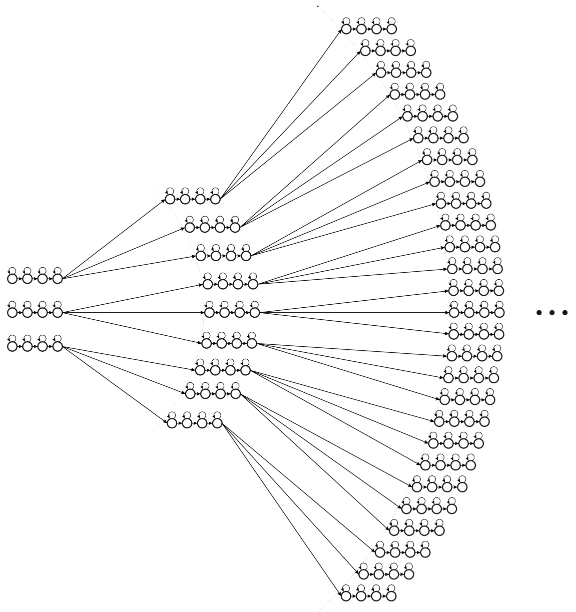

Figure 2.3: The graph shows the search space for sequences of three primitive models.

When traversing from the end of an HMM to the beginning of other HMMs the number of HMM states we have to consider for our search grows exponentially (see Fig. 2.3). To handle the exponentially growing number of HMM states that have to be searched, two optimizations have to be used. The optimizations

1. only traverse the best states and forget about all others.
2. create only one HMM per motion primitive.

The idea of the first optimization is that it is not possible to search all state sequences but we can assume that the states with the best scores (a score is the negative logarithm of a probability or a probability density value) at a certain time step are the ones that most likely lead to the overall motion sequence with the best score. This comparability of different scores is an important advantage of time-synchronous beam search in comparison to other search methods, such as A-star search. Due to the comparability of scores we can apply a *beam*, i. e. applying a threshold relative to the best score and only traversing states that are closer to the best score than the threshold (beam). Using this beam we can vary the trade-off between recognizer speed and recognition accuracy. In general, smaller beams result in faster systems whereas the recognition accuracy decreases at the same time due to search errors resulting from eliminating promising paths.

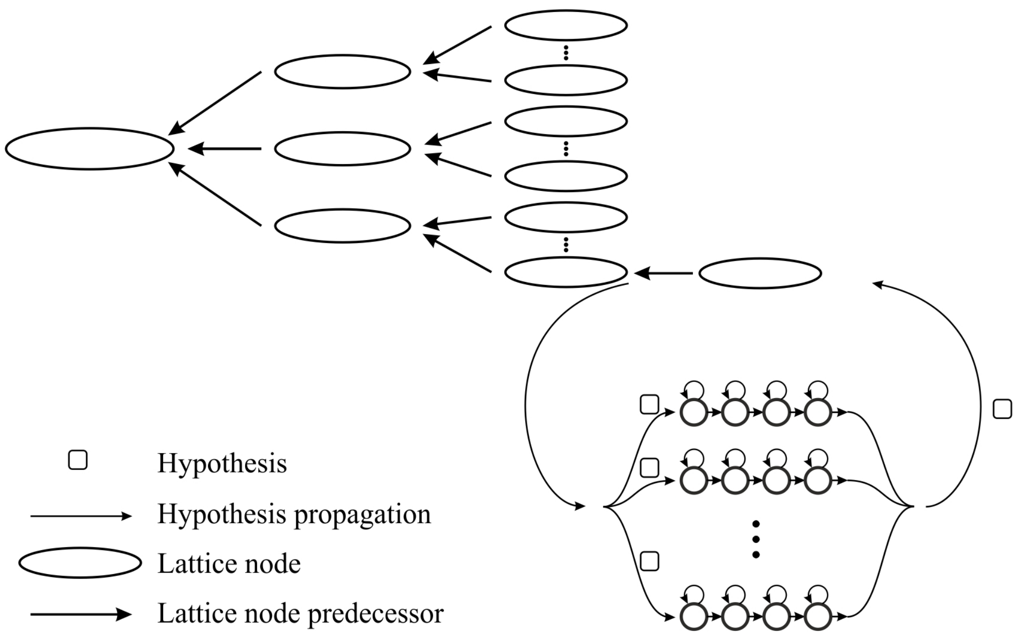

Figure 2.4: A lattice node is created when propagating a hypothesis from the end of an HMM to the beginning of another HMM.

For the second optimization it is necessary to change several things in the search structure. We add history objects (lattice nodes) which store the traversed motion primitives. Each lattice node contains one traversed motion primitive and is connected to its predecessor. A lattice node is created when traversing from one HMM to another (see Fig. 2.4). Instead of storing the current score in an HMM state and knowing the traversed motion primitives implicitly through the connection of the motion primitive HMMs (as in Fig. 2.3) we can use a single HMM for all occurrences of the same motion primitive. We use hypothesis objects of which each stands for a partial motion primitive sequence. Each hypothesis knows its history of motion primitives, i. e. its latest lattice node, and the current score for the HMM state given the history of motion primitives. Each hypothesis is propagated along all outgoing transitions to the successor states once every time step. Only the best hypothesis is kept for each state while all others are removed. Afterwards the hypotheses are pruned, i. e. thresholds are applied and hypotheses that are worse than the thresholds are removed before further propagating the hypotheses. After processing the last feature vector we take the best hypothesis in a final HMM state and trace the motion sequence back by iterating backwards over the lattice nodes. The result is the most likely sequence of motion primitives.

Incorporating Context Models into Beam Search

In the previous section we have explained how to efficiently search the sequence of motion primitives that optimizes $p(X|M)$. We will now introduce how to integrate context models into the search. When integrating context models into Beam Search we have to do two things:

1. Add the context model state to the hypothesis and in each HMM state keep the best hypothesis for each context model state.
2. At motion primitive transitions add the context model score for the current motion primitive.

It is necessary to always keep a hypothesis for each context model state since the *context model score* that will be added when traversing to the next motion primitive could lead to a change in order of hypotheses, i. e. the best hypothesis could be replaced by a previously worse hypothesis. This context model score is always added to the hypothesis score when traversing from one motion primitive to another (see Figure 2.5).

The context model score is often a probability while the HMM score especially the GMM scores are probability density functions. This means that the context model probability is normalized between 0 and 1 while the value of the GMM given a feature vector can be arbitrarily large. Therefore, we use a *context model weight* (cmw), which reduces this discrepancy between the two different scores. We also add a *primitive insertion penalty* (pip) each time we traverse from one motion primitive HMM to another. With this parameter we can additionally configure the tendency of recognizing too many or too few motion primitives. Thus, the final equation that is optimized during the decoding is:

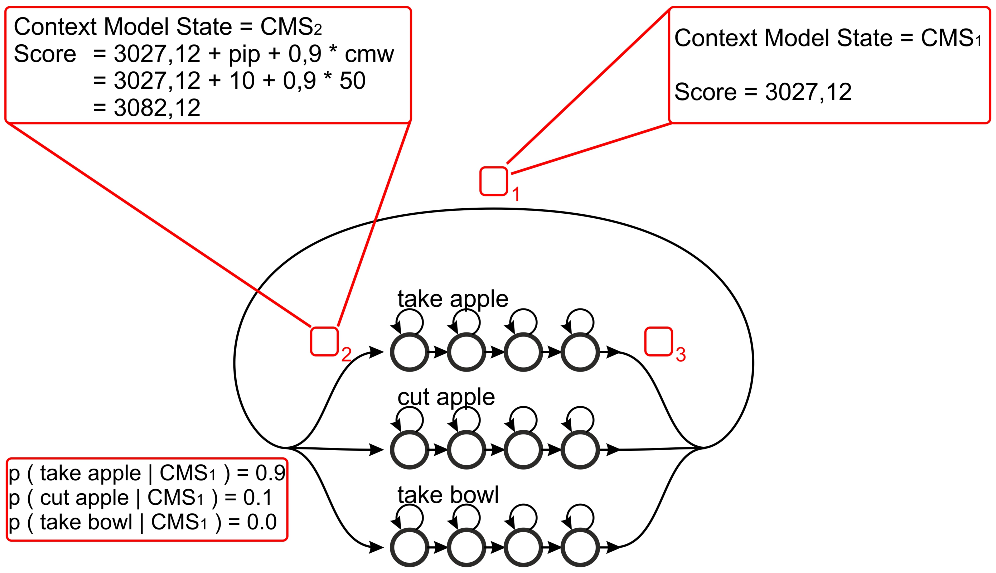
Context Model State = CMS2
Score = 3027,12 + pip + 0,9 * cmw
= 3027,12 + 10 + 0,9 * 50
= 3082,12
Context Model State = CMS1
Score = 3027,12
1
2
3
take apple
cut apple
take bowl
p (take apple | CMS1) = 0.9
p (cut apple | CMS1) = 0.1
p (take bowl | CMS1) = 0.0

Figure 2.5: When propagating a hypothesis from the end of an HMM to the beginning of another HMM the context model score is added to the hypothesis score.

$$\underset{M}{argMax} p(M|X) \approx \underset{M}{argMax}(p(X|M) \cdot \prod_{i=1}^{|M|} p(M_i)^{cmw} \cdot |M|^{pip}). \tag{2.11}$$

2.9 Recognition Toolkits

In this thesis we use two different frameworks for time-synchronous beam search. The first one is the *Janus Recognition Toolkit* (JRTk) [FGH+97] using the ibis decoder [SMFW01]. This framework has been developed primarily for speech recognition. JRTk is used for the sequential recognition systems if not stated otherwise (see Chapters 4, 5, and 8). The second framework that was developed during this thesis and which was used for several experiments is the *Biosignals Recognition Toolkit* (BioKIT) [TWG+14]. BioKIT is designed and used for all kinds of recognition systems based on biosignals such as electromyography, electroencephalography or signals from human motions. This framework is used for the experiments with concurrent motions (see Chapters 6 and 7).

2.9.1 BioKIT

BioKIT [TWG+14] is a software framework for modeling and recognizing feature sequences, especially in the area of biosignals such as speech, electromyography (EMG), electroencephalography (EEG) and motion. It contains modules for loading a variety of data types, modeling and recognizing sequences of primitive models as well as some post-processing. The core framework is written in c++ to be fast enough to compete with other state-of-the-art recognition framework. For easier possibilities of experimenting with new algorithms and performing large experiments, the core framework is wrapped in python. The core recognition engine is a beam search based on token passing as described in 2.8.1. Since our research is partially based on extending the BioKIT framework we will now describe the decoder of the BioKIT framework in more detail.

Pruning in BioKIT

As explained in section 2.8.1 one of the core issues of beam search are the beams (i. e. the pruning). In Figure 2.6 (a) we can see a search graph (in our case a set of HMMs) with multiple hypotheses (with different context model states) at each node. Although we have a rather simple structured search graph the pruning procedure applies the same way for all types of search graphs.
In BioKIT we have 6 different beams for pruning the hypotheses in each time step after having propagated the hypotheses to the successor nodes and updated the hypothesis attributes (e. g. its score). All nodes that have at least one hypothesis are active nodes. The beams are *active node top N*, *active node beam*, *final node top N*, *final node beam*, *hypothesis top N*, and *hypothesis beam*. All top N prunings eliminate all elements of the respective type that are not within the best N elements. The beam prunings eliminate all elements of the respective type that are not within a beam (i. e. a score difference) to the best element.
The first pruning step is the active node top N pruning. Here we keep the best N nodes (from all nodes) and eliminate all others. The score of a node is the score of the best hypothesis within this node. After active node top N pruning we get for example the search graph in Figure 2.6 (b). At the same time the final node top N pruning is performed. We again keep the best N nodes of the initial search graph (a) but this time we apply top N only to the final nodes. This gives us for example the situation in (c). Since the active and final node top N prunings are performed simultaneously we get the result in (d). On the resulting active and final nodes we perform the beam prunings. First we apply the active node beam. Therefore, we need to find the best of all nodes and prune all nodes that are worse than the score of the best node plus the active node beam. As a result we get a search graph like in (e). Again we do the same for final nodes only, which gives us a search graph as in (f). We now have only a few active nodes left and look into the hypotheses in more detail. Up to know we always pruned all hypotheses of a node. We will now perform the hypothesis beam pruning which is applied on each node separately. For each

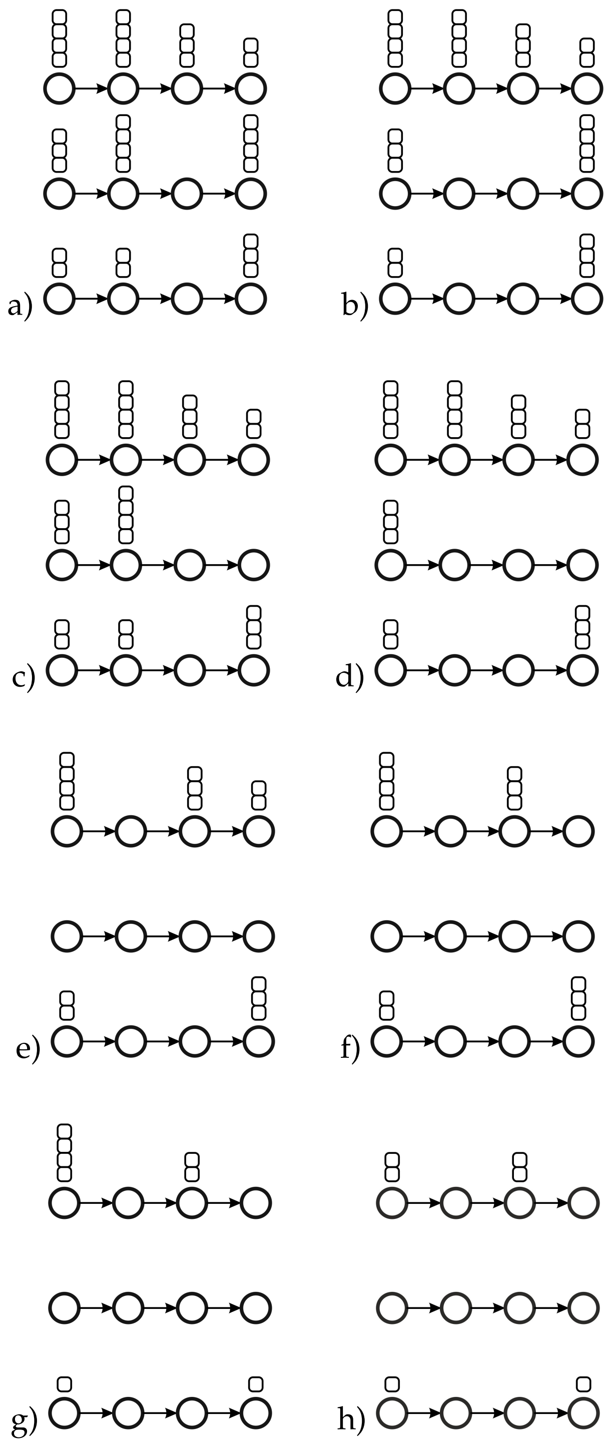

Figure 2.6: Search graph with hypotheses before pruning (a), after active node top N pruning (b), after final node top N pruning (c), after active and final node top N prunings (d), after both node top N and active node beam prunings (e), after both node top N and both node beam prunings (f), after both node top N, active node, final node, and hypothesis beam prunings (g), and after all pruning steps (h).

node we keep the hypotheses that are better than the best hypothesis plus the hypothesis beam as in (g). Afterwards we perform hypothesis top N pruning, e. g. with $N = 2$ we keep only the best 2 hypotheses for each node. The final status of the search graph after the pruning might look like Figure 2.6 (h). As we can see the number of hypotheses has been reduced significantly. This means that the effort for propagating hypotheses in the next time step is a lot less. Selecting the thresholds for the pruning steps is always a trade-off between recognition speed and accuracy.

3

Databases

3.1 Collaborative Research Center 588 - Humanoid Robots

The research in this thesis was part of the Collaborative Research Center 588 "Humanoid Robots" (CRC 588)[1]. The goal of the CRC 588 was the development of concepts, methods and concrete mechatronic components for a humanoid robot. With the help of this partially anthropomorphic robot system, it is possible to perform human-robot interactions in household environments. In order to be accepted by humans the robot needs to look and behave human-like. For a human-like behavior the robot has to have an understanding of its enviroment, e. g. it needs to recognize what humans are doing. One important aspect is the recognition of human motions. To evaluate the approaches developed in this thesis we collected several datasets in household environments.

3.2 Databases

3.2.1 KIT-S - The Staged Kitchen Motions Dataset

For our first experiments we collected a dataset which is a rather straight forward recognition task. We collected upper body motions of a single subject. The subject was asked to only perform one motion at a time, i. e. the left and the right arm are never performing concurrent motions. We also asked the subject to move back to a rest position between each motion. This gives distinct

[1]Collaborative Research Center 588 on "Humanoid Robots – Learning and Cooperating Multimodal Robots" sponsored by the German Research Foundation - http://www.sfb588.uni-karlsruhe.de

pauses within the motion sequence which makes the segmentation of the data and the recognition easier.

Scenario

We focus on human motions as they appear in a kitchen and food preparation scenario of CRC 588, such as placing objects on a table or pouring a liquid into a container. We discriminate the following ten motion sequences: rolling pastry, pouring water, slicing an apple, grinding coffee, sweeping, grating an apple, stirring, cutting a cake, cutting an apple, and mashing. Each motion is described in terms of a sequence of motion primitives, such as fetching, maneuvering, and putting back an object.

Rolling pastry: *Rest position - Take rolling pin - Rest position - Grasp rolling pin - Roll pastry - Release rolling pin - Rest position - Put rolling pin back - Rest position*

Pouring water: *Rest position - Take glass - Rest position - Take bottle - Pour - Put bottle back - Rest position - Put glass back - Rest position*

Slicing an apple: *Rest position - Take slicer - Take apple - Slice apple - Put apple back - Put slicer back - Rest position*

Grinding coffee: *Rest position - Take grinder - Grasp grinder - Grind coffee - Release grinder - Put grinder back - Rest position*

Sweeping: *Rest position - Take broom and dustpan - Sweep - Put broom and dustpan back - Rest position*

Grating an apple: *Rest position - Take grater - Take apple - Grate apple - Put apple back - Put grater back - Rest position*

Stirring: *Rest position - Take bowl - Rest position - Take spoon - Grasp bowl - Stir - Release bowl - Put spoon back - Rest position - Put bowl back - Rest position*

Cutting a cake: *Rest position - Take cake - Rest position - Take bread knife - Grasp cake - Cut cake - Release cake - Put bread knife back - Rest position - Put cake back - Rest position*

Cutting an apple: *Rest position - Take apple - Rest position - Take knife - Grasp apple - Cut apple - Release apple - Put knife back - Rest position - Put apple back - Rest position*

Mashing: *Rest position - Take bowl - Rest position - Take masher - Grasp bowl - Mash - Release bowl - Put masher back - Rest position - Put bowl back - Rest position*

The above described 10 motion sequences consist of sequences of 5-11 motion primitives, where the total number of different primitives is 49. Many motion sequences share the same motion primitives. The subject was asked to perform these motions in a controlled setting (see Fig. 3.1). The objects were placed at fixed positions on the table. The subject stands at the table in a neutral position, i. e. both hands resting on the table. Starting from this rest position the

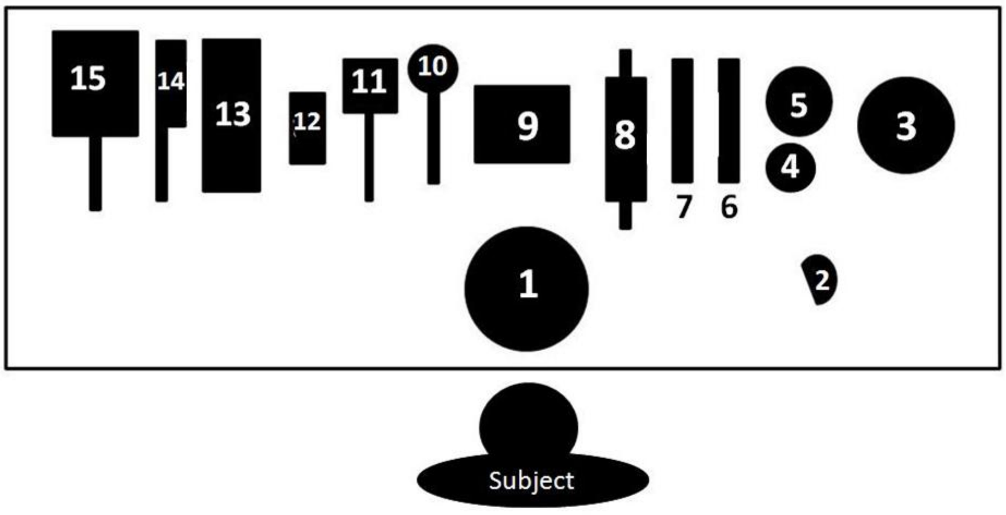

Figure 3.1: The table for the performance of motions in the KIT-S dataset: 1) Working area, 2) Apple, 3) Bowl, 4) Glass, 5) Bottle, 6) Bread knife, 7) Knife, 8) Rolling pin, 9) Grinder, 10) Spoon, 11) Masher, 12) Grater, 13) Slicer, 14) Broom, 15) Dustpan [SGS11].

subject executes a predefined sequence of motion primitives, e. g. fetching an empty glass, fetching a bottle of water, pouring water into the glass, and putting the objects back. In between the motion primitives, the subject is returning to the rest position. An exemplary image sequence of the performed motions is shown in Fig. 3.2.

3.2.2 KIT-F - The Fluent Kitchen Motions Dataset

The scenario of our second dataset is also part of the Collaborative Research Center 588 - Humanoid Robots. Therefore, it also takes place in a kitchen. The motion sequences again comprise taking kitchen utensils from a table, working with them and putting them back to their original positions. The main difference between the KIT-S dataset and the KIT-F dataset is that the subject no longer moved back to the rest position between each motion. Thus motion sequences are fluent. The only constraint is that only one motion primitive is performed at a given time, i. e. no two objects are picked up at the same time. The motion primitives and sequences are as follows:

Pouring water: *Rest position - Take bowl - Take bottle - Pour - Put bottle back - Put bowl back - Rest position*

Grating an apple: *Rest position - Take grater - Take apple - Grate apple - Put apple back - Put grater back - Rest position*

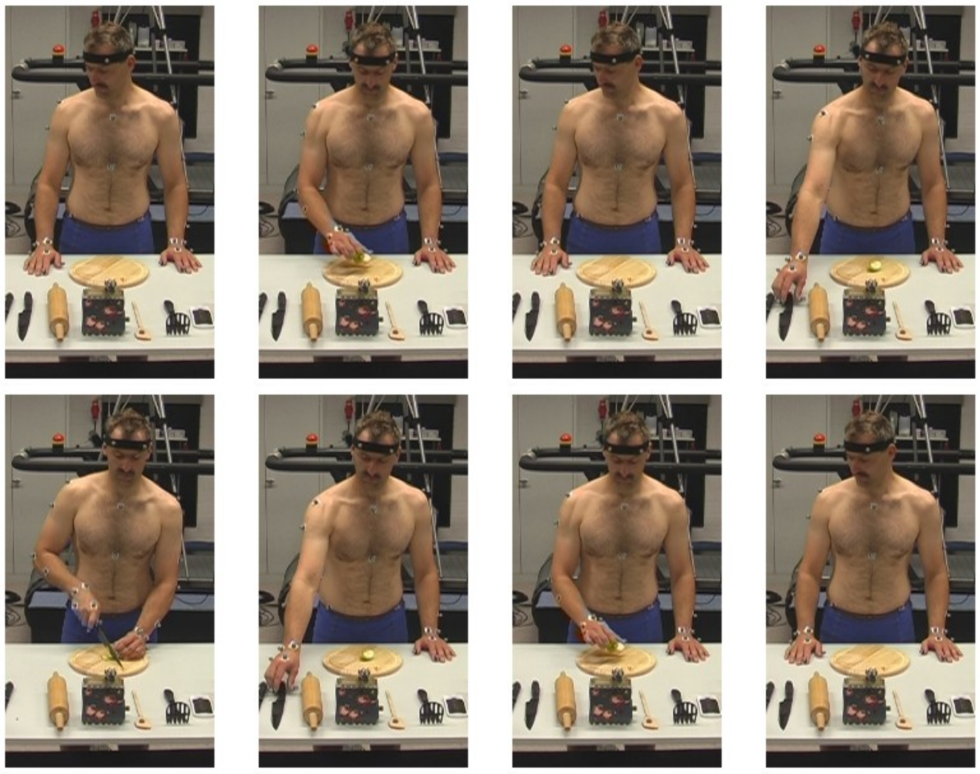
Figure 3.2: Motion sequence for "cutting an apple".

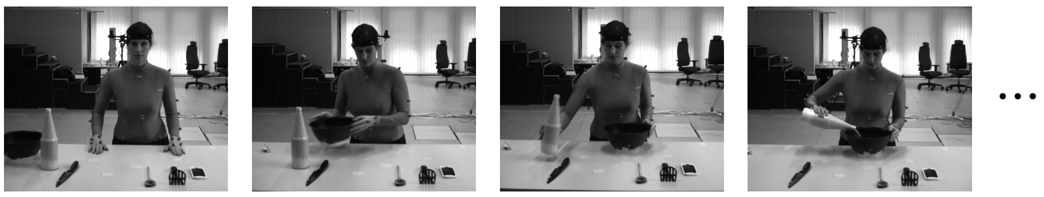

Figure 3.3: Motion sequence for "pouring water".

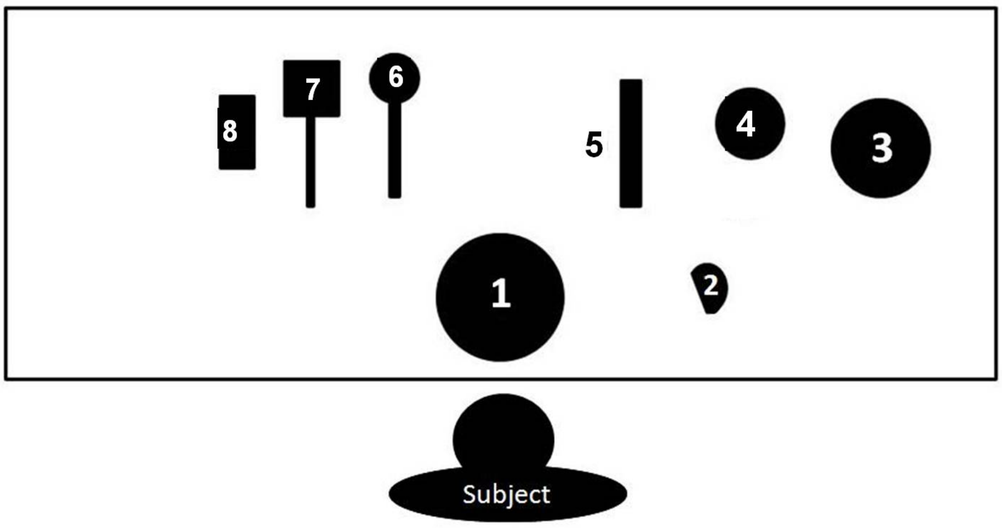

Figure 3.4: The table for the performance of motions in the KIT-F dataset: 1) Working area, 2) Apple, 3) Bowl, 4) Bottle, 5) Knife, 6) Spoon, 7) Masher, 8) Grater [SGS11].

Stirring: *Rest position - Take bowl - Take spoon - Stir - Put spoon back - Put bowl back - Rest position*

Cutting an apple: *Rest position - Take apple - Take knife - Grasp apple - Cut apple - Release apple - Put knife back - Put apple back - Rest position*

Mashing: *Rest position - Take bowl - Take masher - Mash - Put masher back - Put bowl back - Rest position*

If a cyclic motion primitive is involved like stirring or grating, this motion primitive is individually repeated 3-6 times per sequence. The subject was asked to do the motions as natural as possible. The KIT-F dataset was captured in a single session. A female subject performed each task 20 times. The object locations were the same as in the KIT-S dataset, although we used only a subset of the objects for the recordings of the KIT-F dataset (see Fig. 3.4).

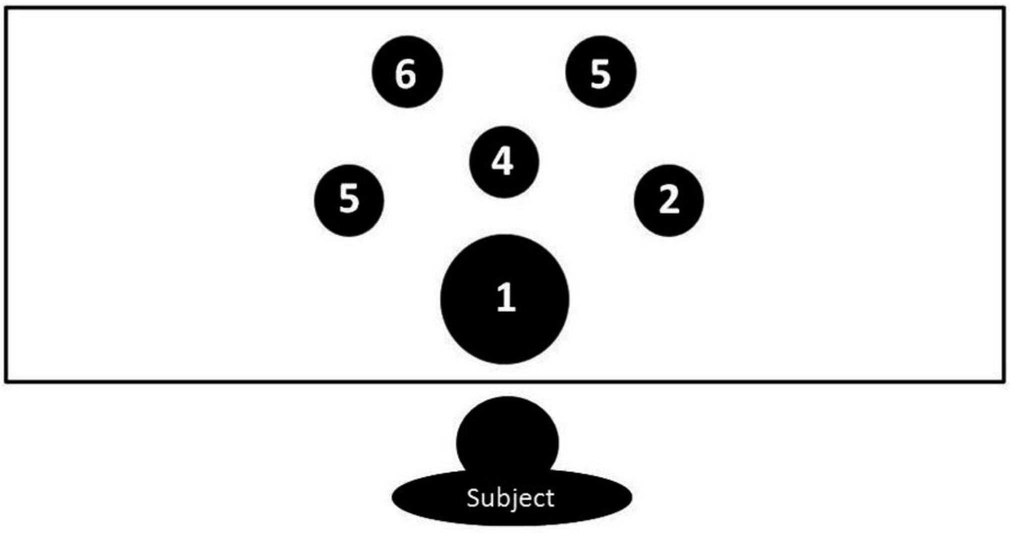

Figure 3.5: Schematic diagram of the possible object position for the KIT-M dataset. Small circles represent object positions at the beginning or end of a motion sequence. The large circle represents the working area.

3.2.3 KIT-M - The Kitchen Motions Dataset with Multiple Positions per Object

We acquired a dataset KIT-M with fluent motions as in the KIT-F dataset for further experiments, with two differences to the KIT-F dataset. In the KIT-M dataset we used varying positions for the objects. Also the subject was allowed to use both hands independently, i. e. she was allowed to perform two independent motions at the same time. Dataset KIT-M contains motion sequences as they happen in every-day life. We collected five types of motion sequences: cutting an apple, grating an apple, mashing, pouring water and stirring. Each type of motion sequence was recorded 75 times, 15 times for each of 5 different object positions. For the 5 different object positions each object was placed once at each of five locations shown in Figure 3.5. Only one object was placed at each position. In order to achieve a good amount of repetitions of each motion primitive we told the subject to grasp an object always with the same hand. In total, we collected more than 1 hour of motion data of one subject for this experiment.

The motion primitives which were recorded for the five different positions are *Take apple, Put apple and grater back, Put apple back, Take bottle, Put bottle back, Take spoon, Put spoon back, Grasp apple during taking knife, Put knife back and release apple, Take grater and apple, Take bowl, Put bowl back, Take masher, Put masher back*. For four of the five positions we also observed the motion primitive *Take knife* *and apple concurrently*. Other motion primitives that were only performed for one locations on the table are *Pour, Stir, Mash, Grate apple, Cut apple, Rest position, Put apple back (center location), Put grater back (right front location).*

3.2.4 The Minta Dataset

For the experiments with the humanoid robot ARMAR, we acquired a dataset with multiple kitchen tasks. The main difference to the previous datasets is the size in terms of recording time and the number of subjects. We also used another set of activities. For the acquisition of this dataset we used a single video camera. The camera view-point was fixed during the recordings to a place in front of a kitchen table, i. e. facing the human. A Point Grey Dragon-Fly Camera with a resolution of 640x480 pixels and a frame rate of 30 fps was used. In the experiments a mix of artificial and day light (9 AM to 8 PM) as well as textured and plain background was used.

The dataset was collected in a kitchen setting, where ten different subjects performed seven kitchen tasks. For each task the subject entered the scene, performed manipulations at the table and left the scene. There were almost no restrictions on how to perform the activities besides always entering the scene from the same direction. Everybody used the same objects for the same tasks and left the scene to the same direction. Every subject performed each task ten times resulting in a total of 700 image sequences. The seven recorded tasks were: lay table, prepare cereals, prepare pudding, eat with spoon, eat with fork, clear table, and wipe table. For a fine-grained recognition of the performed tasks a set of 60 motion primitives, e. g., *Place Object on Table*, *Pour*, or *Stir*, was defined for the motion recognition system.

3.2.5 The Breakfast Dataset

As the Minta and KIT datasets, the Breakfast dataset contains a variety of kitchen activities. We acquired the motion sequences of one subject while performing the activities prepare Bun, prepare cocoa drink, prepare cereals, prepare orange juice, cook scrambled eggs and bake pancake. We captured each of the six activities with five randomly chosen object positions. Each set of object positions was collected three times which gave us a total of about 90 (6x5x3) collected motion sequences. In total the subject manipulated more than 35 objects. Due to the variety of unconstrained kitchen activities this is the most complex dataset we collected for this thesis.

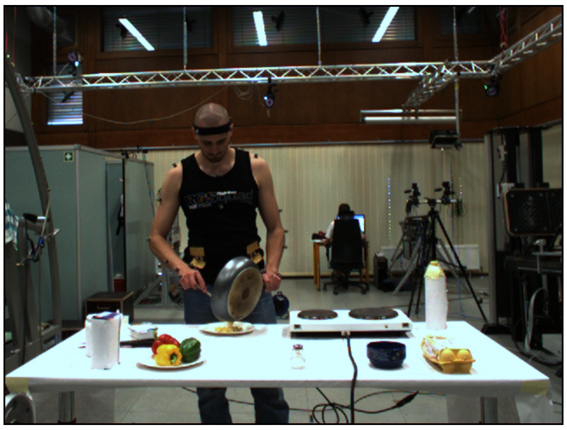
Figure 3.6: Exemplary image of Breakfast dataset

3.2.6 Comparison of Datasets

In this section we give an overview on properties of the collected datasets. The KIT datasets are all rather simple. They were capture to develop the recognition systems. The resulting recognition systems were then applied to the mroe complex datasets, i. e. the Minta dataset and the Breakfast dataset. The Minta dataset is used to show that the recognition system is capable of person-independent online recognition with a humaoid robot. The Breakfast dataset is used for further improvements of the recognition system, specifically the recognition of concurrent and coupled motion primitives. Table 3.7 gives some more details on the properties of the different datasets.

Property	KIT-S	KIT-F	KIT-M	Minta Dataset	Breakfast Dataset
#activities	10	5	5	7	6
#primitives	49	24	82	60	1164
hours of recording	2	0.3	1	10	2
#subjects	1	1	1	10	1
#objects	14	7	7	14	35
data acquisition	marker	marker video	marker	video	marker
fluent	no	yes	yes	yes	yes
concurrency	no	no	yes	yes	yes
object positions	fixed	fixed	varying	varying	varying

Table 3.7: Properties of the collected datasets.

4

Sequential Recognition System for Human Motion

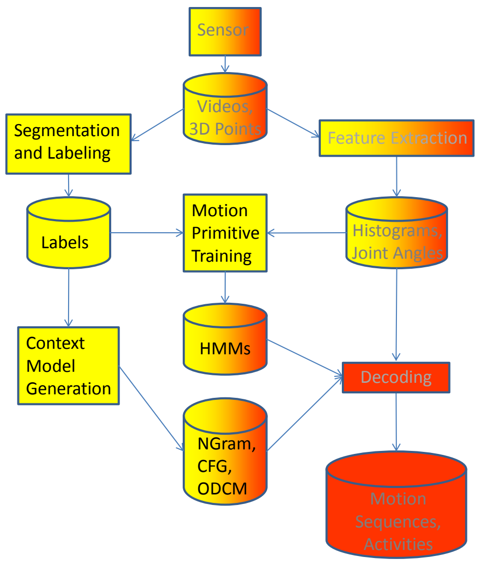

Figure 4.1: The training part of the overall recognition system.

To perform human motion recognition of meaningful motion primitives, e. g. for describing human activities in detail or for doing intention recognition we have to tackle the following tasks as described in Chapter 1:

1. Define a motion primitive alphabet
2. Model motion primitives
3. Extract and select appropriate features
4. Model the motion context
5. Online recognize motion sequences

In this chapter we will explain how we carried out those tasks for sequential human motion recognition. Figure 4.1 shows which components of the system are concerned with carrying out these tasks. In the following we will describe relevant work from literature and how our system is designed that meets some of the existing challenges.

For the generation of our models we assume to have sensor data for a set of motion sequences. A motion sequence usually contains the motions that are performed during a specific activity. As described in chapter 2 we need to split the sensor data of a motion sequence into smaller units to model the motion sequences appropriately. We call those smaller units *motion primitives* which are used to build motion sequences. Firstly, we will describe the system developed to define an appropriate set of motion primitives for a given set of activities. Secondly, we will explain how we segment the sensor data into motion primitives before we explain how we train motion primitive models with the segmented data. Thirdly, we will describe our proposed solutions on how to model the context of motion primitives. Besides using traditional context models, we developed an object dependent context model which allows to incorporate object knowledge into the recognition process of motion primitives.

4.1 Related Work

The focus of our work is on online recognition of human motion primitives, e. g. for a humanoid robot, instead of high-level activity recognition. Recognizing smaller units like motion primitives instead of activities allows a wider range of applications based on the recognized motion primitives such as intention recognition, imitation learning and programming by demonstration. For the recognition of motion primitives it is appropriate to use a single-layered approach, i. e. an approach that represents and recognizes human activities directly based on sequences of images (or other sensory input). Due to their nature, single-layered approaches are suitable for the recognition of motion primitives with sequential characteristics. In contrast, hierarchical approaches represent high-level human activities by describing them in terms of simpler activities or motion primitives. Recognition systems composed of multiple layers are constructed, making them suitable for the analysis of complex activities. However, we will show that we can improve the recognition of motion primitives by applying some methods of hierarchical approaches. Most of the approaches that can recognize concurrent and coupled motion primitives are hierarchical approaches. Due to their modeling techniques most of them are inflexible concerning new activities or even only slightly differently performed activities. They also have difficulties performing online recognition. We will for now stick to single-layered approaches and explain the hierarchical approaches in chapter 6. Aggarwal and Ryoo divide the single-layered approaches into space-time approaches and sequential approaches.

4.1.1 Space-Time Techniques

Space-time approaches are approaches that recognize human activities by analyzing space-time volumes of activity videos. The simplest way would be to concatenate the images of each training video to retrieve a volume and build a volume template for each activity. Those volumes can then be compared to another volume to recognize the activity. Besides the pure volume representation there are other variations of the space-time representation. The activity may be modeled as space-time trajectories or other trajectories instead of modeling the whole volume. To model each activity a set of trajectories is used. Those can then be compared to the trajectories of an activity to be recognized. The second alternative are features extracted from the volume or the set of trajectories. According to [AR11] "Space-time approaches are suitable for the recognition of periodic actions and gestures. Basic approaches using space-time volumes provide a straight-forward solution but often have difficulties handling speed and motion variations. Recognition approaches using space-time trajectories are able to perform detailed-level analysis and are view-invariant in most cases. However, 3-D modeling of body parts from video, which is still an unsolved problem, is required for a trajectory-based approach to be applied. The spatio-temporal local feature-based approaches are getting an increasing amount of attention because of their reliability under noise and illumination changes. The major limitation of the space-time feature based approaches is that they are not suitable for modeling complex motion sequences. The relations among features are important for a non-periodic activity that takes a certain amount of time which most of the previous approaches ignored."

4.1.2 Sequential Techniques

In contrast to the space-time techniques, sequential approaches consider an activity as a sequence of observations (i. e. features). Instead of comparing point trajectories or features of those trajectories they convert the activity into a sequence of feature vectors. To recognize an unknown activity it is also converted into a sequence of feature vectors (e. g. joint angles), describing the persons motions per time step. Once feature vectors have been extracted, sequential approaches analyze the sequence to measure how likely the feature vectors are produced by the person performing the activity. If the likelihood between the sequence and the activity class (or the posterior probability of the sequence belonging to the activity class) is high enough, the system decides that the activity has occurred.

Aggarwal et al. [AR11] divide the sequential approaches into two categories using a methodology-based taxonomy: exemplar-based recognition approaches and state model-based recognition approaches. Exemplar-based sequential approaches describe classes of human actions using training samples directly. They maintain either a representative sequence per class or a set of training sequences per class and match them with a new sequence to recognize its activity. State model-based sequential approaches are approaches that represent a human action by constructing a model which is trained to generate sequences of feature vectors corresponding to the activity. By calculating the likelihood (or posterior probability) that a given sequence is generated by each activity model, the state model-based approaches are able to recognize the activities. In general, sequential approaches consider sequential relationships among features in contrast to most of the space-time approaches, thereby enabling detection of complex activities (i. e. non-periodic activities such as sign language or kitchen activities).

The main advantage of exemplar-based approaches in comparison with state model-based approaches is that they are able to cope with little training data. However, state model-based approaches are able to make a probabilistic analysis on the activity, which enables the modeling and recognition of large sets of motions. A state model-based approach calculates a posterior probability of an activity occurring, enabling it to be easily incorporated with other decisions. In contrast to the training of the exemplar based approaches the training of the state model-based approaches most often does not have a closed solution which can make the training procedure very expensive. An important advantage of the state model-based approaches is the ability to model variations in the data such as temporal and spatial variations. One of the limitations of the state model-based approaches is that the amount of required training data increases with the complexity of the activity. Therefore, it is necessary to divide the activities into simpler motion primitives to reduce the need for more training data.

4.1.3 Motion Primitives

Given the continuous nature of motion, there is an unlimited number of motion sequences that can be performed. Therefore, it is impossible to collect enough traning data to model each motion sequence independently. Boundaries of motion primitives are often arbitrarily defined, making it difficult to automate the motion segmentation process [KTS03]. However, humans tend to perceive motion in terms of discrete motion primitives [GP03, RFG01] and thus motion segmentation is still considered useful for several applications, including imitation learning [PHAS09] and human motion recognition [GKWS09].

When modeling human motions one of the main questions is what entity of a motion should be modeled as one unit. Obviously, human motions have to be split into smaller pieces, since we can not model all motions someone is performing during a day as one entity. As humans we can name the parts of motions and give them a label. Neurological evidence shows that humans also generate motions in terms of motion primitives [RC05].

There is also evidence in the computer science literature that it is a good idea to model motions in terms of primitives [FMJ02, Mat02, KHN09, Bob97, SKKK11]. The definition of motion primitives depends on the application and is often done manually [VKKL07, BA01].

4.1.4 Sequential State Model-based Techniques

State model-based approaches represent human motions as a model composed of a set of states. They are statistical approaches that are designed to generate a sequence of feature vectors with a certain probability. For each model, the probability of the model generating an observation sequence is calculated to measure the likelihood between the motion model and the observation sequence. State model-based approaches can easily be incorporated with other decisions and model the statistical nature of the motion primitives. Therefore, we use a state model-based approach for our recognition of complex human kitchen activities. The most popular method for state model-based recognition are Hidden Markov Models (HMMs). HMMs are widely used and have been proven to be well suited for continuous human motion recognition. One of the earliest works is from Yamato et al. [YOI92] who recognize the tennis actions forehand stroke, backhand stroke, forehand volley, backhand volley, smash, and service. They use vector quantization to create a feature vector for each binarized image. They train one discrete HMM for each action. Tennis actions of three players are recognized by solving the evaluation problem [Rab89]. They get good recognition rates even for player independent recognition. Another early work has been done by Starner and Pentland [SP95]. They recognize American sign language (ASL) based on video input of a tracked human hand (with/without a colored glove). Their system extracts features describing shapes and positions of the hands. Each word of ASL is modeled as one HMM. They perform continuous training and recognition of sentences of ASL. The recognition is performed with 5 frames per second using the Viterbi algorithm. Bobick and Wilson [BW97] recognize gestures based on state models

similar to HMMs. They represent each gesture as a trajectory in 2D-XY space describing the location changes of a hand. The trajectory is modeled using a sequence of fuzzy states each represented by a single Gaussian directly computed by a training sample trajectory. For the recognition of gestures with their system they use the dynamic programming approach. Applying their framework they have successfully spotted and recognized two different types of gestures: 'wave' and 'point' from sequences of motion data. They acknowledge the relation between their work and HMMs, but comment on one important distinction, namely, the existence of a prototype. The time-collapsing procedure discussed in their paper yields a reasonable prototypical model, even if only one training example is present. The statistical nature of an HMM precludes a rapid training phase: Many free parameters need to be adjusted for the model to have some relation to the action. In the approach outlined in [BW97], a model is available immediately.
As a generalization of HMMs some researchers use *Dynamic Bayesian Networks* (DBN). Park and Aggarwal [PA04] recognize actions and interactions using a DBN. They extract body part blobs from video images, estimate the body pose and recognize interactions such as two people approaching each other, two people shaking hands, punching or pushing. Another paper that uses DBNs is the one of Ryoo and Aggarwal. In [RA07] they extend the approach to also considering relations between objects and motions.

4.2 Marker-Based Features

The basis to all high-performing state-of-the-art motion recognition systems are appropriate features.

4.2.1 Motion Capture

Human motions can be captured with several methods, e. g. marker-based methods, methods based on accelerometers, as well as video-based methods. A short overview is given in [Gär09]. In the following section we will describe motion capture with marker-based methods. They are suitable for fine grained motion capture, as it is necessary to recognize motion sequences in a kitchen environment. A marker-based motion capture system simultaneously tracks the positions of multiple markers attached to a human body and provides their positions in 3 dimensional space. Using marker positions as features for human motion recognition is suboptimal since the marker positions of a performed motion dependent on the person's position and orientation. However, the description of an actual motion trajectory should be position and orientation in-

dependently. Therefore, we convert the marker positions to joint angles. Thus, the goal of the motion capture is a mapping of the marker trajectories onto a kinematic human body model to retrieve joint angles.
To capture human motions we use a Vicon motion capture system[1]. We attach reflecting markers to the subject's upper body, head and arms. Reflecting infra red light is simultaneously recorded with 10-14 MX13 Vicon cameras, which are arranged around the subject. The Vicon system outputs 3-dimensional positions and labels of the markers for each time step. We use a very similar marker set (see 4.2) as the Plug-in Gait marker set from Vicon which is the same as the CRC model in [Sim09]. For capturing upper body motions the upper body marker set from Vicon's Plug-In-Gait model is well suited. For body model scaling and the joint angle reconstruction the markers are divided into static and dynamic markers. The static markers are placed on landmarks, i. e. well defined positions on the human body, which have only little relative displacement of the skin in relation to the bones. With this markers we can automatically scale the body model to match the anatomy of the subject. Dynamic markers are placed on the center of body segments and are additionally used to the static markers when reconstructing the joint angles providing redundancy. After scaling the body model the dynamic markers can be calibrated automatically using the Vicon software. The position of the markers can be taken from the manual of the Plug-In-Gait model.

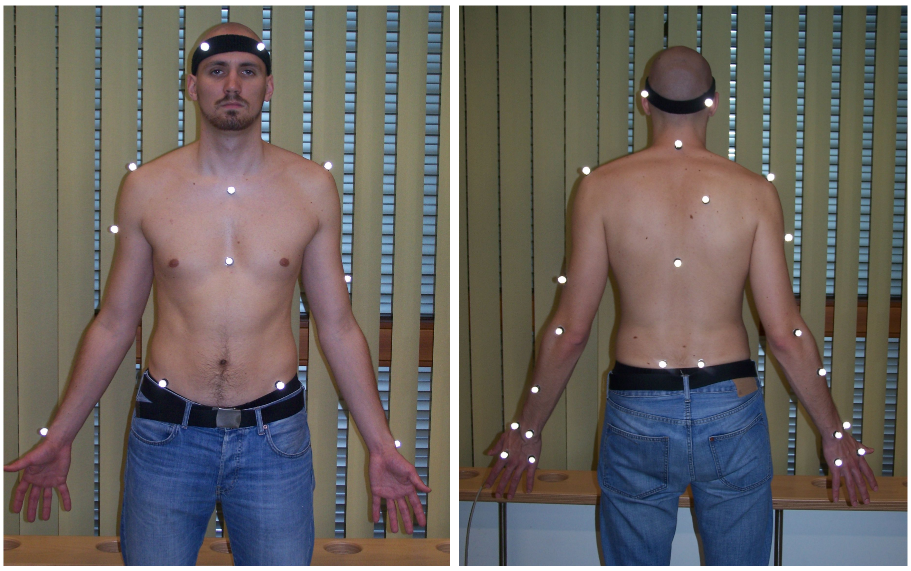
Figure 4.2: Marker set used for motion sequences capturing (See Plug-In-Gait from Vicon).

[1]www.vicon.com

Challenges

When capturing markers with a camera based tracking system, two errors can occur: wrong marker labeling and occluded markers. The marker labels can be assigned to the wrong marker or even to a location where no markers exist. This problem can be avoided by good parameter tuning of the Vicon system, controlled environmental conditions (e. g. light) and with human intervention when detecting wrong marker labels. When detecting an erroneous marker label, the marker can be relabeled using the Vicon system either automatically or if this does not work manually. When Vicon returns gaps in the marker trajectories, those gaps can either be interpolated semi-automatically using the Vicon system or they can simply be ignored when using a kinematic body model for joint angle reconstruction. The body model can compensate most markers for calculating joint angles if only very few are missing at the same time.

4.2.2 Joint Angles

Kinematic quantities of a human body (e. g. joint angles) can be calculated using various automatic methods with different degrees of power. A simple approach is to draw virtual lines between the respective markers and to calculate the joint angles between those lines. The approach used in Vicon's motion capture system works in this way. For a more precise estimate it calculates planes through multiple markers. The main problem of both approaches is their sensitivity to the correct positioning of markers on the human body. If a marker is missing, the joint angle reconstruction algorithm might fail. For our system we used an alternative approach, which is a motion mapping using optimization methods. For this approach we need a kinematic model with joints and model marker positions relative to the joints. Through optimization we can minimize the distance between the measured marker positions and the model's marker positions and retrieve the desired quantities (e. g. joint angles) [Ste08, Gär09]. This approach is very robust against missing and wobbling markers. The approach can be improved by scaling the body model based on markers positioned at landmarks to better fit the anatomy of the human performing motions. Before applying the motion mapping we scaled our model as described in [Köh08].
The problem that has to be solved for motion mapping is a classical non linear optimization problem, which can be solved with a large number of standard methods. For the calculations in this work we used non linear optimization methods of the Matlab Optimization Toolbox, which yielded numerically stable and efficient solutions. An important impact on the quality of the solution have

the definitions of the relative marker positions. For reconstruction we used a rigid upper body multibody model of the human skeleton with 30 DOF. The joints are wrist (2 DOF), elbow (2 DOF) and shoulder (3 DOF) of both arms as well as joints at the lower (neck: 2 DOF) and upper (skull: 3 DOF) neck, and at the lower (lumbar: 2 DOF) and upper (thorax: 3 DOF) spine.

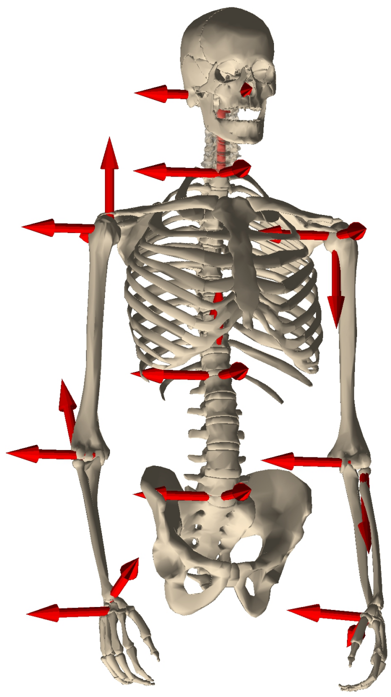

Figure 4.3: The kinematic body model used for joint angle reconstruction from marker points.

As a result, the motion mapping outputs per time step one feature vector consisting of 30 coefficients, 24 joint angles of the kinematic body model as well as 6 values for global position and orientation of the body model. The joint angles of the 24 DOF are used as input features for our motion recognition system.

4.3 Motion Primitive Models

4.3.1 Functional Decomposition of Human Activities

The first step when using motion primitives as the basic unit for motion recognition is the definition of a vocabulary of human motion primitives. Currently, the various systems using motion primitives define them depending on the application the motion recognition is used for. In [GSFS10] we propose a method for standardizing the decomposition of motion sequences into motion primitives. The motion sequences are decomposed based on functional properties. In comparison to biomechanical decomposition which is used in most automatic motion segmentations, we assume to get more meaningful primitives which can be used for higher level reasoning about the current activity. The functional motion primitives can still be used for most tasks the automatically retrieved primitives are used for, e. g. motion synthesis.

As many others we decompose activities into motion primitives manually. This has several advantages:

1. Assures motion primitives with human understandable labels and meanings
2. Assures motion primitives with functional meaning
3. Avoids the lack of grounding the motion primitives
4. Can be assisted by automatic methods (clustering into sets of motion primitives with similar properties, e. g. the same motion primitive for different objects that are handled the same way)

Various different approaches can be found in the literature as to what should be seen as a motion primitive and to how these motion primitives can be modeled [PHAS09]. Various types of motion primitives are used ranging from low-level motions, e. g. moving the hand forward, up to complex motions such as setting the table. In many scenarios it is not sufficient to know the kinematic or dynamic parameters of a motion, since the goal of the motion might not be reached although the execution of the motion is correct. For example, if a robot wants to grasp a glass, it has to make sure that the glass is properly grasped. It is not sufficient, if the robot only performs a motion trajectory based on kinematic and dynamic parameters, which does not result in grasping the glass.

We bridge this gap by looking at the problem from top-down. We belief that a system for decomposing motions into motion primitives should take the goals of a motion into account. It is not suitable to decompose a motion in an arbitrary way, since the goals of the motions have to be fulfilled to perform the motion properly. To the best of our knowledge there is no system for decomposing arbitrary daily-life motions into motion primitives based on functional information. In our work we propose a system which allows us to retrieve a motion decomposition into motion primitives based on functional knowledge. So far relatively few papers dealt with higher abstraction levels of human motions which touch the border of semantics. Some approaches segment the data based on object relationships [AAWD10, SCH08]. We do not apply these approaches since we also want to segment communication gestures which are not object related. Another step in the direction of functional motion representation has been done by Guerra-Filho and Aloimonos [GF06]. They started to close the semantic gap between a WordNet and sensorimotor information by grounding a set of primitive words. Similar ideas have been presented by Ivanov et al. [IB00], whereas they assume a natural decomposition of motions into low-level primitives and higher-level semantic information.

In this section we propose a systematic approach for manually defining motion primitives based on activity goals as presented in [GSFS10]. This approach allows to systematically analyze a given set of activities and retrieve a functionally plausible set of motion primitives. Activities are separated into motion primitives which fulfill subgoals concerning the overall activity goal. Depending on the flexibility of performing an activity and the application, the retrieved set of motion primitives might vary.

The set of motion primitives is the basis for building a motion recognition system. It can be used to segment motion trajectories or to create a motion grammar for the set of activities. The procedure consists of four steps:

1. Retrieve motion contexts (e.g. available objects and their properties)
2. Find functional primitives
3. Retrieve functional and temporal relationships between primitives
4. Define temporal and positional constraints for primitives

These steps are described in more detail in the following section. Before analyzing the motion sequence in detail, the motion context should be considered. As motion context we consider the humans and objects in the environment. The motion context needs to be determined since this information gives the solution space, e. g. one can only grasp an apple if it is within reach otherwise one needs to walk towards the apple first. After the solution space is defined, an analysis of the motion sequence itself has to be carried out. The individual elements of the motion are primitives, which carry a specific function according to the overall goal of the motion sequence. These primitives therefore are called *functional primitives*. We distinguish between *main functional primitives* and *supporting functional primitives*. Main functional primitives determine the goal of the motion. They appear at least once during the motion sequence. In contrast, supporting functional primitives are not directly related to the goal of the motion but rather functionally dependent on other functional primitives. *Preparatory supporting functional primitives* improve the situation for subsequent functional primitives, e. g. Picking up a knife. In contrast, *assistant supporting functional primitives* improve the execution of concurrent functional primitives, e. g. Steadying an apple while cutting it. Finally, *transitional supporting functional primitives* transform the present motion situation into a new situation,

e. g. Putting the knife back to be able to do something else [Göh92]. Besides, this functional relation there is a temporal relation (Fig. 4.4). Temporal relations describe the order in which motion primitives have to be performed. In Fig. 4.6 the primitive *Pick and Place apple* has to be performed before *Pick up knife and prepare cutting* since as soon as we have grasped the knife and carried it to the cutting position we can no longer carry the apple with the same hand. The two axes of the diagram in Fig. 4.4 represent the functional and temporal relationships of motion primitives. Lines and arrows are used for a more precise representation of the dependencies. An arrow specifies a functional relationship between two motion primitives while a line specifies a temporal relationship.

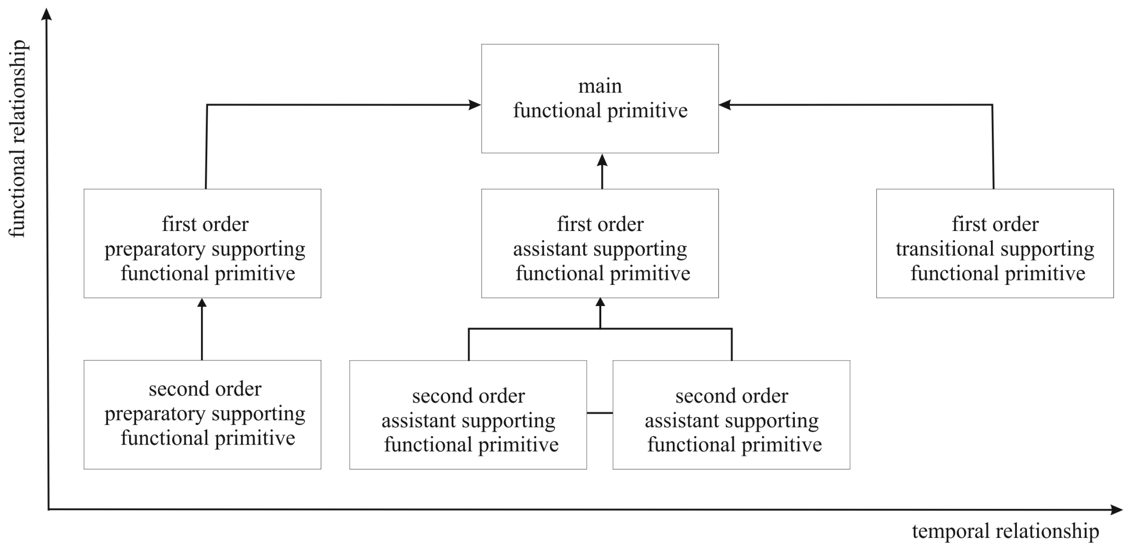
functional relationship
main functional primitive
first order preparatory supporting functional primitive
first order assistant supporting functional primitive
first order transitional supporting functional primitive
second order preparatory supporting functional primitive
second order assistant supporting functional primitive
second order assistant supporting functional primitive
temporal relationship

Figure 4.4: Types and relationships of functional primitives [GSFS10].

There are three possibilities for the decomposition of a motion sequence into functional primitives [Göh92]. In the case of the *inductive functional structuring* observed motions of the performer are the starting point for the decomposition. These motions are performed because they fulfill certain functions in the context of the motion goal. The decomposition into functional primitives that fulfill certain functions is done during the functional decomposition of the observed motions.

The origin of the *deductive functional structuring* are a motion goal and a motion context instead of observed motions. Thereby a motion goal has to be decomposed into sub-goals and according actions of a performer have to be defined.

A third possibility is a *combined functional structuring* which corresponds to a synthesis of the inductive and deductive structuring [Göh92].

For each of the identified functional primitives *temporal* and *positional* constraints have to be examined. This has to be done for the beginning, the middle part and the end of each functional primitive, e. g. it has to be considered where the hands of a person must be, at the beginning, during and at the end of each functional primitive. It also has to be specified, whether a motion primitive has to be performed in a certain period of time. Finally, the segmentation of a motion into different functional primitives can be applied. The procedure does not guarantee that the decomposition always results in the same motion primitives and the same structure. The procedure is mainly a possibility to retrieve functionally plausible motion primitives, whereas the plausibility may depend on the desired application.

Functional Decomposition of Complex Human Activities with Sequential Motions

In this section we apply the functional decomposition procedure to a daily-life motion sequence, cutting apple (see Fig. 4.5). In our setup we assume that a subject stands in front of a table. The result of the decomposition strongly depends on the environmental conditions, e. g. present objects. The objects involved in the activity "cutting an apple" are an apple, a knife, a table and a cutting board. At the beginning and at the end of the motion sequence the objects are placed on the table.

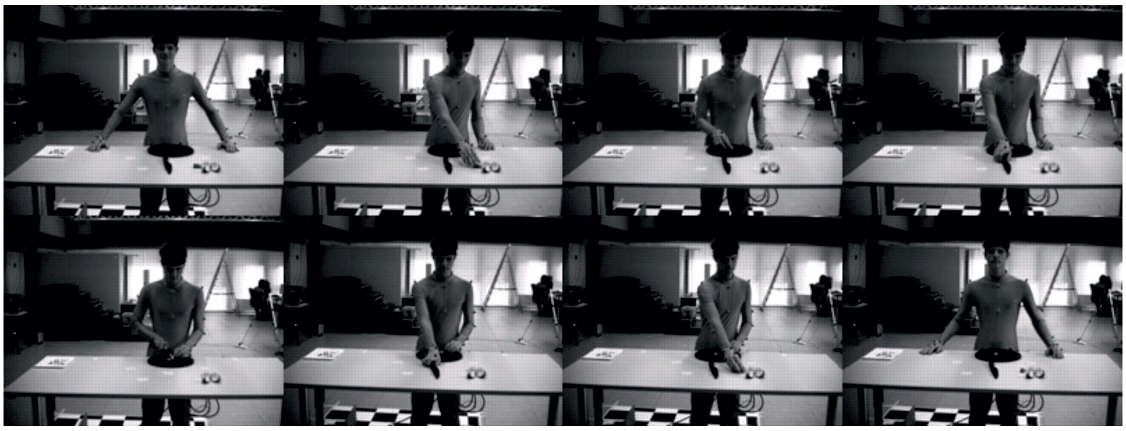

Figure 4.5: Complex human motion sequence in a kitchen scenario: "Cutting an apple".

We use a *deductive functional structuring* to decompose the activity into motion primitives, since this allows us to define motion primitives independent from recording motion data. Fig. 4.6 shows two different decompositions of the activity "cutting an apple". For simplicity reasons we assume that only one motion primitive is performed at a time. Multiple motion primitives performed at the same time will be addressed in Section 6.2.1. The main functional primitive is *Cut apple*. The dotted line represents an arbitrary number of repetitions of the primitive, e. g. cutting an apple multiple times. In order to be able to cut the apple someone needs to get the necessary objects, a knife and an apple. Those motion primitives have no temporal constraints besides the temporal structure shown in Fig. 4.6. At the beginning and at the end of the motion sequence the positional constraints depend on the object locations. The positional constraints during the motion primitives are: FP4: cut at the position of the apple, other hand is at the apple. Constraints at the end of motion primitives are: FP1: apple placed on the cutting board, FP2: knife located on top of apple, FP3: hand at apple, FP6: knife placed at original position, FP7: apple leftover at original position.

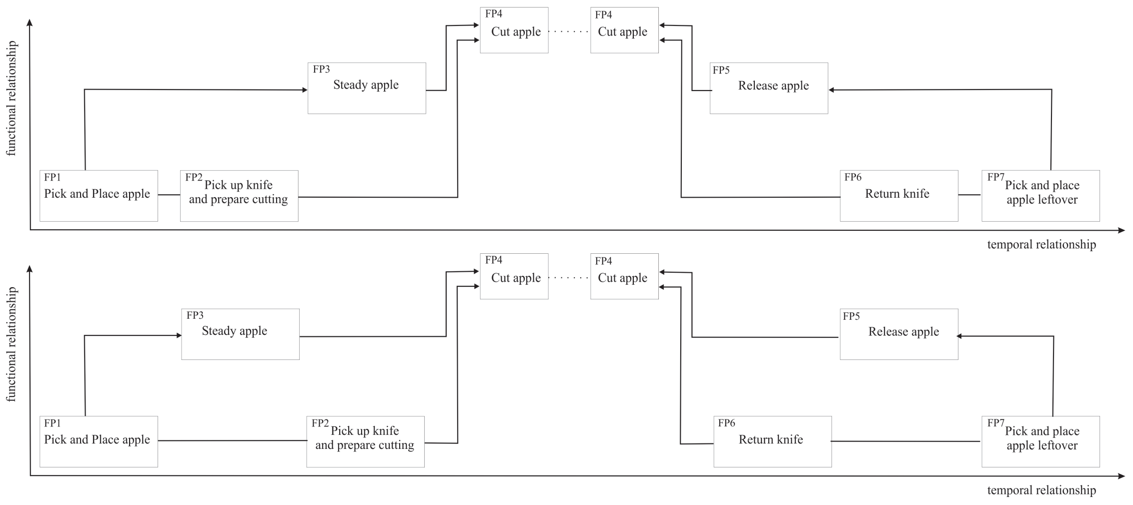

Figure 4.6: Two functional and temporal structures of a complex human motion sequence in a kitchen scenario: "Cutting an apple".

The functional decompositions could be arbitrarily complex. When the subject already picked up the apple someone could call for him from another room, he could go there, come back and then cut the apple. In our experiments we generated the simplest functional decomposition which fully describes our data.

Applying the Functional Decomposition to Segmentation and Grammar Building

The results of the functional decomposition can be applied to two tasks:

- Use it as basis for segmenting the data into motion primitives

- Create a motion grammar describing the typical sequences of motion primitives

Using it for segmenting data a generalized set of motion primitives needs to be defined based on the retrieved ones, e. g. possible start and end positions for the different motion primitives. We retrieved the positions by extracting all possible positions from our training data. The result is a set of motion primitives that can be actually used to segment captured motion sequences into the respective motion primitives. For the segmentation we also need to define how to find the beginning and the ending for each motion primitive, e. g. the motion primitive *Carry* starts when the subject has grasped the object and ends when he starts to release the object. Applying motion primitives as units for segmentation is described in the following Section 4.3.2, the application to grammar building is introduced in Section 4.4.

4.3.2 Segmentation of Human Motion Sequences

Related Work on Automatic Segmentation

In the previous section we described a method to derive and define a set of motion primitives for a given scenario. In order to build models for these motion primitives, we need to segment observation sequences regarding the time slices in which each motion primitive has happened. This segmentation can either be done manually or automatically.

Although many researcher segment their data manually due to reasons that will be discussed below, a lot of work has been done to automatically segment data into motion primitives which we will now briefly discuss.

Most of the automatic approaches segment motion data based on some extremum criteria [WLZ05, WSXZ01] e. g. minimum torque over all joints. After segmentation the motion segments are clustered into a set of motion primitives. For Human motion segmentation it is necessary to find temporal segment boundaries of body parts with many degrees of freedom. Methods that have been proposed often depend on an extremum criteria such as points in time with minimum velocity. The motivation is that human motion does not happen in constant speed since in biological systems the spring effect of muscle is usually exploited to minimize energy costs during movement. Energy can also be saved through turning joints as few as possible, i. e. joints near the end of limbs rotate more than those near the root of body. These characteristics introduce a natural physical measurement of elementary behavior – the torque. Considering the body as articulated, the torque of the whole body at time t is the sum of torques of all joints. The segment boundaries are at the moments with local minima of the torque [WLZ05].

Another approach is governed by the idea that segmentation involves searching natural inconsistent points within the whole observation. A change in the type of human movement usually causes dips in velocity or abrupt variations in moving direction. Wang et al. [WSXZ01] exploit this by finding the local minima of velocity and local maxima of change in direction. The minima (maxima) below (above) a certain threshold are selected as segment points. In practice, they found the calculation of change in direction to be prone to noise. As a result, they apply a Gaussian smoothing filter to reduce noise.
Besides using maxima or minima in velocity or similar as boundaries Shum et al. [LWS02] use linear dynamical systems to find the segmentation boundaries and Jenkins [JM04] proposes a kinematic centroid segmentation. For the kinematic centroid segmentation the center of a body part is observed. When the distance of the center to the original center position is highest, a segment boundary is inserted. Afterwards, the current center position is used as reference and the procedure is repeated.
Most of the automatic segmentation approaches do only return segment boundaries but they do not assign the segments to a class label. Therefore, the segments are usually clustered, e. g. using Nearest Neighbor Clustering with Dynamic Time Warping (DTW) [OSU99] or Hierarchical Clustering using HMMs [WSXZ01].
Segmenting and clustering the data automatically into primitives does not require any human intervention, which makes it feasible for online learning. The main focus of this work is not on online learning but on reliable recognition of human motion, e. g. for further semantical processing of the recognized motion primitives. Therefore, we use motion primitives defined with the functional decomposition described in the previous section. This also solves the problem of assigning motion primitives a human understandable label and meaning which makes the recognition system and its output easier to interpret for a human user. Thus, we decided to use manual segmentation for this work which will be explained in more detail in the following Section 4.3.2.

Manual Segmentation

In this section we describe the approaches used for manual segmentation.
The methods for manually segmenting observation sequences can be separated into different types. Gaps between motion primitives or overlapping of motion primitives can be allowed depending on the application. In our case we want to achieve a segmentation where each point in time belongs to exactly one motion primitive.
According to Kahol et al. [KTS03] the segmentation of motion trajectories is done differently by different persons. Therefore, it is necessary to clearly define

the criteria to segment the motion primitives to get congruent results. The first step when segmenting a set of observation sequences is the definition of criteria for motion primitive boundaries.

For the actual segmentation we visualize the current time step in a motion trajectory. Since we do not allow gaps or overlap we need to specify the motion primitive label and the beginning of each motion primitive while the ending is given implicitly through the start point of the successor primitive. The name of the motion primitive is selected from a list (see Figure 4.7) of motion primitives as defined by the functional decomposition and is used as label for the corresponding segment.

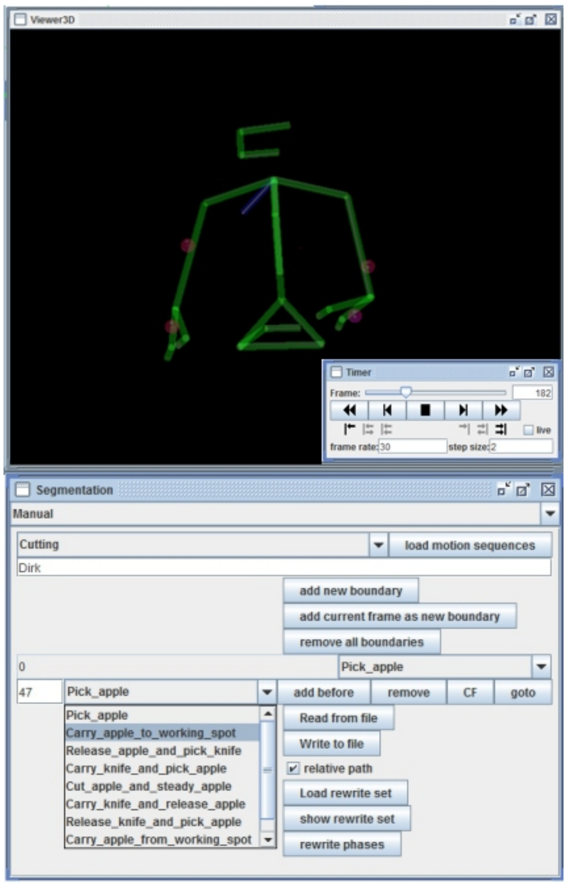
Frame: 182
frame rate: 30
step size: 2
Manual
Cutting
Dirk
0 Pick_apple
47 Pick_apple
Viewer3D
Timer
live
Segmentation
load motion sequences
add new boundary
add current frame as new boundary
remove all boundaries
add before
remove
CF
goto
Pick_apple
Carry_apple_to_working_spot
Release_apple_and_pick_knife
Carry_knife_and_pick_apple
Cut_apple_and_steady_apple
Carry_knife_and_release_apple
Release_knife_and_pick_apple
Carry_apple_from_working_spot
Read from file
Write to file
relative path
Load rewrite set
show rewrite set
rewrite phases

Figure 4.7: User interface for motion sequences segmentation and labeling.

With this approach the segmentation of the data takes 20-40 times real-time for a human annotator, which makes the manual segmentation and labeling of the data a tedious and time consuming task. In [SGS11] we showed that the same recognition rates can be achieved by segmenting only a small amount of data while labeling the rest of the data without specifying the motion primitive boundaries. The labeling without specifying the boundaries takes only about 50% of the time of segmenting the data.

4.3.3 Model Initialization and Training

After segmenting and labeling the observation sequences into motion primitives we can train our recognition system and evaluate its recognition performance. As stated above we use statistical sequence modeling in terms of Hidden Markov Models [Rab89] to model each motion primitive. Hidden Markov Models (HMMs) allow a better modeling of the temporal variations of the motions than templates. For modeling the emission probabilities of the HMM states we use Gaussian Mixture Models (with diagonal covariance matrices).

Initializing the Recognition System

Before we can iteratively improve (train) our HMMs we need to initialize the parameters of the HMMs, namely the means, covariances and weights for the GMMs as well as the state transition probabilities. It is possible to use random initial values and train those to fit the training data. However, as we have manually segmented and labeled data it is better to use the motion primitive boundaries as an initial alignment of the training observation sequences and the HMMs. Therefore, we know which feature vector belongs to which HMM. We could assume that each HMM state models the same number of consecutive feature vectors (which is of course wrong, but it is a good initial guess). With this assumption we partition each observation sequence belonging to an HMM into equally long pieces and assign one piece to each HMM state. Using pieces belonging to a certain HMM state we can estimate the Gaussian Mixture Model using the K-Means algorithm (see e. g. [HAH01] Chapter 4.4) or a fuzzy version of the K-Means algorithm which is called Neural Gas clustering. Both cluster the data into K clusters. The number of clusters is equal to the number of mixture components in the GMM. We then use the clustered data to calculate the Gaussian Mixture Model. The main difference between the K-Means clustering and the Neural Gas clustering is the fact that Neural Gas does assign feature vectors to multiple clusters, which takes more processing time but is more robust against a poor choice of initial cluster centroids. We now have a set of Gaussians and their weights for each HMM state. In order to fully describe the HMM we also need to specify the HMM topology and the transition probabilities. We always use a left to right HMM topology with two transitions for each state. One transition to the successor state and a self loop. The transition probabilities are set to 0.5 for each transition and are not trained.

Training the HMM Parameters

The initial HMMs can be further improved. We use standard HMM training [Rab89] on observation sequences of isolated motion primitive or forced alignment training on sequences of motion primitives to improve the GMMs. Note that in both cases we do not train the HMM transition probabilities. When using forced alignment training we build a sequence HMM representing a training sequence of motion primitives. The sequence HMM is simply a concatenation of the motion primitive HMMs of the given motion sequence. We then align the training observation sequence with the sequence HMM using the Viterbi algorithm as described in Section 2.6.1. The training observation sequence is forced to follow the sequence of motion primitives the sequence HMM is build of. Therefore, the method is called forced alignment. If not stated otherwise, we used forced alignment training featuring the Viterbi algorithm. After aligning the feature vectors to the HMM state sequence the Gaussian Mixture Models are improved using the standard EM-Training for Gaussian Mixture Models. Finally, an HMM for each of our motion primitives has been created.
What we also need is a model on which HMM is more or less likely in a given situation (or context).

4.4 Context Models

It is not sufficient to model motion primitives without modeling their temporal and spatial context. We need to know what happened before the motion or what is happening in the environment while we perform a motion. Several context models have been proposed.

4.4.1 The Different Types of Context Models

Nevatia et al. [NZH03] as well as Ryoo and Aggarwal [RA06] stated that three different types of context have to be modeled to fully describe the context of a motion primitive, i. e. the temporal, spatial and logical contexts. The standard temporal context is Allen's interval temporal logic [All94] which contains the temporal dependencies *before, meets, overlaps, starts, during* and *finishes*. Spatial dependencies are e. g. *near a certain object* or *touching a certain object*. When describing higher level activities based on motion primitives we also need logical elements such as *and, or* and *not*. For our work we concentrated on the first two aspects, i. e. temporal and spatial dependencies, since the logical dependencies contradict the needs of an online recognition system.

4.4.2 Context Free Grammars

One possibility for modeling temporal dependencies which is often used are grammars, especially context free grammars (CFGs). CFGs model the temporal dependencies *before* and *meets*. In the formal language theory, context-free grammars denote a particular class of languages which generate strings of terminals (i. e. symbols). CFGs have first been defined by Chomsky [Cho56]. The advantage of CFGs is that they are expressive enough to describe complex languages like programming languages, while parsing them is computationally tractable. A CFG consists of four components a set of non-terminal symbols, a set of terminal symbols, a set of rules and a start symbol. Beginning with the start symbol the rules can be used to generate all sequences of terminal symbols that follow the given grammar. When modeling motion sequences we can create a CFG that is capable of generating all possible sequences of motion primitives. Each motion primitive in the grammar does have predecessors and successors which specify the context of the motion primitive.

Manual Grammar Generation Based on Functional Decomposition

For small domains it is possible for an expert to manually design a grammar. The advantage of manually defining the grammar is the ability to give names to the non-terminal symbols which have some semantic meaning for a human. In our work we use the functional decomposition described in 4.3.1, which has been created by an expert. The decomposition gives information about the temporal and functional dependencies between the motion primitives. This information can be converted into a context free grammar by an expert. The expert can define non-terminal symbols which belong to subparts of the motion sequence and can also model similar parts of different motion sequences with the same non-terminal in the grammar.

Automatic Grammar Generation Based on Segmentations

Nonetheless Context Free Grammars can also be defined automatically. After retrieving the sequence of motion primitives for the observation sequences, which has to be done anyway to initialize and train our recognition system, it is possible to use these sequences to build a context free grammar. For our work we used a modified version of the Sequitur algorithm [NMW97]. For each pair of consecutive motion primitives that occurs twice in the data a non-terminal is created and the occurrences of the pairs are replaced by the non-terminal symbol. This is done iteratively and recursively investigating one motion primitive after another. Since we have multiple motion sequences we had to modify the Sequitur algorithm to take sequence boundaries into consideration.

4.4.3 Statistical N-Grams

For a small domain with a fixed set of possible motion sequences context free grammars work quite well but they are lacking flexibility concerning spontaneous or unusual motion sequences that were rarely or not seen in the training. We need a large amount of training examples and create a huge grammar to be able to model all motion sequences that can occur. This problem gets even more difficult when we move to larger domains. An alternative to context free grammars are N-Grams. N-Grams do not consider the whole motion sequence but a few consecutive motion primitive, i. e. an N-Gram models the probability of a motion primitive given its N-1 predecessors [BJM83], e. g. $p(mp_i) = p(mp_i|mp_j)$. This probabilities can be calculated as $\frac{count(mp_j, mp_i)}{count(mp_j)}$ using the training data. Since we model the motion primitives depending on the predecessors, we also model some spatial context since the predecessor motion primitive contains some information on the spatial situation. Since this might not be sufficient we also develop an object dependent context model.

4.4.4 Object Dependent Context Model

We should not only model the temporal context of motion primitives, but also the spatial context. Contexts could be:

- The room a motion primitive is performed in
- The situation a motion primitive is performed in
- Objects that are available in the environment
- Objects that have been used shortly
- Typical object positions
- Currently likely object positions/Salient objects

Using this consideration about the environmental context, we decided to developed an *object dependent context model* (ODCM) [Wer11] which models the following aspects for each motion primitive:

1. Do we need objects for the motion primitive?
2. What objects need to be available?
3. What positions do the objects need to be located?
4. What objects have to be grasped to start the motion primitive?

Answering those questions for a motion primitive gives us the context model state the motion primitive can be performed in. The probability of a motion primitive mp_i can be calculated as follows:

If it is an object independent motion primitive then $p(mp_i) = p_{objectIndependentScore}$. If it is a manipulative motion primitive that manipulates objects O_1 through O_N then

$$p(mp_i) = \frac{\sum_{j=1}^{N}(p_{available}(O_j) + p_{atCorrectLocation}(O_j) + p_{grasped}(O_j))}{3 * N} \tag{4.1}$$

whereas all three probabilities are calculated the same way, e. g.:

$$\begin{aligned} p_{available}(O_j) &= p_{available}, if\ O_j\ is\ available \\ &= p_{not\ available}, if\ O_j\ is\ NOT\ available \end{aligned} \tag{4.2}$$

In total we have 7 scores that are optimized during the training of the context model (the score of $p_{objectIndependentScore}$ and two scores each for $p_{available}$, $p_{atCorrectLocation}$, and $p_{grasped}$. During the recognition process we can then compare the desired state with the actual environmental context. We learn the desired states by inferring it directly from the motion primitive, e. g. if a motion primitive is called *Get apple from left back* the context model state is as follows:

1. Yes, we need an object (an apple).
2. An apple needs to be available.
3. The apple needs to be located in the left back corner of the table.
4. No objects must be grasped to start the motion primitive.

If using external sensors/recognizers to model the environment (e. g. using an object recognition system to recognize objects, their location and other properties) it is sufficient to have one universal environment model state for all alternative motion sequences that are part of the current recognition process. If using the recognized motion primitives to estimate the changes in the environment it is necessary to model the environment depending on the performed motion sequences.

5

Evaluating Sequential Human Motion Recognition

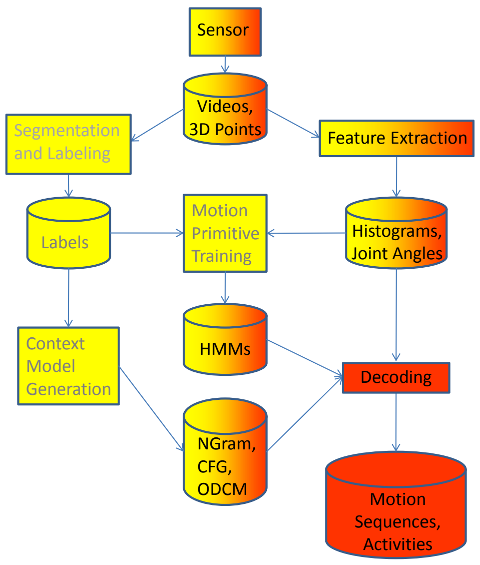

Figure 5.1: The diagram highlights all components involved in the sequential recognition process.

In this chapter we will describe and evaluate our sequential human motion recognition system. We first develop a marker-based baseline human motion recognition system for continuous motion recognition, i. e. we develop suitable features as well as motion primitive models and evaluate various context models. Then we advance the system and evaluate the proposed object dependent context model.

5.1 Improving the Pre-processing

5.1.1 Feature Normalization

For the first experiment we use the KIT-S dataset. Based on the tracked markers and a kinematic body model of the subject, we calculated joint angles of the human upper body for each time frame (see Section 4.2.2). We investigated different features and the influence of features normalization. Since the goal is to optimize recognition accuracy, we performed the evaluation of the features through recognition results. These recognition experiments will now be explained in more detail.

The Recognition System

Based on the collected KIT-S dataset we extracted feature vectors with 24 joint angles each to build a motion recognition system. Our human motion recognition system features the one-pass IBIS decoder [SMFW01], which is part of the Janus Recognition Toolkit (JRTk) [FGH+97]. For HMM motion primitive model training we used about 500 recordings (2 hrs) of human motions from a single subject. For model bootstrapping, we manually segmented two third of the data into motion primitives (see section 4.3.2). In total we have 49 unique motion primitives. Each state of a motion primitive left-to-right HMM has two equally likely transitions, one to the current state and one to the successor state. The emission probabilities are modeled by Gaussian mixtures, initialized by the K-Means algorithm based on the manually segmented and labeled data. HMM training was performed based on forced alignment training (see section 4.3.3) on whole motion sequences. A session independent set of about 100 human motions of the same subject was used as test data. Session independent data refers to a set of recordings from the same subject but collected on another day or in another session. Between sessions markers were repositioned. For decoding we used Time-synchronous Beam Search as described in Section 2.8.1.

Error Measurement

Primitive Error Rate As an error rate we can calculate the Levenshtein distance at motion primitive level between the correct motion primitive sequence (i. e. the reference sequence) and the recognizer output (i. e. the sequence hypothesis). The Levenshtein distance measures the number of edits (i. e. insertions, deletions and substitutions) needed to transform the reference into the hypothesis. The *primitive error rate* (PER) is calculated as

$$PER = \frac{I + D + S}{N} \times 100, \tag{5.1}$$

whereas I is the number of inserted primitives, D is the number of deleted primitives, S is the number of substituted primitives and N is the total number of primitives in the reference sequence. In order to calculate the error rate for a set of motion sequences we calculate the average primitive error rate, i. e. we calculate the error rate for each sequence in the set and build the average over the error rates of all sequences in the set.

Experiments and Results on the KIT-S Dataset

The goal of the following experiments is to find initial features for recognizing continuous human motion sequences. We found that we can not use the joint angle vectors directly since results were very poor (78.4% primitive error rate). Therefore, we first evaluated if we can improve recognition rates by normalizing the feature vectors.

Normalization In our first experiment we used the recognition system described in section 5.1.1. We used standard parameters from the JRTk. Using the standard parameters means that we trained the system with 3 HMM states and 4 Gaussians per state and used a context model weight of 32 and a primitive insertion penalty of 0. For the first experiment we used a 2-gram context model (see section 4.4.3). On the one hand we trained and tested the system with the joint angles described in section 4.2.2 and on the other hand we trained and tested the system on joint angles normalized over time. The 24-dimensional feature vectors were therefore normalized by setting the mean to 0 and the standard deviation to 1 per sequence over time, i. e. $\frac{x-\mu}{\sigma}$. The results for the two different types of feature vectors are as follows:

Normalization	without	with
PER	78.4%	20.9%

Table 5.2: Primitive Error Rate with and without feature normalization.

The above results show that it is essential to use feature normalization for our session-independent motion recognition system, which we did for all following systems.

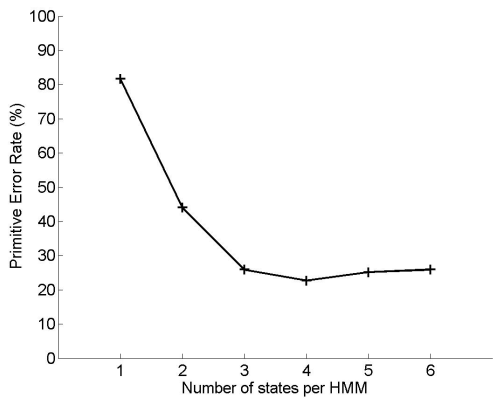

Figure 5.3: Primitive error rates of the KIT-S dataset for varying numbers of states per HMM.

Optimization of amount of HMM states For further improvements of the recognition system we optimize the number of states per HMM. We used the recognition system described in section 5.1.1 with normalized joint angles. This time we used a 0-gram context model (see 4.4.3), i. e. all motion primitives have an a priori probability of 1. We use a 0-gram context model to optimize the amount of HMM states independent of the context model. Figure 5.3 shows the error rates for different numbers of states per HMMs. The best error rate of 22.7 % is achieved when using 4 states (and 4 Gaussians per state).

5.1.2 Feature Comparison

After having shown that the recognition system works well on the KIT-S dataset, we used a dataset with less controlled motions and compare the influence of different features concerning performance. For this purpose we applied a second dataset (KIT-F dataset). We compared three different types of features namely joint angles as well as their first (Δ joint angles) and second ($\Delta\Delta$ joint angles) order derivatives.

Joint Angle-Based Features

In the following experiment we compared three different types of features. Firstly, we used the joint angles from section 4.2.2. Secondly, we calculated the differences of consecutive joint angles (Δ joint angles) which gives us a sort of joint angle velocity. For a sequence of feature vectors $X = x_1, x_2, ..., x_n$ we calculated the joint angle difference as $\dot{x}_i = x_{i+1} - x_{i-1}$. Thirdly, we calculated the second derivative of the joint angles, i. e. $\ddot{x}_i = \dot{x}_{i+1} - \dot{x}_{i-1}$.

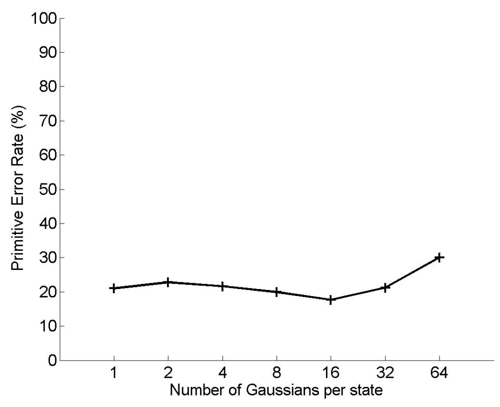

Figure 5.4: Primitive error rates of the KIT-F dataset for varying numbers of Gaussians per state.

The Recognition System

For the following experiments we made some changes to the recognition system described in section 5.1.1. Due to the small size of the dataset we use 10-fold cross-validation to evaluate the system performance. For the bootstrapping of the recognition system all motion sequences had been manually segmented into motion primitives as described in section 4.3.2. In total we had 24 unique motion primitives. Due to the more robust results in comparison with K-Means the models were initialized by the Neural Gas algorithm based on the manually segmented and labeled data. HMM training and decoding was done as before.

Experiments and Results on the KIT-F Dataset

In the previous experiment, 4 states per HMM worked best. We optimize the number of Gaussians per state using 10-fold cross-validation. The number of states was kept at 4. We also optimized the context model weight and the primitive insertion penalty and found that setting both to 2 works best. In Fig. 5.4 we show the recognition error rates on the KIT-F dataset for using different numbers of Gaussians while decoding using a 0-gram context model.
Figure 5.4 shows that we get the best results when using 16 Gaussians where the primitive error rate is 17.6 %. We use this setup to find the best feature type among the three described features. Table 5.5 shows the average primitive error rates over all cross-validation folds on the KIT-F dataset.
We can see that using Δ joint angles works best on the KIT-F dataset. This might be due to the fact that although we normalize the joint angles their variance is still high depending on the subjects position and orientation relative to the table. For Δ joint angles this dependency is a lot smaller. Therefore, we will

Feature	joint angles	Δ joint angles	$\Delta\Delta$ joint angles
PER	17.6%	12.4%	37.2%

Table 5.5: Primitive Error Rate for different feature types using HMMs with 4 states and 16 Gaussians per state for the KIT-F dataset.

stick to Δ joint angles for the following experiments. We will get back to feature optimization in Chapter 7 using larger datasets.

5.2 Evaluating the Context Models

In the first experiment on context models we compared different types of traditional context models namely statistical N-Gram models with a context free grammar on the KIT-F dataset. Those context models are described in section 4.4.
The second experiment on context models is performed on the KIT-M dataset. We also developed a completely new object-dependent context model. We compare this new model with different N-Grams and show that its power is comparable while the modeled properties are complimentary.

5.2.1 Evaluating Traditional Context Models

Experiments and Results on the KIT-F Dataset

Since the optimal number of states per HMM and Gaussians per state could change when using new features we again optimized those parameters for the new features before integrating context models. The average cross-validation results are shown in Fig. 5.6.
In Fig. 5.6 we can see that we achieve the best result with 9 states and 4 Gaussians per state which is an error rate of 4.2 %. The result is also published in [GKS10].
Since 9 states and 4 Gaussians works best on the KIT-F dataset we used this setup for our first experiment on context models. In order to compare the different types of traditional context models, we performed an experiment with the same setup for each of the three context models. First of all, we performed a test with a 0-gram context model, where all transitions from one motion primitive to another are equally likely. In a second experiment we used a simple automatically generated statistical 2-gram context model, where the probability of a motion primitive depends on the predecessor primitive. As a third

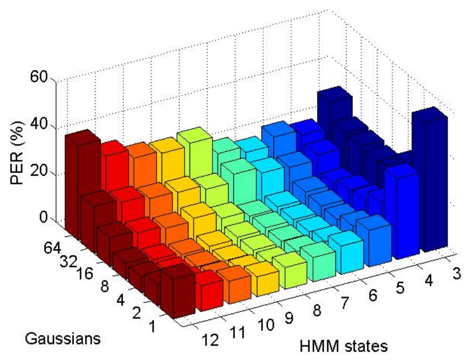

Figure 5.6: Primitive error rates for varying numbers of states per HMM and Gaussians per state on the KIT-F dataset using Δ joint angles as features.

experiment we used a context free grammar (CFG) deduced from the functional decomposition of the motion sequences. As recognition system we used the best system from Section 5.1.2. We only added the different context models during the decoding process.

Context Model	0-gram	2-gram	CFG
PER	4.2%	2.3%	0.9%

Table 5.7: Primitive Error Rates for different context models on the KIT-F dataset.

The results in Table 5.7 [GSFS10] show that context models give significant improvements in primitive error rate. When replacing a 0-gram context model by a 2-gram model the error rate drops by a factor of 2. The same is true when moving from a 2-gram context model to a context free grammar. This suggests that a context free grammar seems to be the best solution for motion recognition. However, a context free grammar also has several drawbacks. While it gave the best results, it had to be created manually. Also, grammars are not as flexible as 2-grams concerning new domains or unseen motion sequences. Since the KIT-F dataset is a rather simple dataset, the drawbacks do not show here.

5.2.2 Evaluating the Proposed Object-Dependent Context Model

In the previous experiment we showed that context models improve the recognition rate of a motion recognizer significantly. All context models that were

used focus on the motion primitives and their typical temporal order. However, they do not take into account any information about the environment, e. g. information about objects that are relevant for the performed motion primitives. Consequently, we developed a context model that models the relations between the performed motion primitives and the objects on the table. This allows the recognizer to react on environmental changes. We compared this new context model with a 0-gram and a 2-gram context model on the KIT-M dataset.
For the experiments in this section we used the KIT-M dataset with fluent motions in contrast to isolated motions in the KIT-S dataset. There are two differences to the KIT-F dataset. The KIT-M dataset used varying positions for the objects and the subject was allowed to use both hands independently, i. e. she was allowed to perform two motions at the same time. This dataset contains motion sequences as they happen in every day life.

The Recognition System

For the evaluation of the ODCM we trained the system as described in section 5.1.2. For the recognition we trained HMMs for 82 motion primitives. The motion primitives are roughly the motion primitives of the previous experiment separated by object positions. We performed a 5-fold cross-validation on the KIT-M dataset. For the recognition we replaced our previous recognizer framework with the Biosignals Recognition Toolkit (BioKIT) (see section 2.9.1) and implemented the ODCM in that framework.

Experiments and Results on the KIT-M Dataset

Before comparing the context models we optimized the numbers of states and Gaussians for the KIT-M dataset. We found that 6 states and 16 Gaussians work best for the KIT-M dataset. Table 5.8 shows the average results for the 5-fold cross-validation as published in [Wer11].

Decoder	JRTk	BioKIT		
Context Model	0-gram	0-gram	2-gram	ODCM
PER	8.0%	7.7%	3.9%	5.0%

Table 5.8: Primitive Error Rate for different context models on the KIT-M dataset.

Table 5.8 shows for the KIT-M dataset that the error rate drops when using the object-dependent context model in comparison to using a 0-gram context model. This indicates that the recognition process can be improved by using object knowledge, although the ODCM results are worse than the 2-gram results. This might be partially due to the nature of the dataset which contains motion sequences that most of the time follow the same sequential order which makes it easy for the 2-gram model to predict the correct successor motion primitive. Due to the complementary nature of the 2-gram context model and the ODCM we expect the error rate to improve when combining both context models. We do not futher investigate this combination due to limit time for this thesis and move on to improving the motion primitive models. Nevertheless, both systems can recognize the correct motion primitive including the correct object and the location on the table in more than 95% of all cases, which is a very good recognition result.

5.3 Online Recognition

5.3.1 Feature Normalization

Probably the most important difference between an offline system and an online system is the limited feature normalization. While we normalized our features in offline mode based on whole sequences, in online mode we can only normalize the features based on the feature vectors that have been captured up to the time step in which we want to recognize the current motion. In section 5.1.1 we calculated the mean and standard deviation for a whole motion sequence and normalized the feature vectors using those values. Now we calculate the mean and standard deviation on the feature vector between the start of the motion sequence and the current time step. As a result the feature normalization is poor in the beginning but improves over time as long as the same activity is performed.

5.3.2 Vicon-Based Online Motion Recognition

For our first online experiment we used a system very similar to the one described in section 5.1.1. There are only a few differences to that system. We limited the number of activities to 8. Thus, the number of unique motion primitives was 41. For the training of the system we used the 400 motion sequences of those eight activities. We initialized and trained the system as described in section 5.1.1. For the decoding we used the according 80 test sequences. The decoding was guided by a context free grammar which was designed to allow online recognition, i. e. the primitive sequences that start with the start symbol are valid parses of the grammar. For the decoding we normalized the features as described in Section 5.3.1 while recognizing the current motion primitive every 1.5 seconds. We achieved an activity error rate (see section 8.1.2) for the whole sequences of 0%, i. e. at the end of each sequence we recognized the activity correctly. This demo system has been published in [GFK$^+$08] and online demonstrations have been presented multiple times.

6

Concurrent and Coupled Recognition for Human Motion

Most motion recognition systems to date recognize motions in a limited domain. Many of those systems do not scale well to larger domains, while others that scale well do not allow online recognition. In this chapter we introduce a novel approach to *concurrent and coupled* (C&C) human motion recognition. This method scales a lot better than most state-of-the-art systems while allowing online recognition of complex human motion sequences. The main difference to the system in chapter 4 is the modeling of motion primitives of individual body parts instead of modeling movements of the whole human body as one combined motion. In order to define a set of motion primitives we extend our method for structured functional motion decomposition from section 4.3.1. The resulting motion primitives can belong to individual body parts or combinations of body parts depending on the motion performed. Thus, we have to model significantly fewer motion primitives than with our previous approach. This enables us to perform online recognition of larger domains.

6.1 Related Work

In this section we want to give an overview on techniques that are capable of modeling concurrency of motions (both single-layered and hierarchical methods) since this is one of the main aspect of this thesis. Only few of the approaches mentioned in Section 4.1 consider concurrency of motions. On the one hand, modeling concurrent motions independently is motivated by findings in neural science concerning human motion generation [KSJ00]. On the other hand, it is motivated by the technical advantages when building a motion recognition system with motion primitive models that occur more frequent in human motion sequences and that are simpler and consequently easier to model. According to [Müs08] humans can perform multiple activities at the same time, i. e. they can perform multiple motions that are independent of each other. The motion generation in the human brain is capable of generating independent motions of a single limb and motions for a combination of limbs [KSJ00].

6.1.1 Single-Layered Techniques

To model concurrent motions some authors use variants of HMMs, such as Coupled HMMs (CHMMs) or Dynamic Bayesian Networks (DBNs). A standard HMM is a sequential model and only one state is activated at a time, preventing it from modeling two or more concurrent motions. Oliver at al. [ORP00] published the concept of CHMMs to model interactions of two agents. Basically, a coupled HMM is constructed by coupling multiple standard HMMs. Each standard HMM represents one of the agents. The coupled HMM has additional transitions between the hidden states of the two different HMMs. They recognize complex interactions of two agent such as 'two persons approaching, meeting, and continuing together'. Brand and Oliver [BOP97] use CHMMs to recognize complex human motions. CHMMs are able to recognize multiple coordinated motion primitives, but their disadvantage is the need for training all possible combinations of concurrent motion primitives.

In contrast, Vogler and Metaxas [VM99] use parallel Hidden Markov Models (PaHMMs), which train motion primitives of different body parts independently and whose likelihoods are only combined for recognition. The PaHMMs are synchronized after every motion primitive and do not allow asynchronous sequences of motion primitives in their application. Each pair of motion primitives stands for a sign of the American sign language.

In [NN07] Natarajan and Nevatia advance the CHMM to Continous Hidden Semi Markov Models (CHSMMs) by adding duration models. They show that those outperform the CHMM and the PaHMM in the domain of sign language. They also present an approximation to the learning and infering algorithms to handle the exponentially increasing computation time for training and applying the CHMM and the CHSMM.

Park and Aggarwal [PA04] use DBNs to recognize interactions of two persons, e. g. pushing, hugging, and kicking. They assume that the temporal evolution of the DBNs for poses of different body-parts is independent of each other. Using the DBNs they evolve the whole-body pose estimation of each person over time. For the recognition of interactions they add spatial and temporal constraint which need to be satisfied for an interaction to be recognized. In order to model the temporal constraints between the two persons' motions they use an adoption of Allen's interval temporal logic [AF94].

6.1.2 Hierarchical Techniques

Aggarwal et al. [AR11] state that statistical and syntactic techniques have difficulties modeling concurrent motions. Only description-based approaches are capable of modeling complex temporal structures.

Statistical

Shi et al. [SHM$^+$04] developed a propagation network to model concurrent motions. With this network they verify the correctness of performing a glucose monitor calibration task. Each node of the network uses a Bayesian Network to evaluate the performed motions. A duration model for each node is used to guide the traversal through the network. With this approach they can model multiple concurrent motions.

Description-based

Gupta et al. [GSSD09] recognize actions in Videos using AND-OR graphs. In contrast to other work they aim to improve the recognition by modeling causality between their primitives. They match tracks from baseball videos to actions in a storyline which includes concurrent actions.
Other researchers such as [GDDD04, ACM$^+$08] use petri nets to recognize human activities. The petri nets are capable of representing Allen's temporal interval logic. Ghanem et al. [GDDD04] developed a system to recognize interactions between humans and vehicles using petri nets. Their system does not yet handle noisy or irrelevant observations. A further disadvantage of their approach is that they do not use confidences of the low-level detection for their activity recognition. Due to the fact that the tokens in a petri net are undistinguishable from each other they can not describle multiple alternative histories. Therefore, the petri nets have difficulties recognizing complex scenes.
Nevatia et al. [NZH03] developed a language-based representation (*VERL*) to recognize interactions between multiple persons. They use three levels of events to model human activities: primitive events, single-thread composite events, and multi-thread composite events. In order to specify constraints between events they use Allen's temporal, spatial and logical predicates. For the low-level recognition of primitive events and single-thread composite events they use Baysian networks and HMMs respectively. For the high-level recognition of multi-thread composite events they developed a heuristic algorithm

to solve constraint satisfaction. A disadvantage of their system was to handle recognition failures of the low-level components. Vu et al. [VBT03] extended the approach of Nevatia et al. to describe any levels of hierarchy, but in contrast to [NZH03] only conjunctions where allowed as logical predicates.

Pinhanez [Pin99] developed a {past, now, future} network (PNF-network) that can model all temporal constraints of Allen's interval temporal logic. Their work focuses on recognizing high-level activities based on simple low-level detectors. They do not try to recognize human motions. Although the network is capable of recognizing activities with arbitrary temporal constraints, the system has several shortcomings. Their system is not capable of handling confidence values of the low-level detectors, i. e. perceptions are converted into a discrete PNF representation. Thus, a miss detection is more likely to happen and results in an erroneous state of the PNF-network. They introduced methods to compensate for a single detection failure. Another disadvantage is the fact that the PNF-network for all activities has to be modeled manually, whereas a sub-network has to be specified redundantly if the sub-event can occur multiple times.

Ryoo et al. [RA09] use a CFG for a high-level description of an activitiy to model the temporal, spatial and logical constraints between motion primitives. Motion primitives are represented as HMMs. As temporal constraints they model the constraints of Allen's interval temporal logic. Their system first detects all low level motion primitives and then evaluates if the detected motion primitives satisfy the CFG of high-level activities. This assumes that the whole activitiy has been performed before recognizing it. In addition to Pinhanez [Pin99], they introduced a way to handle confidence measures of the low-level detectors as well a method for dealing with missing detections of low-level motion primitives. They still have the same drawback as Pinhaned [Pin99] with modeling the activities which has also been done manually.

Zhang et al. [ZTH11] extend stochastic context-free grammars (SCFGs) by Allen's interval temporal logic, but in constrast to Ryoo et al. [RA09] they use the SCFG directly for parsing instead of an abstract representation which needs to be satisfied. HMMs are used as motion primitive detectors in a simple 2D setup. An activity description is retrieved automatically based on training data. For the recognition they use a Multithread parsing algorithm.

6.2 Motion Primitive Models

6.2.1 Functional Decomposition

In section 4.3.1 we proposed a method for structured functional decomposition of human motion sequences. While this method is sufficient for the decomposition of sequential motion sequences it has to be extended for more complex activities, such that it can be applied to the decomposition of human activities with C&C human motion primitives. The main difference is the possibility of resulting in parallel motion primitives. While the previous method results in motion primitives containing independent motions of multiple body parts, the extended version now models motions independently, which is more congruent with the nature of motions. An example for such motion primitives is *Move knife and Steady apple* in Figure 6.1 assuming a right-handed person. It results in the two motion primitives *Carry knife* of the right arm and *Steady apple* of the left arm (see Figure 6.2). The modeling of concurrent motions with separate motion primitives uses the full power of the previously proposed method. The functional decomposition gives us a list of abstract motion primitives which have to be concretized during segmentation.

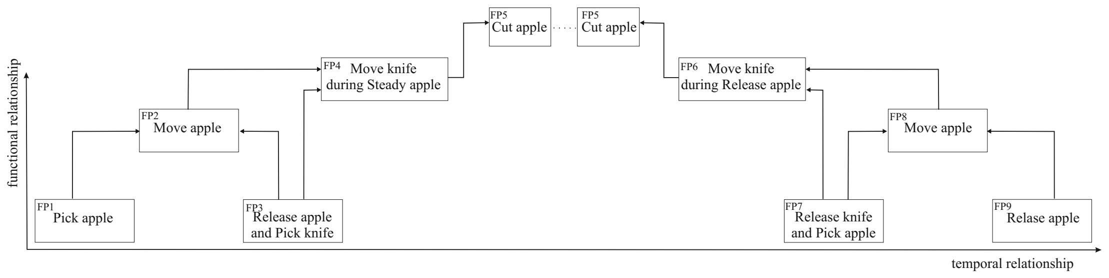
FP5 Cut apple
FP5 Cut apple
FP4 Move knife during Steady apple
FP6 Move knife during Release apple
FP2 Move apple
FP8 Move apple
FP1 Pick apple
FP3 Release apple and Pick knife
FP7 Release knife and Pick apple
FP9 Relase apple
functional relationship
temporal relationship

Figure 6.1: Fine grained functional and temporal structures of a complex human motion sequence ”cutting an apple” with concurrent motions but without distinction of the used body parts.

6.2.2 Segmentation and Labeling

During segmentation, the different properties of the abstract motion primitives have to be specified, e. g. the beginning and ending as well as the inital and final position of a motion. Depending on the category of the motion primitive, different properties have to be specified. The motions can for example be separated into stationary and non stationary motions. Stationary motions may consist of manipulation motions such as *cut apple* which are performed at a certain location. Non stationary motions serve as supporting functional

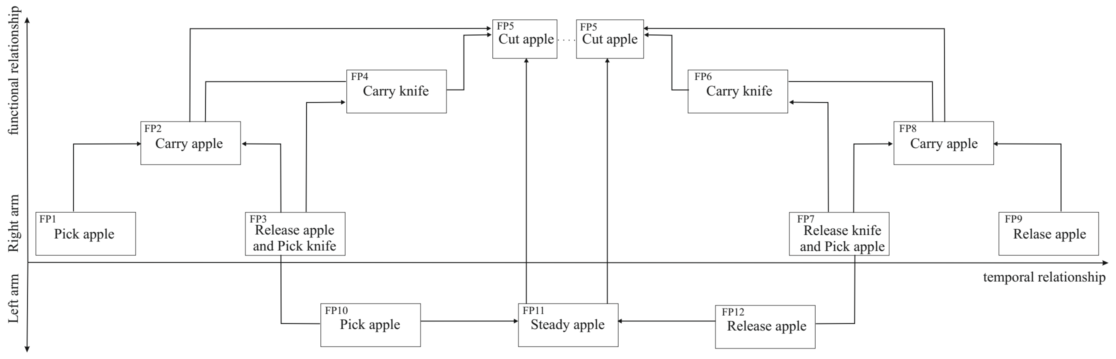

Figure 6.2: Complex human motion sequence with concurrent motion primitives: "cutting an apple".

primitives, which move a body part or an object to the appropriate position, e. g. carrying a knife to the cutting board. For those primitives two locations have to be specified, the start and the end position.
We also need a definition of boundaries between motion primitives with objects and without objects. The intuitive boundary is the contact with the object when grasping or releasing an object. The grasping or releasing motion is always assigned to the motion without object since this motion can not be separated from reaching for the object (see [SB99, Müs08]) due to the anticipation effect. Since the grasping or releasing motion can not be separated from the movement of the hand from one object to another, a motion without object consists of three parts: releasing, moving, and grasping. This is the most general form, but the start or end motion can also be omitted if the predecessor or successor motion primitive is not a motion with an object. For simplicity reasons we do not distinguish different grasps for the releasing or grasping motion.

Concurrent and Coupled Segmentation

Due to the nature of human motions, several motion primitives can be performed at the same time. The set of joints moved when performing a motion primitive varies between different motion primitives. When annotating data with motion primitives for individual body parts, it is necessary to specify the set of joints used for each motion primitive. Therefore, a hierarchy of body parts is used which decomposes the human body into body parts of multiple levels of granularity, e. g. the upper body is separated into left arm, right arm, head, and torso (see Figure 6.3).
Since every label is associated to a body part it has to be decided which body part is the most relevant for the motion primitive. In general, motion primitives are always assigned to the body part that is most important for the performed

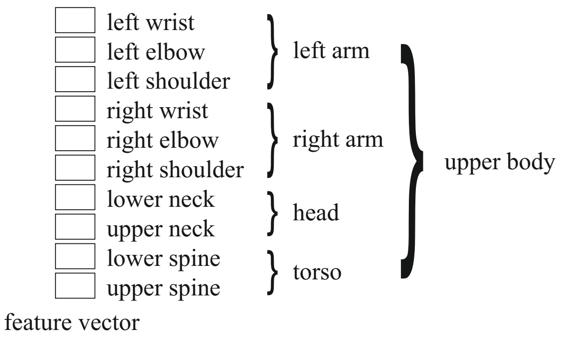

Figure 6.3: Decomposition of a feature vector into feature vectors for individual body parts.

primitive. Motions are only assigned to the whole upper body, if all limbs are performing a coordinated motion serving the same goal which can not be divided further, e. g. the lifting of a heavy object from the floor.

For the segmentation the basic procedure and the software framework are the same as in section 4.3.2 but due to the more complex segmentation process we can not directly use the segmentation module. The segmentation method needs to handle the annotation of motion primitives for multiple layers in a body hierarchy, e. g. upper body (both arms) versus left arm or right arm only. Another challenge is the very large number of possible motion primitive labels, which have to be assigned to the segmented motions. Both aspects will be discussed in the following.

Extension of the Motion Sequence Segmentation Toolkit

The most straight forward extension of the manual segmentation method described in section 4.3.2 is segmenting the data separately for each body part. But this method has multiple drawbacks. If using one column for each body part the segmentation user interface will no longer fit completely on the screen if using more than three or four body parts. In addition to that, it is more difficult to verify the correct start and end times for a motion primitive. As in the sequential manual segmentation, the end time of a motion primitive is specified by the start time of the next primitive to speed up the segmentation process. When the successor primitive is in another column as the motion primitive to be segmented it is difficult to validate the correct end time. We resolved those conflicts by using only one column (for the highest body part in the hierarchy) and split this column only in areas where multiple motions of body parts in a lower level occur. This saves a lot of space on the screen and allows a more intuitive segmentation of the motion primitives, which allows to segment the data faster and with less errors.

Labeling

For the labeling of human motions we aim at a general method. For datasets with realistic daily activities and an sufficiently fine grained segmentation into motion primitives it is not feasible to use a list of all possible motion primitive labels due to the huge number of possible primitives. Therefore, we developed a dynamic generator of motion primitive labels instead of a static list. This results in small lists for different attributes of the motion primitive, e. g. the performed motion such as pouring object O at position A, moving something from A to B etc. In other lists the possible values for O as well as possible positions for A and B are defined. The selection of a motion primitive label from a very large static list of motion primitive labels is transformed in a selection of multiple motion primitive attributes from small lists which can be done faster. It is also easier to find the correct labels. To speed up this process we further use context dependent attributes which means that the set of possible positions for example depends on the performed motion whereby the lists get even smaller. Another big advantage of this method is the possibility of easily transferring the set of possible attributes to another domain. So for example the possible positions for objects can be generically defined and used in multiple domains whereas the combinations of positions for a specific motion will differ between domains. The same is true for other attributes. The set of labels for a specific domain is then computed using all attribute combinations used for this domain.

Motion Primitive Attributes

Based on the functional decomposition and observed motion sequences we created a set of 8 attributes. The possible attributes of a motion primitive are: *action, body part, direct object, indirect object, target object, start position, end position, activity*. Depending on the action of a motion primitive a subset of these attributes has to be specified.

The definition of all possible labels for a given domain is also problematic when using static lists. This list has to be specified beforehand or it has to be extended each time a new primitive occurs in the data. The effort for handling these lists is a lot less when using a list for each attribute. As stated in the previous paragraph these lists are a lot easier to create and maintain due to their more general nature and the shorter lists. A side-effect of modeling each motion primitive as an object with multiple attributes is the easier automatic post-processing of the manually created motion segmentations. The system can for example cluster each attribute individually into coarser classes for the attribute to model the motion sequences in a coarse-grained way, e. g. modeling the motion primitives independent of the used objects. This allows to easily experiment with the set of motion primitives to find the set which models the data best.

6.2.3 Initialization and Training of Concurrent and Coupled Models

Since the main idea of our approach of modeling C&C human motion primitives as independent as possible, the initialization and training of the models is straight forward. We can train the models for each body part independent of the others. The only difference to the training in section 4.3.3 is the training of each motion primitive separately instead of training on whole sequences. And of course only using the features belonging to the respective body part.

6.3 Context Models

For the recognition of motion sequences based on motion primitives as in the sequential recognition (see section 4.4) we need context models for the C&C motion recognition to model the dependencies between multiple motion primitive. When moving from the sequential to the C&C recognition two new challenges have to be dealt with. The challenges are, firstly, the need for modeling temporal context along multiple layers of the hierarchy and, secondly, the necessity of rules for dependencies between concurrent motion primitives. Since the basic idea of our methodology is to model each concurrent motion separately, we dropped the dependencies between concurrent motion primitives which would withdraw one of the main advantages of our methodology. Nevertheless, we need models for sequential dependencies between multiple motion primitives. In order to be flexible with respect to recognizing unseen motion sequences we decided to extend the standard N-Grams for C&C motion primitives.

6.3.1 N-Grams

For the extension of the standard N-Gram context model to a 2-gram context model for the C&C recognition, we calculate the 2-grams and their back-offs for all combinations of predecessors and successor independent of the stream they belong to. At transitions from multiple motion primitives in finer layers

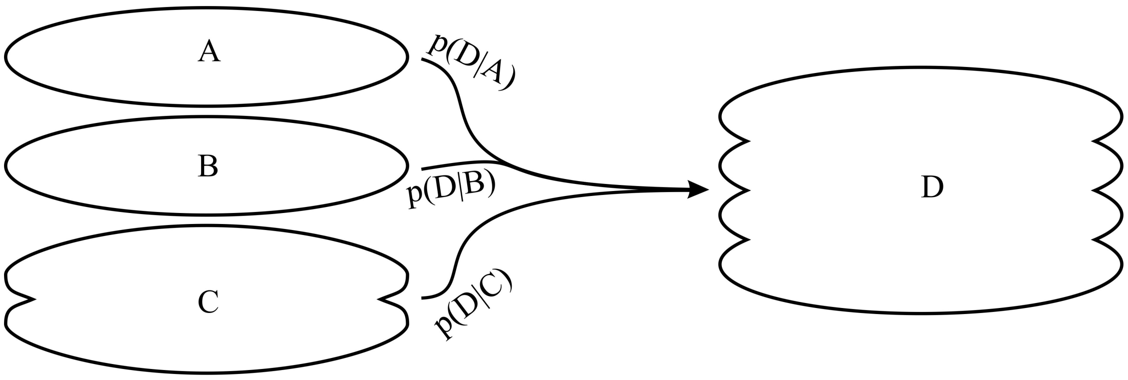

Figure 6.4: Traversing from two motion primitives A and B in layer 2 and a motion primitive C in layer 1 to a motion primitive D in layer 0 we calculate the likelihood for D as $p(D) = \sqrt{\sqrt{p(D|A) * p(D|B)} * p(D|C)}$

to one motion primitive in a coarser layer, we need to weight the likelihoods of the individual 2-grams accordingly. This is done by weighting each motion primitive with the same parent in the body hierarchy equally. This is done recursively for all layers in the body hierarchy. So for example consider the transition in Figure 6.4.

In order to get a probability for a motion primitive, we calculate the average probability for all predecessors (e. g. $p(D) = \sqrt{\sqrt{p(D|A) * p(D|B)} * p(D|C)}$). All probabilities needed for the C&C 2-gram context model are calculated the same way as for the sequential 2-gram context model described in section 4.4.3. Before that we have to convert the sequence of motion primitives into an appropriate format by expressing the whole sequence in the most fine-grained layer. The probabilities can then be calculated for each concurrent stream separately.

6.4 Beam Search for Concurrent and Coupled Human Motion Sequences

6.4.1 Beam Search in BioKIT

In section 2.8.1 we described the general methodology of beam search. In the following we will describe how we designed and implemented beam search for C&C human motion sequences in BioKIT.

Hypothesis Propagation across Body-Parts

When recognizing C&C motion primitives we model motions of different body parts separately, e. g. we model motions of the right arm, motions of the left arm and motions performed with both arms, i. e. the whole upper body. In order to be able to handle the complexity of possible concurrent motions, those motions are modeled independently, i. e. we model motions of the right arm independent of the motion of the left arm if they are not coupled. This gives us the ability to recognize all possible combinations of concurrent motion primitive with a small amount of training data. We are even able to recognize concurrent motion primitives that have not been in the training data.
In order to achieve this independent modeling we create a search graph (a set of HMMs) for each modeled body part, i. e. we have models for coupled motions of the whole upper body, models for motions of the left arm only and motions of the right arm only. Instead of one search graph as before we have multiple search graphs and search independently for the best motion primitives in the different search graphs. As long as we are within a search graph, the hypothesis propagation and pruning is done as before. Once we are at the end of an HMM, things get more complicated. In addition to the propagation of hypotheses from the end of an HMM to the beginning of HMMs of the same search graph as in Figure 2.4, we now have more possibilities of propagating hypotheses. We can propagate them downwards to HMMs of a finer granularity of the current body part (see *downwards propagation* in Figure 6.5) or we can propagate them upwards to a coarser granularity of a set of body parts (see *upwards propagation* in Figure 6.5).
We could, for example, propagate a hypothesis that has finished a motion primitive of the whole upper body to continue in the left and right arm. In this case hypothesis propagation can be done as in the previous chapter. This means we propagate the hypotheses that are in final nodes to all initial nodes of the HMMs in all finer layers. When, for example, propagating upwards from two separate motions of the left and the right arm to a motion of the whole upper body, things get more complicated. Since we propagated the hypotheses independently in each finer body part we need to merge the hypotheses and propagate them upwards. For the merging we always need one hypothesis from each finer body part, e. g. one from the left arm and one from the right arm. Since we have many hypotheses in each finer body part we have a lot of possible combinations, but not all combinations make sense. If, for example, the history of one hypothesis starts with *Take bowl* while the history of the second hypothesis starts with *Rest position* those hypotheses can not be merged since their histories do not fit together. Therefore, we only need to merge hypotheses with matching histories (see Figure 6.6). This can be achieved by

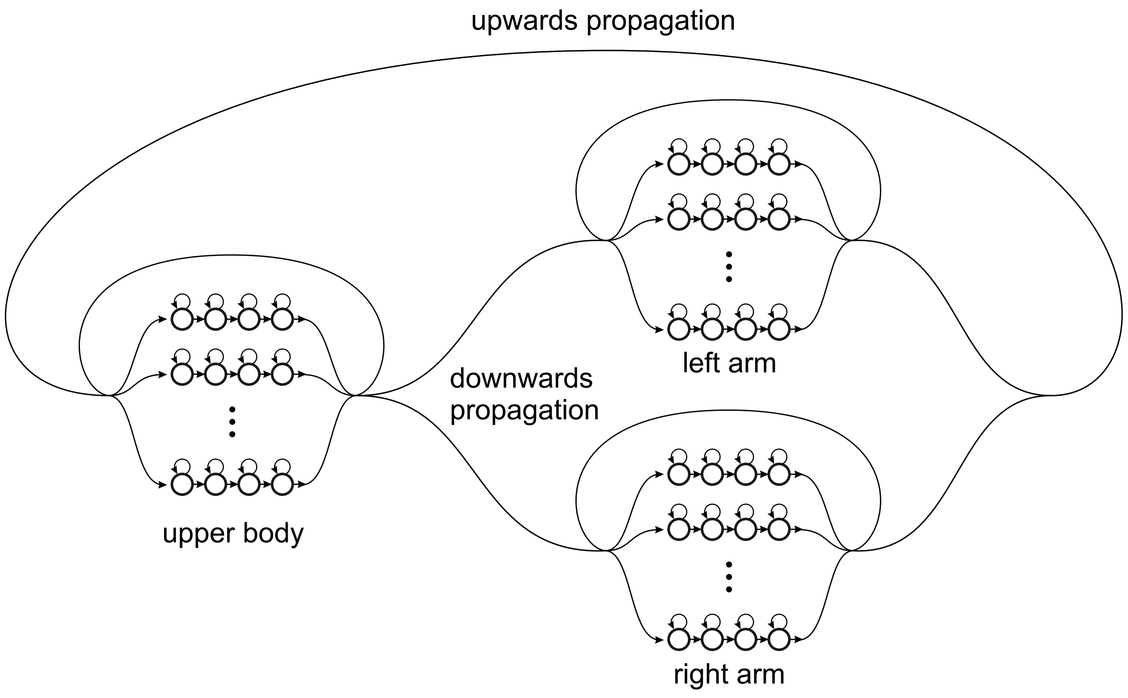

Figure 6.5: Search graphs for C&C motion recognition with possible transitions for propagating hypotheses.

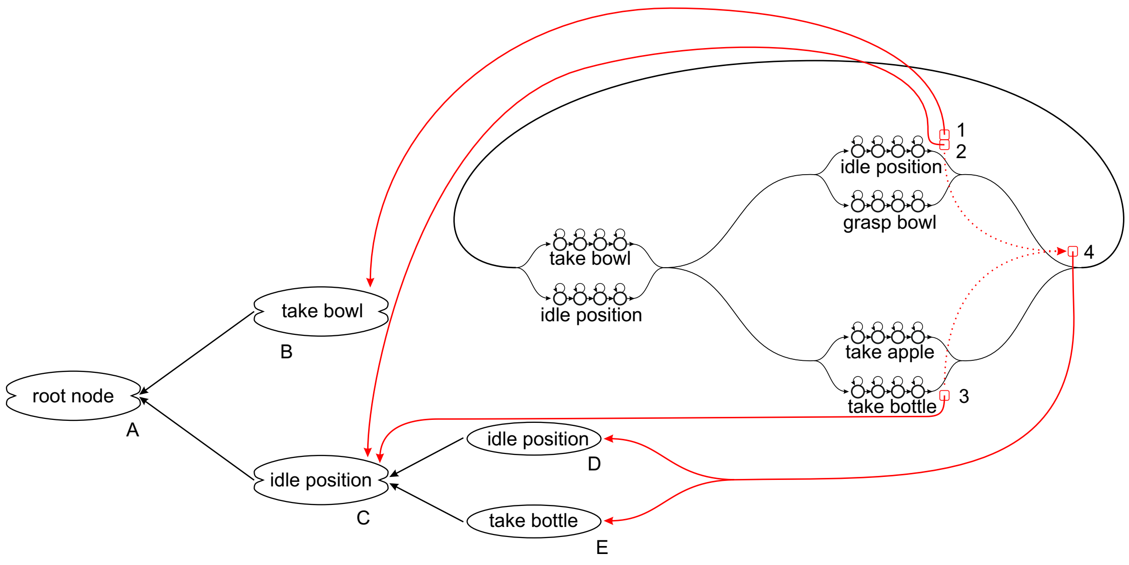

Figure 6.6: Search graphs for C&C motion recognition while propagating hypotheses from a finer to a coarser layer. The hypotheses 2 and 3 can be merged and propagated upwards since they have the same lattice node (C) in the destination layer while the hypotheses 1 and 3 can not be merged due to differing lattice nodes (B,C).

comparing their latest lattice node in the level the hypotheses are propagated to. If they have the same lattice node, the histories match and the hypotheses can be merged. The fact that each lattice node has multiple end times, gives us the flexibility of combining hypotheses with similar but not necessarily exactly equal histories. When merging the hypotheses we also need to merge their scores. This is done by calculating the weighted mean. Each hypothesis is weighted by the fraction of the body part it belongs to. If a body part has two sub body parts each is weighted with $\frac{1}{2}$. If one of those has three sub body parts each of those is weighted by $\frac{1}{2} \cdot \frac{1}{3} = \frac{1}{6}$. Assuming that the score we accumulate in the different search graphs are in the same range, we can compare the newly created hypothesis with other hypotheses in the initial node of the coarser level and perform propagation and pruning within the search graph as before. Unfortunately, the score in the different search graphs dependent on the dimensionality of the GMMs. Therefore, we need to normalize the scores each time we evaluate a GMM.

Score Normalization

Since the motions of different body parts are trained separately and the scores are combined when recognizing the motions, we need to make sure that the scores of the different motion primitives can be compared with each other. The value ranges of the GMMS are different for the different body parts due to the varying dimensionality of the feature vectors. Therefore, the scores of the GMMs have to be normalized to be comparable between body parts with different dimensionality. In the literature there are several normalization techniques proposed. Ozeki [Oze96] proposes to use the score of an ergodic HMM to normalize the scores of the recognition HMMs. They show that this method works well for comparing speech segments with different lengths. They approximate the mutual information of the hypothesis M and the observations X $I(M; X) = logp(X|M) - logp(X)$. Alternatively, it has been proposed to use the sum of likelihoods of a cohort set instead of using an ergodic HMM to approximate $logp(X)$. Both methods require a lot of additional computation power during the decoding since both need additional HMM evaluations. Therefore, we decided to use Z-normalization. For each body part we calculated the average score μ and the standard deviation σ during training and normalize each score during decoding through mean and standard deviation normalization $z = \frac{logp(X|M) - \mu}{\sigma}$.

7

Evaluating Concurrent and Coupled Human Motion Recognition

In this chapter we analyze and evaluate the proposed methods for C&C human motion recognition. We compare different properties for the purely sequential modeling technique as well as the C&C modeling technique. For this comparison we used the Breakfast dataset with a large variety of complex kitchen activities. On this dataset we show the advantages and disadvantages of the developed modeling technique. At the end of this chapter we will present an online system of our C&C motion recognition.

7.1 Motion Primitive Alphabet

The functional decomposition described in 6.2.1 gives a set of primitives.

Since no dataset can contain all motions a human can perform, we need to adapt our segmentation to the actual scenario. The segmentation was based on the motions a human performed. If a human performed two independent motions with the left and the right arm, they were segmented separately. If they always occur concurrently in the scenario, it is better to model them as one primitive. Therefore, we adapted the Breakfast dataset in a way that we searched for motion primitives that occur concurrently most of the time. Those motion primitives are combined into one motion primitive. Exceptions of this rule are static and cyclic motion primitives, where the above rule only applies to the concurrent motion primitive and the static or cyclic motion is splitted temporally. This limits the power of transferring the models to other scenarios but there will be better recognition results in the given scenario.

We also need to adapt our segmentations for the sequential segmentation. We first split the static or cyclic motions based on the boundaries of the concurrently performed motion primitives. We then build new motion primitives wherever concurrent motion primitives have the same start and end time. We combine motions of both arms into one motion of the whole body. If boundaries of multiple motion primitives only match at their beginning and at the very end but not in the middle, the group of motion primitives is combined into one large primitive. This shows the difficulty of modeling arbitrary motions with purely sequential primitives.

7.2 System Properties

When comparing different types of motion recognition system there are multiple important aspects. The recognition systems can be compared in terms of:

- Primitive Error Rate
- Number of Parameters
- Distribution of the data onto the models
- Portability on other scenarios

The first one is probably the most important, but the others give more insight into the recognition system and its properties. The Primitive Error Rate measures the errors for a certain setup of recognition system, while the others are independent of the actual parameter setting of the system and describe the way the recognition system models the data. The main focus of this work lies on the better utilization of the data and the portability to other scenarios which allows to build systems with less training data and for scenarios without any training data.

7.3 Comparing the Parameters of the Sequential and the Concurrent and Coupled Models

7.3.1 Number of Parameters

Given the above definitions of motion primitives for the Breakfast dataset, we get the model properties in Table 7.1.
Table 7.1 shows that the C&C recognition system needs only $\frac{2}{3}$ of the models and parameters respectively than the Sequential recognition system. At the same time, the average length for each motion primitive decreases by 14 % while the minimum length of the motion primitives remains at 3 frames. Having a shorter average length suggests that we can use less states. Thus, we need to train less parameters, i. e. we have more training data for each parameter.

	Sequential	C&C	Reduction
#Primitives	1164	805	30 %
Avg. Length	51 Frames	44 Frames	14 %
Min. Length	3 Frames	3 Frames	-

Table 7.1: System properties for the sequential and the C&C modeling techniques the Breakfast dataset.

7.3.2 Distribution of the Data onto the Models

Fig. 7.2 shows that the data is distributed better on the C&C models since each model on average occurs 3 times more often in the data than for the sequential models. Thus, we have more training data for each parameter that needs to be trained.

	Sequential	C&C
Avg. Frequency	2	6

Table 7.2: Average frequency of the models in the Breakfast dataset for the sequential and the C&C modeling techniques.

7.3.3 Portability to Other Scenarios

Number of Activities	Sequential	C&C
1	1119	615
2	40	107
3	4	42
4	1	20
5	-	13
6	1	5

Table 7.3: The table shows how many unique (object dependent) primitives occur in a certain number of activities.

Table 7.3 as well as Figure 7.4 show that the C&C modeling technique is suited better to be transferred to unknown activities. The tables show the overlap of motion primitives between the recorded activities. Table 7.3 shows that 42 of the 805 C&C motion primitives were performed in 3 different activities whereas 40 of the 1164 sequential primitives were performed in 2 different activities. Figure 7.4 shows how many unique motion primitives need to be modeled for a certain number of activites. Those results show two things. Firstly, if recognizing an activity that has not been part of the training data we can recognize more motion primitives with the C&C system as with the sequential system. Secondly, when scaling the system to larger datasets the number of primitives to be modeled increases a lot faster for the sequential system as for the C&C system.

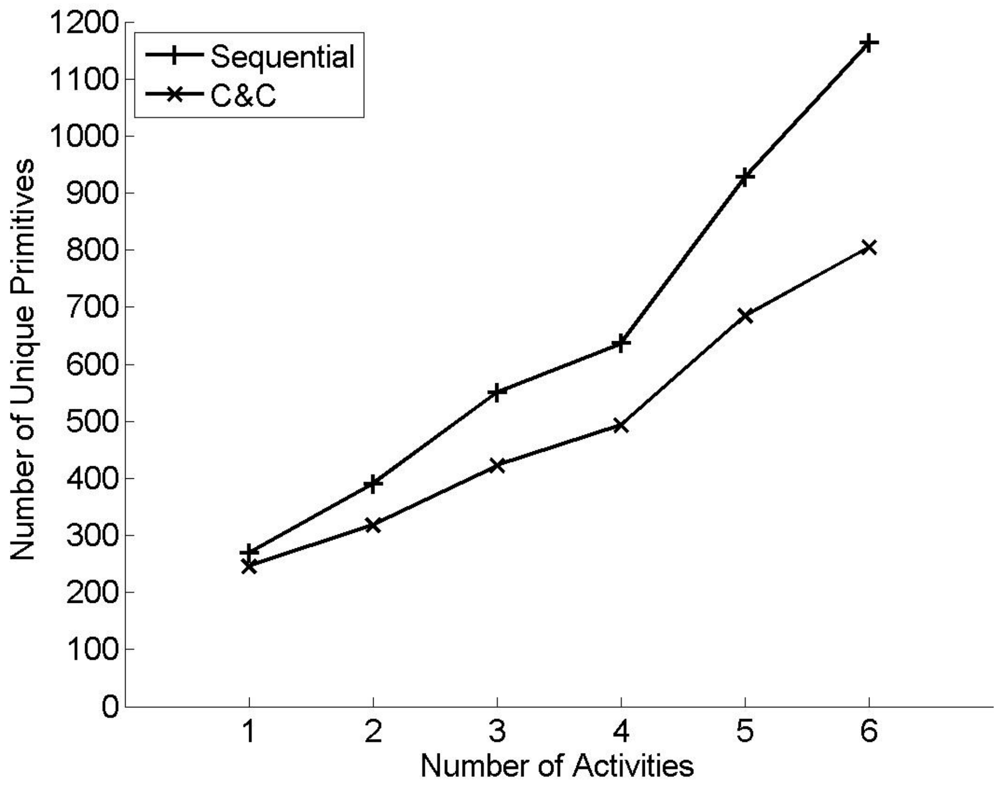

Figure 7.4: Number of unique motion primitives for increasing number of activities for the Breakfast dataset.

7.4 The Recognition System

For the comparison of the sequential and the C&C approach concerning error rate we use the same pre-processing to achieve joint angles as in the previous chapters. In [GS08] we showed that using joint angles of shoulders, arms, and hands is sufficient to achieve good recognition results on our scenarios. For simplicity reasons we start our experiments using those features only.

7.4.1 Primitive Error Rate

In order to compare the recognition results of the sequential and the C&C recognition, we need to define an error rate that gives comparable error rates for both systems. We achieve this by splitting all motion primitives we combined for the recognition into the original manually segmented primitives and calculate the recognition rate on this recognition result. We need to split combined primitives into their parts and merge multiple static or cyclic motion primitives if the same primitive occurs multiple times in a row, since the splitting of those primitive is not of interest for the recognition result and they have been split differently for the sequential and for the C&C recognition. We then calculate the primitive error rate similar to 5.1.1.
In order to achieve comparable error rates for sequential as well as C&C recognition, we calculate the error rate on each finest stream, i. e. if a whole body motion was recognized while two separate motions of the left and of the right arm had been actually performed, we count this as two errors. During the calculation of the Levenshtein distance we make sure that motion primitives of higher streams are not matched only partially to ensure the temporal order of the primitives.
In the calculation of the error rate, it is not an error if e. g. two motion primitives in finer streams are recognized as one motion primitive in a higher stream, if they are named equally. We do not penalize if recognizing motion primitives for the wrong level in the body hierarchy.

7.5 Experiments and Results on the Breakfast Dataset

7.5.1 Parameter Optimization

In our first experiment we optimized the parameters of the C&C recognition system based on Δ joint angles as features with a 0-gram context model. We optimized the number of states between 8 and 16 and the pip between 0 and 40 using 5-fold cross-validation. We achieved the best result with a pip of 10 while the optimal number of states was 14. With this setup we achieved a PER of 68.4% while the corresponding sequential system has a PER of 57.8%.
For further improvements of the recognition system, we investigated the following four topics.

1. Score normalization
2. Tendency towards finer streams
3. Beams optimization

4. Feature set optimization

Score Normalization

We learn the score normalization on the training data which might be suboptimal since the training data fits better to the models than the test data. Therefore, we perform an experiment where we learned the score normalization on the test data. While this could be considered as a "cheating" experiment, we do so in order to get an upper bound for the recognition accuracy. The mean and standard deviation values changed significantly but the primitive error rates did not improve (see Table 7.5). So, the score normalization is not the reason for the worse recognition results of the C&C system.

Training on	Training Data	Test Data
PER	68.4%	68.2%

Table 7.5: Motion Primitive Error Rates for the C&C recognition when normalizing the score based on the training or on the test data.

Tendency Towards Finer Streams

The recognition of motion primitives in finer streams might be more likely than primitives in a higher stream. The recognition of primitives in a finer streams is more flexible concerning start/end time of primitives as well as alignment of primitives between streams. We evaluated if compensating the tendency towards finer streams improves the recognition rate. Therefore, we added a stream penalty which is implemented as an additional offset to the score normalization depending on the stream. We found that we can improve the recognition rate by using an offset of 1 for the finer stream which makes the finer streams less likely. This improves the recognition results as shown in Table 7.6.

Offset	Recognition Error
0	68.4%
0.5	64.9%
1	64.5%
1.5	66.7%
2	77.5%

Table 7.6: Motion Primitive Error Rates for different stream penalties.

Beam Optimization

Since the beams in the previous experiments are a trade-off between accuracy and speed, we will now evaluate the optimal recognition rate without considering the speed issue. Since the hypothesis top N pruning has the most impact concerning accuracy and speed, we define a master beam, which specifies all pruning thresholds besides hypothesis top N. The hypothesis top N threshold is specified independently. In the experiment we varied the master beam between 1 and 6 and the hypothesis top N factor between 5 and 20 whereas $hypo\ Top\ N = master\ beam * hypo\ Top\ N\ factor$. The results in Table 7.7 show that we can improve the recognition rate by almost 12% relative using larger beams, although the recognition speed drops to more than 20 times real time.

7.5.2 Feature Set Optimization

With optimized beams, i. e. a master beam of 4 and a hypothesis top N factor of 5, we use *Sequential Forward Selection* (SFS) to select a better set of features. In [GS08] we showed that SFS works best in comparison with a brute force search and Linear Discriminant Analysis on our data. For the feature selection we use an extended feature set, i. e. joint angles and Δ joint angles. Since the Breakfast dataset is session-dependent we also use the marker positions and the Δ marker positions directly to distinguish between object positions. When selecting the optimal features, we can improve the recognition rate by about 25% relative. The results show that the C&C recognition works comparably well as the sequential recognition (see Fig. 7.8).

Implicit Context Model

In the sequential system we somehow have an implicit context model, i. e. we model which motion primitives of the finer layers are performed concurrently in the training data. If we have a limited dataset and the test data is similar to the training data, this will give better recognition results than recognizing concurrent motions independent of each other. This advantage will turn into a disadvantage, if we recognize more realistic motions. In order to support this theory, we performed several experiments where we train our system on a set of activities and test it on another activity.

7.5.3 Recognizing unseen activity

For this experiment we trained the system on 5 of the 6 activities in the Breakfast dataset and tested it on the activity "Cook Scrambled Eggs" which was not

Beam	hypothesis top N factor	PER	real time factor
1	5	70.9%	0.29
1	10	71.5%	1.25
1	15	71.4%	1.48
1	20	71.4%	1.48
2	5	63.6%	3.57
2	10	61.9%	4.54
2	15	61.3%	6.06
2	20	61.2%	6.06
3	5	63.3%	10.5
3	10	62.9%	13.3
3	15	62.3%	15.4
3	20	61.7%	15.4
4	5	61.6%	20.2
4	10	60.7%	25
4	15	60.3%	33.3
4	20	60.3%	33.3
5	5	61.2%	40.0
5	10	60.1%	100
5	15	59.5%	100
5	20	59.6%	100
6	5	61.0%	100
6	10	60.1%	500
6	15	59.5%	500
6	20	59.3%	500

Table 7.7: Motion primitive error rates for varying master Beam and hypothesis top N thresholds.

part of the training data. Due to the size of our dataset we use object indepedent motion primitives. The results in Table 7.9 show that the sequential system and the C&C system work comparably on an unseen activity.

Sequential	C&C
83.8%	84.4%

Table 7.9: Motion Primitive Error Rates for recognizing unseen activity "Cook Scrambled Eggs" for Δ joint angles.

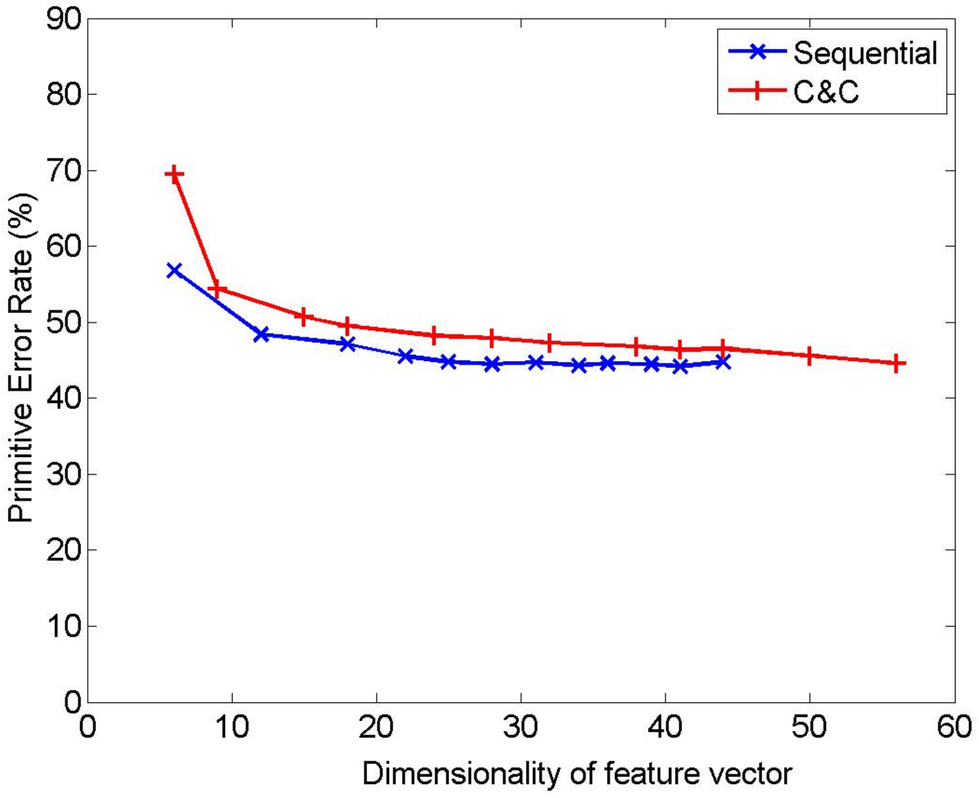

Figure 7.8: Primitive error rates of sequential forward selection on the Breakfast dataset.

For further improvements we again use SFS for the Recognition of the Activity "Cook Scrambled Eggs". The results show that the C&C system retrieves better results than the sequential system if testing on an unknown activity (see Fig. 7.10). In order to further validate this result, we created a synthetic dataset with motion sequences that have not been part of the training data.

Synthetic Test Data

For our synthetic test data we used the original segmentations of the test data of one cross-validation fold and combined it arbitrarily. This can produce gaps between separate motions of the left and right arm and whole body motions, e. g. if the left arm motion is shorter than the right arm motion. We filled these gaps with the motion primitive *wait*. As features we use the 14 Δ joint angles of both arms. Performing a test on the synthetic test data gave us the recognition results shown in Table 7.11. While the recognition rate for the C&C recognition remains the same as the best result for the same cross-validation fold for the

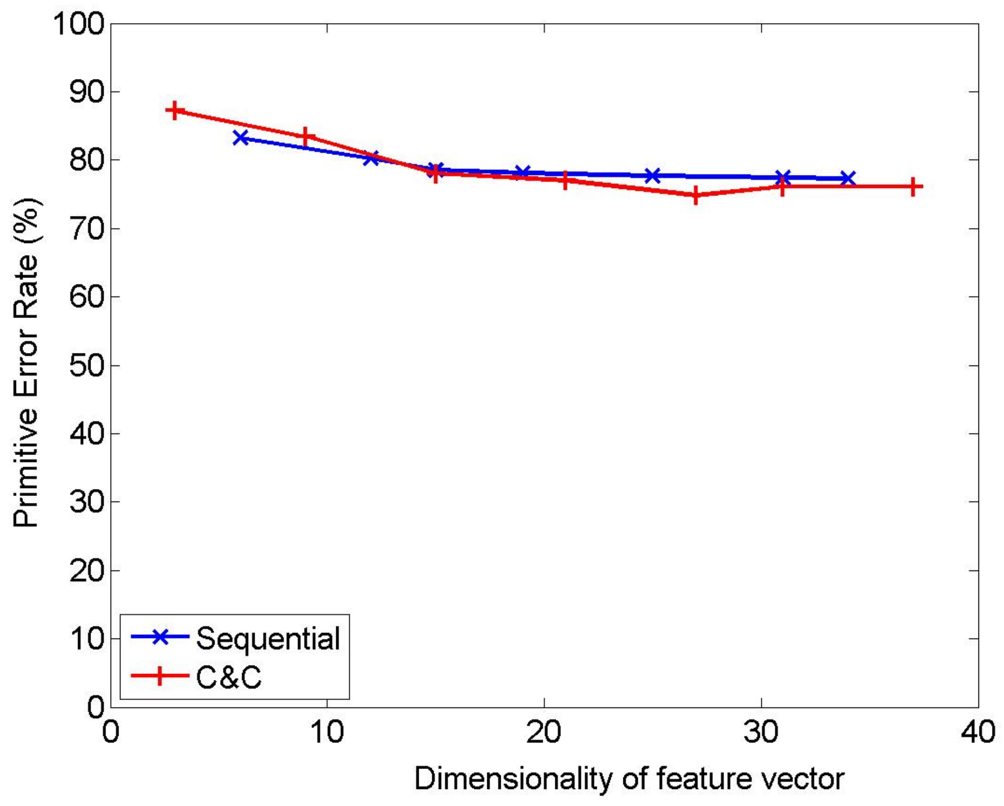

Figure 7.10: Primitive error rates of sequential forward selection for activity "Cook Scrambled Eggs".

tendency towards finer streams experiment at page 80 the recognition rate of the sequential system drops significantly. The recognition results show that the C&C recognition system generalizes better on unseen motion sequences than the sequential system.

System	PER (original data)	PER (synthetic test data)
Sequential	76%	92%
C&C	79%	79%

Table 7.11: Recognition results for testing on synthetic test data and on the original data on cross-validation fold 1.

7.6 Online Recognition

Based on the findings in the previous sections we built an online recognition system on the Breakfast dataset. This system has been demonstrated in the official evaluation of the CRC 588. With this demonstration we have shown that the C&C recognition is capable of recognizing human motions in an online recognition scenario.

8

Marker-less Recognition

In this chapter we will demonstrate the capabilities of our recognition systems concerning marker-less human motion recognition. We transfer knowledge from the marker-based system to a marker-less video-based recognition. We start with two sequential recognition systems, one based on motion capturing and another based on video cameras. We will present performance numbers of different recognition systems as well as describe systems that have been demonstrated in online recognition. Finally, we will present a video-based person-independent human motion recognition system for motions of seven different activities.

8.1 Video-Based Recognition of Human Motion Sequences

The previoulsy described systems have one major shortcoming. They depend on a tracking system with markers attached to the person whose motions are recognized. Markers are not suitable for human-robot interaction in real-life. It is not practical to attach markers to everybody that is in the robots field of view in their everyday environment. In this section we describe the transfer of our knowledge from the marker-based recognition to marker-less motion recognition with video cameras. We have learned from the marker-based recognition that:

- features need to be normalized
- motion features work well
- optimizing the number of states and the number of Gaussians per state improves performance
- a motion grammar gives the best recognition results for restricted tasks

The main challenge is the generation of good features from the video images while the transfer of the recognition system itself is rather straight forward. Our goal in this section is to transfer the marker-based recognition systems to video-based motion recognition. We compare different types of video-based features with each other and with the marker-based motion recognition systems.

8.1.1 Optical Flow Histogram-based Recognition

The first features that we use for our video-based motion recognition systems are weighted histograms of optical flow (OF). Optical flow is a method for calculating motion in a video image. Optical flow yields a motion vector with direction and length of the motion for each pixel between two consecutive video images. In order to be able to use this information for HMM-based human motion recognition we build histograms over the motion directions. More details about the feature calculation are described in the following section.

Optical Flow Histograms

For the image-based representation of motion, histograms of optical flow directions weighted with their norm values are used. The videos of the motion sequences were captured with the left camera of a robot head, which was placed in front of a table. The robot cameras are dragonfly cameras with a resolution of 460 x 680 pixel and a frame rate of 30. Every frame of the video sequence is represented by a histogram of its overall motion directions without any further local information.

The optical flow is computed using a pyramid version of the Lucas Kanade method, as described in [LK81], [Bou99].

The weighted histogram for frame t is calculated from the optical flow OF of images (I_t, I_{t+1}).

$$OF(I_{t+\delta t}(x,y)) = I_t(x + u(\delta t), y + v(\delta t)) \tag{8.1}$$

The motion vector $(u(\delta t), v(\delta t))$ is used to calculate the resulting motion direction θ, indicated by an angle value from $[-\pi, \pi]$ and γ defining the motion intensity.

$$tan(\theta(u(\delta t), v(\delta t))) = \frac{v(\delta t)}{u(\delta t)} \tag{8.2}$$

$$\gamma(u(\delta t), v(\delta t)) = \sqrt{u(\delta t)^2 + v(\delta t)^2} \tag{8.3}$$

Histogram Calculation

In order to simplify the notation the time index δt is presumed to be fixed and omitted in the following. The elements for one bin of the histogram are calculated based on the motion angle θ. As the motion angle ranges from $[-\pi, \pi]$, the vector of elements for the k-th bin $h(k)$ of the histograms with n bins can be defined as:

$$h(k) = \{(u,v)|\theta(u,v) \geq \frac{(k2\pi)}{n} - \pi \cap \theta(u,v) < \frac{((k+1)2\pi)}{n} - \pi\} \tag{8.4}$$

The number of elements in $h(k)$ is indicated by $N(h(k))$ and the elements represent the coordinates (u,v) of the related optical flow vector. So, the i-th element of $h(k)$ is defined as

$$h(k,i) = (u,v) \tag{8.5}$$

The k-th bin for the weighted histogram is calculated from the intensity γ of all elements in the vector as shown in

$$H(k) = \sum_{i=1}^{N(h(k))} \gamma(h(k,i)) \tag{8.6}$$

The histograms are sampled over time resulting in sequence of a multidimensional input vectors for the HMMs. An example for the occurring optical flow vectors as well as for its weighted histogram can be seen in Fig. 8.1.

The Recognition System

The motion sequences of the KIT-F dataset (see 3.2.2) were simultaneously recorded with a Vicon motion capture system and the dragonfly camera system of a humanoid robot head. As video and marker data captures were taken at the same time, the markers are partly visible in the video images but are only used by the Vicon system. We resampled the video to 20 fps for good comparability with the marker-based recognition. We used the same recognition system for the video-based recognition as in section 5.1.2. The only difference are the used features. Instead of the Δ joint angles we use weighted histograms of optical flow.

Experiments and Results on the KIT-F Dataset

Since the number of bins in the histogram is not given beforehand, we have to figure out which number works best. We optimized the dimensionality of the

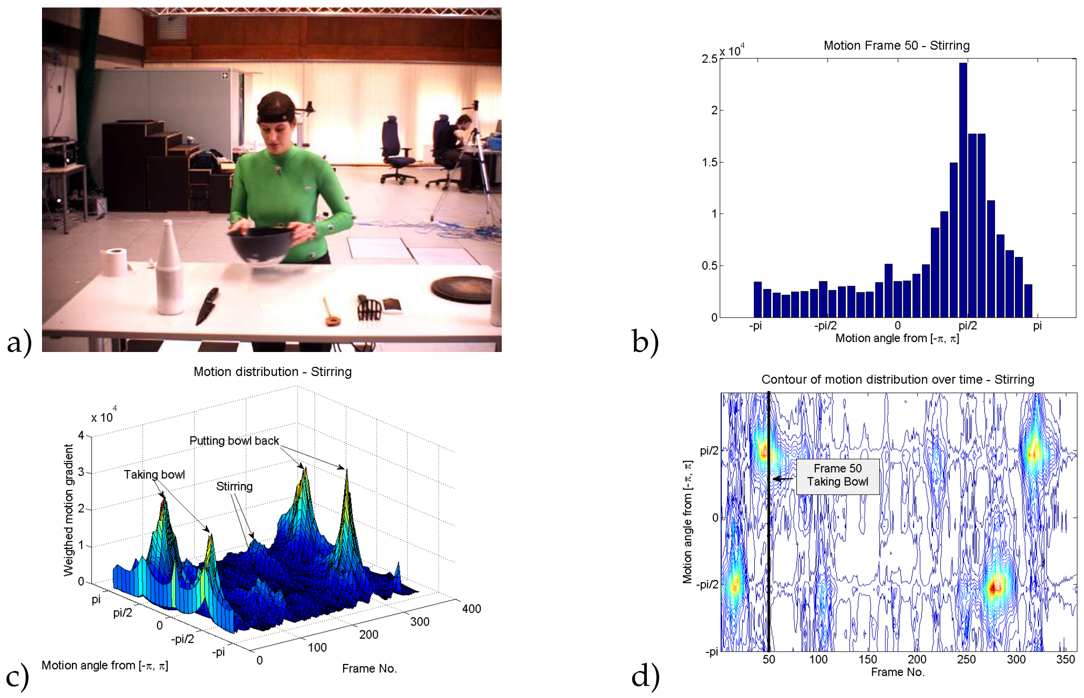

Figure 8.1: Example for motion sequence *stirring*: (a) one frame of motion unit *taking bowl* as part of the motion sequence *stirring*, (b) motion gradient histogram for that particular frame, (c) distribution of optical flow gradients for the complete motion sequence, (d) motion distribution over time of the motion sequence *stirring*.

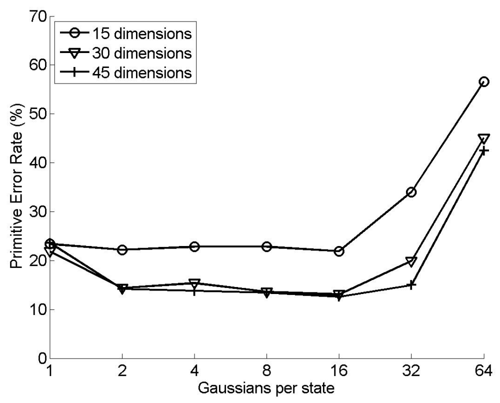

Figure 8.2: Primitive error rates for varying numbers of Gaussians per state and dimension of the feature vector.

feature vector and again the number of Gaussians to see if we can transfer the methods from the marker-based system and to find out how the video-based recognition works in comparison to the Δ joint angles. For better comparability we kept the number of states to 4 since this was the setup for the marker-based recognition for the same dataset. For the experiments we used a 0-gram context model and optimized the primitive insertion penalty and the context model weight. The best values were 64 for the primitive insertion penalty and 32 for the context model weight. Figure 8.2 shows the recognition results over the number of Gaussians with different amount of feature dimensionality (also see [GKWS09]).

We can see that we can achieve comparable error rates as with the marker-based system by simply replacing the Δ joint angle by weighted histograms of optical flow while keeping the rest of the system. The figure also shows that a 30 or more dimensional histogram gives good recognition rates. We decided to use the 30-dimensional feature vectors since 30 dimensions are in the range of the number of Δ joint angles and so they are suited better for comparing both systems. Nevertheless, the 45-dimensional feature vectors give slightly better results. The best recognition rate is achieved when using 16 Gaussians per state as in the marker-based system and we achieve an error rate of 13.1 %. In comparison to the marker-based recognition rate of 12.4 % on Δ joint angles in section 5.1.2 on the same dataset, this is a very good recognition rate. The main drawback of the optical flow histograms is that it is too slow for online recognition (see Table 8.6).

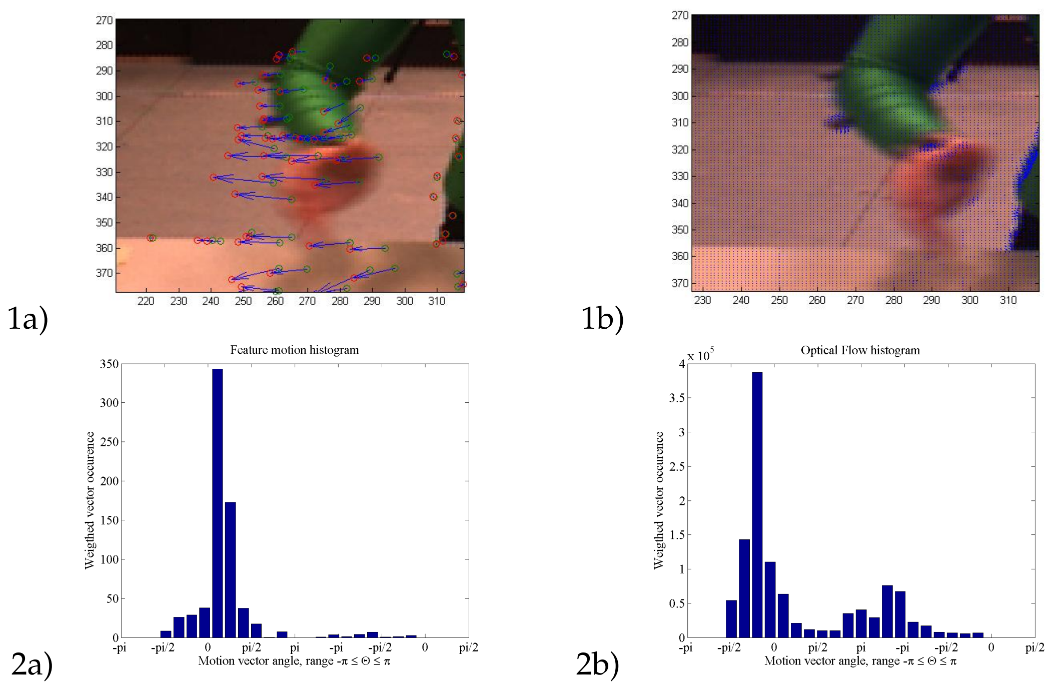

Figure 8.3: Comparison of feature motion (a) and optical flow (b) histogram: 1) example for vector plot, 2) bar plot of weighted motion histograms

8.1.2 Feature Flow Histogram-based Recognition

A possibility of speeding up the motion histogram calculation is the use of feature flow (FF) instead of optical flow as describe in Section 8.1.1. We compare the recognition systems with feature flow and optical flow with respect to the recognition results and speed.

Feature Flow Histograms

Motion information can be gained from dense optical flow fields or from tracking of feature points only. The feature tracking used in this thesis is based on the Lucas-Kanade method described in [LK81] and [Tom91]. The initialization and tracking of features follows the pyramidal KLT feature tracking implementation by [Köh08]. The initialization of new features is done for every frame following the algorithms of Shi and Tomasi [ST94]. Every frame of the video sequence is represented by a global histogram of its overall motion directions without any further local information. The weighted histogram for frame t is calculated from the motion vector of the feature points of images I at time index t and $t+1$ (I_t, I_{t+1}). The motion vector $(u(\delta t), v(\delta t))$ of the feature is used to calculate the resulting motion direction θ, indicated by an angle value from

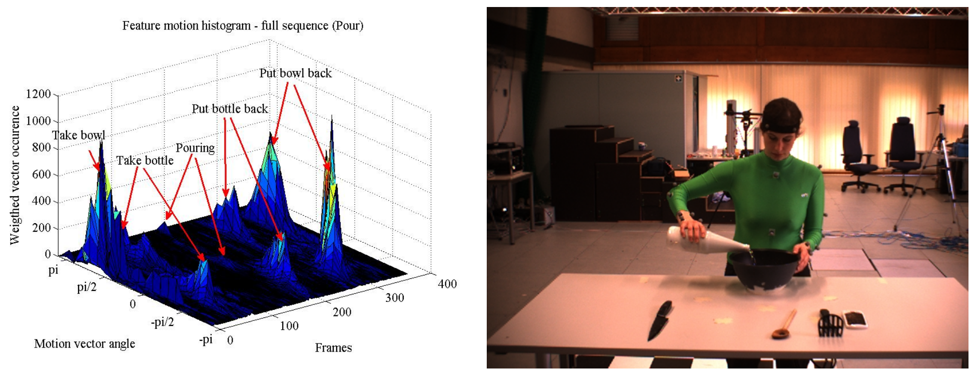

Figure 8.4: Example for feature motion histogram distribution for motion sequence "Pouring water"

$[-\pi, \pi]$ and γ defining the motion intensity. The histograms are calculated are calculated the same way as the optical flow histograms (see page 89).

Examples for the feature flow motion compared to the optical flow motion as well as the resulting histograms can be seen in Figure 8.3. The histograms are sampled over time resulting in a 30-dimensional input vector for the HMMs. An example for the histogram distribution over a complete action sequence can be seen in Figure 8.4.

Comparison with Optical Flow-based and Marker-based Recognition using a Context-Free Grammar on the KIT-F Dataset

In order to evaluate the recognizer performance when using the histograms of feature flow, we compared the results with histograms of optical flow and with Δ joint angles. The results have been calculated with the recognition system described in 5.1.2 using the KIT-F dataset and have been published in [KGSS12]. We used a context free grammar to guide the recognition process. The primitive error rates of the different systems are shown in Table 8.5.

Feature	OF	FF	Δ joint angles
PER	3.1%	2.5%	1.7%

Table 8.5: Comparison of video-based features with Δ joint angles on the KIT-F dataset.

The recognition performance is high since the performed motion sequences can be described very well with a context free motion grammar and therefore the grammar models the motion sequences well and does allow few alternative

sequences. When applying the recognition system to a dataset with more unconstrained motion sequences, we can see that we get worse results due to the more complex grammar and we can see that the OF recognition outperforms the feature flow system.
The runtime is evaluated for feature flow histograms and optical flow histograms on a 2.83GHz Intel Core2Quad processor with 8GB RAM. For the evaluation the processing time per frame for each sequence is analyzed. It can be shown that the optical flow histogram calculation takes around 764 ms and feature flow histograms needs 34 ms, which is about 20 times faster. We can see that feature flow-based recognition is much faster than optical flow-based recognition at the cost of a slightly worse error rate.

Feature	OF	FF
Processing time (per frame)	764 ms	34 ms

Table 8.6: Comparison of the processing time for optical flow (OF) and feature flow (FF) histograms.

Error Measurement

Activity Error Rate Besides calculating the primitive error rate (PER) on motion primitive level, we can also evaluate the results on activity level. Given the recognized activities and the actual performed activities as ground truth, we can compute the activity accuracy. When using a motion grammar as context model for the recognition process, the best way of retrieving the recognized activity is to model the grammar in a way that we have a non-terminal node for every activity, i. e. we can extract this node given the recognized motion primitive sequence to get the recognized activity.

Comparison with Optical Flow-Based Recognition using a Context-Free Grammar on the ADL Dataset

A more complex dataset is the University of Rochester Activities of Daily Living dataset [MPK09]. This set comprises 10 different activities of 5 different persons, which have been manually segmented using a total of 53 motion primitives. We evaluated the dataset using optical flow and feature flow performing a 3-fold cross-validation.
The results show that while optical flow performs quite well for person independent recognition feature flow also gives promising recognition rates. However, the more difficult dataset makes the benefits of OF more prominent.

Feature	OF	FF
PER	36.5%	45.0%
Activity error rate	18.0%	28.7%

Table 8.7: Comparison of video-based features on the Activities of Daily Living dataset.

8.2 Person-Independent Recognition

One of the main advantages of marker-less motion recognition is the possibility of recognizing motions of any person without preparing the person beforehand by attaching markers.

When doing person independent recognition there are two challenges in comparison to person dependent recognition. Firtly, two persons perform the same motion primitive differently, which increases variance and thus makes it more difficult to distinguish two motion primitives. In order to achieve good person independent recognition results we use a motion grammar, which limits the possible sequences of motion primitives and therefore limits the number of motion primitives that have to be distinguished in every time step. We learn the grammar automatically from our segmentations. This eliminates the drawback of manually creating a grammar but we still need segmentations of the motion sequences. Secondly, two persons perform the same activity using different sequences of motion primitives. Assuming that there is a limited set of motion sequences an activity is usually performed, we can improve the recognition system by training the grammar on motion sequences by pooling data from as many persons as possible.

The Recognition System

For the experiments in this section we used the Minta dataset with multiple kitchen activities. The motion recognition uses the same feature flow histograms as in the previous section to recognize the motions of an observed person. The number of states and the number of Gaussians per mixture were optimized in the cross-validation experiments described below. The possible concatenations of the motion primitives are modeled using an automatically learned context-free grammar. We extended the Sequitur algorithm [NMW97] (see section 4.4.2) to work on a set of motion sequences instead of only one sequence. The grammar allows a reliable recognition of the possible motion sequences.

Error Measurement

Online Primitive Error Rate Since this system is intended to run online, we define an online primitive error rate. We also use this error rate for the offline evaluation of the system for better comparability with the online system presented in section 8.3. For an online system we need to recognize the current motion primitive before we have seen the whole activity. Instead of retrieving the best motion sequence after we have finished the sequence of feature vectors, we output the best current motion primitive every second and compare this sequence of motion primitives to the sequence of reference motion primitives. The error rate between those sequences is calculated the same way as for the standard primitive error rate.

Experiments and Results on the Minta Dataset

For the evaluation of our system, we optimized the recognizer on 560 image sequences of eight persons using 8-fold cross-validation (CV). The 140 sequences of the two remaining persons were used as an evaluation set. Recognition results are given as the average recognition rates for the cross-validation and the recognition rate on the evaluation set (EVAL set).

Task	Lay Table	Prepare Cereals	Prepare Pudding	Eat with Spoon
CV	23.1 %	21.1 %	23.9 %	26.9 %
EVAL	16.9 %	12.5 %	19.6 %	57.5 %
Task	Eat with Fork	Clear Table	Wipe Table	*Avg. Rate*
CV	42.0 %	13.5 %	52.7 %	**29.4** %
EVAL	27.2 %	10.3 %	33.0 %	**25.6** %

Table 8.8: Motion primitive error rate given per task for the CV and EVAL sets. The bold numbers are averages over all tasks.

Table 8.8 shows the results of the motion recognition system for the different tasks as published in [GKR$^+$11]. The primitive error rate is low for most of the tasks. Many of the primitives that are not recognized well belong to the task *Wipe Table*, which is the most difficult task to recognize for the motion recognition system due to the large variance in wiping the table. Another problem with the recognition system is the recognition of motions from one specific person, which is due to the nature of our pre-processing and the unusual feature distribution in the videos of this person. The approach is robust against different clothing and appearance of all the other 9 persons. Only unusual appearances which are very different to the ones in the training data are problematic for the recognition system.

8.3 Person-Independent Online Recognition of Intentions, Activities and Motions

We will now describe a system that is capable of online recognizing motion primitives of unknown persons based on video images. It gives an insight on possible application as it is part of the intention recognition system for the humanoid robot ARMAR. The proposed system is based on the one in section 8.2. The main difference is the feature normalization, which is performed for online recognition as in section 5.3.1. For training and for evaluation we used the described feature normalization, whereas we updated and performed the feature normalization in chunks of 1 second. Every second we recognized the most likely performed motion primitives and fed them into the intention recognition system. The motion recognition system does always continue recognizing without considering previous feature vectors again, which allows us to recognize the motion primitive with more than 30 frames per second. This also enables the system to recognize arbitrarily long motion sequences, that are only limited by the expressiveness of the used motion grammar. If we do not detect any motion in the video image for a certain time we restart with the start symbol in the grammar since we can then assume that next time we detect a motion in the image someone is entering the scene and starting with a new activity.

In order to measure the performance of the system we calculate the online primitive error rate described in section 8.2. The results for the cross-validation and the evaluation set are shown in Table 8.9.

The recognition results are quite promising, if we consider that we recognize motions of unknown persons with a single video camera. They are worse than the offline recognition result but far better than guessing the correct motion primitive which would give us an error rate of 98.4%.

The results together with other knowledge such as the recognized activity, available objects and the time of day, are fed into an intention recognition system. The result in [GKR$^+$11] shows the good recognition rate of the intention recognition based on the underlying motion and activity recognition systems. The error rate of the intention recognition system decreases from 19.6% to 16.5%

Task	Lay Table	Prepare Cereals	Prepare Pudding	Eat with Spoon	Eat with Fork	Clear Table	Wipe Table	*Avg. Rate*
CV	37.6 %	38.8 %	40.5 %	33.9 %	55.2 %	40.2 %	57.6 %	**43.7** %
EVAL	33.7 %	34.7 %	43.0 %	55.3 %	36.9 %	38.7 %	44.5 %	**41.4** %

Table 8.9: Average motion primitive error rate given per task for the CV and EVAL sets.

when using the recognized motion primitives in addition to the other knowledge, i. e. the recognized activity, available objects and the time of day. It also shows that the complementarity of the motion and the activity recognition allow the system to be robust against single recognition failures. Therefore, the motion recognition system is a valuable part of the intention recognition system. It has been published in [GKR+11] and demonstrated at various occasions.

9

Conclusion and Future Work

The ultimate goal of this thesis was the development of a recognition system for a humanoid robot. The main aspects that had to be tackled were described in Section 1.2 as follows:

- unobtrusive capturing of human motions
- continuous recognition of human motions
- online recognition of motions to enable immediate robot reaction
- integration of context information into the recognition system
- development of motion modeling technique that scales well to more unconstrained recognition domains

9.1 Summary of Thesis Results

To achieve the above goals we implemented and evaluated several state-of-the-art techniques. For this purpose, we collected multiple datasets to compare our proposed methods with the state-of-the-art in human motion recognition. The main contributions of this thesis are as follows:

Structured Functional Motion Decomposition The basis of a motion recognition system is a way to generate models of human motions. Many motion recognition systems use motion primitives for their recognition but a methodology for structured decomposition of motions into meaningful motion units is missing. To find motion segments that facilitate the modeling and recognition of a large variety of human motions, we developed an approach for human motion decomposition which is based on findings in biology and sports sciences.

Sequential Motion Recognition Based on the functional motion decomposition, we developed a sequential marker-based motion recognition system. This system was evaluated on several datasets. We used the system to optimize the features for motion recognition and to find strategies for generating good motion primitive models as well as context models. The different optimization steps gave an relative improvement of 30% to 70% each (see Figure 9.1). This knowledge was then used for investigations on C&C motion recognition as well as on video-based motion recognition.

C&C Motion Recognition Based on the primitives from the functional motion decomposition, we developed a recognition system that recognizes motions of individual body parts. The improvement includes an innovative modeling and search technique for motion sequences, an appropriate context model and a methodology for error measurement. Together these components allow to create a flexible and scalable motion recognition which allows to tackle more complex activities than other state-of-the-art systems, and improve the Primitive Error Rates by up to 15% relative for a complex recognition task (see Figure 9.2) in comparison to the sequential system.

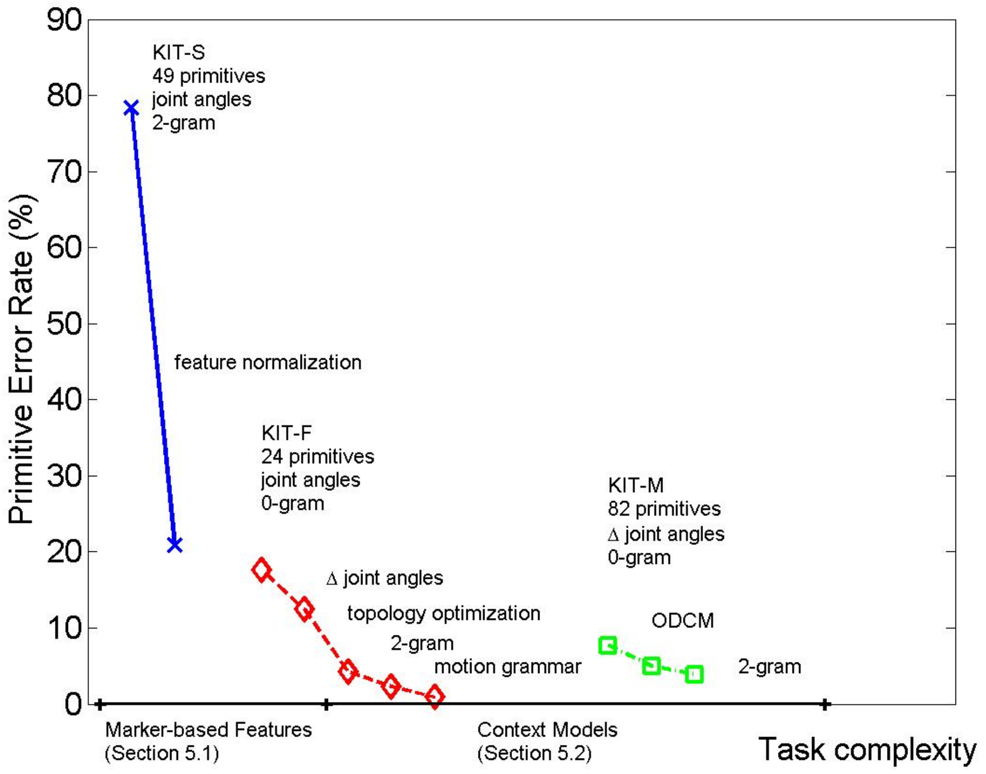

Figure 9.1: Overview of the achieved recognition rate improvements for the sequential marker-based systems.

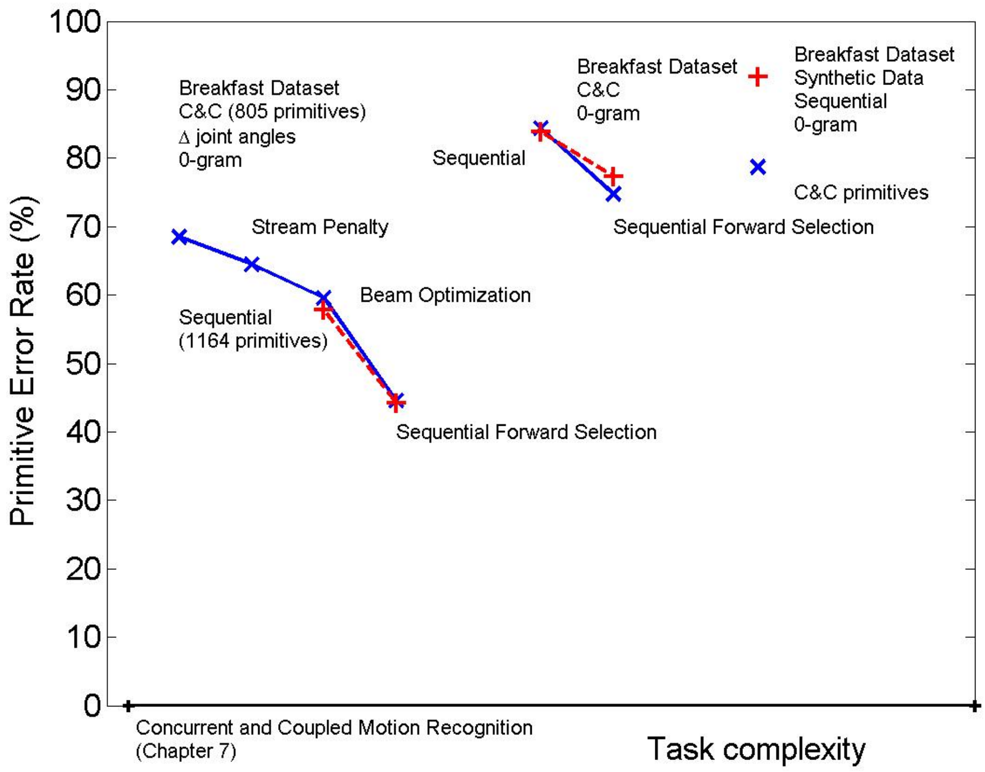

Figure 9.2: Overview of the improvements in recognition rates for the comparison of the sequential and the C&C recognition systems.

Video-based Motion Recognition To prove the viability of our algorithms in real-live scenarios, we used the vision sensors of a humanoid robot for motion recognition experiments. Figure 9.3 shows the improvements in recognition results for the video-based system. We developed a recognition system for a humanoid robot that online recognizes motions of human daily activities. The recognition system was integrated into the robots intention recognition system.

9.2 Suggestions for Future Work

Further Investigations on Integrating Context Information In this thesis we covered a variety of different context models which we used for our motion recognition. Nevertheless, there is far more to be done. We expect the recognition rate to improve further by developing more sophisticated context models which include but are not limited to the following:

1. Combining conventional N-grams with the ODCM.

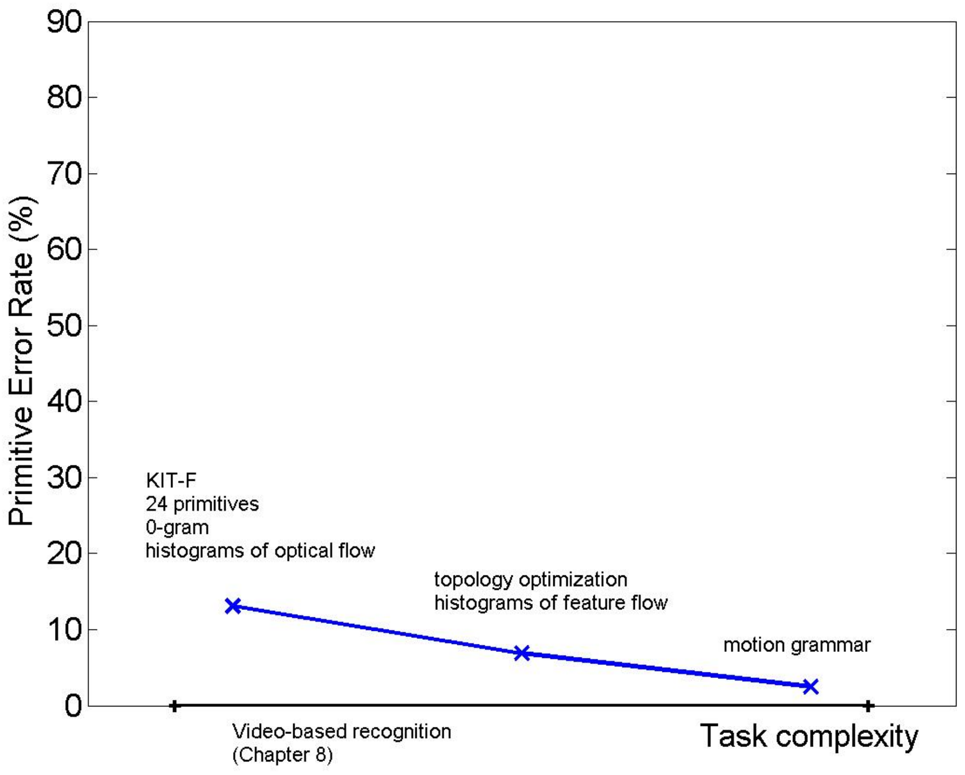

Figure 9.3: Overview of the improvements in recognition rates for the video-based systems.

2. Extending the ODCM to the C&C recognition.
3. Developing context models that include the dependencies of concurrent motion primitives

Using those context models will surely improve the recognition accuracy while hopefully also speeding up the recognition.

Further Investigations on Video-based Recognition of C&C Motion Primitives The next step in recognizing human motions on a humanoid robot should integrate the C&C recognition with a video-based feature extraction. Different methods have been proposed to extract features for individual body parts from videos. Those features vary in terms of robustness and in terms of accuracy concerning the description of human motions.

Investigations on Recognizing Motion Sequences with Unknown Motions In this thesis, we assumed that each observed motion primitive was modeled and trained during the training of the recognizer. In future research it should be investigated how to handle motions that have not been modeled explicitly in the recognition system. This includes modeling HMMs to detect unknown motion primitives as well as extending the context models to handle the results of those HMMs.

Bibliography

[AAWD10] Eren E. Aksoy, Alexey Abramov, Florentin Wörgötter, and Babette Dellen. Categorizing Object-Action Relations from Semantic Scene Graphs. In *2010 IEEE International Conference on Robotics and Automation (ICRA)*, pages 398–405. IEEE, 2010.

[AC97] Jake K. Aggarwal and Qin Cai. Human Motion Analysis: A Review. In *Nonrigid and Articulated Motion Workshop, 1997. Proceedings., IEEE*, pages 90–102. IEEE, 1997.

[ACM$^+$08] Massimiliano Albanese, Rama Chellappa, Vincenzo Moscato, Antonio Picariello, V. S. Subrahmanian, Pavan Turaga, and Octavian Udrea. A Constrained Probabilistic Petri Net Framework for Human Activity Detection in Video. *IEEE Transactions on Multimedia*, 10(6):982–996, 2008.

[AF94] James F. Allen and George Ferguson. Actions and Events in Interval Temporal Logic. *Journal of Logic and Computation*, 4(5):531, 1994.

[All94] Jont B. Allen. How do Humans Process and Recognize Speech? *IEEE Transactions on Speech and Audio Processing*, 2(4):567–577, 1994.

[AR11] Jake K. Aggarwal and Michael S. Ryoo. Human Activity Analysis : A Review. *ACM Computing Surveys*, 2011.

[ARA$^+$06] Tamim Asfour, K Regenstein, Pedram Azad, Joachim Schröder, Alexander Bierbaum, Nikolaus Vahrenkamp, and Rüdiger Dillmann. ARMAR-III : An Integrated Humanoid Platform for Sensory-Motor Control. In *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*, 2006.

[BA01] Darrin C. Bentivegna and Christopher G. Atkeson. Learning from Observation using Primitives. In *Proceedings of the 2001 ICRA. IEEE International Conference on Robotics and Automation*, volume 2, pages 1988–1993. IEEE, 2001.

[BJM83] Lalit R. Bahl, Frederick Jelinek, and Robert L. Mercer. A Maximum Likelihood Approach to Continuous Speech Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(2):179–190, 1983.

[Bob97] Aaron F. Bobick. Movement, Activity and Action: The Role of Knowledge in the Perception of Motion. *Philosophical Transactions of the Royal Society of London. Series B, Biological sciences*, 352(1358):1257–65, 1997.

[BOP97] Matthew Brand, Nuria Oliver, and Alex P. Pentland. Coupled Hidden Markov Models for Complex Action Recognition. *Proceedings* *of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 994–999, 1997.

[Bou99] Jean-Yves Bouguet. Pyramidal Implementation of the Lucas Kanade Feature Tracker Description of the algorithm. *In Practice*, 1(2):1–9, 1999.

[BW97] Aaron F. Bobick and Andrew D. Wilson. A State-based Approach to the Representation and Recognition of Gesture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(12):1325–1337, 1997.

[Cho56] Noam Chomsky. Three Models for the Description of Language. *Information Theory, IRE Transactions on*, 2(3):113–124, 1956.

[DW11] Michael S. Del Rose and Christian C. Wagner. Survey on Classifying Human Actions through Visual Sensors. *Artificial Intelligence Review*, 37(4):301–311, 2011.

[FGH$^+$97] Michael Finke, Petra Geutner, Hermann Hild, Thomas Kemp, Klaus Ries, and Martin Westphal. The Karlsruhe-Verbmobil Speech Recognition Engine. In *ICASSP*, pages 83–86, 1997.

[FMJ02] Ajo Fod, Maja J. Mataric, and Odest Chadwicke Jenkins. Automated Derivation of Primitives for Movement Classification. 12(1):39–54, 2002.

[Gär09] Stefan Gärtner. *Robot Motion Planning Based on Human Motion Capture Data*. PhD thesis, 2009.

[GDDD04] Nagia Ghanem, Daniel Dementhon, David Doermann, and Larry Davis. Representation and Recognition of Events in Surveillance Video Using Petri Nets. In *2004 Conference on Computer Vision and Pattern Recognition Workshop*. IEEE, 2004.

[GF06] Gutemberg Guerra-Filho. Towards a Sensorimotor WordNet SM: Closing the Semantic Gap. In *Proc. of the International WordNet Conference (GWC)*, 2006.

[GFK$^+$08] Dirk Gehrig, Andreas Fischer, Hildegard Kühne, Thorsten Stein, Annika Wörner, Hermann Schwameder, and Tanja Schultz. Online Recognition of Daily-Life Movements. In *8th IEEE-RAS International Conference on Humanoid Robots, Workshop Imitation and Coaching in Humanoid Robots*, 2008.

[GKR$^+$11] Dirk Gehrig, Peter Krauthausen, Lukas Rybok, Hildegard Kühne, Uwe D. Hanebeck, Tanja Schultz, and Rainer Stiefelhagen. Combined Intention, Activity, and Motion Recognition for a Humanoid Household Robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4819–4825, 2011.

[GKS10] Dirk Gehrig, Hildegard Kühne, and Tanja Schultz. Erkennung von menschlichen Bewegungen mit Hidden Markov Modellen. In

Sportinformatik trifft Sporttechnologie, Tagung der dvs-Sektion Sportinformatik in Kooperation mit der Deutschen Interdisziplinären Vereinigung für Sporttechnologie, 2010.

[GKWS09] Dirk Gehrig, Hildegard Kühne, Annika Wörner, and Tanja Schultz. HMM-based Human Motion Recognition with Optical Flow Data. In *9th IEEE-RAS International Conference on Humanoid Robots*, pages 425–430. IEEE, 2009.

[Göh92] Ulrich Göhner. *Einführung in die Bewegungslehre des Sports, Teil 1: Die sportlichen Bewegungen (Introduction to human movement science, part 1: sports movements)*. 1992.

[GP03] Martin A. Giese and Tomaso Poggio. Neural Mechanisms for the Recognition of Biological Movements. *Nature reviews. Neuroscience*, 4(3):179–92, 2003.

[GS08] Dirk Gehrig and Tanja Schultz. Selecting Relevant Features for Human Motion Recognition. In *19th International Conference on Pattern Recognition*. IEEE, 2008.

[GSFS10] Dirk Gehrig, Thorsten Stein, Andreas Fischer, and Hermann Schwameder. Towards Semantic Segmentation of Human Motion Sequences. In *33rd Annual German Conference on Artificial Intelligence*, 2010.

[GSSD09] Abhinav Gupta, Praveen Srinivasan, Jianbo Shi, and Larry S. Davis. Understanding Videos, Constructing Plots Learning a Visually Grounded Storyline Model from Annotated Videos. 2009.

[HAH01] Xuedong Huang, Alex Acero, and Hsiao-Wuen Hon. *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*. Prentice Hall PTR, 1st edition, 2001.

[IB00] Yuri A. Ivanov and Aaron F. Bobick. Recognition of Visual Activities and Interactions by Stochastic Parsing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):852–872, 2000.

[JM04] Odest Chadwicke Jenkins and Maja J. Matarić. Performance-Derived Behavior Vocabularies: Data-driven Acqusition of Skills from Motion. *International Journal of Humanoid Robotics*, 1(2):237–288, 2004.

[KGSS12] Hildegard Kühne, Dirk Gehrig, Tanja Schultz, and Rainer Stiefelhagen. On-line Action Recognition From Sparse Feature Flow. In *International Conference on Computer Vision Theory and Applications*, 2012.

[KHN09] Dana Kulic, Imagawa Hirotaka, and Yoshihiko Nakamura. Online Acquisition and Visualization of Motion Primitives for Humanoid

Robots. In *The 18th IEEE International Symposium on Robot and Human Interactive Communication*, pages 1210–1215, 2009.

[KKUG07] Volker Krüger, Danica Kragic, Ales Ude, and Christopher Geib. The Meaning of Action A review on action recognition and mapping. pages 1–36, 2007.

[Köh08] Hildegard Köhler. Motion-based Feature Tracking For Articulated Motion Analysis. In *Conf. on Multimodal Interfaces (ICMI 2008), Workshop on Multimodal Interactions Analysis of Users a Controlled Environment*, 2008.

[KSJ00] Eric R. Kandel, James H. Schwartz, and Thomas M. Jessell. *Principles of Neural Science*. 2000.

[KTS03] Kanav Kahol, Priyamvada Tripathi, and Panchanathon Sethuraman. Gesture Segmentation in Complex Motion Sequences. *Proceedings 2003 International Conference on Image Processing (Cat. No.03CH37429)*, 3:II–105–8, 2003.

[LK81] Bruce D. Lucas and Takeo Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. *Imaging*, 130(x):674–679, 1981.

[LWS02] Yan Li, Tianshu Wang, and Heung-Yeung Shum. Motion Texture: A Two-level Statistical Model for Character Motion Synthesis. In *siggraph2002*, pages 465–472, 2002.

[MA07] Sushmita Mitra and Tinku Acharya. Gesture Recognition: A Survey. *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, 37(3):311–324, 2007.

[Mat02] Maja J. Mataric. Sensory-Motor Primitives as a Basis for Imitation: Linking Perception to Action and Biology to Robotics. In Kerstin Dautenhahn and Chrystopher Nehaniv, editors, *Imitation in Animals and Artifacts*, pages 392–422. 2002.

[MHKS11] Thomas B. Moeslund, Adrian Hilton, Volker Krüger, and Leonid Sigal, editors. *Visual Analysis of Humans - Looking at People*. 2011.

[MPK09] Ross Messing, Chris Pal, and Henry Kautz. Activity Recognition using the Velocity Histories of Tracked Keypoints. *IEEE 12th International Conference on Computer Vision*, 65(ICCV):104–111, 2009.

[Müs08] Jochen Müsseler, editor. *Allgemeine Psychologie*. 2. edition, 2008.

[NMW97] Craig G. Nevill-Manning and Ian H. Witten. Identifying Hierarchical Structure in Sequences: A linear-time algorithm. *Journal of Artificial Intelligence Research*, 7(1):67–82, 1997.

[NN07] Pradeep Natarajan and Ramakant Nevatia. Coupled Hidden Semi Markov Models for Activity Recognition. 2007.

[NZH03] Ram Nevatia, Tao Zhao, and Somboon Hongeng. Hierarchical Language-based Representation of Events in Video Streams. *Conference on Computer Vision and Pattern Recognition Workshop*, pages 39–39, 2003.

[ORP00] Nuria M. Oliver, Barbara Rosario, and Alex P. Pentland. A Bayesian Computer Vision System for Modeling Human Interactions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):831–843, 2000.

[OSU99] Ryuta Osaki, Mitsuomi Shimada, and Kuniaki Uehara. Extraction of Primitive Motion for Human Motion Recognition. pages 351–352, 1999.

[Oze96] Kazuhiko Ozeki. Likelihood Normalization Using an Ergodic HMM for Continuous Speech Recognition. 1996.

[PA04] Sangho Park and Jake K. Aggarwal. A Hierarchical Bayesian Network for Event Recognition of Human Actions and Interactions. *Multimedia Systems*, 10(2):164–179, 2004.

[PHAS09] Peter Pastor, Heiko Hoffmann, Tamim Asfour, and Stefan Schaal. Learning and Generalization of Motor Skills by Learning from Demonstration. *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 763–768, 2009.

[Pin99] Claudio S. Pinhanez. *Representation and Recognition of Action in Interactive Spaces*. PhD thesis, 1999.

[PSH97] Vladimir I. Pavlovic, Rajeev Sharma, and Thomas S. Huang. Visual Interpretation of Hand Gestures for Human-computer Interaction: A Review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):677–695, 1997.

[RA06] Michael S. Ryoo and Jake K. Aggarwal. Recognition of Composite Human Activities through Context-Free Grammar Based Representation. *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2 (CVPR'06)*, 2:1709–1718, 2006.

[RA07] Michael S. Ryoo and Jake K. Aggarwal. Hierarchical Recognition of Human Activities Interacting with Objects. In *Computer Vision and Pattern Recognition. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[RA09] Michael S. Ryoo and Jake K. Aggarwal. Semantic Representation and Recognition of Continued and Recursive Human Activities. *International Journal of Computer Vision*, 82(1):1–24, 2009.

[Rab89] Lawrence R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, 1989.

[RC05] Debra J. Rose and Robert W. Christina. *A Multilevel Approach to the Study of Motor Control and Learning (2nd Edition)*. Benjamin Cummings, 2005.

[RFG01] Giacomo Rizzolatti, Leonardo Fogassi, and Vittorio Gallese. Neurophysiological Mechanisms Underlying the Understanding and Imitation of Action. *Neuroscience*, 2(September):1–10, 2001.

[SB99] Jeroen B. J. Smeets and Eli Brenner. A New View on Grasping. *Motor Control*, 3(3):237–71, July 1999.

[Sch07] Stefan Schaal. The New Robotics–towards human-centered machines. *HFSP Journal*, 1(2):115–126, 2007.

[SCH08] Muralikrishna Sridhar, Anthony G. Cohn, and David C. Hogg. Learning Functional Object-Categories from a Relational Spatio-Temporal Representation. In *Techniques*, 2008.

[SGS11] Sandro Sitto, Dirk Gehrig, and Tanja Schultz. Developing a Human Motion Recognition System by Applying Automatic Segmentation and Model Transfer, Diploma thesis, Cognitive Systems Lab, Karlsruhe Institute of Technology, 2011.

[SHM$^+$04] Yifan Shi, Yan Huang, David Minnen, Aaron F. Bobick, and Irfan Essa. Propagation Networks for Recognition of Partially Ordered Sequential Action. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2004.*, 2(C):862–869, 2004.

[Sim09] Christian Simonidis. Spezifikationen zu den Ganzkörpermenschmodellen im SFB 588. Technical report, 2009.

[SKKK11] Baby Sanmohan, Volker Krüger, Danica Kragic, and Hedvig Kjellström. Primitive-Based Action Representation and Recognition. *Advanced Robotics*, 25(6-7):871–891, 2011.

[SMFW01] Hagen Soltau, Florian Metze, Christian Fügen, and Alex Waibel. A onepass decoder based on polymorphic linguistic context assignment. In *ASRU*, pages 214–217, 2001.

[SP95] Thad Starner and Alex P. Pentland. Real-time American Sign Language Recognition from Video using Hidden Markov Models. In *Proceedings., International Symposium on Computer Vision 1995*, pages 265–270. IEEE, 1995.

[ST94] Jianbo Shi and Carlo Tomasi. Good Features to Track. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition CVPR94*, 94(June):593–600, 1994.

[Ste08] Günther Stelzner. *Zur Modellierung und Simulation biomechanischer Mehrkörpersysteme*. PhD thesis, 2008.

[Tom91] Carlo Tomasi. Detection and Tracking of Point Features Technical Report CMU-CS-91-132. *Image Rochester NY*, 13(April):1–22, 1991.

[TWG$^+$14] Dominic Telaar, Michael Wand, Dirk Gehrig, Felix Putze, Christoph Amma, Dominic Heger, Ngoc Thang Vu, Mark Erhardt, Tim Schlippe, Matthias Janke, Christian Herff, and Tanja Schultz. BioKIT - Real-time Decoder For Biosignal Processing. In *The 15th Annual Conference of the International Speech Communication Association (Interspeech 2014)*, Singapore, 2014.

[VBT03] Thinh Vu, Francois Bremond, and Monique Thonnat. Automatic Video Interpretation: A Novel Algorithm for Temporal Scenario Recognition. In *18th Int. Joint Conf. on Artif. Intelligence*, pages 1295–1300, 2003.

[VKKL07] Isabel S. Vicente, Ville Kyrki, Danica Kragic, and Martin Larsson. Action Recognition and Understanding Through Motor Primitives. *Advanced Robotics*, 21(15):1687–1707, November 2007.

[VM99] Christian Vogler and Dimitris Metaxas. Parallel Hidden Markov Models For American Sign Language Recognition. *Proceedings of the Seventh IEEE International Conference on Computer Vision*, pages 116–122 vol.1, 1999.

[Wer11] Sergej Werfel. Integration von Objektwissen in die automatische Erkennung menschlicher Bewegungen, Student research thesis, Cognitive Systems Lab, Karlsruhe Institute of Technology. 2011.

[WH99] Ying Wu and Thomas S. Huang. Vision-Based Gesture Recognition: A Review. *Time*, 1999.

[WLZ05] Yi Wang, Zhi-Qiang Liu, and Li-Zhu Zhou. Learning hierarchical non-parametric hidden Markov model of human motion. In *Proceedings of 2005 International Conference on Machine Learning and Cybernetics*, volume 6, pages 3315–3320, August 2005.

[WSXZ01] Tian-Shu Wang, Heung-Yeung Shum, Ying-Qing Xu, and Nan-Ning Zheng. Unsupervised Analysis of Human Gestures. In *PCM '01: Proceedings of the Second IEEE Pacific Rim Conference on Multimedia*, pages 174–181, London, UK, 2001. Springer-Verlag.

[YOI92] Junji Yamato, Jun Ohya, and Kenichiro Ishii. Recognizing Human Action in Time-sequential Images Using Hidden Markov Model. In *Proceedings 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 379–385. IEEE Comput. Soc. Press, 1992.

[ZTH11] Zhang Zhang, Tieniu Tan, and Kaiqi Huang. An Extended Grammar System for Learning and Recognizing Complex Visual

Events. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(2):240–55, 2011.

Zusammenfassung

In den letzten Jahrzenten halten Computer immer mehr Einzug in unser tägliches Leben. Bisher ist die Interaktion jedoch meist auf Maus, Tastatur und Monitor beschränkt. Die Interaktion mit Computern ist daher nicht intuitiv. In den letzten Jahren wurde zunehmend an intuitiveren Mensch-Maschine Schnittstellen geforscht.

Um Aktivitäten oder Bewegungen von Menschen im Alltag automatisch erkennen zu können sind Sensoren notwendig, die nicht invasiv sind und die nicht am Körper getragen werden. Für eine unaufdringliche Erfassung von Bewegungen eignen sich insbesondere optische Sensoren. Optische Sensoren wir Videokameras ermöglichen darüber hinaus die Bearbeitung weiterer Erkennungsaufgaben wie z.B. die Personen- oder Objekterkennung. Obwohl das Erkennungssystem unaufdringlich sein muss, lässt sich die Entwicklung der meisten Systemkomponenten auch mit anderen Sensoren durchführen. Dies ermöglicht die gleichzeitige Entwicklung eines unaufdringlichen Erfassungssystems sowie der Komponenten zur Bewegungsmodellierung und -erkennnung.

Typische Anwendungsgebiete für die Erkennung menschlicher Bewegungen sind die Mensch-Roboter-Interaktion, die Unterhaltungsbranche, der Überwachungsbereich, Sport und Medizin sowie die Indizierung von Videodatenbanken. Vermutlich die schierigste Aufgabe ist die Erkennung von Bewegungen für die Mensch-Roboter-Interaktion, bei der Bewegungen erkannt werden müssen, die von einfachen Gesten bis hin zu komplexen alltäglichen Handlungen reichen. Der Fokus dieser Arbeit sind die Mensch-Roboter Interaktionen, genauer gesagt Interaktionen mit humanoiden Robotern. Angewendet werden die Entwicklungen auf ARMAR III [ARA$^+$06], einen humanoiden Roboter, der im Sonderforschungsbereich 588 "Humanoide Roboter - Lernende und kooperierende multimodale Roboterëntwickelt wurde. Wenn Menschen miteinander interagieren, dann wissen diese intuitiv, was der jeweils andere gerade tut und was dessen Gesten bedeuten. Ihnen ist meistens nicht bewusst, welche komplexen Prozesse in ihrem Gehirn ablaufen um ihnen die Erkennung von einfachen Bewegungen oder auch komplexen Handlungen zu ermöglichen. Um einen humanoiden Roboter mit diesen Fähigkeiten auszustatten, müssen einige Herausforderungen angegangen werden. In dieser Arbeit wird ein System zur online Erkennung menschlicher Bewegungen enwickelt, das mit den eingeschränkten Sensorinformationen eines humanoiden Roboters zurechtkommt. Die erkannten Bewegungen dienen zum Beispiel als Eingabe für das Intentionserkennungssystem von ARMAR.

In dieser Arbeit wurde ein System zur Erkennung menschlicher Bewegungen konzipiert, entwickelt und implementiert. Zu diesem Zweck werden systema-

tisch Methoden und Algorithmen für alle Komponenten des Erkennungssystems entwickelt, d. h. Merkmalsextraktion, Segmentierung und Labeling, Modellierung von Bewegungsprimitive und Kontext sowie die Dekodierung. Es werden Techniken implementiert und evaluiert, die dem aktuellen Stand der Forschung entsprechen. Es werden mehrere Datensätze aufgenommen, um die entwickelten Methoden mit dem Stand der Forschung zu vergleichen. Die wesentliche Beiträge dieser Arbeit sind:

Strukturierte funktionale Bewegungszerlegung: Die Basis eines Systems zur Bewegungserkennung bilden Methoden zur Erstellung von Modellen menschlicher Bewegungen. Viele Erkennungssystem verwenden Bewegungsprimitive, d. h. aussagekräftige Bewegungseinheiten, für ihre Erkennung. Ein strukturiertes Vorgehen zur Definition solcher Einheiten existiert bisher jedoch nicht. In dieser Arbeit wird ein Verfahren zur Bewegungszerlegung vorgestellt, das auf Forschungsergebnissen der Biologie und der Sportwissenschaften basiert. Dieser Ansatz liefert Bewegungsprimitive, welche die Modellierung und Erkennung einer Vielzahl an menschlichen Bewegungen ermöglicht.

Flexible und skalierbare Bewegungserkennung: Basierend auf den Bewegungsprimitiven der funktionalen Bewegungszerlegung wird ein System entwickelt, das Bewegungen einzelner Körperteile erkennt. Dies beinhaltet eine innovative Modellierungs- und Suchtechnik für Bewegungsprimitive, ein geeignetes Kontextmodell sowie Methoden zur Fehlerberechnung. Dieser Ansatz erlaubt die Erkennung von komplexeren Bewegungen als bei vergleichbaren Erkennungssystemen. Auf einem komplexen Datensatz konnte die Erkennungsrate mit diesem Ansatz um 15% relativ gesteigert werden.

Bewegungserkennung auf einem humanoiden Roboter: Um die Anwendbarkeit der entwickelten Algorithemen für alltägliche Szenarien zu demonstrieren, werden die optischen Sensoren von ARMAR für Experimente verwendet. Es wird ein System für einen humanoiden Roboter entwickelt, das online die Bewegungen von alltäglichen Handlungen erkennt. Das Erkennungssystem wird integriert in die Intentionserkennung von ARMAR. Die meisten in dieser Arbeit entwickelten Methoden werden in online Systemen verwendet. Dies zeigt, dass die entwickelten Konzepte einen direkten praktischen Nutzen haben.