Karlsruhe Institute of Technology

Developing a Human Motion Recognition System by Applying Automatic Segmentation and Model Transfer

Diplomarbeit am Cognitive Systems Lab
Prof. Dr.-Ing. Tanja Schultz
Fakultät für Informatik
Karlsruher Institut für Technologie

von

cand. inform.
Sandro Sitto

Betreuer:
Prof. Dr.-Ing. Tanja Schultz
Dipl.-Inform. Dirk Gehrig

Tag der Anmeldung: 01. Oktober 2010
Tag der Abgabe: 31. März 2011

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Karlsruhe, den 31. März 2011

Zusammenfassung

In dieser Diplomarbeit wird der Einsatz von automatischen Segmentierungsverfahren und der Modelltransfertechnik für die Entwicklung eines Erkennungssystems für menschliche Bewegungen untersucht. Komplexe menschliche Bewegungen werden mit Hilfe von Hidden-Markov-Modellen (HMM) modelliert. Die Trainingsdaten für Modelle primitiver Bewegungseinheiten werden durch die Segmentierung von Sequenzen komplexer Bewegungen gewonnen. Da die manuelle Segmentierung mit einem großen Zeit- und Kostenaufwand verbunden ist, werden häufig automatische Segmentierungsverfahren eingesetzt. In dieser Arbeit werden zwei automatische Segmentierungsverfahren untersucht. Das erste Verfahren ist ein überwachtes, HMM-basiertes Segmentierungsverfahren, welches den Einsatz von einer geringen Menge an segmentierten Bewegungen für das Training voraussetzt. Das zweite Verfahren ist ein unüberwachtes Segmentierungsverfahren, welches auf die Hauptkomponentenanlyse (Principal Component Analysis, PCA) basiert und keine Trainingsdaten voraussetzt. Darüber hinaus wird der Einsatz der Modelltransfertechnik untersucht. Diese Technik erlaubt die Übertragung von Modellen eines existierenden Erkennungssystems als Modelle für ähnliche primitive Bewegungseinheiten in einem neuen Erkennungssystem.

Die mit Hilfe der oben genannten Verfahren entwickelten Erkennungssysteme werden durch den Einsatz von mehreren Datensätzen evaluiert. Die Datensätze wurden im Rahmen des Sonderforschungsbereichs 588 - Humanoide Roboter akquiriert. Sie beinhalten Bewegungen, wie sie bei der Arbeit in einer Küche typischerweise vorkommen können.

Eine erste Evaluierung des Erkennungssystems mit manuell segmentierten und transkribierten Daten führte zur Folgerung, dass für die Leistung eines Erkennungssystems die Anzahl von einfach zu gewinnenden Transkriptionen wichtiger ist als die Anzahl von aufwendig zu gewinnenden Segmentierungen. Ein Erkennungssystem initialisiert mit wenig segmentierten Daten kann sogar ein mit mehr Daten initialisiertes System minimal übertreffen, obwohl beide Systeme mit gleicher Anzahl von transkribierten Daten trainiert werden.

Die HMM-basierte Segmentierungsmethode wird mit Hilfe eines Baseline-Systems durchgeführt. Die Leistung des Baseline-Systems wird mit der Leistung des mit Hilfe von automatisch segmentierten Daten entwickelten Systems verglichen. Die Experimente zeigen, dass eine absolute Leistungssteigerung von bis zu 9,4% Fehlerrate mit einem von zwei verwendeten Datensätzen festzustellen ist.

Die Evaluierung der PCA-basierten Segmentierungsmethode führt zu guten Ergebnissen. Dabei ist eine Fehlerrate von bis zu 25,52% erreicht. Die Verteilung der automatisch gewonnenen Bewegungseinheiten unter den komplexen Bewegungen wird verwendet um eine Klassifizierung von komplexen Bewegungen zu ermöglichen. Damit wird eine Fehlklassifikationsrate von bis zu 9,6% erreicht.

Die Evaluierung der Modelltransfertechnik führt zur Folgerung, dass eine anschließende Adaption mit Transkriptionsdaten notwendig ist um eine relativ niedrige Fehlerrate zu erreichen.

Abstract

In this thesis the development of a human motion recognition system using automatic segmentation and model transfer is investigated. Complex human motions are modeled using Hidden Markov Models (HMMs) for primitive motion units. The training data for the motion unit models is provided by segmenting complex motion sequences into primitive motion units. However, manual segmentation is too time and cost consuming. Hence, two automatic segmentation methods are presented. The first one is an HMM-based supervised automatic segmentation method which requires a certain amount of pre-segmented training data. The second one which is an unsupervised automatic segmentation method based on Principal Component Analysis (PCA) does not require any training data. Furthermore, the application of the model transfer technique is investigated in which motion models of an existing recognition system are transferred to a target recognition system in order to be used for recognizing similar motion units.

The recognition systems developed by using the above mentioned approaches are evaluated by using several motion data sets acquired by the Collaborative Research Center 588 - Humanoid Robots. The data sets consist of motions as they appear in kitchen tasks and food preparation scenarios.

An initial evaluation of the recognition system using manually segmented and transcribed data show that the number of less expensive transcriptions for the performance of the recognition system is more important than the number of expensive segmentations. A recognition system initialized using less segmented data might outperform a system initialized using more data although both systems are trained by using the same amount of transcribed data.

The HMM-based segmentation method is carried out by using a baseline system. The performance of the baseline system is compared to the performance of the recognition system developed by using automatically segmented data for training. With one of two data sets used for the evaluation, an absolute performance improvement up to 9.4% error rate is achieved.

The PCA-based segmentation method is successfully applied. A performance up to 25.52% error rate is achieved. The distribution of automatically obtained motion units among complex motions is applied in order to enable a classification of complex motions. A classification error rate up to 9.6% is achieved.

Further experiments show that the model transfer technique only performs well when an additional adaptation using transcription data is applied.

Contents

List of Figures

List of Tables

1. Introduction

Human motion recognition (HMR) has become an important research area in the field of robotics and human–machine interaction. This is due to the fact that HMR is required in many applications, especially in the field of robotics. Human motion recognition is not only essential for recognizing the intention of humans, but also essential for learning human-like behavior. Programming by Demonstration [Dil04] is an important example where human motion recognition is applied.

The challenges behind human motion recognition are enormous; starting with the capturing of motion data and its preprocessing, through the segmentation of complex motion sequences into primitive motion units, and all the way up to the modeling of motion units, the training of motion unit models, and the recognition of modeled motion units in observation data. For all these tasks a lot of effort has to be spent in order to achieve a reliable recognition.

The most important issue in HMR is to find an appropriate modeling technique for modeling motion trajectories. Human motions are often represented as time series data in which a certain time and shape variance in the trajectory is always comprised. This is due to the fact that every execution of a human motion is unique even if it is of the same motion type and it is performed by the same person. Because of the variance in motion data statistical modeling is attractive in order to be applied for modeling human motions. The most well-known and most applied statistical models in the field of HMR are Hidden Markov Models (HMMs). In fact HMMs are successfully applied in many works concerned with modeling human motions. Their statistical characteristics allow them to deal with time and shape variance and they are also suitable for modeling time series data like human motions.

Additional to the consideration about the modeling technique used in the recognition system the abstraction level of representation at which the motions are modeled has to be considered very well. There are different levels of abstraction that can be applied for modeling human motions. Motion models at a high level of abstraction are more complex than motion models at a lower abstraction level and they also have a higher information content. Models at a high abstraction level usually represent complex motions like ”pouring water” whereas models at a low abstraction level

represent primitive motion units within complex motions, e.g. ”take glass” or ”take bottle”.

Modeling at a higher abstraction level requires an enormous amount of training data. However, the collection of human motion data is associated with a great effort. Therefore, the modeling of primitive motions at a lower abstraction level becomes more attractive since a smaller amount of data is needed to train robust models. Complex motions can then be modeled by concatenating models of primitive motions. An additional advantage of this modeling technique is the flexibility in composing and recognizing new complex motions using a relatively small number of existing elementary primitive motion models.

In order to apply motion modeling at a low abstraction level motion data segmentation becomes an essential task in which motion data sequences at a higher abstraction level have to be segmented into primitive motion units. However, this task is also associated with an enormous time and cost effort since it is generally carried out manually.

The effort evolved by the manual segmentation task can be reduced or it can be completely eliminated by applying different approaches. A common approach is to apply automatic segmentation methods instead of manual segmentation. However, developing automatic segmentation methods is very challenging. In particular, unsupervised automatic segmentation algorithms which provide a similar segmentation into meaningful motion units as provided by manual segmentation are hard to design. They are, however, still desirable because they do not require any manually segmented training data. On the contrary supervised model-based automatic segmentation approaches require a certain amount of manually segmented training data, but they are able to provide a similar segmentation as provided by the manual segmentation.

Another approach that can be applied to overcome the manual segmentation effort is known as Cross-Language Transfer (CLT) in the field of Automatic Speech Recognition (ASR). CLT is based on the idea of using existing phoneme models of one or more source languages to built a recognition system for a new target language. For this, expert knowledge can be applied to assign phoneme models in the source recognition systems to similar phonemes in the target language. In this work the CLT is referred to as the model transfer technique.

1.1 Objectives

The main objective of this work is to develop a robust human motion recognition system, and at the same time reduce or rather overcome the great effort associated with the manual segmentation task. For this, three approaches are investigated and evaluated by using three motion data sets. The first two approaches are automatic segmentation methods. The third one is the model transfer technique. However, the first experiments are carried out to investigate the performance of the recognition system using different amounts of segmented data for initialization and different amounts of transcribed data for training.

The first automatic segmentation method is a supervised model-based segmentation method. In this method the segmentation is carried out by using an HMM-based

recognition system as a baseline system. The hypotheses provided by the baseline system are considered as segmentation. Furthermore, the application of confidence measures is investigated in order to enable a confidence-based segmentation filtering. This can be applied to discard bad segments in the hypotheses. This segmentation method requires a certain amount of manually segmented training data.

The second automatic segmentation method is unsupervised. It is based on the Principal Component Analysis (PCA). This segmentation method does not require any manually segmented data which extremely reduces the development effort for the recognition system. The segmentation method consists of two main steps. In the first one a motion data sequence is segmented based on the first principal component of the motion sequence. In the second step the calculated segments are labeled by using the k-means clustering algorithm.

Furthermore, the model transfer technique is investigated by utilizing motion unit models of an existing recognition system to recognize similar motion units in a new target motion data set. This way the target motion data set does not have to be segmented manually.

1.2 Structure of this Work

This work consists of seven chapters. In the first one an introduction of this thesis and its objectives is given. In chapter 2 an overview of existing works related to the topic of this thesis is given. Chapter 3 describes how the human motion data for this work is acquired, represented, and what kind of information it comprises. Also in this chapter the data sets used in this work and their manual segmentations are presented. In chapter 4 the HMM-based HMR approach is described, starting with an overview about the theory of HMMs. Furthermore, a human motion recognition system to be applied as a baseline system is introduced. This is followed by experiments for the evaluation of different baseline and recognition systems using different data sets. Chapter 5 introduces and explains the automatic segmentation methods developed and investigated in this work. Furthermore, the experiments carried out to evaluate the segmentation methods are presented. In chapter 6 the model transfer technique is explained and the experiments conducted for its evaluation are presented. In chapter 7 a summary and a conclusion about this work are given. Furthermore, some aspects of future work are discussed.

2. Related Work

HMM-based Modeling and Recognition of Human Motions

For years, many scientists have investigated and discussed different approaches and methods for modeling and recognizing human motions. The most widely used modeling techniques are based on HMMs. This is due to the fact that HMMs are convenient for modeling time series data. Furthermore, their statistical characteristics enable them to deal with time and shape variance that occurs in motion data.

The HMM-based motion modeling approaches in the literature differ in the type of HMMs applied for modeling, the type of motion data representation, and the abstraction level of representation of motion data [MHK06] [KKUG07].

A simple approach based on discrete HMMs for modeling and recognizing a small number of primitive motions represented in 2D image sequences is presented in [YOI92]. The HMMs are applied for modeling and recognizing six primitive tennis motions. The feature vectors are extracted from the pixel values of the images and vector quantization is applied in order to partition the feature space. The quantized feature vectors which represent the different feature space regions are then applied as output symbols for the discrete HMMs. This approach, however, is probably limited when it comes to modeling and recognizing a larger number of primitive motions. Problems could be the low resolution of the extracted feature vectors, the quantization of the feature space, and the lack of a body-model-based feature extraction. In addition, the application of 2D motion images without a body-model based feature extraction makes the recognition of primitive motions from different viewing angles difficult.

Another approach which allows modeling and recognizing a larger number of primitive motions is introduced by Mori et al [MSSS04]. In this approach the recognition system is based on so-called action models each consisting of a continuous HMM and a feature extraction filter which focuses attention on the typical motion features of the corresponding motion. The feature vectors consist of 36 3D spatial information of different positions in the whole human body. The recognition system has a

hierarchical structure that allows the recognition of primitive motions in different levels of abstraction. The motions at the top level of abstraction are generalizations of special motions at the lower levels of abstraction. The recognition is processed from the top to the lower levels where more details about the primitive motions are comprised. The authors also adopt a concept based on confidence measure to recognize novel motions. Primitive motions with low confidence values are classified as new motions.

The two approaches mentioned above are only applied to model and recognize primitive motions. However, in real-life situations the recognition of continuous complex motions consisting of several primitive motions is required. In [VM99] an approach for the continuous recognition of American Sign Language (ASL) is presented. The approach is based on modeling signs by using concatenations of sign phoneme models in order to allow a continuous recognition of the ASL. The phonemes are modeled by using parallel HMMs. This is due to the fact that using two hands for performing sign language leads to simultaneous occurrences of phonemes. The authors apply 3D motion data to model 89 phonemes for the right hand and 51 phonemes for the left hand. The phonemes are used to model 22 signs.

Segmentation of Human Motion Data

Modeling techniques such like the one presented in [VM99] require the segmentation of complex motions into primitive motions. Since manual segmentation is associated with a great effort, automatic segmentation methods become very interesting. Automatic segmentation methods can be divided into supervised and unsupervised segmentation methods.

Supervised model-based automatic segmentation methods in HMR are hardly known. However, in the field of ASR and Automatic Speech Synthesis many supervised model-based algorithms for automatic segmentation and automatic transcription of speech data have been investigated.

An HMM-based method used for segmenting speech data for the purpose of concatenative speech synthesis is presented in [KC02]. The method is applied to compute phoneme boundaries in speech data by using an HMM-based recognition system in the so-called forced alignment mode. Kim and Conkie [KC02] use speaker independent HMMs to prepare seed phone labels which are then used to train speaker dependent HMMs. The speaker dependent HMMs are used to provide the segmentation task. The HMMs are improved iteratively by using the segmentation of previous iterations. This HMM-based segmentation method is proven to be consistent and accurate, especially when certain post-processing methods are applied to improve the accuracy of the segmentation results. In [KC02] the segmentation system is combined with a so-called spectral boundary correction to improve the segmentation results. For this, spectral features are used to detect spectral mismatches between two successive concatenated units.

The segmentation method in [KC02], however, requires the transcription of speech data to allow the correct concatenation of phoneme models for the given speech data in order to enable the decoding in the forced alignment mode. An automatic segmentation and transcription method based on HMMs for speech data is presented in [KW98] and [ZC98]. In both works the segmentation method is based on the idea

of creating initial HMMs by using a certain amount of transcribed data and of using the initial HMMs to decode the available untranscribed data. In a subsequent step, the hypotheses provided by the decoding process are analyzed for correctness by using a confidence measure. The most confident hypotheses are accepted as transcriptions. In [KW98] Kemp and Waibel apply a confidence measure based on the a-posteriori gamma probability for words. Zavaliagkos et al. [ZC98] use a generalized linear model (GLM) to estimate confidence values.

The automatic segmentation methods mentioned above require a certain amount of pre-segmented and pre-labeled data. Unsupervised automatic segmentation methods manage the segmentation task without the need of pre-segmented and pre-labeled data. Unsupervised automatic segmentation methods for human motion data are discussed in many works. The common approach is to derive a segmentation feature from the multi-dimensional motion data and set segment boundaries in the segmentation feature trajectory where local minima and/or maxima, or zero-crossings are found.

Fod et al. [FMJ02] present an approach based on segmenting motion data consisting of angular velocities. The velocity sequences of each joint are segmented separately. For this, the zero-crossing method is applied to determine the segment boundaries. A total segment boundary over all joints is determined by finding a region where multiple zero-crossings emerge at the same time or nearly the same time. This approach, however, poses some problems. Multiple zero-crossings of some joints can emerge within one segment which leads to segment overlapping. Furthermore, some primitive motions characterized by a small number of active joints and multiple simultaneous zero-crossings within the motion are wrongly segmented. Hence, Fod et al. introduce another algorithm which they call the z-algorithm in order to improve the segmentation task. The algorithm is based on the sum z of the squares of angular velocity and it sets segment boundaries whenever z is lower than an empirically derived threshold.

Wang et al. [WLZ05] consider in their approach expert knowledge from the field of biomechanics. They consider the fact that human motions do not take place at a constant speed and that some effects, e.g. the spring effect of muscles, allow a biological system to minimize energy. Therefore, they decide to consider the torque as segmentation feature. In order to solve the multi-dimensional problem they calculate the torque of the complete body by simply summing the torque of all joints. The segment boundaries are set at local minima of the overall torque.

In their work Jenkins et al. [JM04] present a segmentation method called kinematic centroid segmentation (KCS). In this method a kinematic substructure of the human body, e.g. the arm, is considered as a pendulum. This can be seen as a spatial segmentation of the human body where interrelated joints or degrees of freedom (DOF) are grouped as a pendulum. The segmentation task is based on the centroid of the 3D positions of all involved elements with respect to a so-called base feature, e.g. the shoulder joint. The segmentation trajectory is derived by calculating the Euclidean distance between the centroid value in the first frame and the centroid values in the following frames. A segment boundary is set at frame t where the Euclidean distance is maximum. The same procedure is repeated in order to segment the rest of the motion sequence. For this, frame $t + 1$ is set as the first frame.

Barbič et al. [BSP+04] present three segmentation approaches which are quite different from the above mentioned ones. The first two approaches are based on the principal component analysis (PCA). The authors exploit the data reduction properties of the PCA and show that distinct motion behaviors occur in distinct subspaces of the overall feature space.

In the first approach the PCA is applied to calculate the subspace in which the first motion behavior in the complex motion sequence is best represented. This is done by using a certain number of frames that are assumed to belong to the first motion behavior. In the next step adjacent frames are iteratively projected into the calculated subspace. The authors assume that if the additional frames belong to the same behavior then the projection error stays minimal, and if the additional frames belong to another behavior the projection error increases rapidly. This fact is exploited to set a segmentation boundary where the projection error increases rapidly. The entire process is repeated on the remaining motion sequence.

The second approach is based on a probabilistic extension of the PCA, namely the probabilistic principal component analysis (PPCA). The PPCA is used to model a distinct motion behavior with a Gaussian distribution. The number of frames used for the estimation of the Gaussian distribution is increased iteratively. In each iteration the Mahalanobis distance of adjacent frames to the estimated Gaussian distribution is calculated. The algorithm sets a segment boundary where the Mahalanobis distance becomes radically high.

Under the assumption that the frames of a simple motion form a cluster and that such a cluster can be modeled with a Gaussian distribution, Barbič et al. apply a Gaussian Mixture Model (GMM) to model the entire motion sequence. This way different simple motions are assigned to different Gaussian distributions of the GMM. The boundaries between the Gaussian distributions are then chosen as segmentation boundaries.

The Model Transfer Technique

Besides automatic segmentation techniques, there are other methods that can be applied to overcome the expensive intervention of humans in preparing training data. As mentioned in the introduction, CLT is one of these methods which is based on the idea of using phoneme models of an existing recognition system for a certain language to recognize phonemes of another target language without using training data of the target language.

In [SW01] Schultz and Waibel investigate the application of the CLT technique for recognizing Swedish. For this purpose the application of language dependent and independent acoustic models is applied. In the case of language dependent acoustic models, seven recognition systems of different languages are utilized. In the case of language independent models, acoustic models for the International Phonetic Alphabet (IPA) are used. These models are trained by using speech data of different languages. The phoneme mapping from the target language to the source languages is carried out by applying expert knowledge. The authors also investigate a data-driven mapping approach by using transcriptions of a certain amount of the target language training data. The resulting transcriptions for the target language is used in a following step to bootstrap a new recognition system.

In [VKS10] Vu et al. extend the CLT technique by a subsequent unsupervised training. Their goal is to build a recognition system for Czech without using transcriptions of the speech data. In the first step they apply the CLT technique to build the recognition system by utilizing existing acoustic models for Russian, Bulgarian, Polish, and Croatian. The resulting system is used to decode the un-transcribed Czech speech data. The hypotheses are analyzed for correctness by applying a so-called multilingual a-stabil confidence measure. The confidence measure is based on alternative hypotheses of all involved source languages. According to the confidence measure, the best hypotheses are extracted to be used as training data to build a new recognizer for the target language. The recognizer is improved iteratively by repeating the decoding and the training tasks.

3. Human Motion Data

3.1 Acquisition and Representation of Human Motion Data

Human motions are generally represented in form of time series data which contain a certain kind of motion information at certain time instances. Each time instance in time series data is called frame which is comparable to a frame in animations, i.e. the time series data is a set of consecutive frames, each containing information about a human motion at a certain time instance. The time line of human motion data is then given by the number of frames.

The sampling rate, i.e. the number of frames per second (fps) and the kind of information comprised in a frame is dependent on the task the motion data is used for. Film projectors, for instance, normally work at a frame rate of 24 fps. In some technical applications, however, a higher frame rate is necessary.

The information comprised in a frame differs from application to application, but the most common kinds of information stored in a frame are joint angles or 3D positions of a human body. 3D marker positions, for instance, are usually derived by using optical motion capture systems which are based on infrared cameras that are able to capture the infrared light reflected by markers. Theoretically, only two cameras are required in order to capture the 3D positions of markers. In practice, however, a complex capturing system with a static capturing setup and a larger number of cameras is needed to acquire high-quality motion data.

Joint angle information is often reconstructed out of the 3D marker position cloud by using a virtual kinematic body model of the subject's body. The 3D marker positions are mapped to corresponding positions in the kinematic body model by minimizing the distances between the marker positions in space and the positions in the body model [Ste99]. Once the marker positions are mapped to the kinematic body model, angle values of different joints in the body model can be reconstructed.

The dimension of motion data is dependent on the number of markers attached to the human body or the DOF of the joints considered in the kinematic body model.

Every frame contains either one angle value for each DOF or three values for the XYZ-positions of each marker.

The motion data applied in this work are human motions as they appear in kitchen tasks and food preparation scenarios, e.g. ”pouring water” and ”grating apple”. They include motions concerned with taking kitchen objects or food from a counter top, working with them and putting them back to their places. The motion data sets are acquired by the Collaborative Research Center 588 - Humanoid Robots.

The motions are captured and recorded with a Vicon motion capture system. For this purpose, 10 Vicon cameras are used to capture the infrared light reflected by 35 markers attached to the subject's upper body, head and arms. The Vicon system outputs 3D positions and labels of the markers at a frame rate of 100 fps.

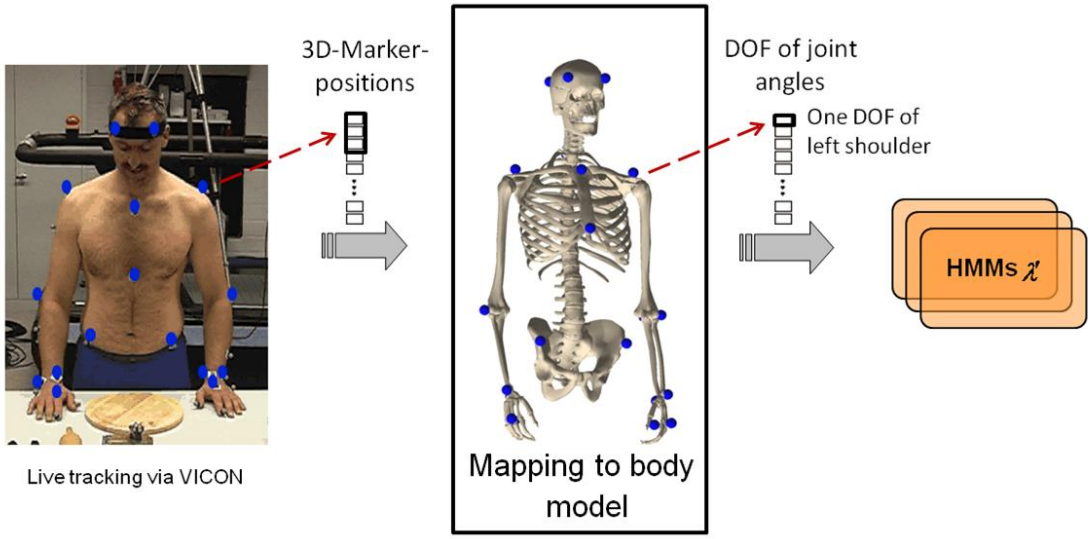

Figure 3.1: The motion data processing chain. Captured 3D marker positions are mapped to a body model. This allows the reconstruction of joint angle information.

The marker-based motion information captured by the Vicon system is utilized in a following step to reconstruct the joint angle information. For this purpose, an optimization-based motion mapping is applied in which marker position trajectories are used to determine the parameters of a kinematic body model in order to enable the reconstruction of the corresponding joint angle trajectories. The motion mapping is achieved by minimizing the distances between the marker positions in space and the corresponding positions in the kinematic body model. The mapping task outputs angle information of 24 DOF, i.e. the dimensionality of the motion data is 24. For the reconstruction of the joint angle information a rigid multi-body model of the human skeleton was applied that consists of the following 24 DOF (global positions and orientations are excluded):

- Arm Joints
  - Wrist Joint ($2 \cdot 2$ DOF)
  - Elbow Joint ($2 \cdot 2$ DOF)
  - Shoulder Joint ($2 \cdot 3$ DOF)

- Lower Neck Joint (2 DOF)
- Upper Neck Joint (3 DOF)
- Lower Spine Joint (2 DOF)
- Upper Spine Joint (3 DOF)

In figure 3.1 the motion data processing chain applied in order to carry out the joint angle reconstruction is shown. The frame rate of the resulting joint angle data is lowered to 20 fps.

3.2 Motion Data Sets

The data used in this work is organized in three different data sets. The data sets differ from each other in respect to the way the motions are performed, the way the kitchen objects are positioned, and the subjects who perform the motions. In the following sections the data sets and their characteristics are explained in more detail.

Data Set A

Data set A consists of 10 complex motion types performed by a male subject. Each complex motion type is recorded 50 times. The total number of motion sequences is 500. The average length of a motion sequence in this data set is 302 frames. This corresponds to a duration of about 15 seconds.

Data set A is characterized by motions that contain artificially added short pauses between certain primitive motions. This is done in order to emphasize the transition between two primitive motions within a complex motion. In the complex motion "pouring water", for instance, a short pause between the primitive motions "take glass" and "take bottle" exists. Another characteristic is the sequential order of primitive motions. Primitive motions are strictly performed one after another, i.e. a started primitive motion is completely performed before another primitive motion is started. The third characteristic is the static positioning of kitchen objects. Each object is positioned at the same place before and after the performance of the motion (see figure 3.2). Data set A consists of the following complex motion types:

- Rolling pastry
- Pouring water
- Slicing apple
- Grinding coffee
- Sweeping
- Grating apple
- Stirring
- Cutting bread
- Cutting apple
- Mashing potatoes

Data Set B

Data set B consists of 5 complex motion types. Each complex motion type is recorded 20 times. The total number of motion sequences is 100. The average length of a motion sequence in this data set is 300 frames which corresponds to a duration of 15 seconds.

As in data set A, the motions in data set B are characterized by the sequential order of primitive motions and the static positioning of kitchen objects as shown is figure 3.3. The difference to data set A is that the motions in data set B are performed fluently without artificial pauses between primitive motions. Data set B consists of the following complex motions which are performed by a female subject:

- Pouring water
- Grating apple
- Stirring
- Cutting apple
- Mashing potatoes

Data Set C

The motions in data set C are performed by the same female subject that performed the motions in data set B. The motions are performed fluently, but the order of primitive motions is in some cases semi-parallel, i.e. some primitive motions that require only the action of one hand or arm are started before another primitive motion performed by the other hand or arm is finished. Complex motions with this characteristic are more realistic and they occur more often in real-life situations. Another characteristic of data set C is the dynamic positioning of kitchen objects. The positions of kitchen objects vary between different recording sessions. Altogether, five different positions on the counter top are allowed. The different positions are shown in figure 3.4. Data set C consists of the following complex motions:

- Pouring water
- Grating apple
- Stirring
- Mashing potatoes

Each of the above mentioned 4 complex motions is performed by using 5 different positioning variations of kitchen objects. Each variation is recorded 5 times. The total number of motion sequences is 100. The average length of a motion sequence in this data set is 207 frames. This corresponds to a duration of about 10 seconds.

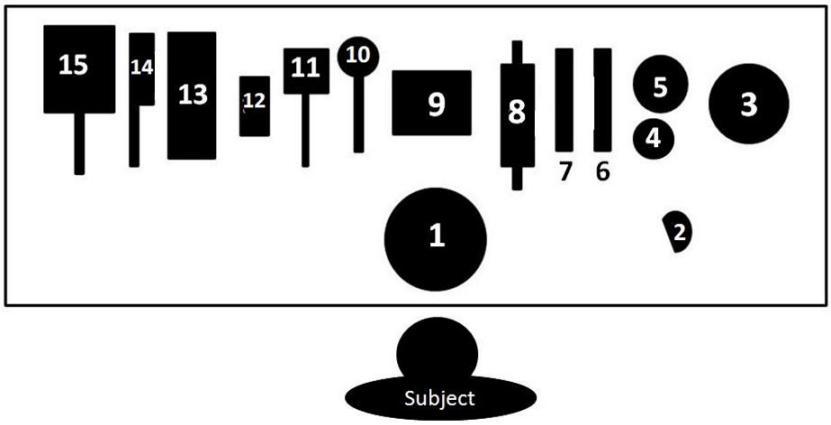

Figure 3.2: The counter top for the performance of motions in data set A: 1) Working spot, 2) Apple, 3) Bowl, 4) Glass, 5) Bottle, 6) Bread knife, 7) Knife, 8) Rolling pin, 9) Grinder, 10) Spoon, 11) Masher 12) Grater, 13) Slicer, 14) Broom, 15) Dustpan.

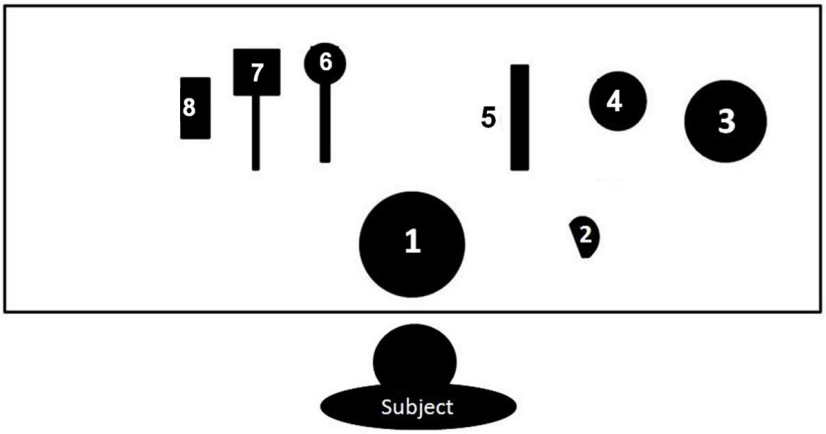

Figure 3.3: The counter top for the performance of motions in data set B: 1) Working spot, 2) Apple, 3) Bowl, 4) Bottle, 5) Knife, 6) Spoon, 7) Masher, 8) Grater.

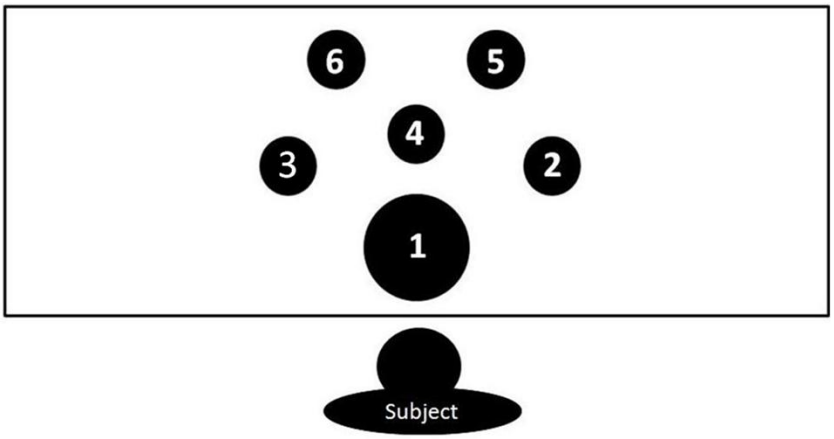

Figure 3.4: The counter top for the performance of motions in data set C: 1) Working spot, 2) Front right, 3) Front left, 4) Center, 5) Back right, 6) Back left.

3.3 Manual Segmentation of Motion Data

In the last section the motion data sets applied in this work were presented. The motion data, however, is provided as complex motion sequences. The motion sequences have to be segmented into motion units in order to allow motion modeling at a lower abstraction level. In this section the manual segmentation of the complex motion sequences into motion units is introduced.

In order to obtain a good segmentation specific requirements have to be followed. An important requirement for the segmentation task is consistency. This can be complied by defining strict rules based on expert knowledge which have to be followed by all operators.

Other important requirements relate to the information content of motion units, and their size which affects their flexibility in composing complex motions. Motion units have to comprise unique and meaningful information and at the same time they have to be small enough in order to be flexibly utilized in composing complex motions. This leads to a problem since a very small motion unit may not contain unique and meaningful information which makes it unusable for modeling. On the other hand a motion unit with too much information content is too special and too long which makes it difficult to be built into different kinds of complex motions. Therefore, a balance between the two requirements has to be found.

The manual segmentation applied in this work is carried out as presented in the next sections. As mentioned in section 3.2, the complex motions in data set A contain artificial pauses between certain primitive motion units. This pauses are referred to as ”rest position”. Cyclic motion units, e.g. ”cut” appear only once in the following description of the segmentations. These motion units, however, may appear several times in the segmentation of the real motion sequences. This is dependent on the duration of the cyclic motion. Since the semi-parallel motion units in data set C are not completely overlapped, the segmentation of such motion units is carried out sequentially as done with data set A and B.

Manual Segmentation of Data Set A

- Rolling pastry: *Rest position - Take rolling pin - Rest position - Grasp rolling pin with both hands - Roll - Release rolling pin - Rest position - Put rolling pin back - Rest position*
- Puring water: *Rest position - Take glass - Rest position - Take bottle - Pour - Put bottle back - Rest position - Put glass back - Rest position*
- Slicing apple: *Rest position - Take Slicer - Take apple right hand - Slice - Put apple back right hand - Put Slicer back - Rest position*
- Grinding coffee: *Rest position - Take grinder - Grasp grinder - Grind - Release grinder - Put grinder back - Rest position*
- Sweeping: *Rest position - Take broom and dustpan - Sweep - Put back broom and dustpan - Rest position*
- Grating apple: *Rest position - Take grater - Take apple right hand - Grate - Put apple back right hand - 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 bread: *Rest position - Take bread - Rest position - Take knife - Grasp bread - Cut bread - Release bread - Put knife back - Rest position - Put bread back - Rest position*
- Cutting apple: *Rest position - Take apple left hand - Rest position - Take knife - Grasp apple - Rest position - Cut apple - Release apple - Put knife back - Rest position - Put apple back left hand - Rest position*
- Mashing potatoes: *Rest position - Take bowl - Rest position - Take masher - Grasp bowl - Mash - Release bowl - Put masher back - Rest position - Put bowl back - Rest position*

Manual Segmentation of Data Set B

- Pouring water: *Rest position - Take bowl - Take bottle - Pour - Put bottle back - Put bowl back - Rest position*
- Grating apple: *Rest position - Take grater - Take apple right hand - Grate - Put apple back right hand - Put grater back - Rest position*
- Stirring: *Rest position - Take bowl - Take spoon - Stir - Put spoon back - Put bowl back - Rest position*
- Cutting apple: *Rest position - Take apple left hand - Take knife - Grasp apple - Cut apple - Release apple - Put knife back - Put apple back left hand - Rest position*
- Mashing potatoes: *Rest position - Take bowl - Take masher - Mash - Put masher back - Put bowl back - Rest position*

Manual Segmentation of Data Set C

- Pouring water: *Rest position - Take bowl from position X - Take bottle from position Y - Pour - Put bottle back on position Y - Put bowl back on position X - Rest position*
- Grating apple: *Rest position - Take grater from position X - Take apple from position Y - Grate - Put apple back on position Y - Put grater back on position X - Rest position*
- Stirring: *Rest position - Take bowl from position X - Take spoon from position Y - Stir - Put spoon back on position Y - Put bowl back on position X - Rest position*
- Mashing potatoes: *Rest position - Take bowl from position X - Take masher from position Y - Mash - Put masher back on position Y - Put bowl back on position X - Rest position*

As mentioned in section 3.2 each motion type is performed by applying 5 different positioning variations of the kitchen objects. X and Y accept one of the following values, where X is not equal Y:

- Front right
- Front left
- Center
- Back right
- Back left

The object positions can be seen in figure 3.4.

4. HMM-based Human Motion Recognition

HMR is a very complex task that requires the consideration of many problems and issues. The modeling method used to model human motions is an important and essential issue that has to be considered very well. Human motions are characterized by their uniqueness. Even professional athletes would not be able to repeat the same motion exactly, i.e. there is a certain time and shape variance in human motions. This characteristic leads to the preference of applying statistical methods for modeling human motions. In this way the statistical properties enable the recognition system to deal with time and shape variance. An important example of statistical models are HMMs.

HMMs are widely used in the field of machine learning particularly in the field of HMR and also in ASR. They are convenient for modeling time series data and their statistical characteristics make them powerful for modeling stochastic processes enabling them to deal with time and shape variance in signal data. These are exactly the characteristics needed to model human motion signals since they are also represented as time series and they vary due to their unique execution property. The effectivity in applying HMMs for HMR is shown in many works, e.g. [GKWS09].

In the next section a brief introduction about the theory of HMMs is given.

4.1 Hidden Markov Models

In his well known tutorial [Rab89] about HMMs, Rabiner distinguishes between two categories of models which can be used for modeling signals. The first one is the category of deterministic models that consider specific information about the signal properties, e.g. the signal might have the form of a sine wave. In this case the specification of the signal model is previously known and only the parameters of the signal model like the amplitude and frequency have to be determined. The second category of signal models is the category of statistical models. Here, it can be assumed that the signal can be characterized as a random process, e.g. a Gaussian

process. After identifying the corresponding statistical model to the given signal the parameters of the model have to be determined or rather estimated.

Hidden Markov Models belong to the category of statistical models. They can be described by two random processes. The first one is referred to as a Markov chain that consists of a set of states. These are connected to each other according to a specific topology and specific state transition probabilities. The states of an HMM are hidden, i.e. they are not observable. Instead, a defined set of symbols can be observed. This is governed by the second random process in which each of the hidden states can emit a symbol out of the symbol set according to a state dependent emission probability.

Formally an HMM is defined as a five-tuple $\lambda = (S, A, V, B, \pi)$ consisting of:

- $S = \{s_1, s_2, \ldots, s_n\}$ is the set of states, where n is the number of states.
- $A = (a_{ij})$ is the state transition probability matrix, where $a_{ij} = P(q_{t+1} = s_j | q_t = s_i)$ is the probability of transition from state s_i at time t to state s_j at time $t+1$ and $1 \leq i, j \leq n$. q_i denotes the system state at time i.
- $V = \{v_1, v_2, \ldots, v_n\}$ is the set of observable symbols (feature vectors) of a **discrete** HMM, where n is the number of distinct symbols, **or** $V = \mathbf{R}^d$ is the observable symbol space (feature space) of a **continuous** HMM.
- $B = \{b_1, b_2, \ldots, b_n\}$ is the set of the state dependent emission probability distributions of a **discrete** HMM **or** the set of the state dependent emission probability densities of a **continuous** HMM, where $b_i(v) = P(o_t = v | q_t = s_i)$ is the probability of observing $v \in V$ from state s_i. o_i denotes the observation at time i.
- π is the initial probability distribution, where $\pi(s_i) = P(q_1 = s_i)$ is the probability of state s_i being the first state q_1 that emits the first symbol of an observation sequence.

Given appropriate values of the above mentioned parameters an HMM can then be applied to generate an observation sequence $O = O_1 O_2 \ldots O_T$, where $O_t \in V$, T is the number of observations in the sequence. O is generated by traversing the sequence of states $Q = q_1 q_2 \ldots q_T$ where $q_i \in S$.

For a given observation sequence $O = O_1 O_2 \ldots O_T$, an HMM can be built by finding appropriate parameters to model the observation O.

As mentioned above, the set of emission symbols V can be a finite set. In this case the HMM is referred to as a discrete HMM, where b_i are discrete probability distributions. This is in contrast to a continuous HMM, where V is $\mathbf{R}^d$ and b_i are continuous probability densities. In many cases a weighted sum of Gaussian densities is used as emission probability densities of a continuous HMM.

In order to apply HMMs, three fundamental problems have to be solved. These problems and their solutions are introduced in the following as stated in [Rab89]:

The Evaluation Problem

Given an observation sequence $O = O_1 O_2 \dots O_T$ and an HMM λ - how to compute $P(O|\lambda)$ efficiently, namely the probability of the observation O given the HMM λ? The evaluation problem can also be formulated as how likely a given HMM λ is able to generate the given observation O. Solving the evaluation problem is important for finding one of many HMMs which best matches the observation sequence.

A theoretical approach to solve the evaluation problem is given by summing the probabilities of all possible state sequences which are able to generate the observation sequence O. One possible state sequence to generate the observation O is given by

$$Q = q_1 q_2 \dots q_T \tag{4.1}$$

The probability of the observation sequence O given the state sequence Q is

$$P(O, Q|\lambda) = P(O|Q, \lambda) P(Q|\lambda) \tag{4.2}$$

Summing the probabilities of all possible state sequences which can generate the observation O leads to the following:

$$P(O|\lambda) = \sum_{all\ Q} P(O|Q, \lambda) P(Q|\lambda) \tag{4.3a}$$

$$= \sum_{q_1, q_2, \dots, q_T} \pi_{q_1} b_{q_1}(O_1)\ a_{q_1 q_2} b_{q_2}(O_2)\ \dots\ a_{q_{T-1} q_T} b_{q_T}(O_T) \tag{4.3b}$$

For an HMM with N states and an observation sequence of length T the computational complexity of (4.3b) is $O(TN^T)$ which means that the computational effort grows exponentially to the observation sequence length T. This makes the computation of (4.3b) infeasible for large T. Thus, another procedure called the Forward-Algorithm is applied to solve the evaluation problem.

The Forward-Algorithm is based on the forward variable which is defined as follows:

$$\alpha_t(i) = P(O_1 O_2 \dots O_t, q_t = s_i|\lambda). \tag{4.4}$$

The forward variable $\alpha_t(i)$ is the probability of the partial observation sequence $O_1 O_2 \dots O_t$ and the state at time t being s_i. The forward variable is then used to calculate $P(O|\lambda)$ recursively as follows:

- Initialization:

$$\alpha_1(i) = \pi_i b_i(O_1), \qquad 1 \le i \le n. \tag{4.5}$$

- Induction:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{n} \alpha_t(i) a_{ij} \right] b_j(O_{t+1}), \qquad \begin{array}{l} 1 \le t \le T-1 \\ 1 \le j \le n. \end{array} \tag{4.6}$$

- Termination:

$$P(O|\lambda) = \sum_{i=1}^{n} \alpha_T(i). \tag{4.7}$$

The computational complexity of (4.7) is $O(TN^2)$. This allows the Forward-Algorithm to become a practicable solution for the evaluation problem.

The Decoding Problem

Given an observation sequence $O = O_1 O_2 \dots O_T$ and an HMM λ - how to compute $argmax_{q_1 q_2 \dots q_T} P(q_1 q_2 \dots q_T | O, \lambda)$ efficiently, namely the most likely state sequence $Q = q_1 q_2 \dots q_T$ that is traversed for generating the observation O? By solving this problem a certain insight into the hidden state structure of HMMs can be gained. In this way, optimal state sequences for certain observations can be found.

To solve this problem the following probability has to be maximized:

$$P(Q|O, \lambda) \tag{4.8}$$

which is the same as maximizing

$$P(Q, O|\lambda) \tag{4.9}$$

An algorithm based on dynamic programming called Viterbi-Algorithm is used to formally solve the decoding problem by finding the most likely single state sequence $Q = q_1 q_2 \dots q_T$ to match the observation $O = O_1 O_2 \dots O_T$. Therefore, the quantity

$$\delta_t(i) = \max_{q_1, q_2, \dots, q_{t-1}} P(q_1 q_2 \dots q_t = s_i, O_1, O_2 \dots O_t | \lambda) \tag{4.10}$$

has first to be defined which is the highest probability along a single path at time t and being in state s_i. The quantity $\delta_t(i)$ can be used to formulate a recursive procedure that is similar to the one used in the Forward-Algorithm. The procedure is defined as follows:

- Initialization:

$$\delta_1(i) = \pi_i b_i(O_1), \qquad 1 \leq i \leq n. \tag{4.11}$$

- Induction:

$$\delta_{t+1}(j) = \left[\max_{1 \leq i \leq n} \delta_t(i) a_{ij} \right] b_j(O_{t+1}), \qquad \begin{array}{l} 1 \leq t \leq T-1 \\ 1 \leq j \leq n. \end{array} \tag{4.12}$$

- Termination:

$$P^* = \max_{1 \leq i \leq n} (\delta_T(i)), \tag{4.13}$$

where P^* is the probability of the best single state sequence given by

$$q_t^* = \arg \max_{1 \leq i \leq n} (\delta_T(i)), \tag{4.14}$$

which is determined by keeping track of the best state sequence produced by the above recursive procedure.

The Training Problem

Given an observation sequence $O = O_1 O_2 \dots O_T$ and an HMM λ - how to adjust the model parameters of λ to maximize $P(O|\lambda)$, i.e. how to modify a given HMM λ to be able to generate a given observation sequence O, also called training sequence, more likely? By solving this problem the HMM λ is optimized to match the observation O more likely.

A well known solution for the training problem is mentioned in [Rab89] which is the Baum-Welch-Algorithm. This algorithm is based on the Forward-Backward algorithm [Rab89]. Here, the transition probabilities and the emission probabilities are optimized. However, in ASR it is common to discard the optimization of the state transition probabilities because it does not have a great impact on the performance of the recognition system. Thus, the training problem can be reduced to only optimize the parameters of the state dependent emission probabilities. This can be achieved by applying a Viterbi-based EM training algorithm that can be carried out in two steps. In the first step the Viterbi-Algorithm mentioned above is applied to determine the most likely state sequence. Then, in the second step the parameters of the model are re-estimated according to the state sequence found in step one. This training algorithm is explained in more detail in section 4.2.2

4.2 HMM-based Motion Recognition System

In section 4.1 a brief introduction to the theory of HMMs was given. In this section more information is given about how an HMM-based recognition system is built and how such a system is applied for HMR.

4.2.1 Modeling of Motion Units

In order to apply HMR several issues have to be considered. One of these issues is the abstraction level at which the motion data is modeled. There are different levels of abstraction that can be applied for modeling human motion data. Motion models at a high level of abstraction are more complex than motion models at a lower abstraction level and they also have a higher information content. Models at a high abstraction level might represent complex motions like ”pouring water” whereas models at a low abstraction level might represent primitive motions like ”take glass” or ”take bottle”. In order to model complex motions at a higher abstraction level a huge amount of training data is needed. This is because models of each complex motion type require a sufficient number of training data that contains different motion variations in order to achieve a robust estimation of the motion. Another disadvantage of modeling complex motions is that new un-modeled complex motions cannot be recognized. Hence, for each new complex motion a new HMM has to be built.

In ASR different levels of abstraction are applied depending on the target application. In a system with limited vocabulary a word abstraction level can be applied. Each single word in the vocabulary is modeled with a single HMM. Sentences which are at a higher level of abstraction can be modeled by concatenating different word models. This approach is adopted by many works concerned with HMR in order to model human motion data. Like a sentence in speech that can be split into several

words a complex motion like "pouring water" can be split into primitive motion units like "take glass", "take bottle", "pour water", "put bottle back" and "put bowl back". This way each motion unit can be modeled by using a single HMM. A complex motion can be modeled by concatenating a certain number of motion unit models. This kind of modeling decreases the amount of training data required by the recognition system. Additionally, the flexibility in composing complex motions using motion unit models increases. New complex motions can be modeled or recognized by concatenating existing motion unit models. Gehrig et al. [GKWS09] investigate such an approach by modeling a set of primitive motion units extracted from complex motion sequences. The motion units are considered as a limited low-level motion vocabulary. This approach as well as the optimized configuration of the HMM-based recognition system presented in [GKWS09] are adopted in this work.

A motion unit model represented by an HMM consists of a certain number of states. Each state represents a part of the motion unit. In this work every motion unit is modeled with an HMM consisting of 4 states: 1) the beginning of the motion unit, 2) middle part I, 3) middle part II, and 4) the end of the motion unit.

The states in an HMM are connected to each other by a certain kind of topology. In this work a linear left-to-right topology is applied. This topology is often used in ASR and is successfully applied for HMR, e.g. in [GKWS09]. As shown in figure 4.1, the topology only allows state transitions to the current state or to the right neighboring state. As mentioned in section 4.1 the transition probabilities have no impact on the performance of the recognition system. Hence, all transition probabilities are set to be equal.

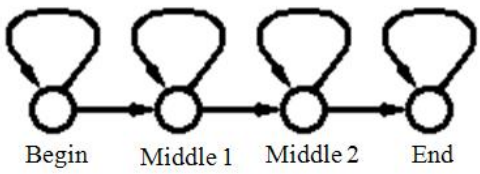

Figure 4.1: An HMM with linear left-to-right topology.

As mentioned in chapter 3 the information comprised in the human motion data used in this work are angle values of 24 DOF of different joints. The angle values of one DOF lie in the interval $I = [-\pi, \pi]$. The angle values of all DOF are then elements of the space $V = I^{24}$. Consequently, V builds the continuous feature space or rather the continuous symbol space of HMMs used to model the motion units. As mentioned in section 4.1 this kind of HMMs is called continuous HMMs.

The HMM states emit feature vectors $v \in V$ according to the state dependent emission probability densities b_i, where i refers to state i. The emission probability densities used in this work are Gaussian mixtures. The emission probability density b_i of state i is then defined as follows:

$$b_i(x) = \sum_{j=1}^{m} c_{ij} N(x|\mu_{ij}, \Sigma_{ij}), \tag{4.15}$$

where the parameters c_{ij}, μ_{ij} and Σ_{ij} have to be estimated in the training phase. In this work, a Gaussian mixture of 16 Gaussians with diagonal covariance matrices

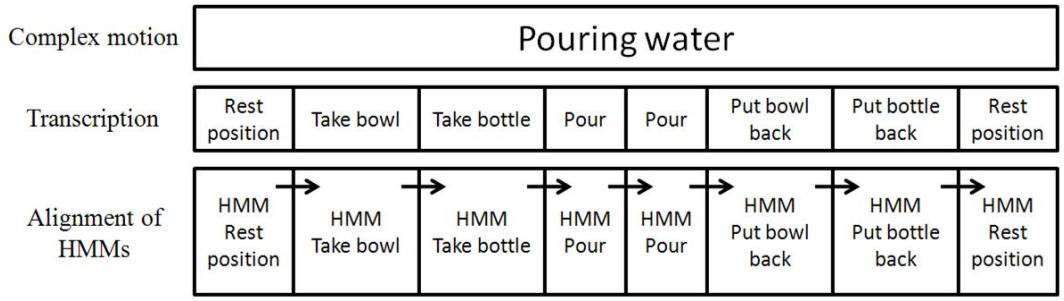

Figure 4.2: The forced alignment of a complex motion sequence. According to a motion transcription, the motion unit models (HMMs) are concatenated or aligned in order to build a model for the entire complex motion. The arrows illustrate the transition from the end state of the predecessor HMM to the begin state of the successor HMM.

is used as an emission probability density function. The number of Gaussians was optimized in experiments conducted in [GKWS09].

4.2.2 Initialization and Training of Motion Unit Models

The training process is essential when HMM-based modeling is applied. In the training process the parameters of the models, especially the parameters of the emission probability densities, are estimated to fit the underlying training data.

As mentioned above the algorithm applied for training in this work is a Viterbi-based EM-algorithm. In order to apply the Viterbi-based EM-algorithm initial motion unit models are required. A good initialization of motion unit models can be achieved by applying pre-segmented motion data. According to the segmentation different data segments in motion data sequences can be assigned to specific motion units.

The initialization task is carried out in three steps. The first step is to divide each data segment associated with a certain motion unit by the number of HMM states. This way a set of feature vectors is assigned for each state to be used for initializing the parameters of the Gaussian mixtures. The second step is to cluster the feature vectors associated with an HMM state among a number of clusters equal to the number of Gaussians in the Gaussian mixture. In the third step of initialization the mean and the covariance matrix of each Gaussian are calculated out of the feature vectors located in the corresponding cluster. In this work the neural gas algorithm [MBS93] is applied as clustering method in the initialization task of HMMs.

After initialization of the motion unit models the actual training procedure can be started. In this work a Viterbi-based EM-algorithm is applied. The algorithm is based on two main steps. The first one can be referred to as the expectation step of the EM-algorithm and the second one as the maximization step.

The expectation step itself consists of several steps. Firstly, a forced alignment of motion units is applied. In this step the motion unit models which appear in a certain complex motion training sequence are concatenated according to the transcription of the complex motion. A transcription in this context means the ordered sequence of motion unit labels applied to compose the complex motion. In figure 4.2 the forced alignment of the complex motion ”pouring water” is illustrated. In a subsequent step the Viterbi-Algorithm is applied to find the best state sequence within

the concatenated HMM sequence given the training data sequence. This way each feature vector in the training sequence is assigned to one of the states of the concatenated HMMs. In the maximization step the assigned feature vectors are used to re-estimate the Gaussians of the associated HMM state by using an EM-algorithm.

The expectation and maximization steps of the Viterbi-based EM-algorithm are iteratively repeated for a defined iteration number in order to find a good estimation of the Gaussian Mixtures. In this work five iterations were applied.

4.2.3 Decoding using IBIS

The decoding is the process of finding the most likely hypothesis for a given observation. In this work the observation is a complex motion sequence X which consists of a certain amount of frames (feature vectors). The sequence can be subdivided into several primitive motion units. Thus, an optimal hypothesis provided by the decoding process has to be a sequence of motion unit labels $\hat{U}$ (transcription) which match the motion units in the observation most likely. This can be defined as follows:

$$\hat{U} = \arg\max_{U} P(U|X). \tag{4.16}$$

By applying the Bayes' theorem for conditional probabilities the following is obtained:

$$P(U|X) = \frac{P(X|U)P(U)}{P(X)}, \tag{4.17}$$

where $P(X|U)$ is the probability of observing feature vector sequence X given the sequence of motion units U, $P(U)$ is the prior probability of motion unit sequence U and $P(X)$ is the prior probability of observing the feature vector sequence X. Since $P(X)$ is a denominator which has no impact on maximizing $P(U|X)$ (4.16) can then be written as follows:

$$\hat{U} = \arg\max_{U} P(X|U)P(U). \tag{4.18}$$

Since X is a sequence of feature vectors and U a sequence of motion unit labels (4.18) can be written as follows:

$$\hat{U} = \arg\max_{U_1,U_2,\ldots} \prod_i P(X_i|U_i)P(U_i), \tag{4.19}$$

where $P(X_i|U_i)$ is the probability of observing the partial observation X_i given the HMM with the label U_i and $P(U_i)$ is the prior probability of motion unit U_i which is determined by using a statistical motion unit model. An example of a statistical motion unit model is the bi-gram model in which the occurrence probability of a certain motion unit given a certain motion unit as its predecessor is comprised. In this work the bi-gram statistical model is applied.

The IBIS decoder used in this work is based on the Viterbi-Algorithm. It integrates the decoding of the optimal state sequence within an HMM for a single motion

unit and the transition between distinct motion units in one process. The prior probabilities of motion units in the statistical bi-gram model are considered when a transition between two motion units is possible. The transition is possible only if the current state of the HMM is the end state.

The IBIS decoder carries out the decoding as a time-synchronous beam search in which only the state paths with the highest probabilities within the beam are considered. During time-synchronous search the considered hypotheses are expanded simultaneously frame by frame.

The output of the IBIS decoder is a hypothesis consisting of a sequence of motion unit labels and their boundaries in the observation sequence.

4.2.4 Performance Measures

4.2.4.1 Motion Unit Error Rate

In order to measure the performance of the recognition system the hypothesis provided by the decoder is compared to a reference motion unit sequence (manual transcription). The more a hypothesis matches the corresponding reference the lower is the recognition error. The comparison is achieved by using dynamic programming in which the motion units of the hypothesis and the reference are aligned. The alignment outputs the number of insertions (I), deletions (D), and substitutions (S) of motion units needed to align the hypothesis and reference sequences. These quantities can be used to define a performance measure.

Similar to the word error rate (WER) measure which is a common measure in ASR, a measure for motion unit recognition can be defined in terms of motion unit error rate (MUER) which can be calculated as follows:

$$MUER = \frac{I + D + S}{N} \cdot 100\%, \tag{4.20}$$

where N is the number of motion units in the reference sequence.

The MUER can be greater than 100% since the number of motion units N in the reference can be smaller than the sum of I, D, and S.

4.2.4.2 Precision, Recall, and F-score

A performance measure for a recognition system with respect to individual motion units can be helpful in many situations. This, however, cannot be achieved by using the MUER introduced in the last section. This is because the MUER can only be applied to measure the error rate with respect to complete sequences of concatenated motion units.

In order to measure the performance with respect to individual motion units the precision, recall, and F-score measures are applied. The precision measure with respect to a certain motion unit type u is defined as follows:

$$precision_u = \frac{N_{u,c}}{N_{u,h}}, \tag{4.21}$$

where $N_{u,c}$ indicates how many times motion unit u is recognized correctly and $N_{u,h}$ is the occurrence frequency of motion unit u in the hypothesis. A motion unit is considered to be recognized correctly if it occurs in the hypothesis in the same position as in the reference sequence after aligning the two sequences by using dynamic programming.

The precision measure, however, is not sufficient for measuring the performance because it does not consider the deletions that occur when the alignment of the hypothesis and reference sequences is applied. This can be balanced by using the recall measure which is defined as follows:

$$recall_u = \frac{N_{u,c}}{N_{u,r}}, \tag{4.22}$$

where $N_{u,c}$ indicates how many times motion unit u is recognized correctly and $N_{u,r}$ is the occurrence frequency of motion unit u in the reference sequence.

In order to enable the comparability of performance of different motion unit types the precision and recall measures are combined to define the F-score measure. The F-score measure is the harmonic mean of the precision and recall measures and it is defined as follows:

$$F\text{-}score_u = \frac{2 \cdot precision_u \cdot recall_u}{precision_u + recall_u}. \tag{4.23}$$

Unlike the MUER, the higher the F-score value the higher is the performance of the recognition system.

4.2.5 Confidence Measures

In the last section a performance measure for the recognition system is given in terms of the MUER. This can be achieved by comparing the hypothesis with a reference motion sequence, but in real-world applications there are no reference data which can be used in order to assess the correctness or rather the confidence of a hypothesis. This kind of measure, however, can be very useful for many applications. Actually, this kind of measure has been investigated by many scientists in many fields of artificial intelligence. The most common term for such a measure is the confidence measure.

The definition of a confidence measure depends on the type of the underlying decoding process. In this work the applied decoding process is a time-synchronous beam search that allows the generation of a so-called lattice. A lattice is graph with all retrieved recognition results. It contains the best hypotheses provided by the decoding process. The hypothesis with the highest a posteriori probability in a lattice is chosen as the recognition result. The a posteriori probability is a first indication about the confidence of the provided hypothesis, but in this case the a posteriori probability corresponds to the above mentioned complete sequence $\hat{U}$ of motion unit labels. Yet, a confidence measure for each motion unit within the sequence is more beneficial for many applications.

In their work [KS97], Kemp and Schaaf introduce different confidence measures based on a lattice as an information source. In the following sections two different

confidence measures investigated in [KS97] are introduced. After the introduction of the confidence measures, a method for the confidence measure evaluation is introduces.

4.2.5.1 Gamma-Probability

A lattice provided by the above mentioned IBIS decoder consists of nodes that represent the motion units at different time instances in different hypotheses and links that represent the transition among different motion units. The topology of a lattice can be compared to that of an HMM; the nodes of the lattice can be seen as states of an HMM, the probability $P(X_i|U_i)$ provided by a motion unit model in a certain node in the lattice can be seen as the emission probability of an HMM state, and the prior probabilities provided by the statistical motion unit model can be seen as the transition probabilities among HMM states. This analogy leads to the possibility of applying the Forward-Backward-Algorithm in order to enable the calculation of a probability for each link in a lattice. The link probabilities can then be interpreted as a posteriori probabilities of motion units. This probability measure is called the gamma probability and it can be applied as a confidence measure for hypotheses in the motion unit abstraction level.

4.2.5.2 A-Stabil

Another confidence measure which is based on a lattice of hypothesis is the acoustic stability. As the name suggests, this confidence measure is based on the stability of the acoustic score. The acoustic score in ASR is analogous to the probability $P(X_i|U_i)$ provided by a motion unit model given an observation. The term acoustic stability is retained in this work.

Consider an extension of (4.18) with a weighting factor lz and a motion unit insertion penalty lp as follows:

$$\hat{U} = \arg\max_{U} \left[lp^{|U|} \cdot P(X|U) \cdot P(U)^{lz} \right], \tag{4.24}$$

where $|U|$ is the count of motion units in U.

In order to determine the acoustic stability a set H of alternative hypothesis is computed by setting different values for lz and lp. In a subsequent step the best hypothesis provided by the recognition system is used as a reference hypothesis to be aligned with each of the remaining hypotheses in H. Finally, for each motion unit in the reference hypothesis the occurrence frequency in the hypotheses of H is calculated and normalized by the size of H to indicate the acoustic stability of the given motion unit. This can be considered as a confidence measure in the motion unit abstraction level.

4.2.5.3 Evaluation of the Confidence Measures

In order to make meaningful use of a confidence measure it has to be evaluated first. The evaluation shows whether recognized motion units with a high confidence value are actually recognized correctly and whether recognized motion units with a low confidence value are recognized wrong. This can be achieved by comparing the confidence value of a recognized motion unit with its precision value.

The precision measure mentioned in section 4.2.4.2 is calculated with respect to a certain motion unit type u. In order to make use of the precision measure in conjunction with the confidence measure the precision measure is calculated with respect to confidence values within a specific confidence value range.

$$precision_x = \frac{N_{x,c}}{N_x}, \tag{4.25}$$

where $N_{x,c}$ is the number of correctly recognized motion units with confidence values within the range x and N_x is the number of all motion units with confidence values within the range x.

Equation 4.25 gives the rate of correctly recognized motion units with respect to confidence values within a specific confidence value range. A meaningful confidence measure is found when $precision_x$ increases and decreases proportionally to the confidence values within the range x.

4.2.6 Cross-Validation

In order to avoid noisy recognition results because of the relatively small size of the data sets the cross-validation technique is utilized. In this work the k-fold cross-validation technique is used in which k different partitions of a data set are applied, each containing a different subset of training and test data. Each different partition is called fold. The test data set in each fold is complementary to the test data set in other folds. The test data set of each fold consists of N_{test} motion sequences:

$$N_{test} = \frac{1}{k} N_{all}, \tag{4.26}$$

where N_{all} is the amount of motion sequences in the complete data set. The training data set is given by the remaining $N_{training}$ motion sequences:

$$N_{training} = \frac{k-1}{k} N_{all}. \tag{4.27}$$

Consequently, the complete data set is a conjunction of the test and the training subsets in a specific fold.

In order to determine the overall recognition performance the recognition performance of each fold is measured separately. Then the average of the performance of all folds is considered as the overall performance. An example of a 5-fold cross-validation is illustrated in figure 4.3.

The evaluation of the recognition system is carried out by using the data sets presented in section 3.2. The evaluation using data set A and B is carried out by using a 10-fold cross-validation. With regard to data set A, in each fold 45 motion sequences of each motion type can be used to initialize and train the system. The resulting 5 motion sequences of each motion type can be used to test the system. The total number of motion sequences in the training set in each fold is 450. The total number of motion sequences in the test set in each fold is 50.

Figure 4.3: A 5-fold cross-validation. The black area represents the motion sequences used for testing. The motion sequences in the remaining white area are used for initialization and training.

With regard to data set B, in each fold 18 motion sequences of each motion type can be applied to initialize and train the system. The remaining 2 motion sequences of each motion type can be applied to test the system. The total number of motion sequences in the training set in each fold is 90. The total number of motion sequences in the test set in each fold is 10.

The evaluation using data set C is carried out by using a 5-fold cross-validation. In each fold 20 motion sequences of each complex motion type (4 for each positioning configuration) are used to initialize and train the system. The remaining 5 motion sequence of each motion type (one for each positioning configuration) are used to test the system. The total number of motion sequences in the training set in each fold is 80. The total number of motion sequences in the test set in each fold is 20.

4.2.7 Experiments

In the previous section an HMM-based human motion recognition system was introduced. In this section several recognition systems of two different types are developed. The first system type is referred to as the baseline system. A baseline system is developed by using the same amount of manually segmented data for initialization and training. The amount of manually segmented data is varied between different experimental sessions. The second system type is a recognition system that can be developed by using different amounts of manually segmented data for initialization and manually transcribed data for training. The performance of both system types is measured in terms of MUER introduced in section 4.2.4.1.

A baseline system is defined as follows:

$$BS(data, c_{i,t}), \tag{4.28}$$

where *data* is the applied data set and $c_{i,t}$ is the number of motion sequences of each complex motion type used for initialization and training. In this work the development of baseline systems serves two purposes; firstly, the performance of a baseline system is considered as a baseline for the performance of other recognition systems developed by applying the approaches investigated in this work; secondly, the developed baseline systems are utilized for segmentation in the supervised HMM-based segmentation method.

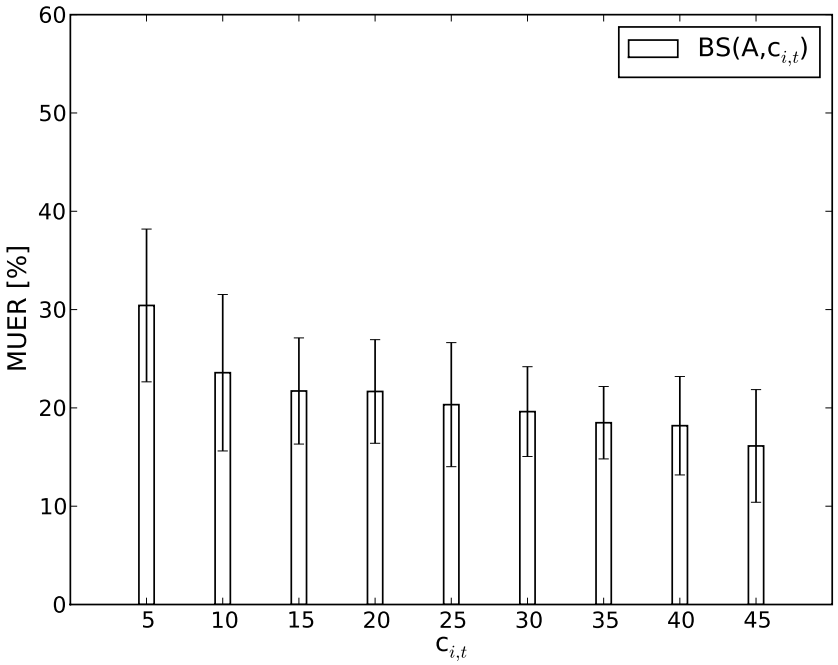

Figure 4.4: The performance of the baseline systems using different amounts of initialization and training data from data set A. $c_{i,t}$ is the number of manually segmented motion sequences of each complex motion type used for initialization and training.

The first experiment conducted in this section deals with the evaluation of different baseline systems on different data sets. The evaluation is carried out by using different data amounts for the initialization and training tasks.

The evaluation of baseline systems using data set A is conducted by applying a 10-fold cross-validation as described in section 4.2.6. For each fold 9 experimental sessions are carried out. In each session a new baseline system is developed by using a different amount of initialization and training data. In the first session the baseline system is developed by using 5 motion sequences of each complex motion type. In each following session the sequence number of each complex motion type is increased by 5.

The results of the evaluation are shown in figure 4.4. As expected the performance increases by increasing the number of motion sequences used for initialization and training. A performance difference of 7% can be noticed between $BS(A, 5)$ and $BS(A, 10)$. This difference decreases between $BS(A, 10)$ and $BS(A, 15)$ to become about 2%. In the following sessions the performance difference between almost all two consecutive baseline systems decreases to a value smaller than 1% MUER. The best recognition result which is a MUER of 16.13% is achieved by using all 45 motion sequences of each complex motion type for initialization and training ($BS(A, 45)$). This result is accompanied by a standard deviation of 5.73%.

With regard to the evaluation using data set B, the experiment is also carried out by using a 10-fold cross-validation. For each fold, 9 experimental sessions are carried out. The first session is carried out by using 2 motion sequences of each complex

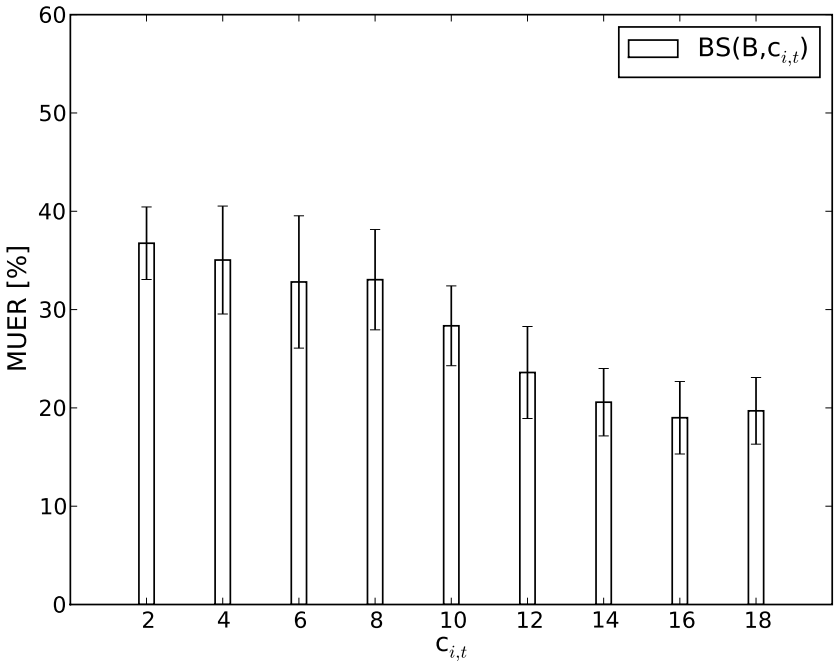

Figure 4.5: The performance of the baseline systems using different amounts of initialization and training data from data set B. $c_{i,t}$ is the number of manually segmented motion sequences of each complex motion type used for initialization and training.

motion type for initialization and training. In the following sessions the number of sequences is increased by 2.

The results of the evaluation are shown in figure 4.5. The same trend as in the evaluation using data set A can be noticed. The performance increases by increasing the number of motion sequences used for initialization and training, although the results in session 4 and 9 do not quite follow the trend. The standard deviation increases firstly then decreases with increasing the training sequence count. The best recognition result is a MUER of 19% which is achieved by using 16 motion sequences for training ($BS(B, 16)$). The standard deviation amounts to 3.69%.

The evaluation using data set C is carried out by using a 5-fold cross-validation. For each fold the performance is measured in 4 different sessions. The first session is carried out by using 5 motion sequences of each complex motion type (one sequence of each positioning configuration). In each following session the number of motion sequences is increased by 5. The last session is carried out by using 20 motion sequences.

The results of the evaluation are shown in figure 4.6. The performance difference in the first three sessions is minimal, but a high performance difference of 5.5% between the last two sessions can be noticed. The best performance of 32.51% MUER is achieved in session 4 ($BS(C, 20)$) in which the entire training set is applied. This is accompanied by a standard deviation of 3.65%.

The evaluation of the baseline systems using different amounts of training data shows that the higher is the amounts of initialization and training data the better is the

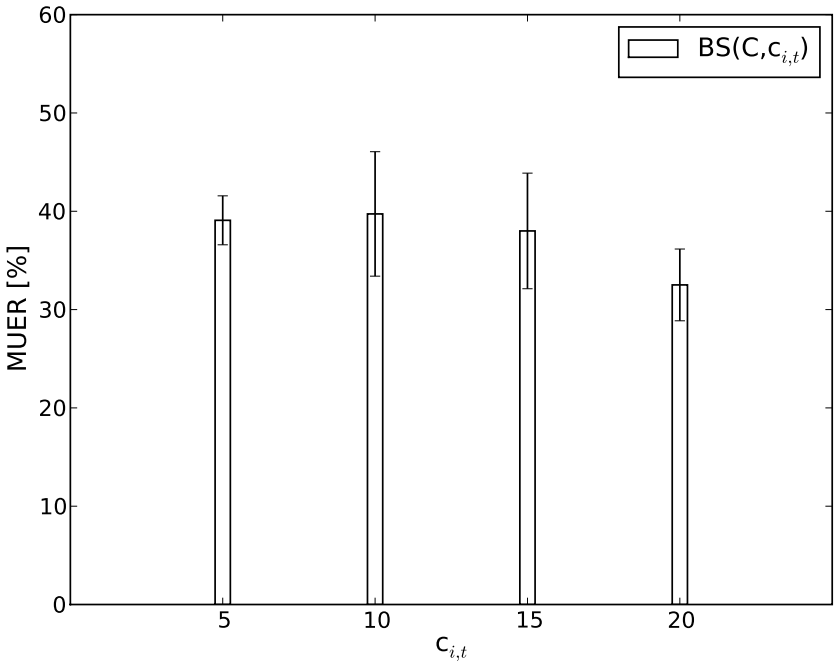

Figure 4.6: The performance of the baseline systems using different amounts of initialization and training data from data set C. $c_{i,t}$ is the number of manually segmented motion sequences of each complex motion type used for initialization and training.

performance of the baseline system. This is due to the fact that the Gaussians of the emission probability density functions of the HMM states can be estimated more robustly when more data is provided.

The effort spent for the preparation of manual transcriptions is much lower than that of manual segmentations. The segmentation data is only applied in the initialization task. In the training task only the transcriptions of data sequences is required. Therefore, using a little amount of manually segmented data for the initialization and more transcription data for the training might lead to a reliable performance of the recognition system and at the same time the effort spent for preparing manual segmentations can be reduced.

In the following experiment several recognition systems are developed which are initialized and trained by using different amounts of manually segmented data for initialization and different amounts of manually transcribed data for training. The developed recognition systems are defined as follows:

$$RS_m(data, c_i, c_t), \tag{4.29}$$

where *data* is the applied data set, c_i is the number of motion sequences of each complex motion type used for initialization, and c_t is the number of motion sequences of each complex motion type used for training.

The evaluation using data set A is carried out in 5 sessions. In each session a 10-fold cross-validation is applied. In the first session 5 sequences of each complex

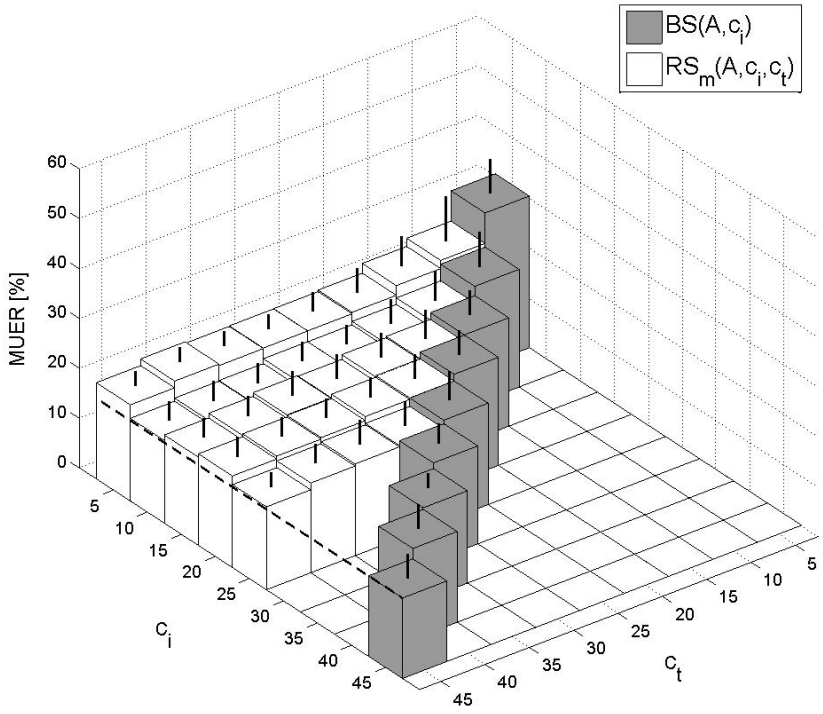

Figure 4.7: The performance of the recognition systems using different amounts of training data from data set A. c_i is the number of manually segmented motion sequences of each complex motion type used for the initialization. c_t is the number of manually transcribed motion sequences used for training.

motion type are used for the initialization task. This number is increased by 5 in each following session. In each session several recognition systems are developed and evaluated. The first recognition system in each session is trained by using 5 more motion sequences of each complex motion type than used in the initialization task. This number is increased by 5 for every following recognition system in the session. The maximum number of training sequences of each complex motion type is 45.

In figure 4.7 the results of the evaluation of the developed recognition systems and the results of the evaluation of the baseline systems using data set A are shown. The gray bars represent the performance of the baseline systems developed in the last experiment. The white bars represent the performance of the recognition systems trained with different amounts of data sequences in conjunction with their transcriptions. The lines on the bars represent the standard deviation of MUERs between different cross-validation folds. It can be seen that in all sessions the performance of the recognition systems increases by increasing the number of manually transcribed motion sequences used for training. The performance is almost as good as the performance of the baseline systems initialized and trained with the same number of motion sequences. Using many manually transcribed motion sequences reduces the MUER even if the number of motion sequences used for initialization is low. A good example for this behavior is the recognition system $RS(A, 10, 45)$ which has a performance of 16.57% MUER. This is very close to the performance of the baseline system $BS(A, 45)$ (16.13% MUER) which is initialized by using 35

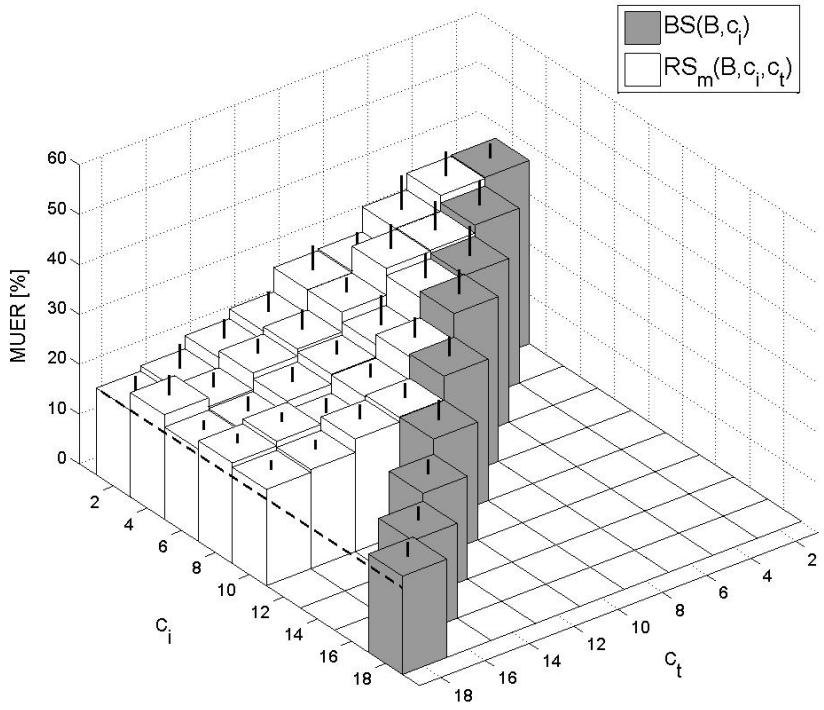

Figure 4.8: The performance of the recognition systems using different amounts of training data from data set B. c_i is the number of manually segmented motion sequences of each complex motion type used for the initialization. c_t is the number of manually transcribed motion sequences used for training.

additional motion sequences of each complex motion type. In almost all cases the standard deviation decreases by increasing the training data.

The evaluation using data set B is also carried out in 5 different sessions. For this, a 10-fold cross-validation is applied. In the first session 2 sequences of each complex motion type are used for the initialization task. This number is increased by 2 in each following session. In each session several recognition systems are developed and evaluated. The first recognition system in each session is trained by using 2 more motion sequences of each complex motion type than used in the initialization task. This number is increased by 2 for every following recognition system. The maximum number of training sequences of each complex motion type is 18.

In figure 4.8 the results of the evaluation of the developed recognition systems and the evaluation of the baseline systems using data set B are shown. The same trend as in the evaluation using data set A can be noticed. In all sessions the performance of the recognition systems increases by increasing the number of transcribed motion sequences used for training. The performance is almost as good as the performance of the baseline systems initialized and trained with the same number of segmented motion sequences. In some cases the recognition systems outperform the baseline systems initialized by using a higher number of manually segmented motion sequences, but trained using the same number of transcribed motion sequences. Recognition system $RS_m(B, 2, 18)$ which has a performance of 17.37% MUER out-

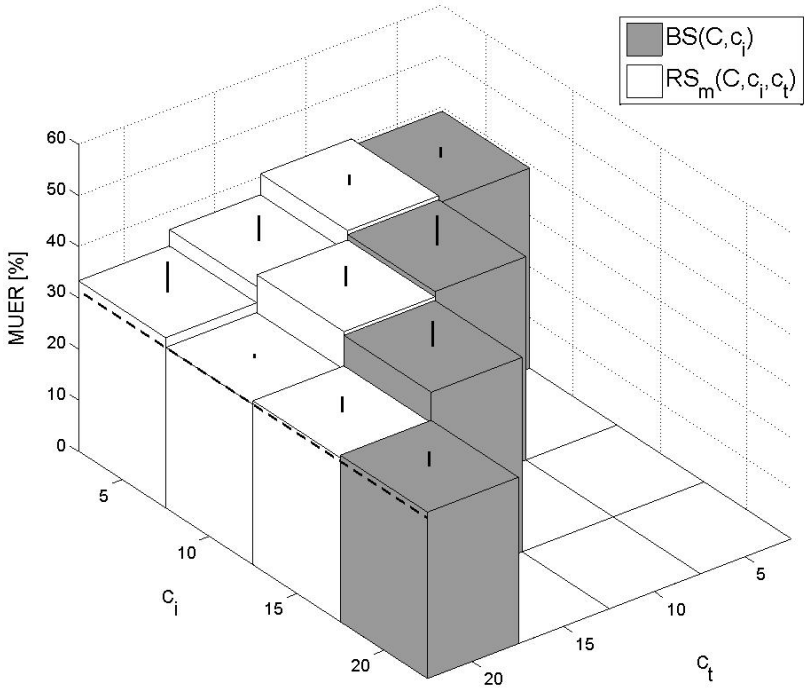

Figure 4.9: The performance of the recognition systems using different amounts of training data from data set C. c_i is the number of manually segmented motion sequences of each complex motion type used for the initialization. c_t is the number of manually transcribed motion sequences used for training.

performs the baseline system $BS(B, 18)$ (19.7% MUER) even though it is initialized by using 10 more segmented motion sequences of each complex motion type. In almost all cases the standard deviation decreases by increasing the training data.

The evaluation using data set C is carried out in 3 sessions. In each session a 5-fold cross-validation is applied. In the first session 5 sequences of each complex motion type (one of each positioning configuration) are used for the initialization task. This number is increased by 5 in each following session. In each session several recognition systems are developed and evaluated. The first recognition system in each session is trained by using 5 more motion sequences of each complex motion type than used in the initialization task. This number is increased by 5 for every following recognition system. The maximum number of training sequences of each complex motion type is 20.

The results of the evaluation using data set C are illustrated in figure 4.9. The gray bars represent the performance of the baseline systems developed. The same trend as in the last two evaluations can be noticed. Except for recognition system $RS_m(C, 5, 10)$ all other recognition systems perform better than the baseline systems initialized with the same amount of data. $RS_m(C, 10, 20)$ with a performance of 31.33% MUER performs slightly better than the baseline system $BS(C, 20)$ (32.03% MUER). The standard deviation increases only in the first session. In the remaining sessions the standard deviation always decreases.

The results of the last experiment leads to the conclusion that the amount of training data is more significant for the performance of the recognition system than the amount of initialization data. Even if a little amount of data is applied for initialization the performance of the recognition system can be significantly increased by applying more data in the training task. This is a great advantage since for the training task only the data sequences and their transcriptions is required. The effort spent for the preparation of manual transcriptions is much lower than the effort spent for preparation of manual segmentations. Thus, the first possible approach for reducing the development effort for a motion recognition system is to reduce the number of manual segmentations used for initialization and increase the number of manual transcriptions used for training.

5. Automatic Segmentation

Segmentation is an important task applied for the initialization and for the training of many recognition systems. It becomes essential when signals are modeled at an abstraction level of representation lower than that of the provided signal data used for training. With regard to HMR, complex motions at a higher level of abstraction, e.g. "pouring water", have first to be segmented into primitive motion units at a lower abstraction level, e.g. "take bowl", "take bottle", "pour", "put bottle back", and "put bowl back", in order to enable motion modeling at a lower abstraction level; namely at the primitive motion unit abstraction level.

Segmentation is mostly done manually which is very time-consuming, and therefore, costly. Moreover, manual segmentation is not applicable for real-time applications and there have to be strict rules for operators to avoid inconsistency. These reasons make the application of automatic segmentation methods desirable. However, developing automatic segmentation methods is very challenging. In particular, unsupervised automatic segmentation algorithms which provide a similar segmentation into meaningful motion units as provided by the manual segmentation are hard to design. They are, however, still desirable because they do not require any manually segmented training data, i.e. there is no need for human intervention. On the contrary supervised model-based automatic segmentation approaches require a certain amount of manually segmented training data, but they are able to provide a similar segmentation as provided by the manual segmentation.

In this chapter two automatic segmentation approaches are introduced. The first one is supervised and HMM-based. For this approach a little amount of manually segmented training data is required to initialize and train HMMs that are applied for automatic segmentation. The second approach is unsupervised and PCA-based. In this approach the first principal component provided by the PCA is used to derive a segmentation feature.

5.1 Supervised Automatic Segmentation

Generally, supervised model-based automatic segmentation is based on the idea of using a set of previously trained models for different motion units to find similar

motion units in un-segmented motion data. The models are usually trained by using manually segmented data. In this section a supervised automatic segmentation approach based on HMMs is introduced.

5.1.1 HMM-based Automatic Segmentation

An advantage of signal modeling using HMMs is the comprised segmentation task within the decoding process. A decoding algorithm, e.g. the Viterbi-Algorithm, provides the best state sequence given an observation. This means that sequences of feature vectors in an observation are assigned to specific HMM states. Using the IBIS decoder mentioned in section 4.2.3, it is possible to decode an observation that represents a complex motion built by concatenating several motion units. Given such an observation the IBIS decoder is able to provide the best HMM sequence that can be concatenated in order to built the observation. This way sequences of feature vectors in the observation are assigned to complete HMMs. Hence, the hypothesis $\hat{U}$ provided by the IBIS decoder consists of a sequence of motion unit labels representing the respective HMMs. The hypothesis $\hat{U}$ also comprises the boundaries between motion units in the observation (transitions between concatenated HMMs). Consequently, the hypothesis $\hat{U}$ can be considered as segmentation or transcription (without boundaries) of an observation that represents a complex motion sequence. This fact can be exploited to apply automatic motion segmentation by using a previously trained HMM-based recognition system.

As mentioned in section 4.2.7 the applied recognition system for segmentation is referred to as the baseline system. The baseline system is initialized and trained by using a certain amount of manually segmented data. The system is utilized to decode/segment un-segmented motion data.

The hypothesis provided by the baseline system can be erroneous, especially when a little amount of data is used for training. Thus, quality information about the provided hypothesis can be very useful in order to avoid a false training of a recognition system caused by using erroneous segmentation data. Therefore, a confidence measure such as mentioned in section 4.2.5 can be applied to identify and filter correctly recognized data segments in order to allow collecting clean segmentation data.

The HMM segmentation task can be applied in different configurations. A possible configuration is to apply the baseline system to segment a complete data set and then use only filtered transcriptions in conjunction with the old training data to re-train the baseline system. This can be repeated for several iterations in order to improve the quality of the transcriptions. In this work this segmentation configuration is referred to as configuration I and is formally defined as follows:

1. Define initial train set T_{ini}
2. Define test set T_{test}
3. Define segmentation set S
4. Define threshold θ_C for confidence values
5. Define structure for recognition system Θ

6. Initialize and train Θ using T_{ini}
7. Repeat the following steps for n iterations
   (a) Decode/Segment S using Θ
   (b) Write segmentation results with confidence value $\geq \theta_C$ into set T_S
   (c) Re-train Θ using $T_{ini} \cup T_S$
   (d) Test Θ using T_{test}

Another segmentation configuration is similar to configuration I. The only modification is to apply the automatic segmentation for both re-initialization and re-training instead of only re-training using transcriptions. This configuration is referred to as configuration II and it is formally defined as follows:

1. Define initial train set T_{ini}
2. Define test set T_{test}
3. Define segmentation set S
4. Define threshold θ_C for confidence values
5. Define structure for recognition system Θ
6. Initialize and train Θ using T_{ini}
7. Repeat the following steps for n iterations
   (a) Decode/Segment S using Θ
   (b) Write segmentation results with confidence value $\geq \theta_C$ into set T_S
   (c) Re-initialize and re-train Θ using $T_{ini} \cup T_S$
   (d) Test Θ using T_{test}

The modification can be found in step 7c of configuration II.

The third configuration of the HMM-based segmentation algorithm used in this work is based on the idea of using the baseline system to segment the data in several consecutive sessions. In each session only a subset of the data is segmented. The gained filtered segmentation is used in conjunction with the old training data to re-train the baseline system. The segmentation and re-training steps can be repeated for several iterations in each session. In the following sessions the re-trained baseline system is used to segment another subset of the remaining data. The new segmented data is used to re-train the baseline system in conjunction with the training data of the previous sessions. This is repeated until all available data is segmented and used to improve the baseline system. As in configuration I, only the transcriptions are used for the re-training task. Configuration III of the HMM-based segmentation algorithm is formally defined as follows:

1. Define initial train set T_{ini}

2. Define test set T_{test}
3. Define segmentation set S
4. Define threshold θ_C for confidence values
5. Define structure for recognition system Θ
6. Initialize and train Θ using T_{ini}
7. Repeat the following steps until S is empty
   (a) Select m motion sequences from S and insert them into set S_m
   (b) Repeat following steps for n iterations
      i. Decode/Segment S_m using Θ
      ii. Insert segmentation results with confidence value $\geq \theta_C$ into set T_i of iteration i
      iii. Re-train Θ using $T_{ini} \cup T_i$
      iv. Test Θ using T_{test}
   (c) Remove S_m from S

In the next section experiments are carried out in order to examine the efficiency of each segmentation configuration mentioned above.

5.1.2 Experiments

5.1.2.1 Evaluation of Confidence Measures

In section 4.2.5 two confidence measures were introduced which can be applied to estimate the confidence of recognized motion units. This kind of measure is very useful for the HMM-based segmentation process. A recognized motion unit with a high confidence value can be considered as a correct segmentation that can be applied for training a recognition system or improving an existing one.

The experiments introduced in section 4.2.7 are carried out in order to examine the performance of the baseline systems using different data sets and different amounts of initialization and training data. In addition, the a-stabil and the gamma probability confidence measures were calculated in order to enable their evaluation. This is carried out in order to examine the validity of the confidence measures. In this section the results of this evaluation are presented. The evaluation is conducted by using the confidence specific precision measure introduced in section 4.2.5.3.

In figures 5.1 A - C the precision values with respect to confidence values in different ranges provided by the baseline systems $BS(A, 5)$, $BS(A, 25)$, and $BS(A, 45)$ are plotted. The black dots represent the precision values with respect to gamma probability confidence values within different ranges. The precision values with respect to the a-stabil confidence values are represented by squares. For the a-stabil confidence measure it can be noticed that when the baseline system is trained by using 5 motion sequences, the a-stabil values are spread within the range (0.25,1] and the respective precision values are spread between 0 and 0.9. Actually, the expected positive correlation between confidence values and precision values exists.

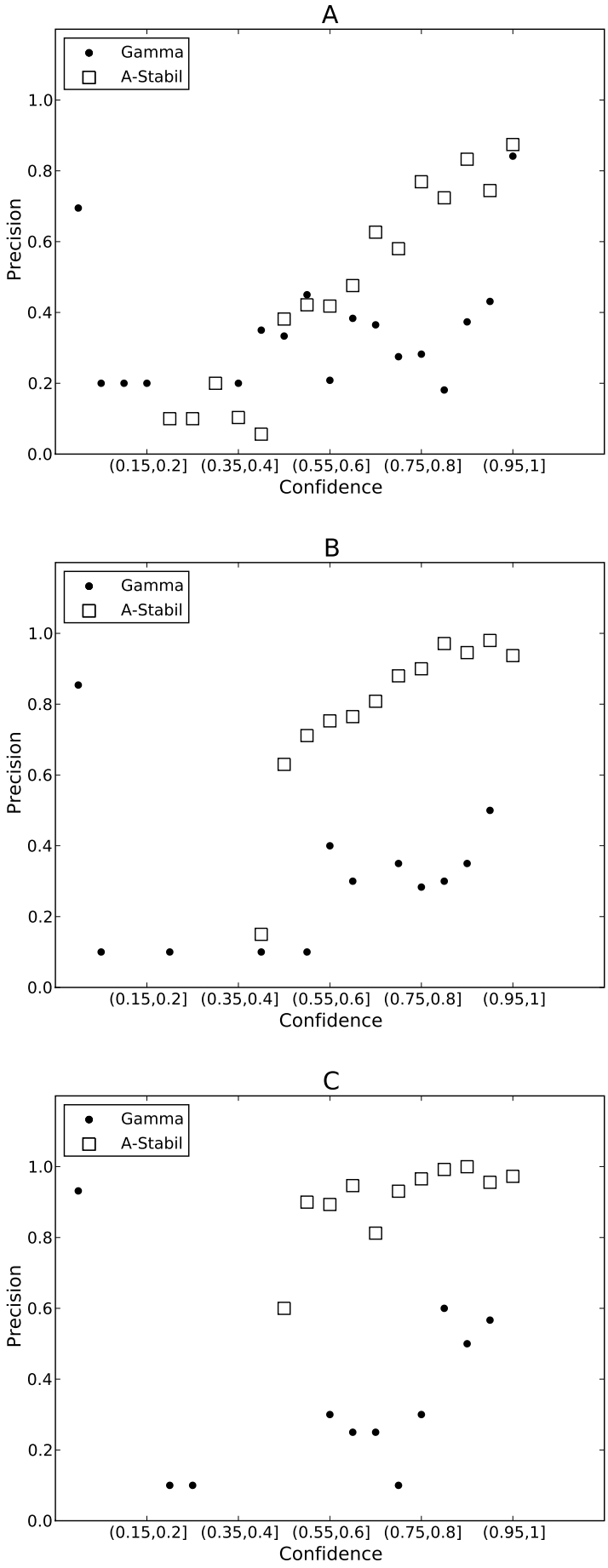

Figure 5.1: The precision values with respect to confidence values provided by the baseline systems trained using data set A. The values are provided by the baseline systems A) $BS(A, 5)$, B) $BS(A, 25)$, and C) $BS(A, 45)$.

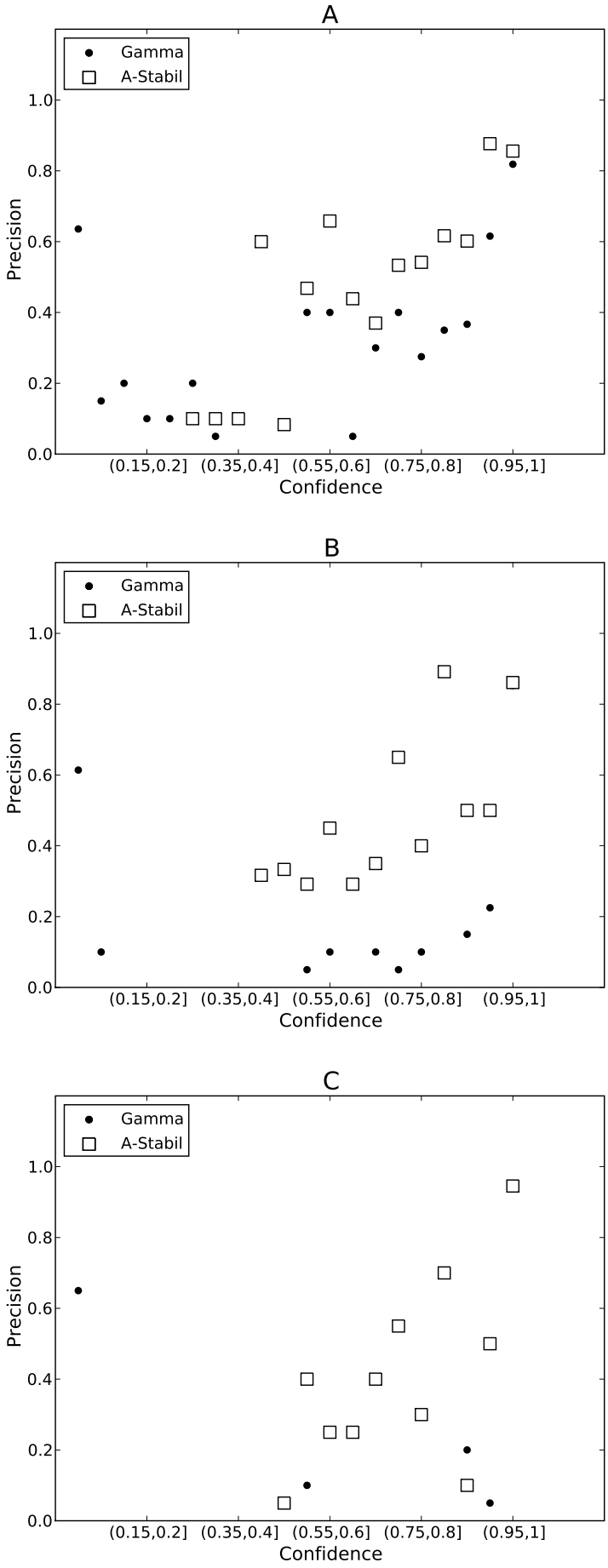

Figure 5.2: The precision values with respect to confidence values provided by the baseline systems trained using data set B. The values are provided by the baseline systems A) $BS(B, 2)$, B) $BS(B, 10)$, and C) $BS(B, 18)$.

Besides the outlier within the range (0.4,0.45] the positive correlation can also be noticed when 25 motion sequences are used for training, but in this case the a-stabil values are spread within the range (0.4,1.0]; almost all respective precision values are spread between 0.6 and 1. This indicates that the baseline system becomes more confident when more training data is used. This can also be seen in figure 5.1 C in which the precision and confidence values provided by the baseline system $BS(A, 45)$ are plotted. The a-stabil values are spread within the range (0.45,1] and almost all respective precision values are spread between 0.8 and 1. However, in this case an explicit correlation cannot be noticed.

The evaluation of the gamma probability confidence measure turns out to be quite different. No matter how much data from data set A is used for training the baseline system, the values of the gamma probability are spread within the range [0,1], but there are different gaps in which no gamma probability values exist. Almost all respective precision values are spread between 0.1 and 0.5. A correlation between precision and confidence values cannot be noticed no matter how many motion sequences are used for training the baseline system.

The precision values with respect to confidence values provided by baseline systems trained by using data set B are illustrated in figures 5.2 A - C. The precision values are calculated with respect to the confidence values provided by the baseline systems $BS(B, 2)$, $BS(B, 10)$, and $BS(B, 18)$. The values of the a-stabil measure using 2 motion sequences for training are spread within the range (0.25,1]. The precision values are spread between 0.1 and 0.85. Increasing the number of training data leads to results similar to those of the evaluation with data set A. The a-stabil values when using 10 motion sequences for training are found within the range (0.4,1] and when using 18 training sequences the values are spread within the range (0.45,1]. In all mentioned cases no explicit correlation between precision and confidence values is noticed.

The values of the gamma probability provided by the baseline systems $BS(B, 2)$, $BS(B, 10)$, and $BS(B, 18)$ are spread within the range [0,1], but different gaps in different ranges within the range [0,1] are noticed. The baseline system trained with 18 motion sequences provides gamma probability values only within the ranges [0,0.05], (0.5,0.55], and (0.85,1]. The respective precision values are low except for the precision value of 0.64 for the confidence range [0,0.05]. In all mentioned cases no explicit correlation between the precision values and the confidence values can be noticed.

The results of the evaluation of both confidence measures mentioned above do not turn out to be as expected. A clear correlation between the confidence values and the precision values does not exist. A correlation between the gamma probability values and the precision values does not exist at all. In terms of the a-stabil confidence measure a general statement about a stable correlation cannot be made since a certain correlation is only found in figure 5.1 A and B where a small amount of data set A is used for training. This leads to the assumption that either the confidence measures do not provide a meaningful estimate of the quality of the recognized motion units or the amount of confidence values used for evaluation within specific confidence value ranges is too low to enable a meaningful evaluation. Consequently, the frequency of confidence values within each confidence value range as well as the

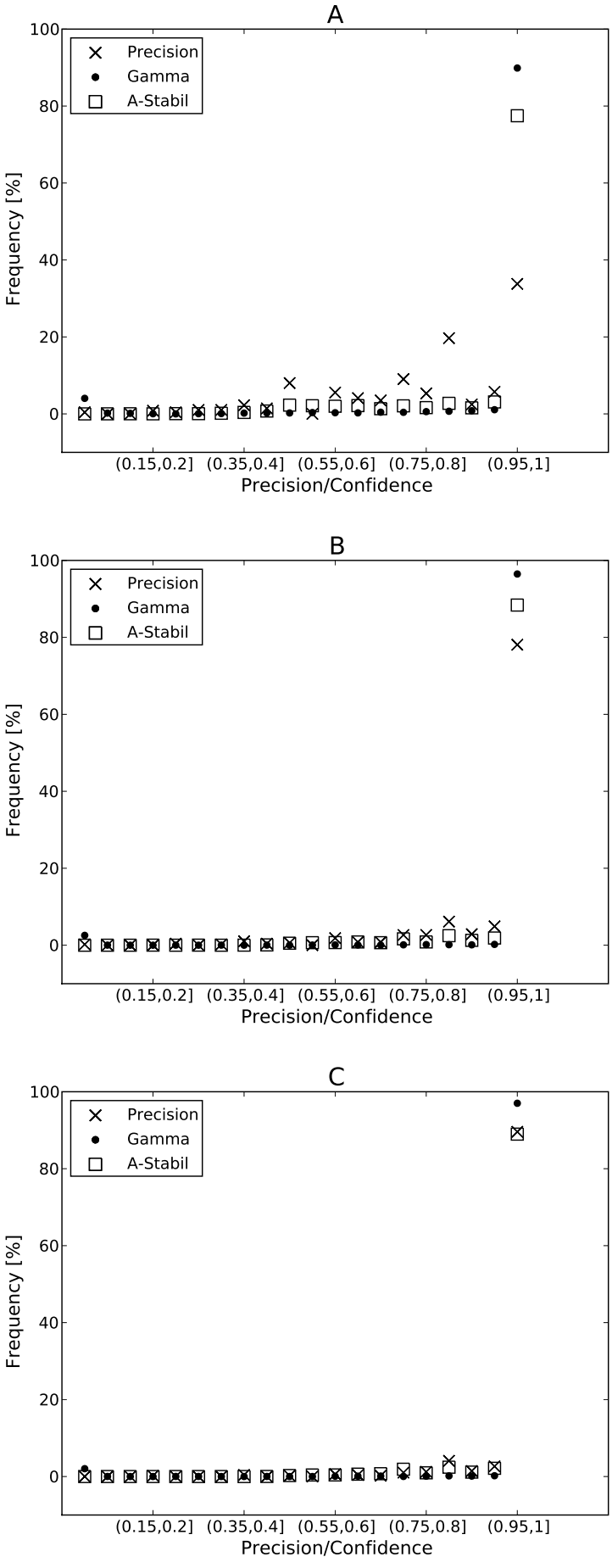

Figure 5.3: The frequency of confidence and precision values provided by the baseline systems trained using data set A. The values are provided by the baseline systems A) $BS(A, 5)$, B) $BS(A, 25)$, and C) $BS(A, 45)$.

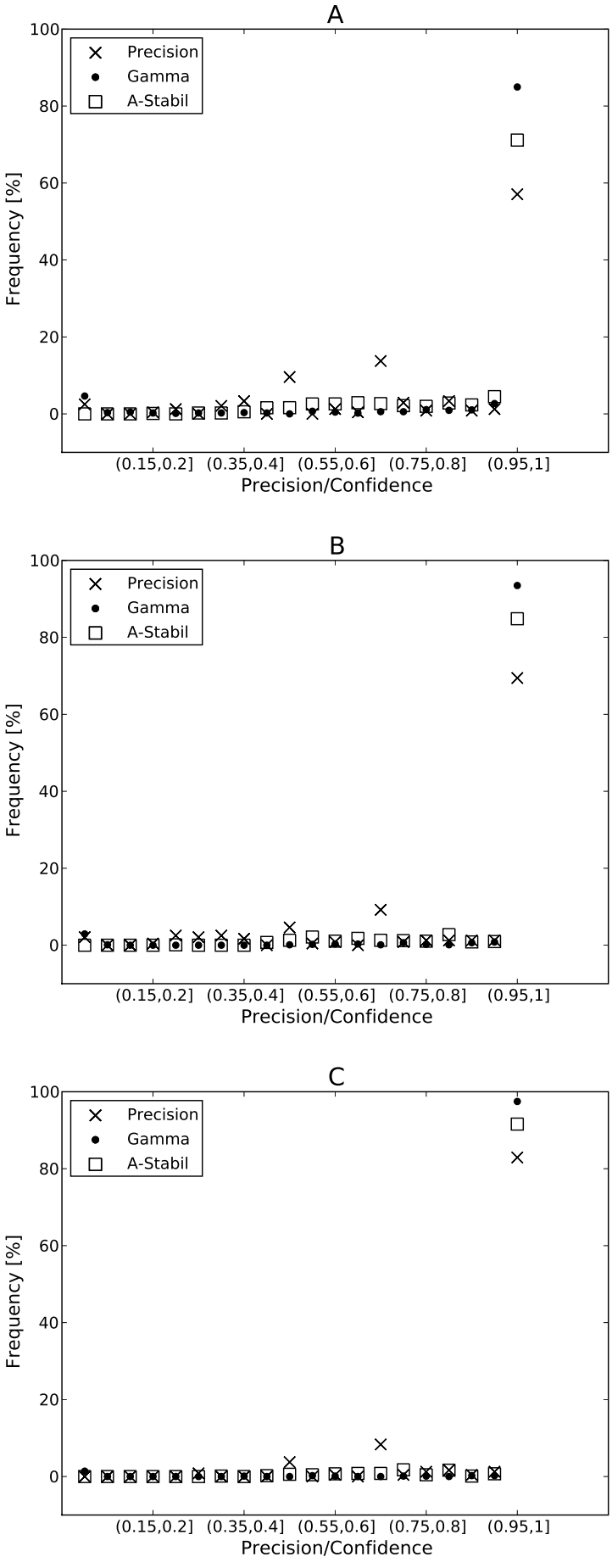

Figure 5.4: The frequency of confidence and precision values provided by the baseline systems trained using data set B. The values are provided by the baseline systems A) $BS(B, 2)$, B) $BS(B, 10)$, and C) $BS(B, 18)$.

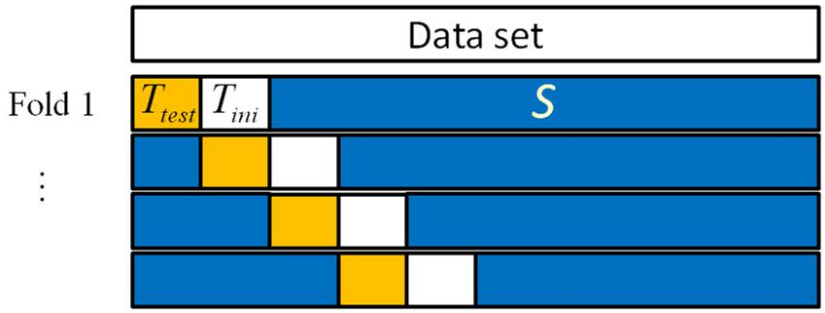

Figure 5.5: The first 4 folds of the 10-fold cross-validation used for the HMM segmentation. T_{test} indicates the test set, T_{ini} the initial train set, and S the segmentation set.

frequency of precision values of all different motion units within different ranges is calculated.

In figures 5.3 A - C and 5.4 A - C the frequency of the precision values as well as the frequency of both confidence values are plotted. The cross marker represent the frequency of precision values of all motion units within different ranges, the black dots represent the gamma probability confidence values, and the squares represent the a-stabil confidence values. It can be seen that for both confidence measures almost all confidence values lie within the range (0.95,1]. Within the range [0,0.95] the number of confidence values is very low. The same trend can be noticed with the frequency of precision values in particular when more data is applied for training the baseline system. This fact confirms the above mentioned assumption and leads to the conclusion that a meaningful evaluation of the confidence measures cannot be carried out because the number of precision values and confidence values within all ranges except the range (0.95,1] are to low.

The baseline systems are confident (high confidence values) even if a little amount of training data is applied. When a little amount of training data is used the frequency of confidence values within the range (0.95,1] is always greater than the frequency of precision values. This means that the system is overconfident (confidence values higher than objective precision). When more training data is applied the baseline system provides almost only high confidence values.

All above mentioned facts lead to the conclusion that a meaningful selection of correct recognized motion units cannot be guaranteed by using the a-stabil and the gamma probability confidence measures. This leads to the decision to discard the filtering of segmented data and investigate the performance of the HMM segmentation method without confidence-based filtering.

5.1.2.2 Evaluation of the HMM-based Segmentation Method

In this section different experiments concerning the evaluation of the HMM-based segmentation method are presented. The experiments are carried out by using the segmentation configurations I-III described in section 5.1.1. As mentioned in the last section, the confidence-based filtering is discarded and the entire automatically obtained segmentation is used to improve the baseline system.

In order to distinguish between the different developed recognition systems, specific notations are assigned to different types of recognition systems. A recognition de-

veloped by applying the HMM segmentation method in configuration I is defined as follows:

$$RS_{hmm,I}(data, c_m, c_a, i_s), \tag{5.1}$$

where *data* is the applied data set, c_m is the number of manually segmented motion sequences of each complex motion type used to initialize and train the baseline system, c_a is the number of automatically segmented motion sequences of each complex motion type used for re-training, and i_s is the applied number of segmentation and re-training iterations. The developed recognition systems in configuration II and III have the same notation, but a different index ($RS_{hmm,II}$ and $RS_{hmm,III}$).

The experiments are carried out in several sessions. In each session a different baseline system is used for the segmentation task. The 10-fold cross-validation mentioned in section 4.2.6 is modified to enable the evaluation of the segmentation method. The data set is divided into three subsets. The first one is the test data set whose size remains one tenth of the complete data set as explained in section 4.2.6. The second one is the training data set which is utilized to initialize and train the baseline system. The size of this subset is varied in different experimental sessions. The remaining part is the segmentation subset. This subset is segmented by the baseline system. The partition of the subsets in every fold is different from the partition of other folds. Figure 5.5 shows the first 4 folds of the 10-fold cross-validation used in session 1. The test subset consists of 5 sequences, the training subset of 5 sequences, and the segmentation subset of 40 sequences of each complex motion type.

The evaluation of the HMM segmentation method in configuration I and II using data set A is carried out in several sessions. In each session the number of data used to initialize and train the baseline system is increased by 5, starting with 5, and ending with 40 sequences of each complex motion type. The number of data remaining for segmentation is decreased by 5, starting with 40, and ending with 5 sequences. In each session 3 segmentation and re-training iterations for improving the quality of the segmented data are applied.

The results of the evaluation are shown in figures 5.6 A - C in which the performance of the baseline systems used for the segmentation task is compared to the performance of the resulting recognition systems. The gray bars represent the performance of the baseline systems. The white bars represent the performance of the recognition systems using configuration I and the black bars represent the performance of the recognition systems using configuration II. The lines on the bars represent the standard deviation of MUERs between different cross-validation folds. c_m is the number of motion sequences of each complex motion type used to initialize and train the baseline systems. c_a is the number of additional automatically segmented motion sequences of each complex motion type used for developing the new recognition system. As expected the performance of the recognition systems increases with increasing the number of motion sequences used for training the baseline systems. It can be noticed that there is no significant difference in the performance between configurations I and II. Re-training the baseline system without a new initialization using the segmented data performs slightly better when the baseline system is trained with a little amount of data. Using several segmentation and re-training

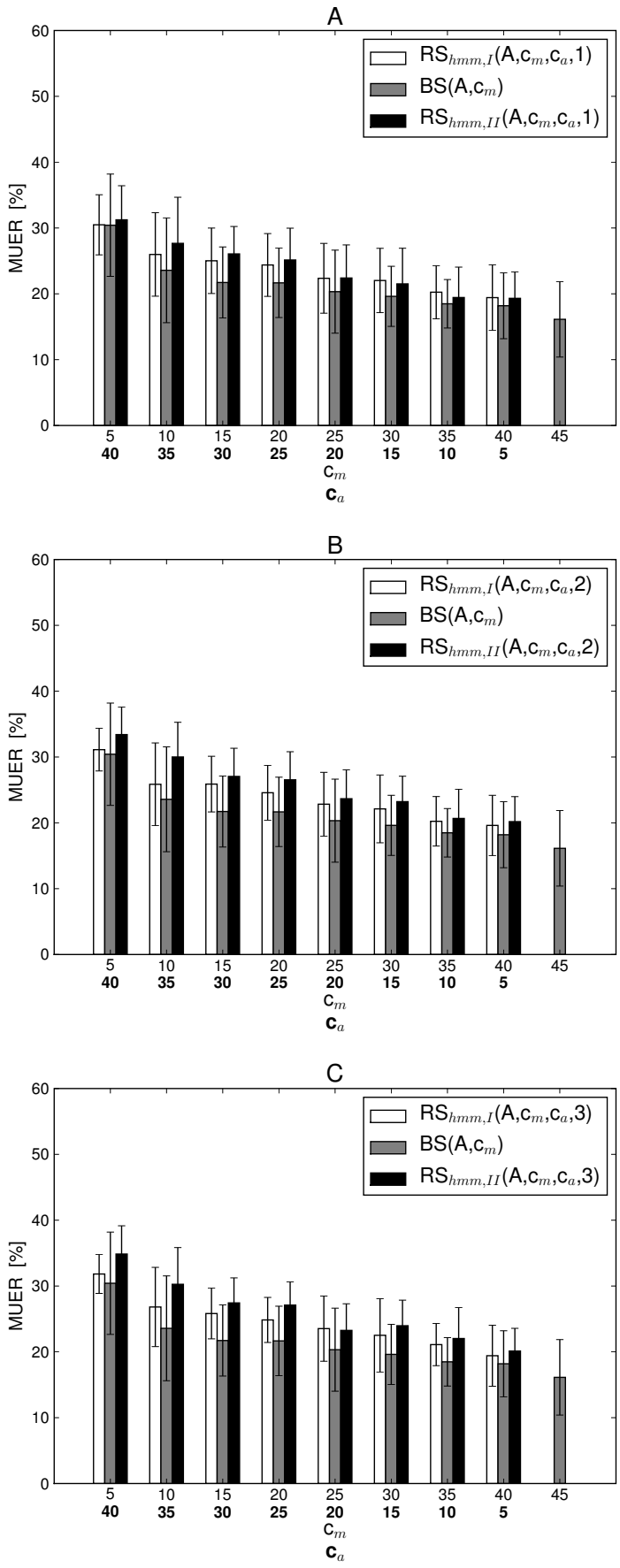

Figure 5.6: The performance of the recognition systems using HMM segmentation in configuration I and II and data set A. In A) 1, B) 2, and C) 3 segmentation and re-training iterations are applied.

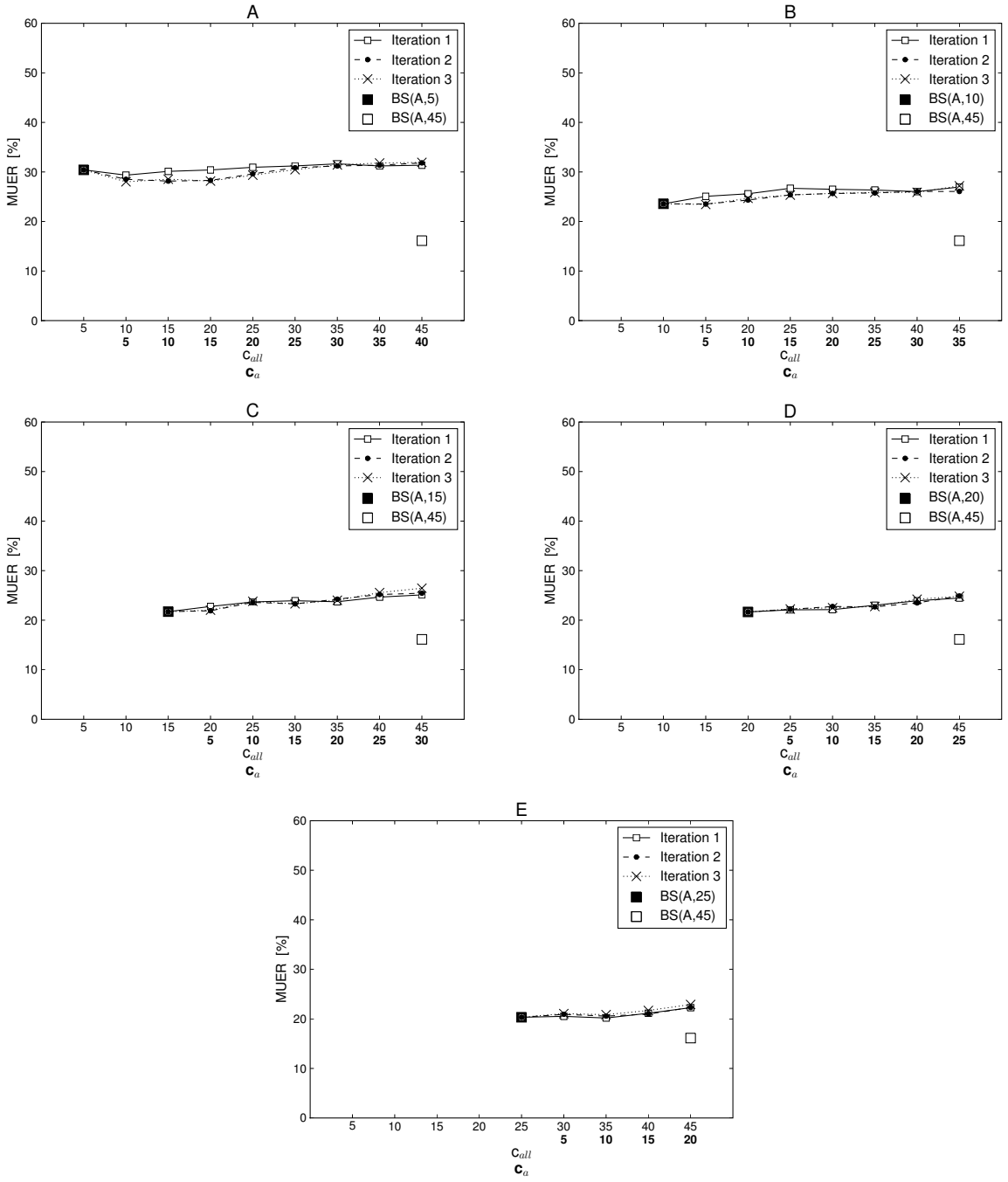

Figure 5.7: The performance of the recognition system using HMM segmentation in configuration III. c_{all} is the number of motion sequences of each complex motion type used for training. c_a is the number of HMM-segmented motion sequences used for re-training. In each plot the performance using 3 different segmentation and re-training iterations is illustrated. In A) $BS(A, 5)$, B) $BS(A, 10)$, C) $BS(A, 15)$, D) $BS(A, 20)$, and E) $BS(A, 25)$ are used for the initial segmentation task.

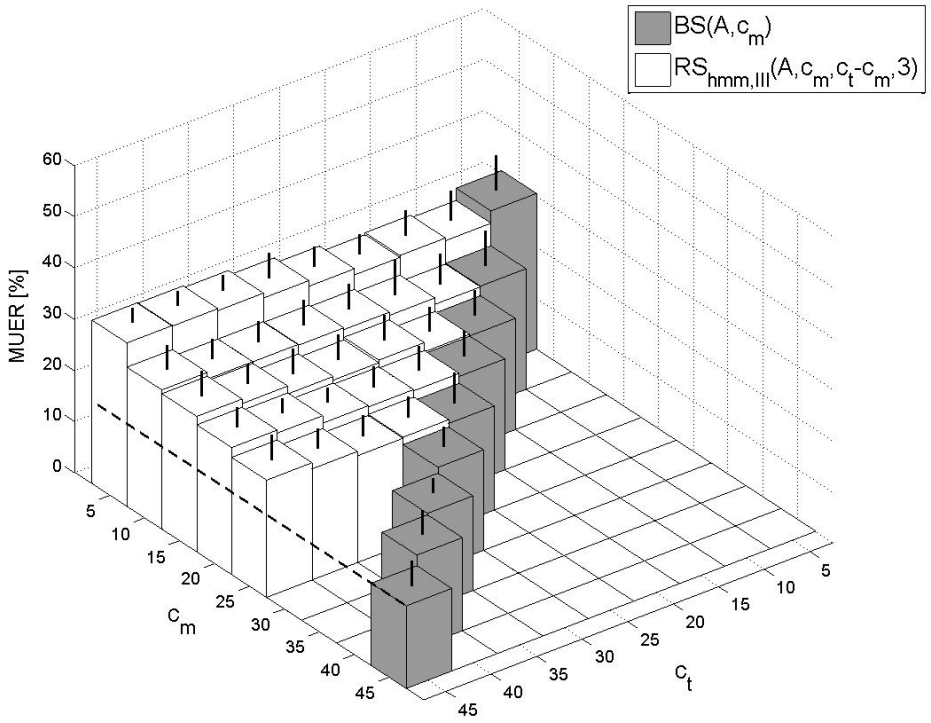

Figure 5.8: The performance of the recognition systems using 3 segmentation and re-training iterations in configuration III and data set A.

iterations does not improve the recognition systems. Actually, the recognition systems perform better when one segmentation and re-training iteration is applied. The most interesting observation is that the baseline systems always perform better than the resulting recognition systems.

The evaluation of the HMM segmentation in configuration III using data set A is carried out by applying the same baseline systems as in configurations I and II. The remaining data is segmented in several sessions. In each session 5 new sequences of each complex motion type are segmented. The segmentation results are used to re-train the recognition system. In each session 5 segmentation and re-training iterations for improving the quality of the recognition system are applied.

In figures 5.7 A-E the performance of the resulting recognition systems using different baseline systems for the segmentation task are illustrated. As one can see the resulting recognition systems show a small improvement only when the baseline systems $BS(A, 5)$ and $BS(A, 10)$ is applied for the segmentation task. This improvement, however, only remains for a few following segmentation sessions. Otherwise, the performance decreases when more data are segmented and used for re-training. In sessions with more initial training data for the baseline system the performance decreases in every following segmentation session. Using several segmentation and re-training iterations improves the performance slightly. In figure 5.8 an overview of the performance of different recognition systems developed by using different baseline systems and 3 segmentation and retraining iterations is given. The gray bars represent the performance of the baseline systems. The white bars represent the

performance of the resulting recognition systems. The lines on the bars represent the standard deviation of MUER between different cross-validation folds.

The evaluation of the HMM segmentation method in configuration I and II using data set B is carried out by starting with 2 motion sequences of each complex motion type to initialize and train the baseline system. In each following session the number of motion sequences is increased by 2. In the last session 16 motion sequences are used to initialize and train the baseline system. The number of data remaining for segmentation is decreased by 2 starting with 16 and ending with 2 sequences.

The results of the evaluation are shown in figures 5.9 A - C in which the performance of the baseline systems is compared to the performance of the resulting recognition systems. As one can see the resulting recognition system using one segmentation and re-training iteration performs better than the baseline system when 2, 4, 6, 8, and 10 motion sequences are used to initialize and train the baseline system. In the remaining sessions the performance of the baseline systems is slightly better. Using more segmentation and re-training iterations improves the performance slightly. In the first three and in the fifth session the resulting recognition system performs better when configuration II is applied with one segmentation and re-training iteration. This is different when more segmentation and re-training iterations are applied. The segmentation with configuration I performs better in almost all sessions. The best performance gain using configuration I is an absolute improvement of 9.39% MUER obtained by re-training the baseline system $BS(B, 6)$ (32.81% MUER) with 12 additional HMM-segmented motion sequences of each complex motion type; the resulting recognition system $RS_{hmm,I}(B, 6, 12, 3)$ has a performance of 23.42% MUER. Using configuration II the best performance gain is an absolute improvement of 7.1% MUER obtained by re-training the baseline system $BS(B, 8)$ (33.04% MUER) with 10 additional HMM-segmented motion sequences of each complex motion type; the resulting recognition system $RS_{hmm,II}(B, 8, 10, 3)$ has a performance of 25.94% MUER.

The evaluation of the HMM segmentation method in configuration III using data set B is carried out by using the same baseline systems for the segmentation task as in configurations I and II. The remaining training data is segmented in several sessions. In each session 2 new motion sequences are segmented. The segmented data is used to re-train the recognition system. This is repeated for 5 iterations in order to improve the quality of the recognition system.

The results of the evaluation are shown in figures 5.10 A-E and 5.11. It can be seen that in all sessions a certain improvement of the recognition systems compared to the baseline systems exists. There is an obvious performance improvement when at least 2 segmentation and re-training iterations are applied. Applying more than 2 iterations does not lead to a significant performance difference. The best performance gain is an absolute improvement of 9.1% MUER obtained by re-training the baseline system $BS(B, 6)$ (32.81% MUER) with 10 additional HMM-segmented motion sequences of each complex motion type; the resulting recognition system $RS_{hmm,III}(B, 6, 10, 3)$ has a performance of 23.71% MUER.

The HMM segmentation method leads to performance improvements when it is applied on data set B. The best performance improvements are achieved when only a re-training without re-initialization is applied (configuration I and III). For this, only

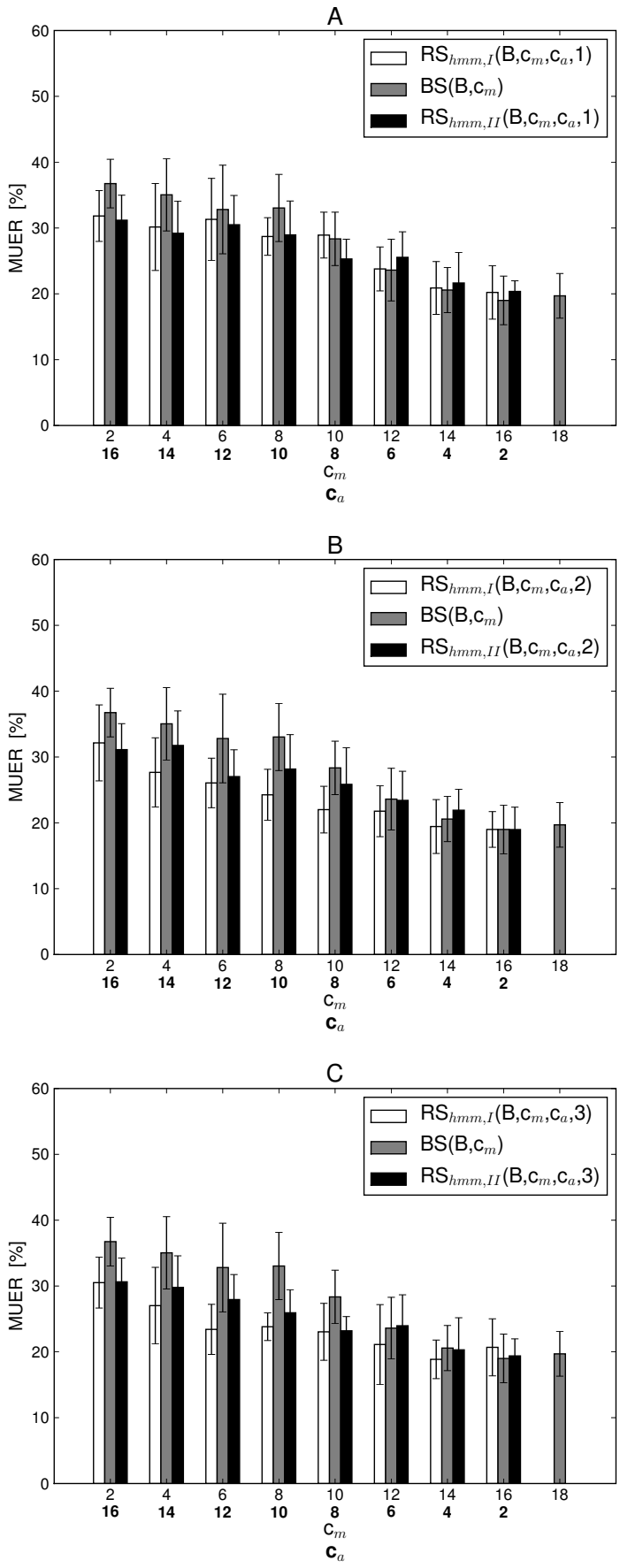

Figure 5.9: The performance of the recognition systems using HMM segmentation in configuration I and II and data set B. In A) 1, B) 2, and C) 3 segmentation and re-training iterations are applied.

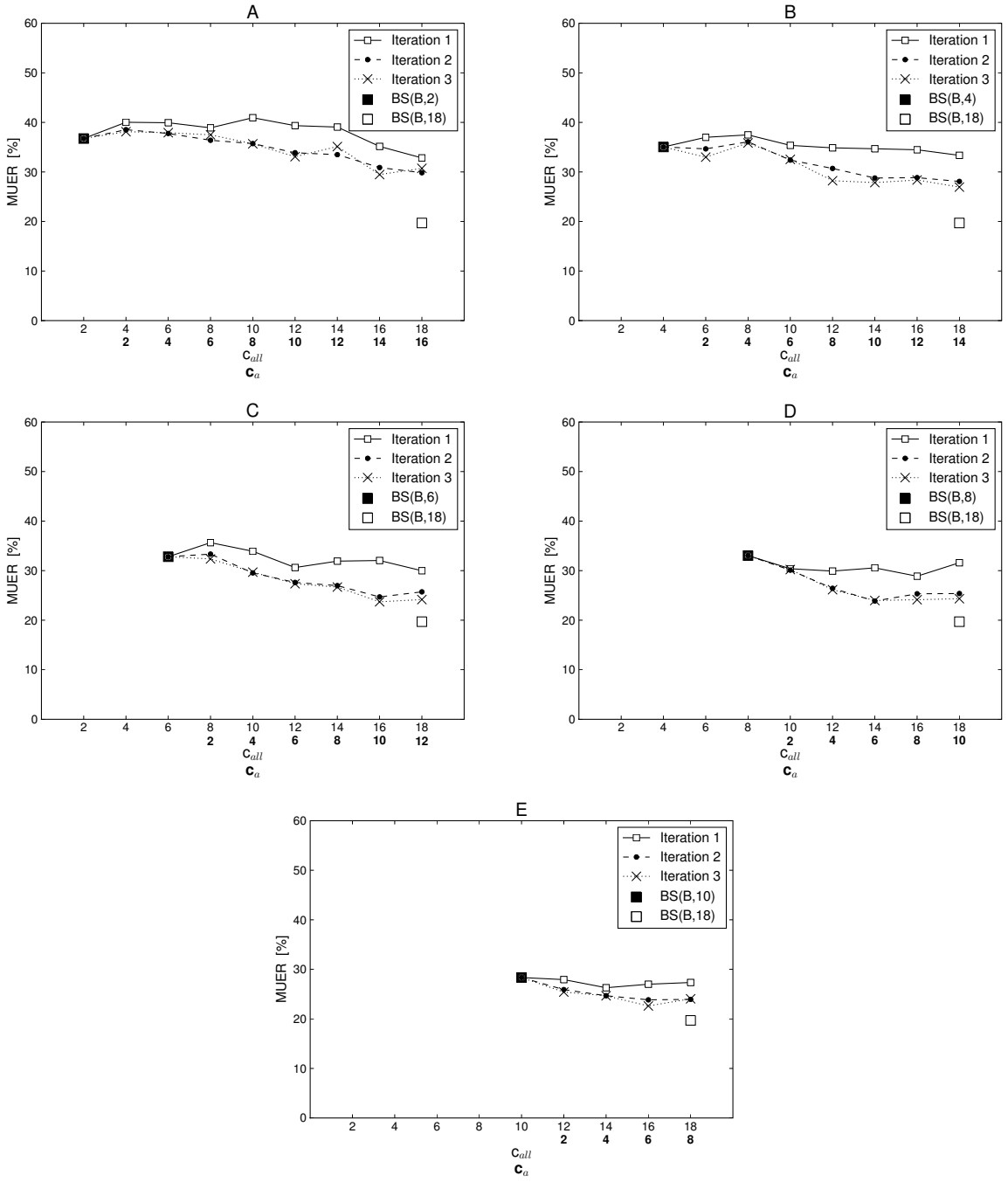

Figure 5.10: The performance of the recognition system using HMM segmentation in configuration III. c_{all} is the number of motion sequences of each complex motion type used for training. c_a is the number of HMM-segmented motion sequences used for re-training. In each plot the performance using 3 different segmentation and re-training iterations is illustrated. In A) $BS(B, 2)$, B) $BS(B, 4)$, C) $BS(B, 6)$, D) $BS(B, 8)$, and E) $BS(B, 10)$ are used for the initial segmentation task.

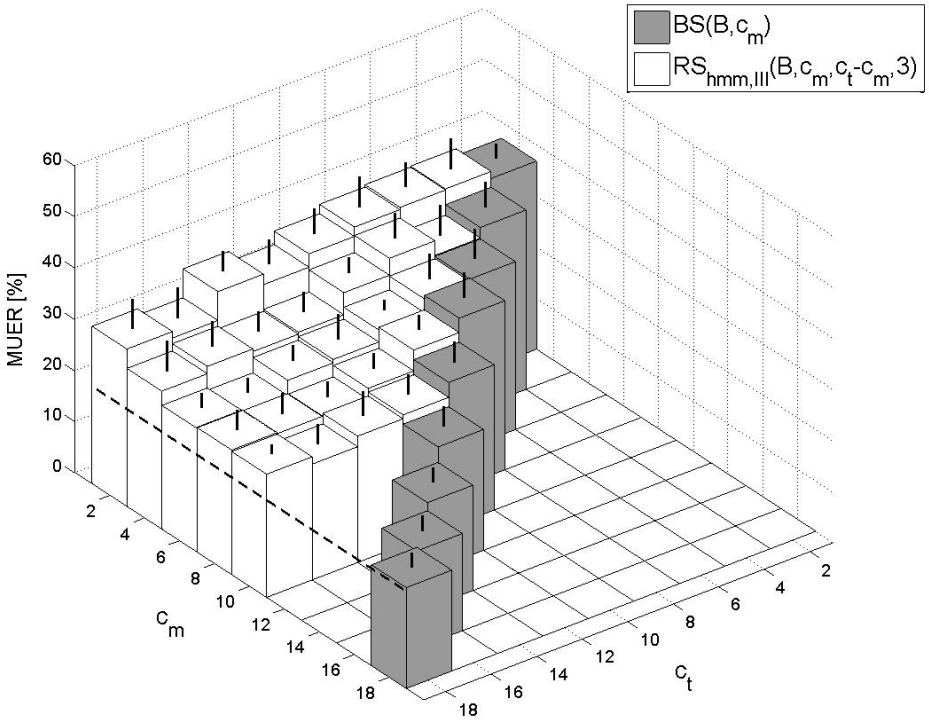

Figure 5.11: The performance of the recognition systems using 3 segmentation and re-training iterations in configuration III and data set B.

the automatically computed transcriptions for motion sequences are needed. However, the HMM-based segmentation method leads to a reduction of the performance when it is applied on data set A. In order to explain this behavior a further analysis is carried out in which the recognition performance with respect to individual motion units in data set A is measured.

In figure 5.12 F-score values provided by two different systems are illustrated. The black colored dots represent the F-score values provided by the the baseline system $BS(A, 5)$. The white squares represent the F-score values of the recognition system after a re-training using 40 additional HMM-segmented motion sequences of each complex motion type. It can be noticed that the F-score values of almost all motion units are higher when HMM-segmented data is used for re-training. Only the F-score values of a few motion units either remain equal or they decrease slightly. The greatest decrease of 0.26 from F-score 0.91 to 0.65 is that of the motion unit with index 36 which is the motion unit ”rest position”.

Motion unit ”rest position” in data set A represents the pauses at the beginning and the end of a complex motion, but it also represents the artificially added short pauses as mentioned in section 3.2. Therefore it occurs very often in data set A, which means that a decrease of its F-score value has a great negative impact on the overall performance of the recognition system. The low F-score value of 0.65 is derived from a precision value of 0.99 and a recall value of 0.49. Such a high precision value implies that almost all ”rest positions” that occur in the hypotheses are recognized correctly. The low recall value indicates that the correctly recognized

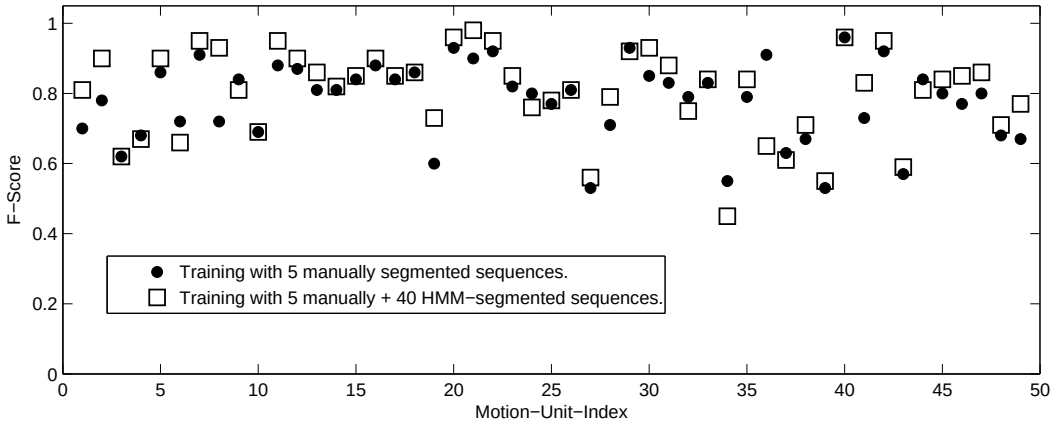

Figure 5.12: A comparison between F-score values of motion units provided by a recognition system trained with only 5 manually segmented motion sequences of each complex motion type and a recognition system trained with 5 manually and 40 HMM-segmented motion sequences of each complex motion type.

"rest positions" are about half the existing ones. These facts lead to the conclusion that the hypotheses provided by the recognition system only contain half of the existing "rest positions". This can be seen in figure 5.13 in which the occurrence frequency of all motion units in the reference sequences as well as in the hypotheses of two different recognition systems is illustrated. In 5 reference sequences of each complex motion type the mean occurrence frequency (10 cross-validation folds) of "rest position" amounts to 162.3. A similar frequency of 153 is found in hypotheses provided by a baseline system trained with 5 motion sequences of each complex motion type. The frequency, however, extremely decreases to 81.2 in hypotheses provided by the baseline system retrained by using 40 additional HMM-segmented motion sequences of each complex motion type. This leads to a high number of deletions during the calculation of the MUER.

A high number of deletions can arise because the respective HMM is somehow defective. This can happen when an HMM is mis-trained by using false or unsuitable training data. Consequently, the HMM-based segmentation of motion unit "rest position" was analyzed by comparing it to the manual segmentation.

The comparison is carried out by using two samples of segmented data. The first sample (sample I) contains 40 HMM-segmentations of each complex motion type computed by a baseline system initialized and trained with 5 motion sequences of each complex motion type. The second sample (sample II) consists of 35 HMM-segmentations of each complex motion type computed by using a baseline system initialized and trained with 10 motion sequences of each complex motion type. The segmentation process is carried out by using segmentation configuration I. The performance obtained using sample I is 32.7% MUER and that obtained using sample II accounts 26.7%.

The comparison of the HMM-segmentations in samples I and II and the respective manual segmentations is carried out frame-wise. Each frame of the motion unit "rest position" in the HMM-segmentation is compared to the respective frame in the manual segmentation. This way the matching rate of the motion unit "rest position"

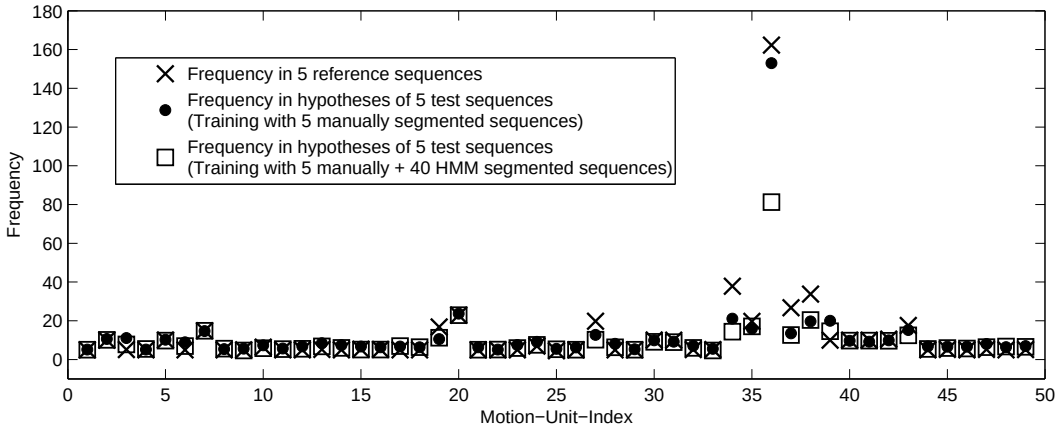

Figure 5.13: The frequency of motion units in reference sequences and in two different types of hypotheses. The first type is provided by the baseline system trained with 5 manually segmented sequences of each complex motion type from data set A. The second type is provided by the recognition system re-trained by using additional 40 HMM-segmented sequences of each complex motion type.

in the HMM-segmentation and all motion units in the manual segmentation can be determined. In addition, the average lengths of the motion unit "rest position" in the HMM-based and in the manual segmentation are computed and compared.

In figure 5.14 the results of the frame-wise matching is illustrated. One can see that only 61.7% of the "rest position" frames in the HMM-segmentation actually match the "rest position" frames in the corresponding manual segmentations. The remaining 38.3% of frames are distributed among the other motion units. The average length of the motion unit "rest position" in the HMM-segmentations is 18.1 frames which increases compared to the average length of 12.2 frames in the manual segmentation. This is a growth of 148%. A similar result is obtained by the comparison using sample II. In this case the matching rate of the motion unit "rest position" accounts 64.1% and the average length of the HMM-segmentations accounts 16.6 frames. In this case the growth of average length amounts to 136%.

In order to show that the matching rate calculation is principally meaningful the matching rate for a motion unit with a high F-score is calculated. For this calculation the motion unit with index 40 (see figure 5.12) is chosen which is the motion unit "take bowl". The matching in conjunction with sample I provides a matching rate of 98.9%. The length of the motion unit decreases from 73.5 frames in the manual segmentation to 56.8 frames in the HMM-segmentation. This is a negative growth to 77%. The matching result associated with sample II is almost identical. The matching rate accounts 98.7%. The average length of the motion unit in the HMM-segmentations accounts 67.1 (91%) which is closer to the average length in the manual segmentation.

The above mentioned analyses confirm the assumption that the HMM of the motion unit "rest position" is mis-trained by using defective and unsuitable segmentations. More than one third of the frames in the HMM-segmentations of "rest position" belong to other motion units. Consequently, the resulting HMM does not represent a pure version of the motion unit "rest position" and therefore the recognition sys-

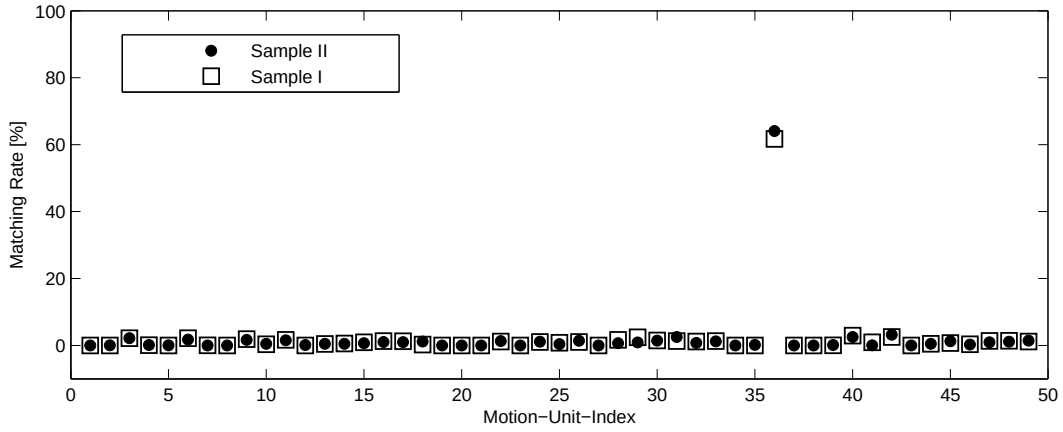

Figure 5.14: The matching rate between the automatically obtained motion unit "rest position" and all motion units in the manual segmentation. Motion unit index 36 represent motion unit "rest position" in the manual segmentation.

tem does not recognize it appropriately. This leads to a great number of deletions and to a significant decrease of the overall performance of the recognition system. Nevertheless, the increased F-score values of many other motion units after the retraining using HMM-segmentations (see figure 5.12) and the results of the evaluation using data set B imply that the HMM-based segmentation method is an appropriate automatic segmentation method and it has a great potential to be applied as a semi-automatic segmentation method.

5.2 Unsupervised Automatic Segmentation

In the previous section an approach for supervised model-based automatic segmentation was introduced. In this section an unsupervised automatic segmentation approach based on PCA is presented and evaluated.

Unsupervised motion segmentation can be divided into two tasks. The first one deals with finding segment boundaries in motion sequences in order to divide complex motions at a high level of abstraction into several primitive motion units at a lower abstraction level.

Labeling is the second task required when unsupervised segmentation is applied. The labeling task is concerned with the assignment of detected motion segments to different classes of primitive motion units.

In the next sections the two above mentioned tasks are discussed in more detail.

5.2.1 PCA-based Automatic Segmentation

The most common approach for unsupervised automatic segmentation is to set segment boundaries in data trajectories where local minima and/or maxima or where so called zero-crossings are found. The main problem with this approach is the multi-dimensionality of motion data. The segment boundaries have to be set at the same time instance for all features, but the local minima, maxima, and zero-crossings are shifted relatively to each other. One approach is to segment the trajectories of

all data features separately and merge those which reside in the same interval. As mentioned in [FMJ02] this approach has proven to be problematic because segment overlapping cannot be avoided. In other approaches a single segmentation feature is derived out of all data features by either applying a weighted sum of all data features or by calculating new quantities, e.g. a centroid of different joints, or by choosing or extracting a single feature out of all data features that is most representative for the whole data. The derived segmentation feature can then be explored for minima, maxima, or zero-crossing.

Typically, in large feature spaces the data variability in the feature trajectories is only found in a specific low-dimensional subspace dependent on the motion type. Finding the subspace, where the most data variability takes place, would reduce the multi-dimensional problem of segmentation because only a small number of features (in the low-dimensional subspace) have to be taken into account for deriving a new segmentation feature. This can be achieved by applying feature space reduction or rather feature extraction techniques. One of the most effective approaches for feature space reduction and feature extraction is the PCA.

The PCA is a transformation of the feature space into a new space where the new features are least correlated and the variance is maximized. In his book [Bis06], Bishop defines the PCA in two ways:

1) in the maximum variance definition the PCA is described as an orthogonal projection of data onto a lower dimensional linear vector space where the variance of projected data is maximal.

2) in the minimal error definition the PCA is described as a linear projection, where the projection error is minimal; the projection error is defined as the average square distance of a data point and its projection.

To explain the first definition as stated in [Bis06] in more detail consider the data set

$$x = \{x_1, x_2, \ldots, x_n\} \tag{5.2}$$

with dimensionality d, and a projection vector u ($u^T u = 1$) which projects the data onto a one-dimensional space ($d' = 1$). The direction of the one-dimensional space is defined in dependence of the direction of the d-dimensional projection vector u. The result of the projection is given by the scalar values

$$x' = \{u^T x_1, u^T x_2, \ldots, u^T x_n\} \tag{5.3}$$

The mean of x' is

$$\bar{x}' = u^T \bar{x}, \tag{5.4}$$

where

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{5.5}$$

is the mean of data set x. The variance of the projected data is given by

$$var(x') = \frac{1}{n}\sum_{i=1}^{n}(u^T x_i - u^T \bar{x})^2 = u^T C u \tag{5.6}$$

where C is the covariance matrix of data set x defined by

$$C = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(x_i - \bar{x})^T. \tag{5.7}$$

By using a Lagrange multiplier λ with the constraint $u^T u = 1$, the projected variance $u^T C u$ can be maximized with respect to u by maximizing

$$u^T C u + \lambda(1 - u^T u). \tag{5.8}$$

This can be achieved by setting the derivative of (5.8) with respect to u equal to zero, where a stationary point is found when

$$Cu = \lambda u \tag{5.9}$$

which means that u must be an eigenvector of C. By left-multiplying (5.9) by u^T, the variance is given by

$$u^T C u = \lambda \tag{5.10}$$

This means that the projection variance is maximized when the projection vector u equals the eigenvector with the largest eigenvalue λ. This eigenvector is called the first principal component.

To consider the case, where $d > d' > 1$, the optimal linear projection for which the projected data is maximized, is given by the d' eigenvectors $u_1, \ldots, u_{d'}$ of the data covariance matrix C that correspond to the d' largest eigenvalues $\lambda_1, \ldots, \lambda_{d'}$.

The first extracted principal component or a linear combination of a few first components can be applied to derive a segmentation feature which can be analyzed for segmentation criteria like minima, maxima, or zero-crossings.

A segmentation criterion has to be selected depending on the information type comprised in the motion data. For motion data which comprises velocity information of joint angles or markers, the zero-crossing segmentation criterion might be more appropriate because a velocity equal zero indicates a short time non-activity of the joint or marker. Whereas for motion data which comprise marker positions or joint angle information a minima and/or maxima segmentation criterion might be more convenient because a minimum or maximum value of a marker position or a joint angle indicates a maximum deflection or a turning point of a certain motion.

In this work the first principal component is used to derive a segmentation feature since it can be assumed that it already contains enough information to carry out a

meaningful segmentation process. The segmentation criterion is to set a segmentation boundary where relevant local minima and maxima are found.

Consider a complex motion sequence x as follows:

$$x = x_1 x_2 \dots x_t, \tag{5.11}$$

where x_i is frame i and t is the motion sequence length. The motion frames are considered as data points in order to calculate the PCA. The first component u_1 is applied as projection vector in order to derive a one dimensional segmentation feature. The trajectory x' of the segmentation feature is given as follows:

$$x' = u_1^T x = x'_1 x'_2 \dots x'_t. \tag{5.12}$$

In figure 5.15 the feature trajectories of a motion sequence of complex motion type "rolling pastry" are plotted. The motion sequence projected by using the first component as projection vector is illustrated in figure 5.16.

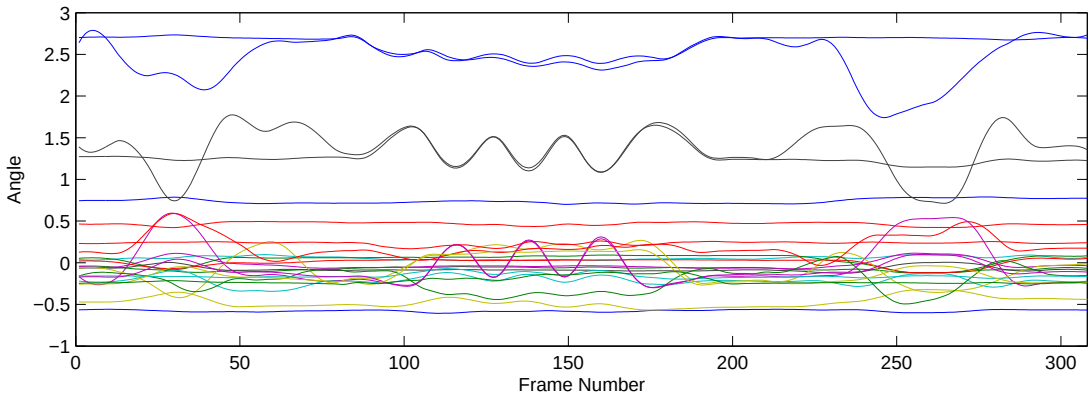

Figure 5.15: A motion sequence of type "rolling pastry". It can be seen that only a few features are involved in the motion activity.

In the first step of the segmentation algorithm the local minima and maxima in x' are localized by using the sliding window technique. A sliding window of a specific length is moved over x' to localize maxima and minima where the value in the middle of the window is either greater than all neighboring values in the window for a maximum or less than the neighboring values for a minimum. The sliding window technique works fine only if smooth data trajectories are available. Hence, either a smoothing algorithm has to be applied to smooth the data trajectories or other algorithms for localizing minima and maxima have to be used. In this work the Savitzky-Golay smoothing filter [Orf96] implemented in MATLAB is applied to smooth the data trajectories. The set of found local minima and maxima is given as follows:

$$M = \{x'_i | x'_i \textit{ is a local minimum or a local maximum}\}, \tag{5.13}$$

where M is sorted and the values are alternately arranged, i.e. a minimum is followed by a maximum and vise versa. In figure 5.16 an example of detected local minima and maxima in the segmentation feature trajectory is illustrated.

In the second step segments between a minimum and a maximum or vice versa are localized. The absolute value of the minimum-maximum difference has to exceed a threshold value θ_s which is dependent on the range r of all feature values. The range r of feature values x' is given by

$$r = max(x') - min(x') \tag{5.14}$$

The threshold is given by the function

$$\theta_s(a) = \frac{r}{a}, \tag{5.15}$$

where a has to be chosen appropriately.

A segment is detected when

$$|M(i) - M(i+1)| \geq \theta_s. \tag{5.16}$$

In figure 5.17 an example of segment detection is illustrated.

This kind of segment detection leads to gaps between the detected segments. In the third step of the segmentation algorithm, the gaps among the found segments are either closed or considered as segments. For this, a second threshold value θ_g has to be chosen. If the gap is greater than θ_g then the gap is considered as a segment, whereas if the gap is smaller than θ_g the boundaries of the neighboring segments are merged to close the gap by setting the new boundary at a previously found maximum or minimum that is as close as possible to the center of the gap.

In order to avoid the building of small segments a third threshold θ_l for the segment length has to be applied. The segmentation algorithm removes each segments with a length less or equal θ_l. Figure 5.18 illustrate the boundary detection of the motion sequence plotted in figure 5.15

The result of the segmentation algorithm is a set B of m boundaries which are given as follows:

$$B = \{b_1, b_2, \ldots, b_m | 1 \leq b_i < b_j \leq t\} \tag{5.17}$$

The PCA-based segmentation algorithm is formally defined as follows:

1. Calculate the first component u_1 of motion sequence x using PCA
2. Compute the segmentation feature x' by projecting x using u_1 as projection vector
3. Find minima and maxima in x'
4. Find relevant segment boundaries among the found minima and maxima
5. Update segment boundaries by removing gaps between found segments
6. Update segment boundaries by removing small segments

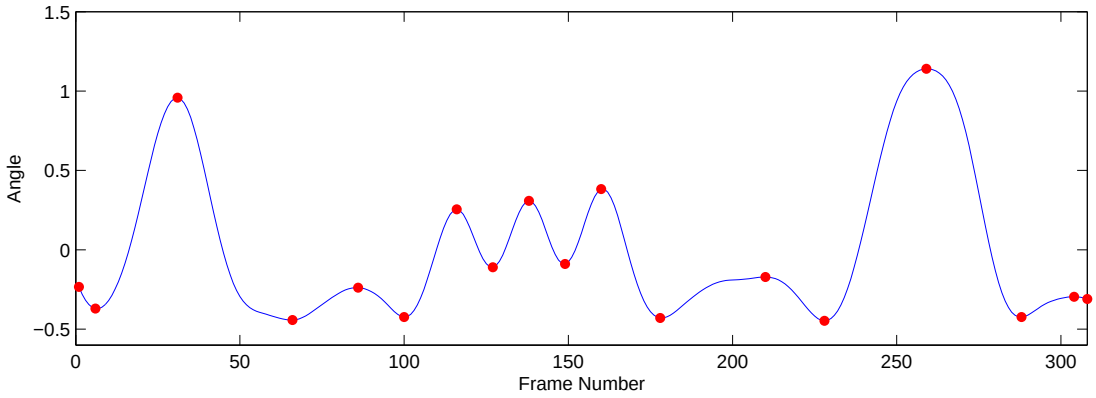

Figure 5.16: The projection of complex motion "rolling pastry" using the first principal component as projection vector. The first component is computed by using the motion data plotted in figure 5.15. The black dots are local minima and maxima detected by applying the sliding window technique.

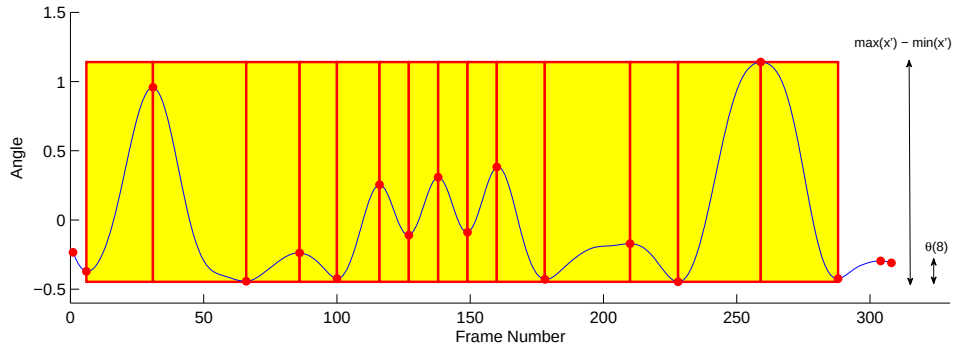

Figure 5.17: Segment detection using threshold $\theta_s(8)$. As can be seen all segments are detected except the first and the last two ones which do not satisfy the threshold condition.

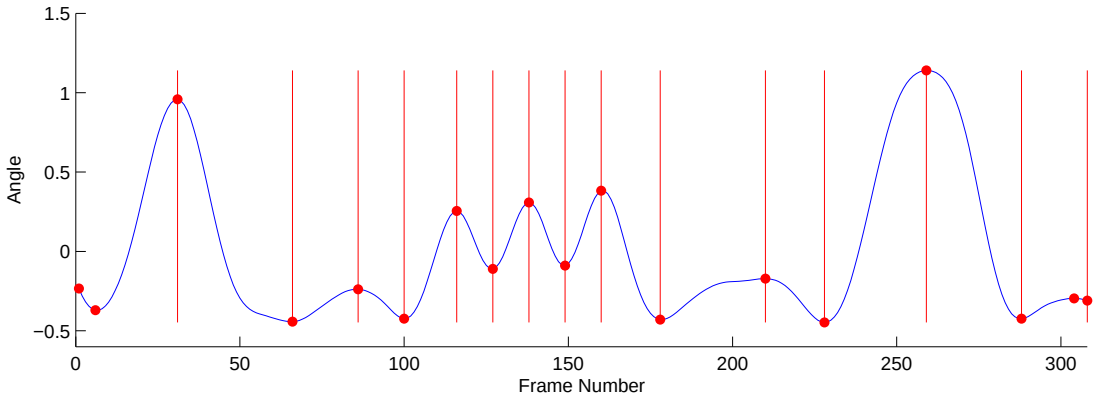

Figure 5.18: Boundary detection after removing gaps and small segments.

5.2.2 Labeling of Motion Unit Segments

In the previous section a method for detecting segment boundaries in motion data trajectories was introduced. However, boundary detection is only one of two tasks required by the segmentation process. In the second task the segments among the detected boundaries are labeled in order to be applied as training data. Labeling in this context means the assignment of detected segments to certain segment categories or rather segment classes which contain similar kinds of motion segments. The distinct segment classes represent the distinct kinds of motion units. The assignment of segments to distinct segment classes can be achieved by applying a cluster analysis.

Cluster analysis is known as an unsupervised learning problem which can be used to organize a set of objects into groups or classes each containing a subset of objects that are somehow similar. The similarity between objects is defined by a similarity or a distance measure which depends on the task, the type of objects, and the feature space. The definition of the distance measure has a big impact on the effectiveness of the clustering algorithm.

There are several options for defining a distance measure for human motion data segments, e.g. dynamic time warping or probabilistic distance measures. These, however, increase the development and computational effort compared to simpler distance measures like the Euclidean distance. Moreover, it is not guaranteed to get better results if complex distance measures are applied. Hence, in this work the Euclidean distance measure in conjunction with a segment length normalization is applied.

Consider a set S of n motion data segments which have to be clustered

$$S = \{S^1, S^2, \ldots, S^n\}, \tag{5.18}$$

where S^i is the motion segment i in set S consisting of t_i frames.

A motion data segment S^i of length t_i can be described as follows:

$$S^i = (f_1^i f_2^i \ldots f_d^i)^T, \tag{5.19}$$

where f_j^i is the trajectory of feature j in segment i and d is the number of features in the motion data. The feature trajectory f_j^i can be given as follows:

$$f_j^i = x_{j1}^i x_{j2}^i \ldots x_{jt_i}^i, \tag{5.20}$$

where x_{jk}^i is the value of feature j at time k in segment i.

To calculate the Euclidean distance of two Segments S^a and S^b a length normalization has to be applied first. This is achieved by changing the length of one segment to fit the length of the other segment. This is done by either adding interpolated data points to the trajectories of the data segment if it has to be enlarged or by removing data points from the trajectories of the data segment if it has to be downsized. After the normalization of the segment length to t frames the distance of

each feature trajectory f_j^a in segment S^a to the according feature trajectory f_j^b in segment S^b can be calculated as follows:

$$D(f_j^a, f_j^b) = \sqrt{\sum_{k=1}^{t}(x_{jk}^a - x_{jk}^b)^2}. \tag{5.21}$$

Since it is not needed for distance comparison the root of the summation in (5.21) is not calculated in order to minimize the computational effort. Thus, the distance measure is then defined as follows:

$$D(f_j^a, f_j^b) = \sum_{k=1}^{t}(x_{jk}^a - x_{jk}^b)^2. \tag{5.22}$$

In order to calculate the distance between the two segments S^a and S^b the distances of all trajectories of segments S^a and S^b are summed as follows:

$$D(S^a, S^b) = \sum_{j=1}^{d} D(f_j^a, f_j^b) \tag{5.23a}$$

$$= \sum_{j=1}^{d}\sum_{k=1}^{t}(x_{jk}^a - x_{jk}^b)^2. \tag{5.23b}$$

After defining a distance measure for motion data segments, an appropriate clustering algorithm has to be found. There are several algorithms which can be applied for clustering. One of the most known and most applied clustering algorithms is the k-means clustering algorithm.

The k-means algorithm is a simple and at the same time an effective clustering algorithm. The main idea behind it is to determine a number of clusters or classes a priori and represent each of them by a centroid in the feature space determined either manually or randomly. Each object in the set is then assigned to a centroid with the least distance according to the defined distance measure. The centroid of each cluster is then recalculated to be centered among the objects in the respective cluster. The last two steps are repeated for a specific number of iterations or until no changes in the recalculation of centroids are noticed.

The recalculation or update of the centroid location is usually done by calculating the mean of objects in the respective cluster. In this work the location update of the centroids is achieved in three steps. In the first step the length of the new centroid is calculated to become the average length of all segments in the respective cluster. In the second step the length of all segments in the cluster is normalized according to the new length of the centroid. Finally, in the third step, the mean of all segments is calculated by summing the trajectories of all segments in the respective dimensions and divide the sum by the number of segments. The mean of all segments becomes the new centroid of the respective cluster.

In order to avoid the underrepresentation of some motion units in the recognition system the number of clusters is updated at the end of the clustering task. Clusters containing a segment number smaller than a defined threshold θ_C are discarded. The segments in the discarded clusters are assigned to the remaining clusters according to the distance measure mentioned above.

The k-means algorithm applied in this work is formally defined as follows:

1. Define the number of clusters n
2. Define the number of iterations i
3. Define threshold θ_C
4. Choose n segments out of the segment set randomly to become the centroids of the clusters
5. Assign each segment to the nearest centroid according to the defined distance measure
6. Update the location of each centroid by calculating the mean of segments in the respective cluster
7. Repeat steps 5 and 6 i times
8. For all clusters: if the amount of segments in the cluster is smaller than θ_C then discard the cluster and assign segments to the remaining clusters

5.2.3 Experiments

In section 5.2 an unsupervised automatic segmentation approach for human motion data was introduced. In this section the experiments carried out to evaluate this segmentation approach are presented. The experiments are conducted by using data set A and B.

For each data set (A and B) the experiments are conducted in three different phases. In the first phase the segmentation method introduced in section 5.2.1 is applied to find segmentation boundaries within the complex motion sequences. In the second phase the motion segments between the detected boundaries are labeled by using the clustering method presented in section 5.2.2. Hence, the segmentation task is completed by segmenting the complex motion sequences and by assigning a specific motion unit label to each motion segment. In the third and last phase the HMM-based recognition system introduced in section 4.2 is evaluated by using the segmented data provided from phases one and two.

The first phase is carried out in different experimental sessions. For each data set A and B 6 different experimental sessions are conducted. In each session a different threshold value $\theta_s(a)$ is used by setting different values for a in (5.15). The values used for a are 4, 6, ..., 14. Threshold θ_g is set to 20 and threshold θ_l is set to 10.

In order to start the second phase, namely the clustering phase, the number of initial clusters has to be specified first. In this work the number of clusters is chosen to

a	Motion Unit Count	Total Segment Count	Average Segment Length	MUER (%)	σ (%)
4	39	4346	34.8	26.29	4
6	43	4586	32.9	26.85	3.26
8	41	4686	32.2	25.52	4.47
10	44	4747	31.8	27.23	4.53
12	44	4817	31.4	28.38	4.16
14	45	4862	31.1	27.84	4.41

Table 5.1: The results of the recognition systems using PCA-based unsupervised automatic segmentation of data set A. The evaluation is carried out by using different values of $\theta_s(a)$.

a	Motion Unit Count	Total Segment Count	Average Segment Length	MUER (%)	σ (%)
4	18	801	37.4	43.76	13.47
6	17	857	35	40.78	16.76
8	18	880	34.1	40.91	18.69
10	22	909	33	30	16.13
12	21	921	32.6	34.35	17.67
14	21	934	32.1	31.42	16.7

Table 5.2: The results of the recognition systems using PCA-based unsupervised automatic segmentation of data set B. The evaluation is carried out by using different values of $\theta_s(a)$.

be similar to the motion unit number obtained by the manual segmentation task. Thus, for data set A 50 and for data set B 25 initial clusters are used. The clustering task for data set A is executed with 15 clustering iterations. The clustering task for data set B is executed with 10 iterations.

The third and last phase is carried out by using the HMM-based motion recognition system mentioned above in conjunction with data sets A and B, and the segmentation results obtained by the unsupervised segmentation method. The evaluation is conducted by using a 10-fold cross-validation. The configuration of the recognition system is the same as described in section 4.2.

The recognition systems developed by applying the PCA segmentation method are defined as follows:

$$RS_{pca}(data, c_a, a), \tag{5.24}$$

where *data* is the applied data set, c_a is the number of automatically segmented motion sequences of each complex motion type used to initialize and train the recognition system, and a is the value required to compute threshold $\theta(a)$.

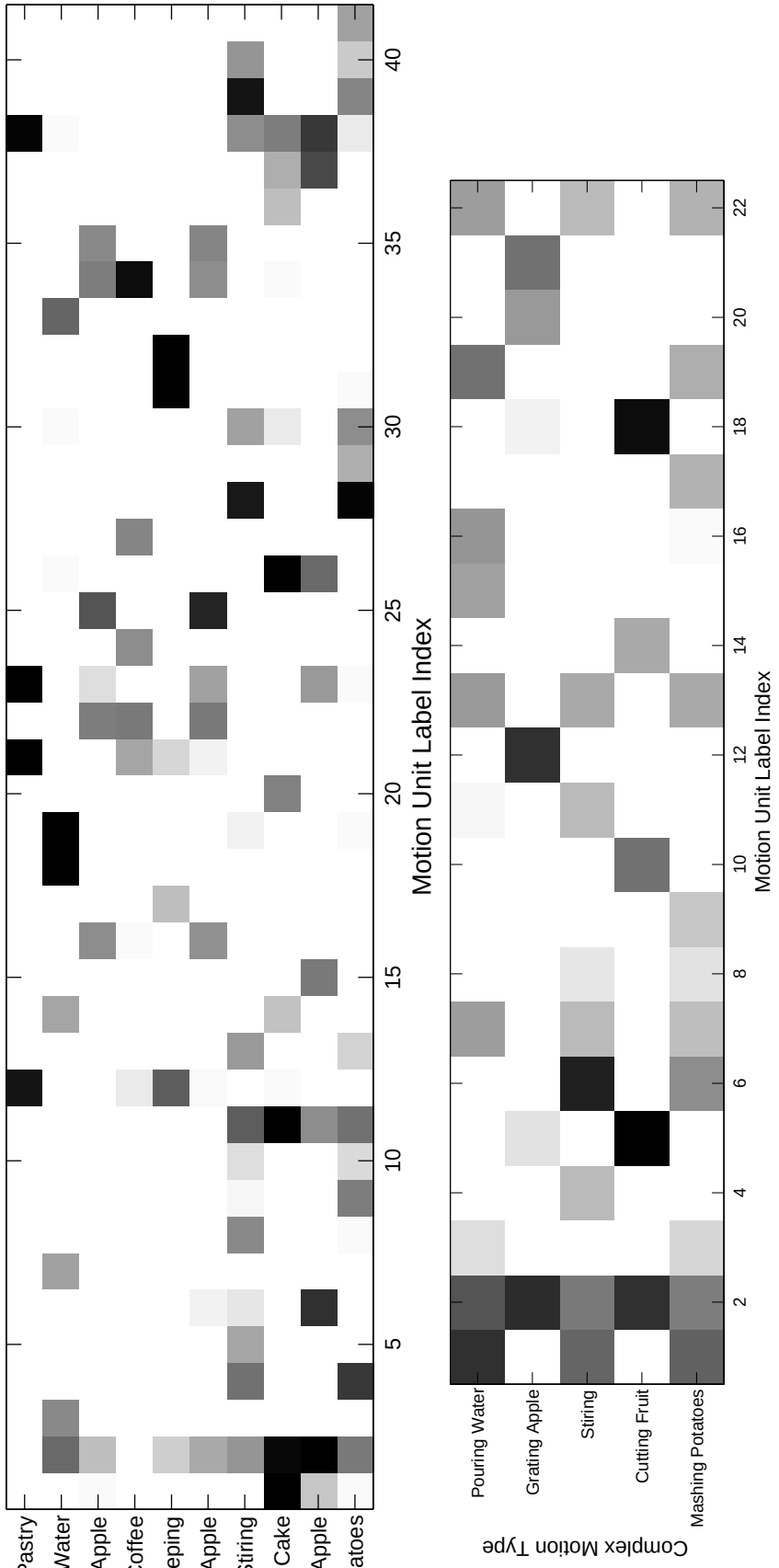

Figure 5.19: The distribution of motion units derived by the PCA-based segmentation among complex motion types. The upper figure shows the distribution for data set A. The lower figure shows the distribution for data set B. The darker the area the more often a motion unit is represented in the respective complex motion type.

The results of the evaluation are listed in table 5.1. As one can see the result does not deviate much. The MUER ranges from 25.52% to 28.38%, the motion unit count ranges between 39 and 45, and the average segment length lies between 31.1 and 34.8 frames. The best recognition performance of 25.52% MUER for data set A is achieved by using $\theta_s(8)$. The initial amount of 50 clusters is reduced to 41 through the clustering process. The segment count in the whole data set is 4686 which is much smaller than the segment count of 6148 in the manual segmentation. Consequently, the average length (32.2 frames) of the automatically obtained motion units is greater than the average length of the manually obtained segments (24.6 frames).

The results of the evaluation with data set B are listed in table 5.2. As can be seen the deviation of the performance using different threshold values $\theta_s(a)$ is greater compared to the deviation of the performance using data set A. The MUER ranges between 30% and 43.76%, the motion unit count ranges from 17 to 22, and the average motion unit length lies between 32.1 and 37.4 frames. The best recognition result of 30% MUER is achieved by using $\theta_s(10)$. The initial cluster number of 25 is reduced to 22 through the clustering task. The segment count in the whole data set is 909 which is not quite far from the segment count of 964 in the manual segmentation. The average segment length of 33 frames is similar to the average segment length in the manual segmentation (31.1 frames).

Figure 5.19 shows the distribution of motion units derived by the unsupervised segmentation task among the different complex motion types. One can see that each complex motion type has a different distribution of motion units. Consequently, a classification of the complex motion type can be achieved by applying a statistical model that contains the distribution of available motion units for every complex motion type. In order to prove this assumption an experiment with a simple classification algorithm is carried out. The algorithm relies on the following score function:

$$score_c(h) = \frac{1}{n} \sum_{i=1}^{n} d_c(h_i), \tag{5.25}$$

where $score_c$ is the score function for the complex motion type c, h is the hypothesis generated by the recognition system consisting of n motion units, h_i is motion unit i in hypothesis h, and $d_c(h_i)$ is the distribution of motion unit h_i for complex motion c. The distribution function is given as follows:

$$d_c(u) = \frac{c_{c,u}}{c_c}, \tag{5.26}$$

where $c_{c,u}$ is the occurrence frequency of motion unit u in the segmentations of complex motion c, and c_c is the number of all motion units in segmentations of complex motion c.

The above mentioned classification algorithm computes the score of a specific hypothesis for every complex motion type. The complex motion type with the highest score is chosen as the classification result. For every cross-validation fold the provided hypotheses by the recognition system can be applied for classification. Consequently, hypotheses for the entire data sets A and B exist in order to be applied

by the classification algorithm. For data set A a classification error rate of 9.6% is achieved. Almost all mis-classifications are due to confusions between complex motion types "grating apple" and "planing apple" which are very similar motions. For data set B a classification error rate of 12% is achieved. Almost all mis-classifications are due to confusions of complex motion type "Mashing Potatoes" with complex motions "pouring water" and "stirring".

In order to compare the motion units obtained by the unsupervised segmentation method to the motion units in the manual segmentation a frame-wise comparison is carried out. For every frame in every motion sequence the corresponding motion unit types in both segmentations are compared. In this way the frame count for each motion unit matching combination can be computed. Consequently, two normalized matrices

$$A = (a_{ij}) = \left(\frac{c_{ij}}{c_i} \cdot 100\% \right) \tag{5.27}$$

and

$$M = (m_{ij}) = \left(\frac{c_{ij}}{c_j} \cdot 100\% \right) \tag{5.28}$$

can be created, where c_{ij} is the count of frames which simultaneously belong to the automatically obtained motion unit i and the manually obtained motion unit j, c_i is the overall frame count of the automatically obtained motion unit i, and c_j is the overall frame count of the manually obtained motion unit j.

The matching frame count in A is normalized by the frame count of the automatically obtained motion units. This means that each row in A contains the matching rate distribution of a single automatically obtained motion unit among the motion units in the manual segmentation. The sum of a row in A equals 100%.

M contains the matching frame count normalized by the frame count of the motion units in the manual segmentation. In this case the columns in M contain the matching rate distributions of the manually obtained motion units among the motion units in the automatic segmentation. The sum of a column in M equals 100%.

In order to apply a matching algorithm to carry out an explicit matching between automatically obtained motion units and manually obtained ones the harmonic mean of the matching rates in matrices A and M is calculated. The harmonic mean is an appropriate measure to calculate the average of rates. The result of the harmonic mean calculation is a matrix H defined as follows:

$$H = (h_{ij}) = \left(2 \cdot \frac{a_{ij} \cdot m_{ij}}{a_{ij} + m_{ij}} \right). \tag{5.29}$$

Figures 5.20 and 5.21 show the matching distribution matrices for data set A and B. It can be noticed that there is no explicit matching of a single automatically obtained motion unit to a single manually obtained one, but there is rather a motion unit multi-matching. Multi-matching in this context means that several motion units in

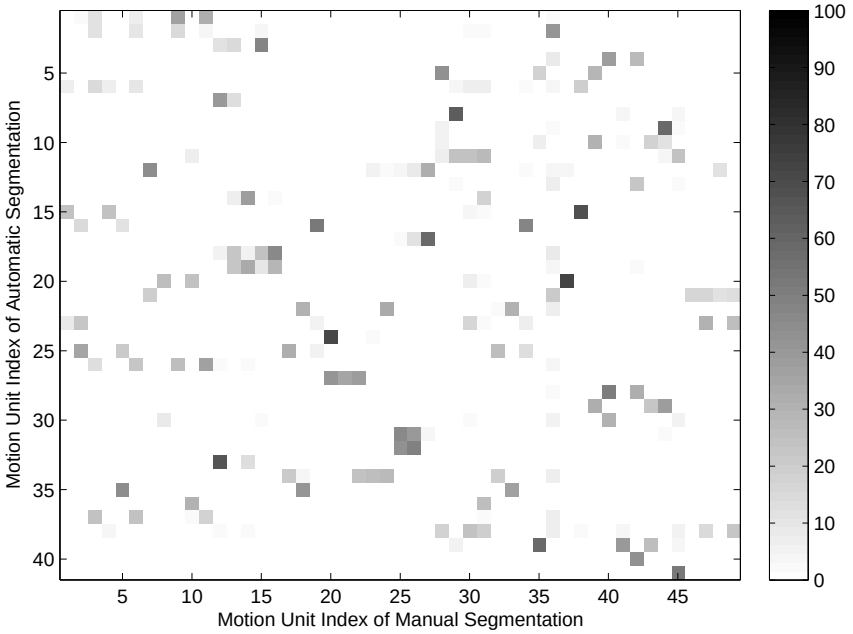

Figure 5.20: The matching between the automatically obtained motion units and the manually obtained ones in data set A. The darker the area the higher is the matching. A black area indicates a matching of 100% whereas a white area indicates a matching of 0%.

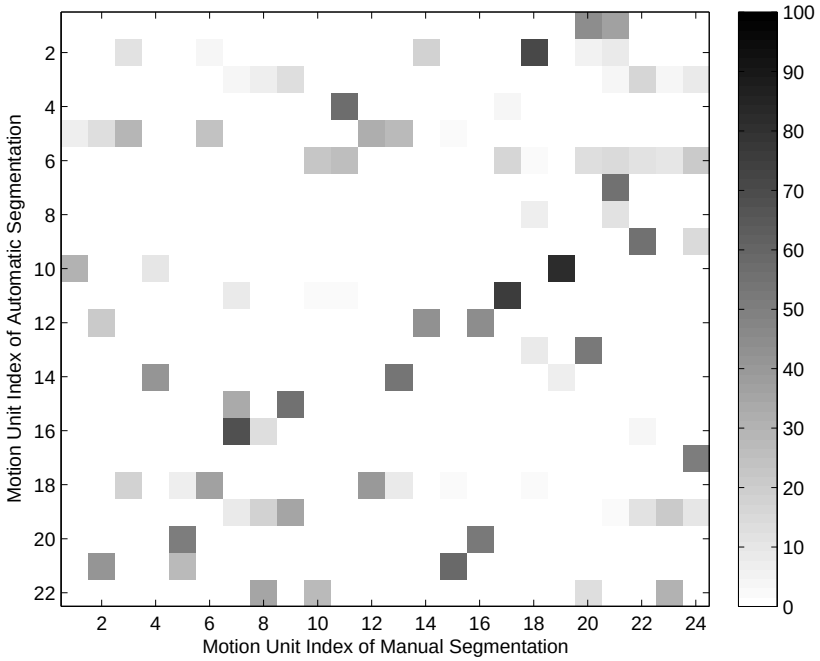

Figure 5.21: The matching between the automatically obtained motion units and the manually obtained ones in data set B. The darker the area the higher is the matching. A black area indicates a matching of 100% whereas a white area indicates a matching of 0%.

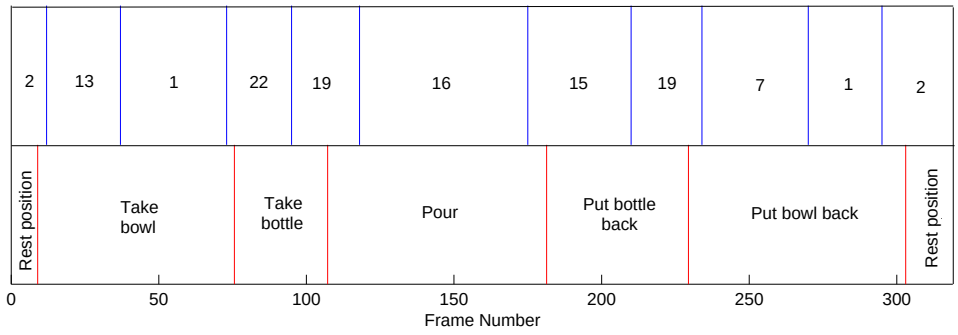

Figure 5.22: A comparison of a PCA-segmentation in the upper part of the figure and a manual segmentation in the bottom of the figure. The numbers in the unsupervised segmentation indicate the index of motion unit labels.

the automatic segmentation can either match a single motion unit in the manual segmentation and/or vice versa.

In the first case two or more motion unit types can match a single motion unit in the manual segmentation. This is due to the segmentation criterion used in the PCA segmentation method. In order to allow a robust segmentation the segmentation criterion is to set segment boundaries where minima or maxima occur in the trajectory of the segmentation feature, i.e. a segment is between a minimum and a maximum. In some cases, however, some motion units in the manual segmentation lie between two maxima or two minima. Each of these motion units corresponds to two neighboring motion units in the automatic segmentation. This can be seen in figure 5.22; the motion unit ”take bowl” in the manual segmentation, for instance, matches the motion units ”13” and ”1” in the automatic segmentation and the same can be noticed with motion unit ”put bowl back” which matches the motion units ”7” and ”1” in the automatic segmentation.

In the opposite case a single automatically obtained motion unit can match several manually obtained motion units. This can be seen in figure 5.22; motion unit ”1”, for instance, match a part of motion unit ”take bowl” and a part of motion unit ”put bowl back”. This is due to the similarity of motion units ”take bowl” and ”put bowl back”. The similarity of the trajectories leads to the assignment of motion units or parts of them to the same clusters by the clustering algorithm.

Due to the impossibility of a one-to-one motion unit matching a double motion unit matching algorithm is applied in order to enable a double matching in each direction. The algorithm allows each motion unit to match at most two motion units in the other segmentation type. In order to avoid meaningless matchings a threshold θ_h for the harmonic mean is applied. Matchings with a higher harmonic mean than θ_h are considered as valid matchings. This way the motion units can have a double matching, only one matching, or no matching at all. The matching algorithm is formally described as follows:

1. Create a one-to-one matching list L containing the matches in H with harmonic mean greater than θ_h
2. Sort L in descending order

Automatic Segmentation Motion Unit Index	Manual Segmentation Motion Unit Label	Harmonic Mean (%)
1	Take bread	36.8
	Put bread back	32.5
2	Rest position	41.9
3	Take glass	48.0
4	Take bowl	37.5
5	Take spoon	43.1
7	Pour	40.6
8	Put spoon back	63.0
9	Take masher	59.3
12	Roll	44.8
	Sweep	31.5
14	Put bottle back	38.5
15	Cut apple	68.0
16	Slice	52.1
	Grate	48.2
17	Sweep	58.5
18	Put glass back	45.5
19	Put bottle back	33.3
20	Cut bread	72.5
22	Put grinder back	34.1
	Put Slicer back	31.1
23	Take rolling pin	30.2
24	Grind	71.7
25	Take apple right hand	35.8
	Take Slicer	31.3
26	Put bread back	36.7
27	Grind	41.8
	Take grinder	37.9
28	Take bowl	50.2
	Put bowl back	32.3
29	Take masher	38.5
	Grasp bowl	31.5
31	Take broom and dustpan	46.0
	Put back broom and dustpan	40.5
32	Put back broom and dustpan	48.5
	Take broom and dustpan	43.6
33	Pour	66.8
35	Put apple back right hand	44.4
	Put Slicer back	40.7
39	Stir	58.7
	Release bowl	40.2
40	Put bowl back	43.4
41	Put masher back	52.2

Table 5.3: The results of the double matching between automatically and manually obtained motion units in data set A.

3. For each matching i in L

   (a) If both motion units in i have less than two valid matchings then

      i. Set matching i as valid

      ii. Increase the number of valid matchings for both motion units in i by one

4. Set all valid matchings as matching result

Tables 5.3 and 5.4 show the results of the matching algorithm applied to the segmentations of data set A and B. The matching applied to the segmentations of data set A results in matching 31 of 41 (75.6%) automatically obtained motion units with 32 of 49 (65.3%) motion unit in the manual segmentation. The total number of valid matchings is 43. The discarded motion units in the manual segmentation are ”grasp

Automatic Segmentation Motion Unit Index	Manual Segmentation Motion Unit Label	Harmonic Mean (%)
1	Take bowl	44.9
	Put bowl back	37.2
2	Rest Position	70.8
4	Put spoon back	56.4
5	Take knife	32.1
7	Put bowl back	55.1
9	Mash	56.1
10	Cut apple	82.7
	Grasp apple	30.8
11	Stir	75.5
12	Put grater back	43.9
	Take grater	43.5
13	Take bowl	52.0
14	Put knife back	54.6
	Release apple	41.0
15	Put bottle back	54.9
	Pour	33.0
16	Pour	67.8
17	Put masher back	50.1
18	Take knife	40.1
	Put apple back left hand	37.2
19	Put bottle back	34.7
20	Put grater back	52.0
	Put apple back right hand	51.5
21	Grate	58.2
	Take apple right hand	41.3
22	Take bottle	35.7
	Take masher	30.3

Table 5.4: The results of the double matching between automatically and manually obtained motion units in data set B.

apple", "grasp bread", "grasp grinder", "grasp rolling pin with both hands", "mash", "put apple back left hand", "put grater back", "put knife back", "put rolling pin back", "release apple", "release bread", "release grinder", "release rolling pin", "take apple left hand", "take bottle", "take knife", and "take grater". 12 automatically obtained motion units have a double matching. In the opposite case 11 motion units in the manual segmentation have a double matching. The best matching according to the harmonic mean of 72.5% is the matching between the automatically obtained motion unit with index 20 and the motion unit "cut bread".

The number of valid matchings of motion units in the segmentations of data set B is 28. 19 of 22 (86.4%) automatically obtained motion units are matched with 22 of 24 (91.7%) motion units in the manual segmentation. Only motion units "Take apple left hand" and "Take spoon" in the manual segmentation do not have an appropriate matching. The number of automatically obtained motion units with double matching is 9. The number of double matchings of motion units in the manual segmentation is 6. The best matching with a harmonic mean of 82.7% is the matching between the automatically obtained motion unit with index 10 and motion unit "cut apple" in the manual segmentation.

As a conclusion, it can be stated that the unsupervised segmentation method works fine with both data sets A and B. Compared to the performance of the baseline systems $BS(A, 45)$ (16.13% MUER) and $BS(B, 16)$ (19% MUER) the performance of the recognition systems $RS_{pca}(A, 45, 8)$ (25.5% MUER) and $RS_{pca}(B, 18, 10)$ (30% MUER) can be considered as reliable precisely because no human intervention is required at all. The manual segmentation provides better results only if at least 10 complex motion sequences of each complex motion type are segmented and used

for initialization and training. The distribution of motion units among the different complex motion types is distinct which allows a classification of complex motion sequences if needed. The segmentation matching of both data sets is not optimal. This is due to the fact that the unsupervised segmentation method provides a different kind of motion units which is dependent on the segmentation criterion. However, the applied double matching algorithm based on the frame-wise motion unit comparison provides an appropriate matching considering the multi-matching characteristics of the motion units.

6. Model Transfer

6.1 Model Transfer Technique

The model transfer technique is based on the idea of developing a recognition system by transferring models from one or more existing recognition systems to a new recognition system. This method can be applied in order to avoid the manual segmentation effort.

The model transfer technique is known in the field of ASR as the CLT technique. In [SW01] it is applied to transfer existing phoneme models from one or more source recognition systems to a recognition system for a new target language. For this, phonemes of the source languages are assigned to similar phonemes of the target language. This assignment is called phoneme mapping.

In [SW01] two methods of phoneme mapping are presented. The first one is a manual mapping method which relies on expert knowledge. In this method an expert has to decide which phonemes of the source languages are most similar to the phonemes of the target language. This way the phonemes of the source languages are assigned or mapped to phonemes of the target language. The other method is an automatic data-driven mapping method where little amount of training data of the target language is used to calculate the optimal mapping. Both methods require a certain effort of time and costs. However, it is much lower than the effort caused by the transcription and segmentation processes for the training data of the target recognition system.

Similar to the CLT technique, motion unit models in a recognition system build by using motion data from a certain data set could be transferred to another motion recognition system that is applied to recognize similar motion units in another motion data set. In this work the model transfer technique is investigated using three different variants: the first one is the pure model transfer technique; the second variant is a combination of the model transfer technique and the HMM-based automatic segmentation method introduced in section 5.1.1; in the third variant the model transfer technique is combined with a further adaptation using different amounts of transcription data from the target data set.

All three variants of the model transfer technique investigated in this work are evaluated by using motion data sets B and C. The motion units in motion data set B are used as source motion units. The target recognition system is built to recognize motion units of data set C. Both data sets are performed by the same female person. The complex motion types in both data sets are equal, but the motions are performed somehow differently. In data set B primitive motions (motion units) are always performed consecutively while in motion data set C some primitive motions are performed almost simultaneously. In data set B the objects on the counter top are always positioned on the same places (see figure 3.3) while in data set C the object positions are varied between different recording sessions (see figure 3.4). In data set C only the following positions are permitted to be occupied: *back right, back left, center, front right, front left.*

Since the motion information of fingers is not covered by the motion data applied in this work, object specific grasping motions are not relevant in the recognition process. Thus, it can be assumed that the motions of the complete arms toward specific object positions on the counter top are more important. This means that the motion development in certain motion units depends on the object positions. Therefore it might be more appropriate to carry out the motion unit mapping according to the object position in the motion unit and not according to the type of object. In this work the motion unit mapping is carried out depending on the object positions on the counter top. Since in data sets B and C the covered object positions are not exactly the same (see figures 3.3 and 3.4) the nearest covered object positions in data set B are chosen for the positions in data set C.

In data set B the positions on the counter top are covered by at most one object. Thus, a multi-object mapping has to be carried out. The multi-object mapping in this context means that only one motion unit model associated with a certain object position in the source data set (data set B) is applied for mapping multiple motion units associated with the same object position in the target data set (data set C). Actually, the multi-object mapping is one of the problems that arise in the mapping process and which can decrease the performance of the recognition system. The second problem concerning the mapping process is the multi-position mapping. The multi-position mapping takes place when a certain object position in the source data set has to be used to map at least two different object positions in the target data set.

As mentioned above the performance of the recognition system can decrease when multi-object and multi-position mapping are applied. This is due to the fact that the recognition system is not able to accurately distinguish between different motion units which have the same source model. However, applying a statistical motion unit model can help to overcome the mapping problems. Such a statistical motion unit model that contains the occurrence probabilities of motion units supports the IBIS decoder mentioned in section 4.2.3 in order to enable a reliable decoding. In this work a bi-gram model built by using transcription data from the target data set (data set C) is applied.

In the first variant of the model transfer technique applied in this work the transferred models are applied for recognizing motion units in the target data set without further modifications. This pure variant of the model transfer technique can be formally defined as follows:

1. Define train set T_{source} from source data set
2. Define test set T_{test} from target data set
3. Define structure for recognition system Θ_{source}
4. Initialize and train Θ_{source} using T_{source}
5. Define structure for recognition system Θ_{target}
6. Map HMMs of Θ_{source} to HMMs of Θ_{target}
7. Test Θ_{target} using T_{test}

The second variant of the model transfer technique is combined with the HMM-based automatic segmentation method (Configuration I) presented in section 5.1.1. The transferred models are applied to generate hypotheses/segmentations for the target data set. The generated segmentations can be filtered by applying confidence based filtering. The filtered segmentations are utilized to re-train the target recognition system. A similar approach for ASR is presented in [VKS10]. The second variant of the model transfer technique applied in this work can be formally defined as follows:

1. Define train set T_{source} from source data set
2. Define test set T_{test} from target data set
3. Define segmentation set S from target data set
4. Define threshold θ_C for confidence values
5. Define structure for recognition system Θ_{source}
6. Initialize and train Θ_{source} using T_{source}
7. Define structure for recognition system Θ_{target}
8. Map HMMs of Θ_{source} to HMMs of Θ_{target}
9. Repeat the following steps for n iterations
   (a) Decode/Segment S using Θ_{target}
   (b) Write segmentation results with confidence value $\geq \theta_C$ into set T_S
   (c) Re-train Θ_{target} using T_S
   (d) Test Θ_{target} using T_{test}

The third variant of the model transfer technique applied in this work is combined with a further adaptation of the transferred HMMs by using different amounts of transcription data. The adaptation, however, is carried out by applying the Viterbi-based EM-algorithm for training mentioned in section 4.2.2. This way, the Gaussians in the HMMs are not only modified by applying a transformation as usually done using conventional adaptation techniques, but they are completely re-estimated. This means that this variant of the model transfer technique replaces the initialization task which is usually carried out by using manually segmented data. This variant of the model transfer technique can formally be defined as follows:

1. Define train set T_{source} from source data set
2. Define test set T_{test} from target data set
3. Define adaptation set A from target data set
4. Define structure for recognition system Θ_{source}
5. Initialize and train Θ_{source} using T_{source}
6. Define structure for recognition system Θ_{target}
7. Map HMMs of Θ_{source} to HMMs of Θ_{target}
8. Re-train Θ_{target} using T_A
9. Test Θ_{target} using T_{test}

6.2 Experiments

In the last section different variants of the model transfer technique were introduced and explained. In this section the experiments for the evaluation of this technique in conjunction with human motion data is presented. The experiments are conducted by using data sets B and C.

In order to apply the model transfer technique to the entire data set C a motion unit mapping which includes the multi-object and the multi-position mapping problems has to be carried out. The resulting motion unit mapping is shown in table 6.1. As one can see only one motion unit in the source data set associated with a certain object position is applied to map all motion units associated with the same object position (multi-object mapping). Motion units "take masher" and "put masher back" are used for mapping two positions namely *front left* and *back left* (multi-position mapping). The motion units "take bowl from position x" and "put bowl back on position x" in the target data set are mapped to motion units "take bowl" and "put bowl back". This is due to the fact that these motion units in both data sets are performed by using both arms. Therefore, it can be assumed that a two arm motion is similar to another two arm motion even if the position of the object "bowl" is different. The total number of mapped motion units is 65. This is exactly the number of motion units in data set C.

The first experiment is carried out to evaluate the first variant of the model transfer technique applied in this work. The source recognition system for this experiment is initialized and trained by using all 100 motion sequences in data set B. The trained models of the source recognition system are mapped to motion units in data set C according to mapping table 6.1. The resulting target recognition system is tested by using all 100 motion sequences of data set C. The recognition result is a MUER of 78.08% which is a very poor performance of the resulting recognition system. This recognition system is referred to as

$$RS_{mt}(B, C, 65), \tag{6.1}$$

Target Motion Unit	Source Motion Unit
Grate	Grate
Mash	Mash
Pour	Pour
Rest position	Rest position
Stir	Stir
Put object back on position back left	Put masher back
Put object back on position back right	Put bottle back
Put object back on position center	Put spoon back
Put object back on position front left	Put masher back
Put object back on position front right	Put apple back right hand
Put bowl back on position x	Put bowl back
Take object from position back left	Take masher
Take object from position back right	Take bottle
Take object from position center	Take spoon
Take object from position front left	Take masher
Take object from position front right	Take apple right hand
Take bowl from position x	Take bowl

Table 6.1: The mapping of motion units from data set B to all motion units in data set C. Altogether 6 different objects are utilized to perform the complex motions. Position x can be one of the 5 permitted positions in data set C.

where B and C are the source and the target systems, and 65 is the number of transferred models.

The second experiment is carried out to evaluate the second variant of the model transfer technique. In this experiment the model transfer technique is combined with the HMM-based supervised segmentation method introduced in section 5.1.1. The confidence-based filtering is discarded again and the complete hypotheses are applied as segmentations for the re-training task. The experiment is conducted by using a 5-fold cross-validation on data set C as mentioned in sections 4.2.6 and 4.2.7. The initial target recognition system is used to segment the training sets in the cross-validation folds. The segmented data is then used to re-train the target recognition system. The segmentation and the re-training is repeated for 10 iterations. The results are shown in figure 6.1 A. The MUER using one segmentation and re-training iteration after the model transfer amounts to 74.48%. The performance is worse when 2 iterations are applied (76.78% MUER). The best performance of 73.12% MUER is achieved by using 10 segmentation and re-training iterations. This is a minimal absolute performance improvement of 4.96% MUER compared to the performance of the initial target recognition system. The application of more than one segmentation and re-training iterations only leads to minimal performance improvements. The performance of the resulting recognition system is still poor which makes further improvements necessary. The resulting recognition system is referred to as

$$RS_{mt,s}(B, C, 65, 20, i_s), \tag{6.2}$$

where B and C are the source and target systems, 65 is the number of transferred models, 20 is the number of automatically segmented motion sequences of each complex motion type used for re-training, and i_s is the number of applied segmentation and re-training iterations.

In the next experiment the third variant of the model transfer technique is evaluated. In this variant the model transfer technique is combined with a further adaptation of the transferred models using transcription data for the target data set. The experiment is conducted in 4 sessions. In each session different amounts of transcription data is applied. In the first session 5 motion sequences of each complex motion type are used (one sequence for each positioning configuration). In each following session the number of motion sequences is increased by 5. In the last session 20 motion sequences are applied for adaptation. The test set consists of 5 motion sequences of each complex motion type. The experiment is carried out by using a 5-fold cross-validation. The results of this experiment as well as the results of the two other experiments mentioned above are shown in figure 6.1 B. As one can see the MUER decreases significantly even if a little amount of transcription data is applied for adaptation. The performance of the transferred recognition system after adaptation using the transcriptions of 5 motion sequences of each complex motion type is 59.31% MUER. This is an absolute performance improvement of 18.77% compared to the performance of the initial transferred system. The performance using 20 sequences of each complex motion type for adaptation amounts to 36.02% MUER. This is an absolute performance difference of 42.06% MUER. The resulting recognition system is referred to as

$$RS_{mt,a}(B, C, 65, c_t), \tag{6.3}$$

where B and C are the source and the target systems, 65 is the number of transferred models, and c_t is the number of motion sequences of each complex motion type used for adaptation.

In figure 6.1 B the performance of the adapted recognition systems are compared to the performance of the baseline systems $BS(C, c_{i,t})$ and the recognition systems $RS_m(C, 5, c_t)$ presented in section 4.2.7. One can see that the baseline systems always performs better than the adapted recognition systems. This is also the case for the recognition systems initialized by using only 5 motion sequences of each complex motion type and trained by using the same amounts of training data as used for adaptation.

In order to investigate the effect of the multi-object and the multi-position mapping problems on the performance of the transferred recognition system a further experiment is carried out in which only a subset of motion data set C is applied. The subset consists of complex motion types ”stirring” and ”mashing potatoes”. For each of these complex motion types only the motion sequences of one object position per object is applied. This means that the total number of motion sequences to be recognized is 10. The motion unit mapping is carried out as shown in table 6.2. As one can see the multi-object and the multi-position mapping problems do not occur in this mapping configuration which should lead to better recognition results. However, it must be noted that the number of motion units to be recognized is lowered from 65 mapped motion units when the entire data set C is applied to 9

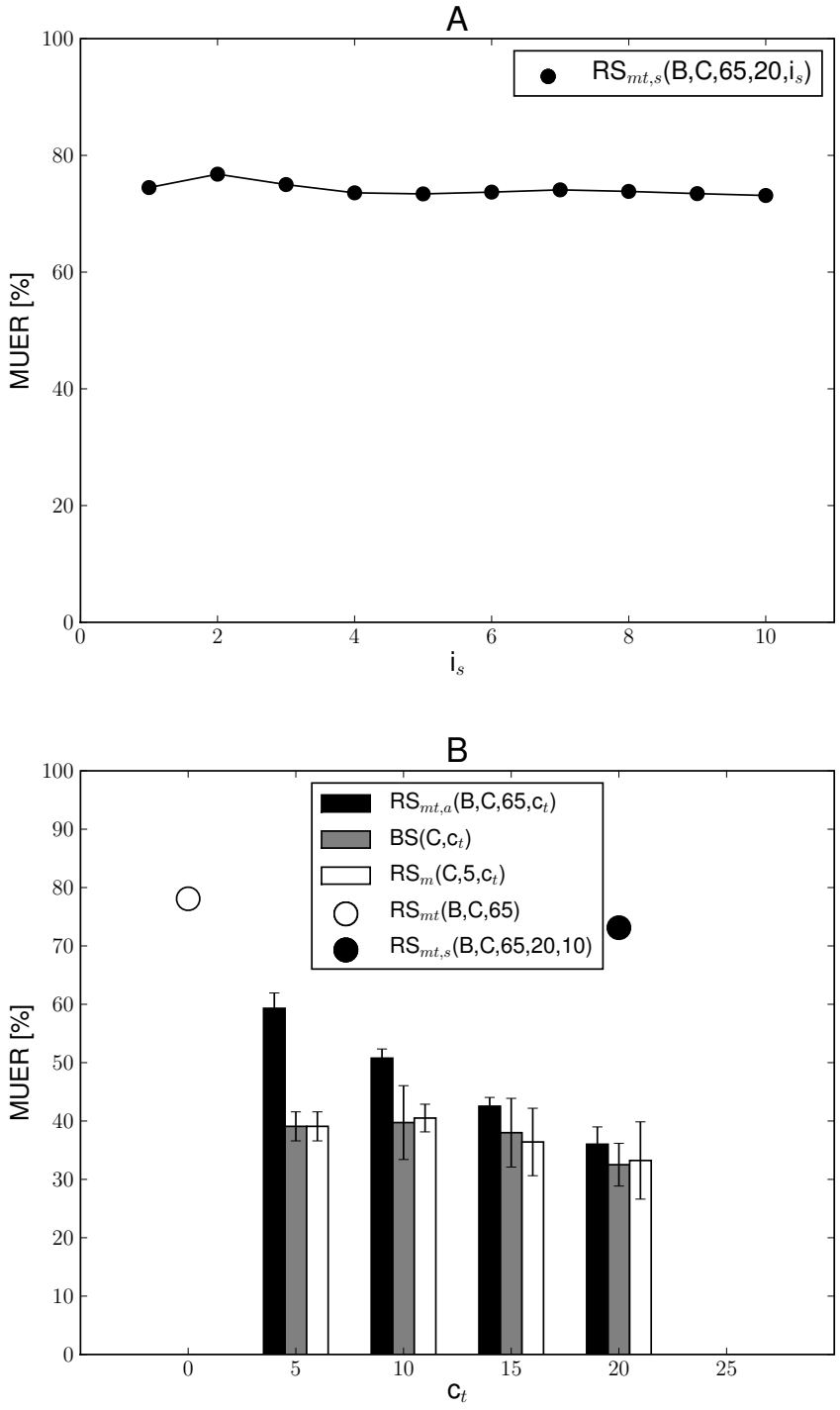

Figure 6.1: The model transfer technique applied on target data set C. In figure A the performance of the target recognition system using different numbers i_s of segmentation and re-training iterations is illustrated. Figure B illustrates the performance of the target recognition system combined with adaptation.

Target Motion Unit	Source Motion Unit
Mash	Mash
Rest position	Rest position
Stir	Stir
Put bowl back on position front right	Put bowl back
Put masher back on position back left	Put masher back
Put spoon back on position center	Put spoon back
Take bowl from position front right	Take bowl
Take masher from position back left	Take masher
Take spoon from position center	Take spoon

Table 6.2: The mapping of motion units from data set B to motion units of two complex motions in data set C.

mapped motion units in this configuration. The source recognition system for this experiment is initialized and trained by using all motion sequences in data set B. The result is a MUER of 57.45% which is a better result compared to the result of the first model transfer experiment mentioned above. However, this has to be treated with caution because the number of mapped motion units is far lower than in the first model transfer experiment and because the MUER of 57.45% is still too low to be considered as a reliable performance.

The low MUER leads to the assumption that the motion units in data sets B and C are too different to allow a meaningful application of the model transfer technique. In order to prove this assumption an additional experiment concerned with transferring models within the same data set is carried out. The experiment is conducted by applying the model transfer technique on two complex motions from data set C. The HMMs of one complex motion type are used as source models. The data of the other complex motion type are used as target data. The source complex motion type is "mashing potatoes" while the target complex motion type is "stirring". For every complex motion 25 sequences exist which are divided among 5 different object positioning configurations. The motion unit mapping is carried out as shown in table 6.3. The number of mapped motion units is 22. The source recognition system is initialized and trained by using all 25 motion sequences of the complex motion type "mashing potatoes". All 25 sequences of the complex motion type "stirring" are used for recognition. The result of this experiment is a MUER of 37.91%. This is a good performance compared to the performance of the baseline systems $BS(C, c_{i,t})$ built by using manually segmented data (see figure 4.6).

The results of all experiments in this section imply that the model transfer technique cannot be applied reasonably on data set B and C. The technique did not perform well even though the multi-object and multi-position mapping problems were eliminated. Transferring models of similar motion units within the same data set (data set C) led to appropriate results. These facts leads to the conclusion that the motion units in data sets B and C are too different. This might be a consequence of the modeling abstraction level of motion units applied in this work. As mentioned before, in ASR the model transfer technique is applied for transferring phoneme models. The modeling abstraction level of phonemes is lower than the modeling abstraction level of motion units applied in this work. Motion units applied in this

Target Motion Unit	Source Motion Unit
Rest position	Rest position
Stir	Mash
Put bowl back on position x	Put bowl back on position x
Put spoon back on position x	Put masher back on position x
Take bowl from position x	Take bowl from position x
Take spoon from position x	Take masher from position x

Table 6.3: The mapping of motion units of "mashing potatoes" to motion units of "stirring" in data set C. Position x can be one of the 5 permitted positions in data set C.

work are comparable to words in speech recognition since they contain more specific information and they can surely be modeled at a lower abstraction level.

The application of adaptation after the model transfer (from data set B to data set C) using transcriptions for motion sequences in data set C improved the recognition system quite reasonably. However, using the same amount of adaptation data in conjunction with a little amount of manually segmented data (5 motion sequences of each complex motion type) for initialization leads to a better performance.

7. Conclusion and Future Work

7.1 Summary and Conclusions

In this thesis two automatic segmentation methods for human motion data and the model transfer technique were investigated in order to enable the development of a robust human motion recognition system, and at the same time reduce or rather overcome the great effort associated with the manual segmentation task. The evaluation was carried out by applying three motion data sets A, B, and C acquired by the Collaborative Research Center 588 - Humanoid Robots. The data sets consist of motions as they appear in kitchen tasks and food preparation scenarios.

The first experiments were conducted to develop baseline systems initialized and trained by using different amounts of manually segmented data. The baseline systems served two purposes: firstly, the performance of the baseline systems was considered as a baseline for the performance of other developed systems; secondly, the baseline systems were utilized for the HMM-based segmentation approach.

Furthermore, other experiments were carried out in order to investigate the performance of recognition systems trained by using more manually transcribed data than manually segmented data for initialization. The experiments showed that the amount of transcribed training data is more significant for the performance of the recognition system than the amount of initialization data. This is a great advantage since transcriptions used in the training task can be prepared with a much lower effort than segmentations.

A supervised HMM-based segmentation method was evaluated by using data sets A and B. For both data sets the evaluation was conducted in several experimental sessions. In each session an HMM baseline system was initialized and trained with a different amount of manually segmented data. The baseline systems were utilized for segmentation. The HMM-based segmentation method was applied in three different configurations.

In order to improve the quality of the automatically obtained segmentation the application of confidence-based filtering was investigated. For this, two confidence measures, namely the gamma probability and the a-stability, were evaluated. In this

work, however, the evaluation showed that a stable correlation between confidence values and actual recognition results does not exist and that the system is too confident even if a little amount of training data is applied. This led to the decision to discard the confidence-based segmentation filtering and to apply the entire obtained segmentations for developing the recognition systems.

The application of the HMM-based segmentation on data set A did not lead to performance improvements of developed recognition systems compared to the performance of the baseline systems. This is due to fact that the HMM of the artificially added motion unit "rest position" in data set A was corrupted by applying unclean automatically obtained segmentation data for re-training. Thus, the recognition system was only able to recognize about 50% of this motion unit. However, F-score values of many other motion units in data set A increased after applying a re-training using additional HMM-segmented training data.

The application of the HMM-based segmentation in conjunction with data set B led to successful performance improvements. In almost all experimental sessions the developed recognition system outperformed the baseline system. An absolute performance improvement up to 9.39% MUER was achieved. In this case the performance of the baseline system $BS(B, 6)$ (32.81% MUER) was improved to 23.42% MUER by using 12 additional automatically segmented motion sequences of each complex motion type ($RS_{hmm,I}(B, 6, 12, 3)$).

A successful application of the HMM-based segmentation method as presented in this work is dependent on the data set. Applying an appropriate segmentation filtering method will surely lead to better improvements and results. The best improvements were achieved by applying a re-training using the transcriptions provided by the segmentation method. The HMM-based segmentation method can be applied as a semi-automatic segmentation and transcription method combined with manual correction.

A PCA-based segmentation method was introduced and evaluated by using data sets A and B. The segmentation method was carried out in two main steps. In the first step motion unit boundaries were detected by finding minima and maxima in the trajectory of the segmentation feature derived by using the first component provided by the PCA. In the second step the detected segments were clustered and labeled by applying the k-means clustering algorithm.

The evaluation of the PCA-based segmentation method in conjunction with data set A led to MUERs up to 25.52%. The evaluation in conjunction with data set B led to performance results up to 30% MUER. The classification of complex motion types based on motion unit distributions led to a classification error rate of 9.6% for data set A and 12% for data set B. In addition, the PCA-segmentations were compared to the manual segmentations by applying a frame-wise comparison. The comparison results were employed in order to determine matchings between automatically obtained motion units and motion units in manual segmentations. Due to the fact that no explicit one-to-one matching exists a double motion unit matching algorithm was applied. The algorithm allows the matching of at most two motion units in both directions. For data set A the matching algorithm was able to find valid matchings for 75.6% of the automatically obtained motion units and 65.3% of the motion units in the manual segmentation. For data set B the matching re-

sulted in matching 86.4% of the automatically obtained and 91.7% of the manually obtained motion units.

The PCA-based segmentation method led to reliable results in both motion unit recognition and complex motion classification. Recognition systems trained with PCA-segmented data outperformed the baseline systems trained with less than 10 manually segmented sequences of each complex motion type. These reasons make it a serious alternative to other segmentation methods especially when no manually segmented data exists and a recognition system for a limited number of motions has to be developed rapidly. A drawback of the PCA segmentation is that the obtained motion units do not have a semantic meaning as the motion units in the manual segmentation. This problem could not be solved by applying explicit one-to-one matchings between motion units in the automatic segmentation and motion units in the manual segmentation. Instead, double motion unit matchings were applied. The PCA segmentation method can be applied when the semantic meaning of motion units is not important for the application. It can be reasonably applied for the classification of complex motions.

The model transfer technique was applied on data sets B and C. HMMs of motion units in data set B were transferred to a new recognition system for data set C. The motion unit mapping was carried out in dependence to the object positions on the counter top. Due to the characteristics of data sets B and C multi-object and multi-position mappings had to be applied. All 65 motion units of data set C were mapped. The result of the transferred recognition system is a MUER of 78.08%. A combination of the model transfer technique and HMM segmentation led to an improvement up to 72.03% MUER. Combining model transfer and transcription data adaptation led to significant performance improvements up to 36.02% MUER. An experiment including motion unit model transfer of only two complex motion types in which no multi-object and multi-position mappings existed led to a performance of 57.45% MUER. A further experiment including only motions from data set C was carried out. In this experiment motion units of one complex motion in data set C was mapped to motion units of another complex motion in data set C. The result was a MUER of 37.91%.

The model transfer technique applied in this work did not perform well even though the multi-object and multi-position mapping problems were eliminated. Transferring models of similar motion units within the same data set led to appropriate results. Applying adaptation after the model transfer using transcription data improves the recognition system quite reasonably. However, using the same amount of adaptation data in conjunction with a little amount of manually segmented data (5 motion sequences of each complex motion type) for initialization leads to a better performance. Thus, the application of the model transfer technique is only worthwhile if the initialization models using pre-segmented data is not possible, but still a certain amount of transcription data is available.

The presented approaches in this work are compared to each other according to the expected performance of the developed recognition systems and the effort required for the development. The comparison can be seen in figure 7.1. The best performance can be achieved by applying manually segmented data for initialization and training. However, this option require the highest effort due to the manual segmentation task. A more cost-efficient option is to use a little amount of manually

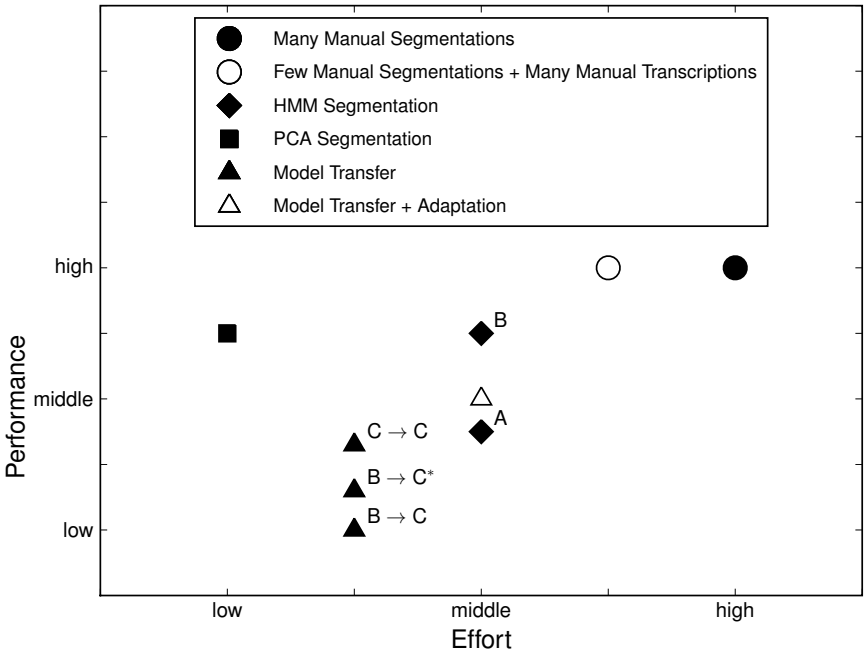

Figure 7.1: An overview of the expected performance of recognition systems using different development approaches and the effort required for the development. The letters A and B next to the diamond markers refer to the data sets A and B. The letters next to the triangle markers refer to the source and target recognition systems. B $\rightarrow$ C* refers to the model transfer without multi-position and multi-object mapping.

segmented data for the initialization task and use more manually transcribed data for the training task. This way the effort can be lowered and at the same time the performance is kept high. The lowest effort required to overcome the manual segmentation task combined with a good performance can be achieved by applying the automatic PCA segmentation method. In this case motion units without semantic meaning has to be accepted or some effort has to spent in order to give them a semantic meanig. The automatic HMM segmentation method provide motion units with semantic meaning, but a a certain effort has to be spent for the preparation of manually segmented data; depending on the data set, reliable recognition results can be achieved. The pure model transfer technique require a relatively low effort for the motion unit mapping task, but the performance is very low. Combining the model transfer technique with adaptation leads to a performance improvement. For this an additional effort has to be spent in order to prepare transcriptions of motion sequences in the target data set.

7.2 Future Work

The most promising approaches investigated in this work are the HMM-based and the PCA-based automatic segmentation methods.

The HMM-based automatic segmentation method could be exploited to build the core of a semi-automatic segmentation and transcription framework. However, it

must be ensured that the effort spent for correcting automatically generated segmentations and transcriptions must not exceed the effort spent for generating new manual segmentations and transcriptions. This could be achieved by applying a graphical user interface (GUI). Generated segmentations and transcriptions can be provided for operators in form of concatenated graphical elements that can be moved, deleted, inserted, and resized. Operators may be able to choose one of the n best alternative hypotheses provided by the system. Most likely motion units should be provided in a drop-down list as insertion options.

Furthermore, the HMM-segmentation method could be improved in order to generate more accurate transcriptions. Instead of a flexible bi-gram statistical model a more restrictive option could be applied in order to enforce certain motion unit compositions. This can be achieved by applying a motion grammar.

A drawback of the PCA-based segmentation method is the lack of semantic meaning of obtained motion units. However, systems based on such motion units can be improved in order to allow the assignment of semantic meaning through interaction. Motion units which require a semantic meaning (from the viewpoint of humans) can be labeled using different kinds of human-machine interaction methods. One simple possibility would be the integration of the PCA-based segmentation method into a framework like mentioned above. System-generated labels could be replaced by entering new meaningful labels. A more interesting scenario is to enable a robot to learn the humane meaning of the most important automatically segmented and composed motions through natural speech.

Bibliography

[Bis06] C.M. Bishop. *Pattern recognition and machine learning*, chapter 12. Springer, 2006.

[BSP+04] J. Barbič, A. Safonova, J.-Y. Pan, C. Faloutsos, J.K. Hodgins, and N.S. Pollard. Segmenting motion capture data into distinct behaviors. In *Graphics Interface*, pages 185–194, 2004.

[Dil04] R. Dillmann. Teaching and learning of robot tasks via observation of human performance. *Journal of Robotics & Autonomous Systems*, 47:2004, 2004.

[FMJ02] A. Fod, M.J. Mataric, and O.C. Jenkins. Automated derivation of primitives for movement classification. *Autonomous Robots*, 12:39–54, 2002.

[GKWS09] D. Gehrig, H. Kuehne, A. Woerner, and T. Schultz. HMM-based human motion recognition with optical flow data. pages 425 –430, dec. 2009.

[JM04] O.C. Jenkins and M.J. Mataric. Performance-derived behavior vocabularies: data-driven acquisition of skills from motion. *International Journal of Humanoid Robotics*, 1(2):237–288, 2004.

[KC02] Y.-J. Kim and A. Conkie. Automatic segmentation combining an HMM-based approach and spectral boundary correction. 2002.

[KKUG07] V. Krüger, D. Kragic, A. Ude, and C. Geib. The meaning of action: a review on action recognition and mapping. *Advanced Robotics*, 21:1473–1501(29), September 2007.

[KS97] T. Kemp and T. Schaaf. Estimating confidence using word lattices. In *Proceedings of EuroSpeech*, pages 827–830, 1997.

[KW98] T. Kemp and A. Waibel. Unsupervised training of a speech recognizer using tv broadcasts. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP-98*, pages 2207–2210, 1998.

[MBS93] T.M. Martinetz, S.G. Berkovich, and K.J. Schulten. 'Neural-gas' network for vector quantization and its application to time-series prediction. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, 4(4):558–569, 1993.

[MHK06] T.B. Moeslund, A. Hilton, and V. Krüger. A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding*, 104(2-3):90 – 126, 2006.

[MSSS04] T. Mori, Y. Segawa, M. Shimosaka, and T. Sato. Hierarchical recognition of daily human actions based on continuous hidden markov models. pages 779 – 784, may. 2004.

[Orf96] S.J. Orfanidis. *Introduction to signal processing*, chapter 8. Prentice-Hall, 1996.

[Rab89] L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, pages 257–286, 1989.

[SK97] T. Schaaf and T. Kemp. Confidence measures for spontaneous speech recognition. In *Proc. ICASSP*, pages 875–878, 1997.

[Ste99] G. Stelzner. Zur Modellierung und Simulation biomechanischer Mehrkörpersysteme. In *Schriftenreihe des Instituts für Technische Mechanik, Universität Karlsruhe (TH)*, volume 11, 1999.

[SW01] T. Schultz and A. Waibel. Experiments on cross-language acoustic modeling. In *Proc. of Eurospeech'01, Alborg*, pages 2721–2724, 2001.

[VKS10] N.T. Vu, F. Kraus, and T. Schultz. Multilingual a-stabil: a new confidence score for multilingual unsupervised training. In *IEEE Workshop on Spoken Language Technology, SLT 2010, Berkeley, California, USA*, 2010.

[VM99] C. Vogler and D. Metaxas. Parallel hidden markov models for american sign language recognition. In *ICCV*, pages 116–122, 1999.

[WLZ05] Y. Wang, Z.-Q. Liu, and L.-Z. Zhou. Learning hierarchical non-parametric hidden markov model of human motion. volume 6, pages 3315 –3320 Vol. 6, aug. 2005.

[YOI92] J. Yamato, J. Ohya, and K. Ishii. Recognizing human action in time-sequential images using hidden markov model. pages 379 –385, jun. 1992.

[ZC98] G. Zavaliagkos and T. Colthurst. Utilizing untranscribed training data to improve performance. In *DARPA Broadcast News Transcription and Understanding Workshop, Landsdowne*, pages 301–305, 1998.