Diplomarbeit im Fach Informatik

Appearance-Based Gesture Recognition

RHEINISCH-WESTFÄLISCHE TECHNISCHE HOCHSCHULE AACHEN
Lehrstuhl für Informatik VI
Prof. Dr.-Ing. H. Ney

vorgelegt von:
Philippe Dreuw
Matrikelnummer 217722
E-Mail philippe.dreuw@gmx.net

Gutachter:
Prof. Dr.-Ing. H. Ney
Prof. Dr. T. Seidl

Betreuer:
Dipl.-Inform. D. Keysers

Januar 2005

Erklärung

Hiermit versichere ich, dass ich die vorliegende Diplomarbeit selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel verwendet habe. Alle Textauszüge und Grafiken, die sinngemäß oder wörtlich aus veröffentlichten Schriften entnommen wurden, sind durch Referenzen gekennzeichnet.

Aachen, im Januar 2005

Philippe Dreuw

Abstract

This diploma thesis investigates the use of appearance-based features for the recognition of gestures using video input. Previously, work in the field of gesture recognition usually first segmented parts of the input images — for example the hand — and then used features calculated from this segmented input. Results in the field of object recognition in images suggest that this intermediate segmentation step is not necessary and we can instead employ features directly obtained from the input images, so-called appearance-based features. In this work, we show that using these features and appropriate models of image variability, we can obtain excellent results for gesture recognition tasks. Very good results can be obtained using a downscaled image of each video frame and tangent distance as a model of image variability. Also a new dynamic tracking algorithm is introduced which makes its tracking decisions at the end of a video sequence using the information of all frames. This tracking method allows for tracking under very noisy circumstances. Finally, a new database with the German fingerspelling alphabet was recorded which will be freely available for further research.

Acknowledgment

I would like to thank Prof. Dr.Ing. Hermann Ney for the interesting possibilities at the Chair of Computer Science VI of the RWTH Aachen University of Technology where I have been a member since January 2003.

I would also like to thank Prof. Dr. Thomas Seidl, who kindly accepted to co-supervise the work.

Specially, I would like to thank Daniel Keysers, Thomas Deselaers and Morteza Zahedi for the helpful suggestions, discussions and assistances this work received. Their ideas and suggestions were a great help.

I would like to thank Jan Bungeroth and all volunteers for the participation in creating the i6-Gesture database.

Also, I would like to thank the other members of the image recognition group at the Chair of Computer Science VI – Andre Hegerath, Ilja Bezrukov, Helga Velroyen and David Rybach – for many discussions, helpful tips and comments or proof reading the manuscript. Also I would like to thank Alexander Zimmerman for many LaTeX-tips.

Of course special thanks go to my parents. They gave me the possibilities doing these studies. Last but not least I would like to thank my girlfriend Cindy for supporting me while doing this work and who mostly saw me only late at night.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This diploma thesis investigates the use of appearance-based features for the recognition of gestures using video input. Previously, work in the field of gesture recognition usually first segmented parts of the input images — for example the hand — and then used features calculated from this segmented input. Results in the field of object recognition in images suggest that this intermediate segmentation step is not necessary and we can instead employ features directly obtained from the input images, so-called appearance-based features. In this work, we show that using these features and appropriate models of image variability, we can obtain excellent results for gesture recognition tasks. For example, on the LTI-Gesture database, very good results can be obtained using a downscaled image of each video frame and tangent distance as a model of image variability.

Sign language is the natural language of deaf people. Two main reasons for studying gesture and sign language recognition are:

- to create applications to help deaf people
- to use this know-how for gestural input devices in the field of human computer interfaces.

Developing sign language applications for deaf people can be very important, as many of them, being not able to speak a language, are also not able to read or write a spoken language. Ideally, a translation systems would make it possible to communicate with deaf people. — Imagine a system for blind and deaf people ...

In human computer interfaces, one of the main advantages of using visual input is the possibility to communicate without need for physical contact with the equipment to be controlled (so-called "10 foot" user interfaces). This is achieved by eliminating input devices such as joysticks, mice, and keyboards e.g. allowing the unrestricted body to give signals to the computer through gestures such as finger pointing.

Compared to speech commands, hand gestures are advantageous in noisy environments, in situations where speech commands would be disturbing, as well as for communicating quantitative information and spatial relationships.

Unlike haptic interfaces, gesture recognition does not require the user to wear any special equipment or attach any devices to his body. The gestures of the body are read by a camera instead of sensors attached to a device such as a data glove. Using cameras and looking at users is a powerful technique to facilitate the human-computer interaction.

In addition to hand and body movement, gesture recognition technology can also be used in combination with facial and speech expressions (i.e. lip reading), or eye movements.

Gestural input devices are in great demand today, as the growing amount of functions of e.g. a television, a car, or a simple mobile phone confronts the users with new problems. Previous approaches which added a new button or wheel for each new function on the teleguidance or the car dashboard ended in complex and no longer usable input devices. Today's mainstream trend goes in the contrary direction, making input devices smaller and overloading functions: the best example might be the buttons of a mobile phone where each button can have plenty of functions in different contexts. Creating gestural input devices offers the possibility to extend often used devices without the need of overloading functions or buttons. Intuitively usable gestures are more easily accepted and learned by users and do not distract them by the habitual use of the device, e.g. integrating a gesture controlled unit into a car dashboard might be more easily controlled than searching small buttons while driving. The well known Sony-EyeToy of Playstation is an example where such a gestural input device is already fully accepted by the users and offers new possibilities for interacting with a game console.

The main focus in this work is set on using simple appearance-based features with no need for complex feature extraction. Appearance-based features in combination with hidden Markov model classifiers known from speech-recognition and newly integrated distance measures known from image and optical character recognition (e.g. being invariant against affine transformations) are investigated. Also a new tracking algorithm was developed which traces the best tracking path at the end of an entire observation sequence.

Chapter 1 explains the differences between gestures and sign language and gives a short overview of current notation system for sign language. The remainder of this work is organized as follows: Chapter 2 gives a survey of sign language recognition and human computer interfaces systems available to show what has already been developed and to show the differences between these two fields. In Chapter 3 we introduce appearance-based features used in gesture recognition and in this work. Chapter 4 gives a short overview of the hidden Markov model (HMM) theory. We tried to be as general as possible in developing our methods for recognizing gestures and focused especially on hidden Markov models (HMMs) properties and the use of different distance measures. Especially the theory part important for this work is explained and the different distance measures and used inside the HMMs are introduced. In Chapter 5, a new tracking algorithm based on dynamic programming techniques is introduced. This tracking method makes its decision at the end of a sequence. Chapter 6 presents the databases used to test our gesture recognition system. Also a new database of finger-spelling letters of German Sign Language (GSL), which was created in the course of this work and recorded neither under clothing constraints nor under constant lighting conditions, is presented. Chapter 7 shows the interesting results obtained with appearance-based features and different distance measures or tracking on these databases. Finally, we conclude and summarize this work in Chapter 8.

1.1 Gestures and Sign Language

A **gesture** is a form of non-verbal communication made with a part of the body and used instead of verbal communication (or in combination with it). Most people use gestures and body language in addition to words when they speak. These gestures include acts such as pointing, one of the few gestures whose meaning varies little from one country to the next, as well as using the hands and body to keep time with the rhythms of speech and emphasize certain words or phrases. Most of these gestures have no invariable or specific meaning.

A **sign language** is a language which uses gestures instead of sound to convey meaning — combining hand-shapes, orientation and movement of the hands, arms or body, facial expressions and lip-patterns. Sign languages are usually developed in deaf communities, which include interpreters and friends and families of deaf people as well as people who are deaf or hearing-impaired themselves.

Contrary to popular belief, sign language is not international. Wherever communities of deaf people exist, sign languages develop. As with spoken languages, these vary from region to region. They are not completely based on the spoken language in the country of origin. Other simple forms of signed communication have been developed in situations where speech is not practical to speak, such as between scuba divers, in television recording studios, in loud workplaces, or while hunting.

Sign language is a visual language and consists of 3 major components:

- finger-spelling: used to spell words letter by letter
- word level sign vocabulary: used for the majority of communication
- non-manual features: facial expressions and tongue, mouth and body position

Today three famous notation systems for gesture and sign language exist. A short overview of these systems is given in the following sections. Figure 1.1 shows the difference between their notation forms.

(a) (b) (c) (d)

Figure 1.1. Some examples of the different sign language notion systems taken from `http://signwriting.org/forums/linguistics/ling001.html` visited 17.12.2004: (a) a person signing "What", (b) in Stokoe notation, (c) in SignWriting notation, (d) in HamNoSys notation

1.2 Stokoe Notation System

According to the sign linguist Stokoe [Stokoe 80], one can represent a sign as a kind of "chireme", as vowels and consonants are kinds of phonemes in spoken language. He invented a written notation for sign language in 1960 as ASL[1] had no written form at that time. Such a chireme includes three visual features:

1. **DEZ (designator):** hand shape or configuration of the hand involved in the sign
2. **SIG (signation):** movement executed by the hands
3. **TAB (tabula):** location of the sign in relation to the body

By using symbols to represent the component parts of American Sign Language, he was able to demonstrate how these parts fit together to form a linguistic structure identical with that of spoken language. The original notation consisted of 55 symbols in three groups, each representing one of the formational parameters of a sign: location, hand shape, and movement (see Figure 1.2).

1.3 SignWriting Notation System

Another method to describe signs called "SignWriting" was developed in 1974 by Sutton [Sutton 77], a dancer who had developed "DanceWriting" two years earlier, a notation system for representing dance movements. When Sutton applied this iconic method to recording signed languages, she realized that recording the movement is also recording

[1]ASL = American Sign Language

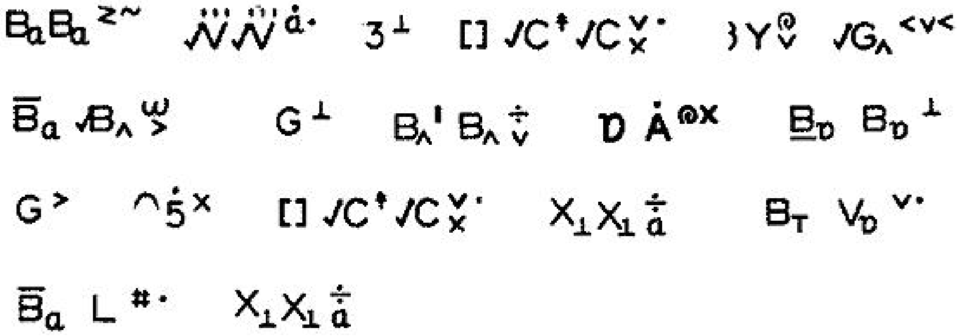


Figure 1.2. Example of sentences in Stokoe notation taken from `http://www.signwriting.org/forums/linguistics/ling006.html`, visited 21.09.2004

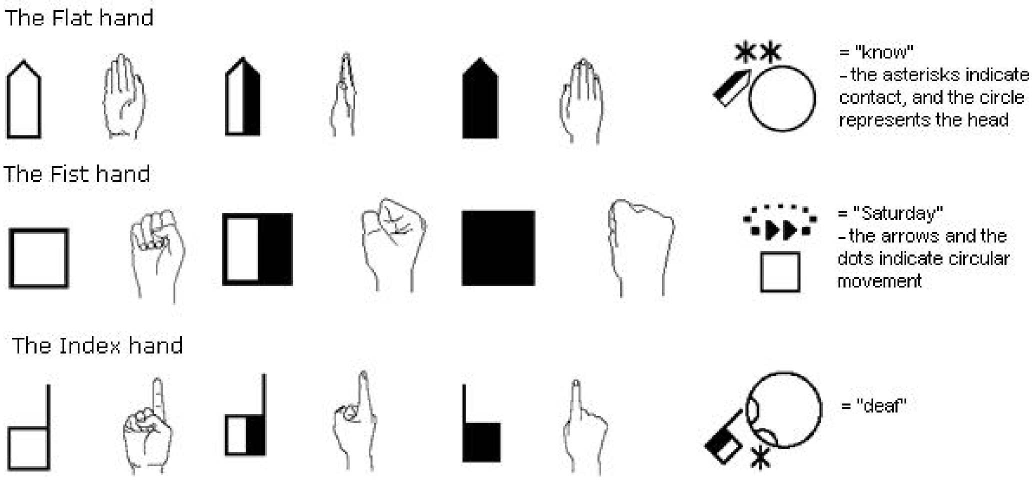


Figure 1.3. A selection of basic ASL SignWriting signs taken from `http://www.omniglot.com/writing/signwriting.htm`, visited 21.09.2004

the language. It uses visual symbols to represent the hand shapes, movements, and facial expressions of sign languages (see Figure 1.3).

1.4 The HamNoSys Notation System

HamNoSys [Prillwitz & Leven^{+} 89] was developed by a group of hearing and deaf people as a scientific/research tool and first made publicly available in 1989. The purpose of HamNoSys, unlike SignWriting, has never been an everyday use to communicate (e.g. in letters) in sign language. It was designed to fit a research setting and should be applicable to every sign language in the world.

It consists of about 200 symbols covering the parameters of hand shape, hand configuration, location and movement (cf. Stokoe Notation). The symbols are as iconic as possible and are easily recognizable. The order of the symbols within a string is fixed, but still it is possible to write down one and the same sign in lots of different ways.

Hence, the transcriptions are very precise, but on the other hand also very long and cumbersome to decipher. It is possible to transcribe facial expressions, but their development is not quite finished yet. HamNoSys is still being improved and extended all the time as the need arises. Figure 1.4 shows some example sentences in HamNoSys notion.

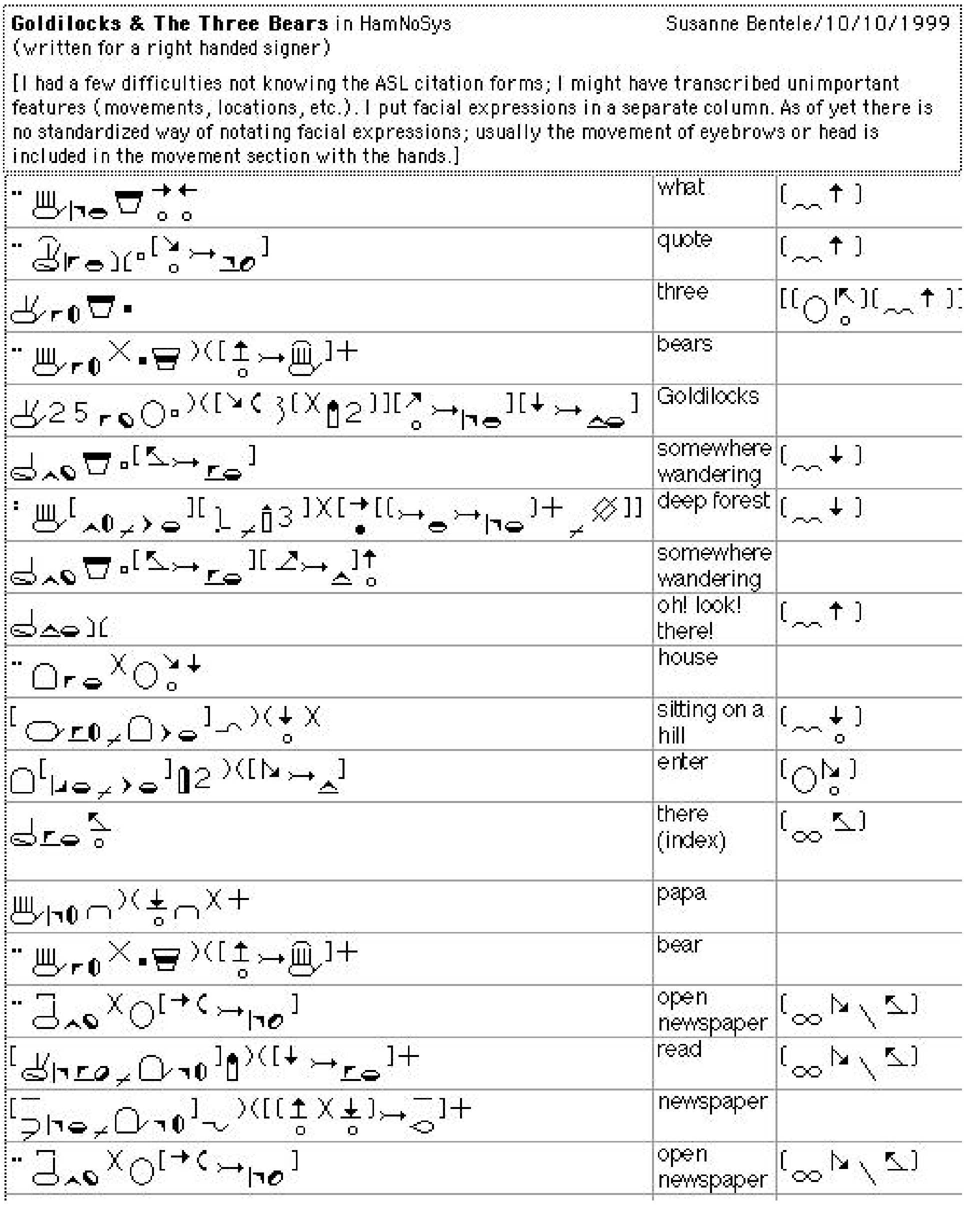


Figure 1.4. Some example sentences in HamNoSys notation taken from `http://www.signwriting.org/forums/linguistics/ling007.html`, visited 17.12.2004

Chapter 2

State of the Art in Gesture Recognition

The research in computer gesture recognition primarily focuses on sign language recognition and human-computer interaction (HCI). People frequently use gestures to communicate, and HCI studies usually focus on the computer input/output interface. The researchers who use gestural input use their own definitions of gestures. These gestures can be translated by computers into either symbolic commands or trajectory motion commands to control their experimental systems.

In sign language the gestures are part of a visual language and well defined. The gestures are used to communicate in form of finger-spelling, as complete words, or as non-manual features.

Many disciplines must be combined to achieve a reliable recognition system as one has to deal with e.g. capturing problems like varying lighting conditions, skin colored clothes or tracking of multiple objects.

Most of the systems presented in the remainder assume constant environment variables for their systems, e.g. constant lighting conditions. The sign language recognition systems presented in Section 2.1 mostly assume persons wearing non-skin-colored clothes with long sleeves and a fixed camera position under constant lighting conditions. The systems presented in Section 2.2 are often too person-dependent and use only gestures which exhibit great differences to be easily recognizable.

The remainder of this chapter gives an overview on systems and methods available for gesture and sign language recognition. Results from [Rigoll & Kosmala$^+$ 98] and [Pelkmann 99] are also given in Chapter 7 in comparison to the results obtained in this work.

2.1 Related Work in the Context of Sign Language

One of the first working real-time sign language recognition systems was developed by Starner et. al [Starner & Pentland 95]. Their HMM-based system works without explicitly modeling the fingers and recognizes sentence level American Sign Language. The tracking module (see also [Wren & Azarbayejani$^+$ 97]) can be used with or without colored gloves, where the resultant shape, orientation and trajectory information is input to an HMM for recognition of the signed words. With a 40 word lexicon they achieve an error rate of 8 % for the skin color tracking case.

The idea of an automatic sign language recognition system using subunits rather than models for whole signs was presented in [Bauer & Kraiss 02]. The advantage of such a system will be a future reduction of necessary training material and simplified enlargement of the existing vocabulary. Their system is able to detect sign subunit boundaries automatically by a so called fenomic model, which is completely data driven. The system was trained and tested by one person and thus is highly person dependent.

[Triesch & von der Malsburg 02] presented a system for person-independent classification of hand postures in grayscale images against complex backgrounds in video images. They use the elastic bunch graph matching method to model variance in hand posture appearance between different subjects and variance in backgrounds. Their local image descriptions are based on a wavelet transform with complex Gabor-based kernels.

[Bowden & Windridge+ 04] present in their paper a two-stage classification procedure where an initial classification stage extracts a high level description of hand shape and motion. A second stage of classification is then used to model the temporal transitions of individual signs using a classifier bank of Markov chains combined with Independent Component Analysis. The system performs well with single instance training.

[Wu & Chiu+ 04] propose an error-tolerant approach to retrieving sign words from a Taiwanese Sign Language database in their paper. The database is tagged with visual gesture features, which are defined in terms of the visual characteristics of sign gestures (see also [Stokoe 80]). The maximum a posteriori estimation is exploited to retrieve the most likely sign word given the input feature sequence and an error tolerant mechanism based on mutual information criterion is proposed to retrieve a sign word of interest efficiently and robustly.

2.2 Related Work in the Context of Human Computer Interaction

Many interface systems have been developed for intelligent room systems. For example the ALIVE system [Maes & Darrell+ 97] allows unencumbered full-body interaction between a human participant and a rich graphical world inhabited by autonomous agents. PFINDER [Wren & Azarbayejani+ 97], a real-time system for tracking people in arbitrarily complex but single-person, fixed-camera situations and interpreting their behavior, is a descendant of the vision routines originally developed for the ALIVE system.

[Freeman & Anderson+ 98] used computer vision techniques to find the user's open hand across a room. This was used to control a television such as volume change or switching channels.

[Black & Jepson 98] extended the condensation algorithm by [Isard & Blake 98] to recognize gestures, which where modeled as temporal trajectories of some estimated parameters over time. The condensation algorithm is used to incrementally match the gesture models to the input data. They combined Dynamic Time Warping and Hidden Markov Model properties for recognition and matching of the model trajectories to the input trajectories. Their system cannot distinguish between different shapes as it recognizes only trajectories.

[Rigoll & Kosmala$^+$ 98] presented a person-independent real-time system for gesture recognition. The system uses global motion features, extracted from each difference image of the image sequence, and HMMs as a statistical classifier. These HMMs are trained on a database of 24 isolated gestures, performed by 14 different people. An error rate of 7.1% is achieved for a person and background independent recognition, but the system can only distinguish between gestures, which can be characterized by their movement.

[Little & Boyd 98] developed a vision system that can recognize people by the way they walk. The system computes optical flow for an image sequence of a person walking, and then characterizes the shape of the motion with a set of sinusoidally-varying scalars. Feature vectors composed of the phases of the sinusoids are able to discriminate among people.

[Pelkmann 99] extended a gesture recognition system to control a car-navigation system, where one infrared camera is installed inside a car to capture the gestures. The system uses Hidden Markov Models as statistical classifiers and is trained on a set of 14 isolated gestures using several geometric features such as compactness, Hu-Moments and global motion as gesture features. It achieved an error rate of 4.5%.

[Bobick & Davis 01] presented a view-based approach to the representation and recognition of human movement using temporal templates, which are static vector-images where the vector value at each point is a function of the motion properties at the corresponding spatial location in an image sequence. They explore two versions of those templates (motion-energy- and motion-history-images) and develop a recognition method matching temporal templates against stored instances of views of known actions.

Sony developed the USB EyeToy camera [Sony 04], a commercial system, aimed at the player and plugged into a PlayStation(R)2 computer entertainment system, the gamer enters the short calibration process before beginning the fast-paced, active gameplay. The camera uses a combination of advanced facial and motion tracking technology to lock onto the player and capture movement to enable gamers to become physically part of the game. The gamer's movements are then used to control steering, jumping, ducking, grinding, acceleration, braking, and tricks, as they interact with the environment and avoid obstacles. Sony has sold more than 4 million EyeToy units worldwide since its release.

[Wilson & Oliver 03] created a stereo-vision based system and demonstrate their algorithm in combination with speech recognition to perform several basic window management tasks. They use a fast stereo vision algorithm for recognizing hand positions and gestures. Their system uses two inexpensive video cameras to extract depth information. This depth information enhances automatic object detection and tracking robustness and can be used also in applications (e.g. painting programs).

[Zobl & Nieschulz^{+} 04] present the gesture part of their multimodal system, which consists of a gesture-optimized user interface, a real time gesture recognition system and an adaptive help system for gesture input. The system can deal with eleven dynamic hand gestures classes and four hand poses. They use an adaptive background to segment and threshold the images. After filtering those images with a forearm filter, they calculate moment based features like area, centroids and Hu-Moments [Hu 62] (i.e. trajectory and hand form). The system was trained and tested by using data of one person and thus is highly person dependent.

[Morrison & McKenna 04] compare trajectory-based and history-based methods for visual recognition of gestures. They use skin colour as a common visual cue, recognition methods based on hidden Markov models, moment features and normalised template matching. They propose skin history images as a useful history-based representation and report results on a database of sixty gestures.

Chapter 3

Features for Appearance-Based Gesture Recognition

Many research groups in gesture recognition use quite complex methods to recognize the gestures, like fingertip detection, calculating the angles between the fingers or matching of 3D-models. They split up their systems into several subsystems. This approach has the disadvantage that a possible error is propagated through the whole system when the parent system makes a wrong decision.

Often used features for gesture recognition are:

- Color: brightness, skin color models, ...
- Texture: Gabor-filters, gradients, ...
- Shape: Active Shapes, Active Contour Models, ...
- Motion: centroids, difference images, optical flow, ...

A short survey on features for the visual analysis of human movement can be found in [Gavrila 99]. Spatio-temporal segmentation of video sequences is also an often used and essential step in video analysis. It attempts to extract backgrounds and independent objects in the dynamic scenes captured in the sequences. A survey can be found in [Megret & DeMenthon 02].

In an appearance-based approach one does not create such modular systems which have to extract features, even though all the information one needs to recognize a gesture is the image itself — segmenting images is very difficult and never perfect.

As we wanted to create a system which is person- and background-independent, we specially analyzed difference images and centroid features. In Section 3.1, we present and discuss different image-subtracting methods and features and in Section 3.2 some centroid features.

3.1 Image Features

In an appearance-based approach the image itself and transformations (distortion, filtering, sub-sampling, ...) of the image are usually used as features.

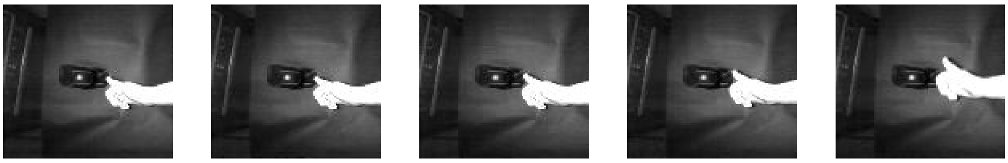


Figure 3.1. Original infrared-image sequence of the gesture "Daumen Rechts" seen from car interior roof

In this thesis, we will denote an original image X in a sequence at time $t = 1, ..., T$ by X_t and by $X_t(x, y)$ the pixel value at the position (x, y). Any transformed image will be denoted by $\widetilde{X}$.

Although this notation is slightly overloaded, it will be clear from the context which transformation is being used.

3.1.1 Original Images

When working for example with gray valued images (e.g. infrared-images like in Figure 3.1), a (thresholded) original image can be used as a feature.

This allows a simple description of the gesture without any movement properties as one image feature at time t contains only information about this time frame. Also a performer (in front of a camera) should wear the same clothing as seen in the training process.

If more exact information is needed, e.g. to distinguish between gestures which differ only in one finger, the area of interest with the signing hand can be extracted and used as a feature with higher resolution (see tracking in Chapter 5).

Using original image sequences as feature without any thresholding or tracking can already lead to very good results (see Chapter 7). Another motivation to use this feature is that nearest neighbour classifiers perform very well in a number of image recognition cases by simply comparing the original images as they are.

Other often useful features are the spatial derivative images of such an original image sequence as in Figure 3.2. They allow to detect important image features such as fingers or fingertips which then appear as edges.

3.1.2 Difference Images

Calculating difference images is one of the simplest methods to detect motion in an image sequence.

Motion is a very important feature in image sequences, which demonstrates us the relation between local properties and time variation. This method is fast and the optical flow in the motion field can be used in further processing steps and applications.

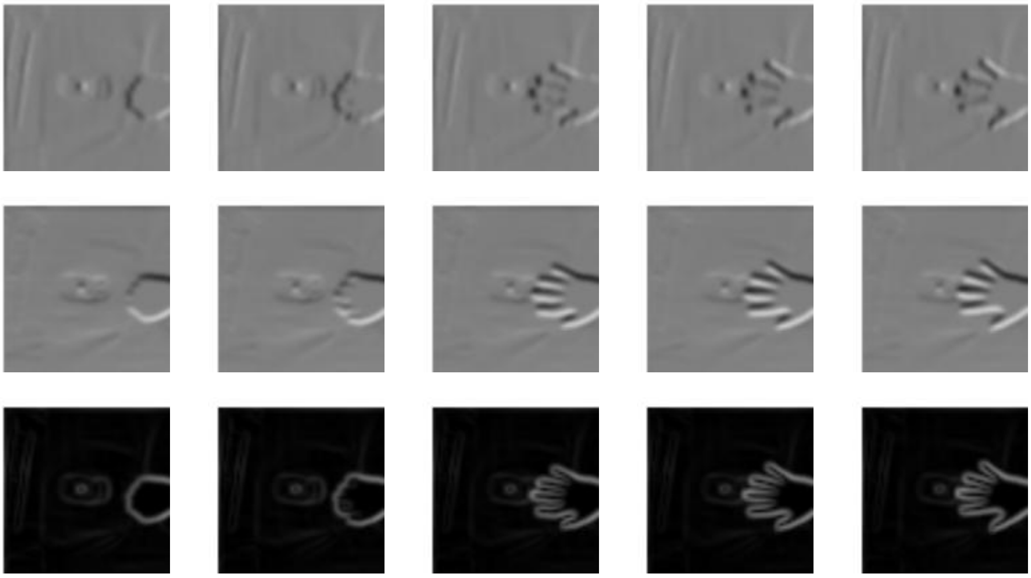


Figure 3.2. Spatial derivative image sequence of the gesture "Five" using a Sobel filter which detects edges and suppresses background. The first row uses a vertical Sobel filter so that the fingertips appear as short dark edges. The second row uses a horizontal Sobel filter so that the fingers appear as longer dark and light edges. The last row uses a Sobel filter so that the hand shape appears as light edges.

Common applications of image differencing include object tracking [Yang & Levine 92], or intruder and vehicle surveillance systems [Koller & Weber$^+$ 94]. There are also examples of its use for analysing satellite images [Singh 89] to measure land erosion, deforestation, urban growth and crop development. Other applications are motion detection for gait analysis [Little & Boyd 98], data compression (e.g. MPEG), 3D-scene reconstruction, traffic control, or analysing medical images to measure cell distribution [Knoll & Brinkley$^+$ 85].

A difference calculation usually represents a first derivative in movement direction of the objects whereas the gradients are perpendicular to the movement direction.

For some applications these difference images are sufficient, but they are very susceptible to noise, changes in the lighting conditions or camera position because these transformations also lead to an intensity change in the difference image. On the other hand not every movement results in a image change, e.g. difference images of homogeneous surfaces.

The use of temporal data is mostly based on the assumption of a static background and a fixed camera position. A good survey on these assumptions and recognition systems can be found in [Moeslund & Granum 01].

Subtraction is widely used by simply subtracting the previous image from the current

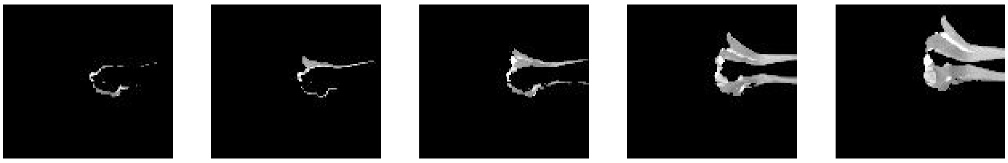


Figure 3.3. Absolute difference sequence of the gesture "Daumen Rechts"

image in a pixel-by-pixel fashion, and the use of three consecutive frames instead of two is an extended version [Haritaoglu & Harwood+ 98].

Also the use of background subtraction is very popular, and a more advanced version is to update the background image during processing but using this technique the system has to be initialized before recognizing any gesture. Kim et al. [Kim & Chalidabhongse+ 04] models the background using quantization/clustering techniques and a comparison of this technique and other background subtraction techniques are presented.

One of the research topics of this thesis is to investigate if different subtraction methods play a role in the recognition of gestures. We analysed three methods for subtracting images: absolute difference, first derivative and second derivative.

Absolute Difference

The absolute difference image $\widetilde{X}_t$ corresponding to the original image X_t is calculated as follows:

$$\widetilde{X}_t(x,y) = |X_{t-1}(x,y) - X_t(x,y)| + |X_t(x,y) - X_{t+1}(x,y)| \tag{3.1}$$

This feature allows to detect shape and motion, but not the direction of the motion. Figure 3.3 shows five absolute difference frames of the gesture "Daumen Rechts" where the right hand with extended thumb is moving to the right inside a car, i.e. to the top of the image.

First Derivative

The first time derivative difference image $\widetilde{X}_t$ corresponding to the original image X_t is calculated as follows:

$$\widetilde{X}_t(x,y) = X_{t+1}(x,y) - X_{t-1}(x,y) \tag{3.2}$$

This feature allows to detect shape, motion, and the direction of motion in each frame disregarding the current frame X_t.

Changes in the number of subtraction images or the temporal distance between them influences for example the bounding-box tracking (see Section 5.1): with normal

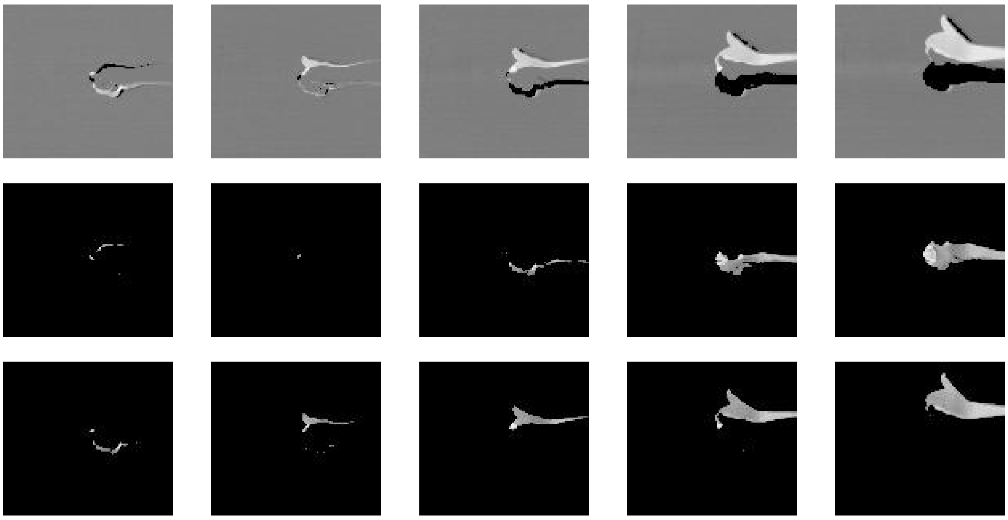


Figure 3.4. First derivative difference sequence of the gesture "Daumen Rechts": the first row shows the first time derivative sequence $\widetilde{X}_t$, the second row shows the negative part $\widetilde{X}_t^-$ and the third row shows the positive part $\widetilde{X}_t^+$

difference function $t - (t+1)$ worse results are obtained than with first derivative $(t+1) - (t-1)$ function. On the other hand using $t-(t+1)$ can improve the error rate obtained with the COG-features presented in Section 3.2 on the DUISBURG-Gesture database (see Section 6.2) from 13% to 10%.

Additionally, this difference image can be split into a negative and positive difference image as follows:

$$\begin{aligned} \widetilde{X}_t^-(x,y) &= |\widetilde{X}_t(x,y)|, \quad \text{if } \widetilde{X}_t(x,y) < 0, \text{ otherwise } 0 \\ \widetilde{X}_t^+(x,y) &= \widetilde{X}_t(x,y), \quad\;\; \text{if } \widetilde{X}_t(x,y) > 0, \text{ otherwise } 0 \end{aligned} \tag{3.3}$$

Figure 3.4 shows the same sequence as in Figure 3.3 but calculated with the first derivative difference method.

Thresholding the obtained time difference image by an appropriate value can reduce noise which can e.g. emerge from camera noise. This can also be helpful for tracking but the results from Section 7.1.3 show that the outcome of thresholding the difference images results in a loss of features, too.

Second Derivative

The second time derivative difference image $\widetilde{X}_t$ corresponding to the original image X_t is calculated as follows:

$$\widetilde{X}_t(x,y) = X_{t-1}(x,y) - 2 \cdot X_t(x,y) + X_{t+1}(x,y) \tag{3.4}$$

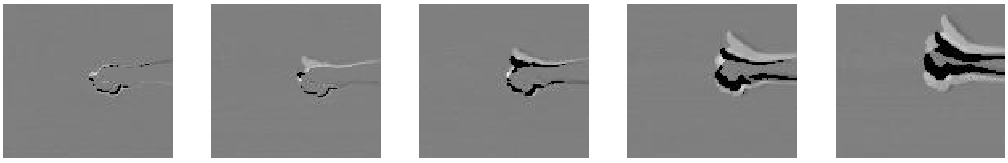


Figure 3.5. Second derivative difference sequence of the gesture "Daumen Rechts"

This feature allows to detect motion and the direction of motion in each frame, taking into account the current frame X_t. Also this difference image can be split into a negative and positive difference image.

Figure 3.5 shows the same sequence as in Figure 3.3 but calculated with the second derivative difference method.

3.1.3 Skin Color Images

The skin color model used in this work is based on the Compaq Cambridge Research Lab image-database presented in [Jones & Rehg 98] and [Jones & Rehg 02]. The skin color probability histograms were generated from 3077 pictures containing masked skin regions and 6286 pictures not containing skin, i.e. from a dataset of nearly 1 billion labelled pixels.

The probability s of a specified color c being skin color is calculated according to the Bayes formula:

$$p(s|c) = \frac{p(c|s) \cdot p(s)}{p(c|s) \cdot p(s) + p(c|\bar{s}) \cdot p(\bar{s})} \tag{3.5}$$

where c is a representation of the RGB of the image at a specific position, $p(s)$ is the overall skin probability and $p(\bar{s}) := 1 - p(s)$. The probabilities $p(c|s)$ and $p(c|\bar{s})$ are read from the given skin- and non-skin color models which were estimated from a larger collection of pictures.

Skin probability images denoted as S were created according to their skin probability maps. Therefrom one can also segment the original image X by its own skin color probability where T_p is a suitable skin color probability threshold:

$$X(x, y) = \begin{cases} S(x, y) & \text{if } S(x, y) > T_p \\ 0 & \text{otherwise} \end{cases} \tag{3.6}$$

Applying a Gaussian filter on the skin color probability map before segmenting (thresholding) the original image can improve the segmentation as gaps in contiguous skin regions are reduced (smoothed). Instead of a fixed threshold, the segmentation

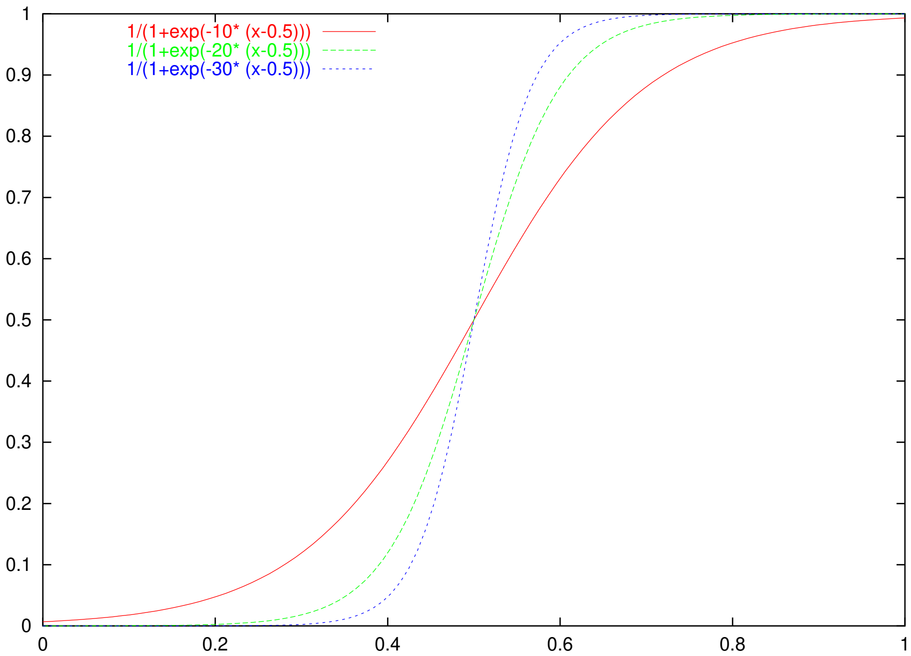


Figure 3.6. Example of three different sigmoid functions with $\alpha = 10, 20, 30$ and $T_p = 0.5$ which improve the segmentation of original images with skin color probability maps

can be improved once again by using a sigmoid function like:

$$X(x,y) = \begin{cases} \frac{1}{1+exp(-\alpha\cdot(S(x,y)-T_p))} & \text{if } S(x,y) > T_p \\ 0 & \text{otherwise} \end{cases} \tag{3.7}$$

Some sigmoid functions usable for skin color segmentation are shown in Figure 3.6.

Figure 3.7 shows the differences between normal skin color probability maps and Gaussian filtered skin color probability maps with their thresholding results. One can see that the original image thresholded by a sigmoid function with a smoothed skin color probability map has less artifacts and gaps.

These Gaussian and sigmoid smoothing functions to segment skin regions are not necessarily the optimal methods and many alternative algorithms have been suggested (e.g. [Raja & McKenna$^+$ 98], [Sigal & Sclaroff$^+$ 00] or [Zhu & Yang$^+$ 00]).

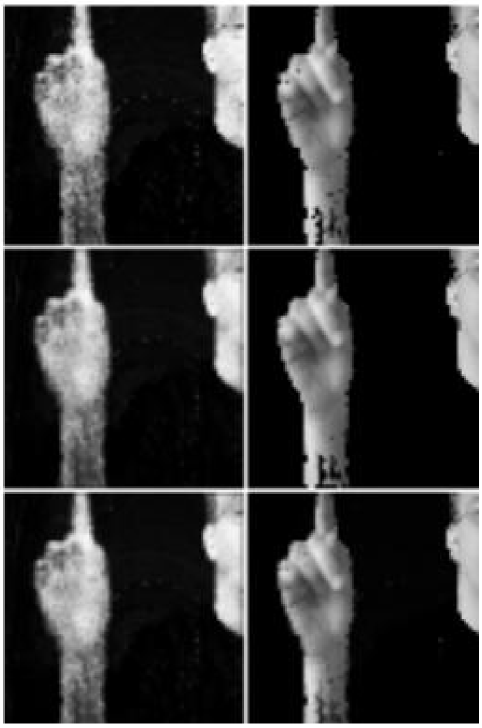


Figure 3.7. Skin color segmentation: the first row shows the skin color probability map and the original image thresholded by a fixed value, the second row shows the skin color probability map modified by a Gaussian filter and the original image thresholded by a fixed value, the third row the same skin color probability map but the original image thresholded by a sigmoid function. The original image thresholded by a sigmoid function with a smoothed skin color probability map has less artifacts and gaps.

Figure 3.8 shows some examples of possible features derived from skin color probability maps.

3.1.4 Energy and History Images

The notions motion-energy-image (MEI) and motion-history-image (MHI) were introduced by [Bobick & Davis 01]. The basic idea is to construct a vector-image that can be matched against stored representations of known movements. This image is used as a temporal template.

A binary MEI $E_\tau(x, y)$ is defined as follows:

$$E_\tau(x, y, t) = \bigcup_{i=0}^{\tau-1} \widetilde{X}_{t-i}(x, y) \tag{3.8}$$

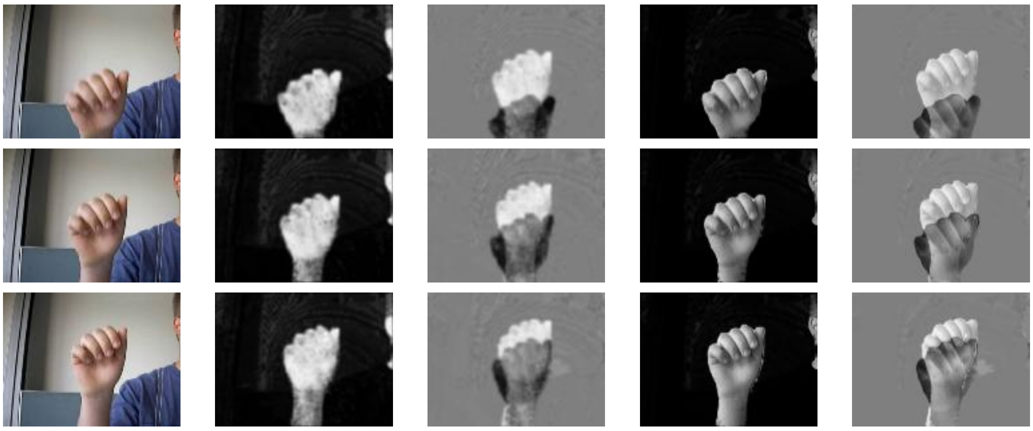


Figure 3.8. Skin color image features: original, skin probability, 1^{st} time derivative of skin probability, original thresholded by skin probability and 1^{st} time derivative of original thresholded by skin probability

The duration τ is critical in defining the temporal extent of a movement and they derive a backward-looking (in time) algorithm that dynamically searches over a range of τ in recognition. Thus a MEI describes *where* motion occurs.

To represent *how* (as opposed to where) motion in the image is moving, a motion-history image (MHI) is formed. In an MHI H_τ , pixel intensity is a function of the temporal history of motion at that point and τ is used as a simple replacement and decay operator (with $1 \leq \tau \leq N$ for a sequence of length N):

$$H_\tau(x,y) = \begin{cases} \tau & \text{if } \widetilde{X_t}(x,y) > T_0 \\ \max(0, H_\tau(x,y,t-1)-1) & \text{otherwise} \end{cases} \tag{3.9}$$

The result is a scalar-valued image where more recently moving pixels are brighter. Note that the MEI can be generated by thresholding the MHI above zero. Figure 3.9 shows a key frame with its corresponding MHI and MEI.

Skin history image (SHI) features are an extension of the MHI and were presented by [Morrison & McKenna 04]. Analogously to the motion history image of [Bobick & Davis 01] a skin history image is defined using a replacement and decay factor:

$$H_\tau(x,y) = \begin{cases} \tau & \text{if } S_t(x,y) > T_p \\ \max(0, H_\tau(x,y,t-1)-1) & \text{otherwise} \end{cases} \tag{3.10}$$

This results in a scalar-valued image in which pixels that are currently skin coloured are brightest, pixels that have not been skin coloured for some time are darker and pixels that have never been skin coloured during the last τ frames are black. Figure 3.10 shows a key frame with his corresponding SHI and SEI.

(a) (b) (c)

Figure 3.9. Motion energy and history image examples on the DUISBURG-Gesture database: the original key frame (a) at time $t = 47$ of the gesture "Round-Clockwise" with the corresponding motion-history-image (b) and motion-energy-image (c)

(a) (b) (c)

Figure 3.10. Skin energy and history image examples on the i6-Gesture database: the original key frame (a) at time $t = 44$ of the gesture "J" with the corresponding skin-history-image (b) and skin-energy-image (c)

3.2 Centroid Features

The center of gravity (COG) is a common feature to trace the movement path of a gesture: e.g. the difference between two consecutive COGs provides information about the movement speed and direction.

[Rigoll & Kosmala$^+$ 98] also analyzed other centroid features with which they were able to differentiate between 24 dynamic gestures (see Section 6.2).

They analyzed the following centroid features, where $\widetilde{X}_t$ is an ordinary time derivative image $X_t - X_{t-1}$:

- the center of gravity in x direction

$$M_t^x = \frac{\sum_{x,y} |x \cdot \widetilde{X}_t(x,y)|}{\sum_{x,y} |\widetilde{X}_t(x,y)|} \tag{3.11}$$

- the center of gravity in y direction

$$M_t^y = \frac{\sum_{x,y} |y \cdot \widetilde{X}_t(x,y)|}{\sum_{x,y} |\widetilde{X}_t(x,y)|} \tag{3.12}$$

- the mean absolute deviation from the center of gravity in x direction

$$A_t^x = \frac{\sum_{x,y} |(M_t^x - x) \cdot \widetilde{X}_t(x,y)|}{\sum_{x,y} |\widetilde{X}_t(x,y)|} \tag{3.13}$$

- the mean absolute deviation from the center of gravity in y direction

$$A_t^y = \frac{\sum_{x,y} |(M_t^y - y) \cdot \widetilde{X}_t(x,y)|}{\sum_{x,y} |\widetilde{X}_t(x,y)|} \tag{3.14}$$

- the positive center of gravity in x direction

$$M_t^{x^+} = \frac{\sum_{x,y|\widetilde{X}_t(x,y)>0} |x \cdot \widetilde{X}_t(x,y)|}{\sum_{x,y|\widetilde{X}_t(x,y)>0} |\widetilde{X}_t(x,y)|} \tag{3.15}$$

- the positive center of gravity in y direction

$$M_t^{y^+} = \frac{\sum_{x,y|\widetilde{X}_t(x,y)>0} |y \cdot \widetilde{X}_t(x,y)|}{\sum_{x,y|\widetilde{X}_t(x,y)>0} |\widetilde{X}_t(x,y)|} \tag{3.16}$$

- the negative center of gravity in x direction

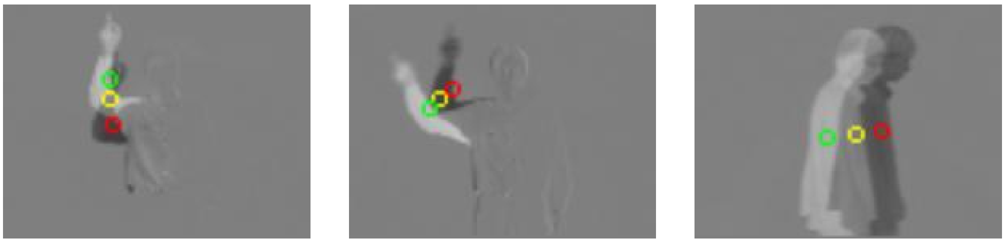
$$M_t^{x^-} = \frac{\sum_{x,y|\widetilde{X}_t(x,y)<0} |x \cdot \widetilde{X}_t(x,y)|}{\sum_{x,y|\widetilde{X}_t(x,y)<0} |\widetilde{X}_t(x,y)|} \tag{3.17}$$

- the negative center of gravity in y direction

$$M_t^{y^-} = \frac{\sum_{x,y|\widetilde{X}_t(x,y)<0} |y \cdot \widetilde{X}_t(x,y)|}{\sum_{x,y|\widetilde{X}_t(x,y)<0} |\widetilde{X}_t(x,y)|} \tag{3.18}$$

- the distance between the positive and negative center of gravity in x direction

$$dM_t^x = M_t^{x^+} - M_t^{x^-}; \tag{3.19}$$



(a) (b) (c)

Figure 3.11. Some examples of the centroid features on the DUISBURG-Gesture database, where a green circle means the positive COG, a yellow circle the COG and a red circle the negative COG: (a) Centroids of one frame of gesture “To-Top”, (b) Centroids of one frame of gesture “Round-Counterclockwise”, (c) Centroids of one frame of gesture “To-Right”

- the distance between the positive and negative center of gravity in y direction

$$dM_t^y = M_t^{y^+} - M_t^{y^-}; \tag{3.20}$$

- the overall intensity of motion

$$G_t = \sum_{x,y} |\widetilde{X}_t(x, y)| \tag{3.21}$$

An error rate of 7.1 % (with hold out method) was achieved by [Rigoll & Kosmala+ 98] on the DUISBURG-Gesture database with the seven features (3.11), (3.12), (3.13), (3.14), (3.19), (3.20), and (3.21). In the following, this feature set will be called “COG-features”.

Figure 3.11 shows a difference image with an overlayed feature vector.

Chapter 4

Hidden Markov Models

The ability of Hidden Markov models (HMMs) to compensate time and amplitude variations has been proven for speech recognition [Jelinek 98], gesture recognition [Schlenzig & Hunter$^+$ 94], [Pavlovic & Sharma$^+$ 97], [Bobick & Wilson 97], sign language [Starner & Weaver$^+$ 98], [Vogler & Metaxas 01], and human actions like walking or acrobatic [Brand & Oliver$^+$ 97], [Moore & Essa 02], [Nguyen & Bui$^+$ 03].

Hidden Markov models have a number of parameters, whose values are set to best explain training patterns for the known category. Test patterns are classified by the model that has the highest posterior probability, i.e. that best explains the test pattern [Duda & Hart$^+$ 01].

During the last 10 years many research groups have used HMMs in gesture recognition [Yamato & Ohya$^+$ 92], [Darrell & Pentland 93], [Schlenzig & Hunter$^+$ 94], [Wilson & Bobick 95], [Yang & Xu$^+$ 97], [Orio 99], [Wilson & Bobick 00], [Bretzner & Laptev$^+$ 01], [Starner & Leibe$^+$ 03]. Instead of words or phonemes, as in speech recognition, we have to consider gestures in this case.

In this work, we considered only whole gestures for models as opposed to speech recognition, where most of the speech recognition systems work on phoneme basis. This can be problematic when extending a corpus, because each time a word is added to the corpus, training material has to be added in order to train the according model. [Bauer & Kraiss 02] made experiments with HMMs using self-organizing subunits.

Recognizing sign language is much more complicated than recognizing gestures, that is why we decided to use whole word models. We focused especially on distance measures being invariant against slight affine transformations or distortions.

In Section 4.1 we present an overview of the theory of the HMMs and in Section 4.2 we present an overview on the distance measures.

4.1 The Theory of Hidden Markov Models

Problems that have an inherent temporality (a process that unfolds in time) may have states at time t that are influenced directly by a state at $t-1$. The idea of a HMM is to represent a signal by a state of a stochastic finite state machine. A more detailed description can be found in [Jelinek 98], and an often cited tutorial on HMM can be found in [Rabiner 89].

To classify an observation sequence X_1^T, we use the Bayesian decision rule:

$$\begin{aligned} X_1^T \longrightarrow r(X_1^T) &= \operatorname*{argmax}_k \left\{p(k|X_1^T)\right\} \\ &= \operatorname*{argmax}_k \left\{p(X_1^T, k)\right\} \\ &= \operatorname*{argmax}_k \left\{p(k) \cdot p(X_1^T|k)\right\} \end{aligned} \tag{4.1}$$

where X_1^T is a sequence with images $X_1, ..., X_t, ..., X_T$. Here, $p(k)$ is the a priori probability of class k, $p(X_1^T|k)$ is the class conditional probability for the observation X_1^T given class k and $r(X_1^T)$ is the decision of the classifier. This decision rule is known to be optimal with respect to the expected number of classification errors if the required distributions are known [Duda & Hart+ 01].

The class conditional probability of this observation is then defined using an HMM as follows:

$$p(X_1^T|k) = \sum_{s_1^T} p(X_1^T, s_1^T|k) \tag{4.2}$$

$$p(X_1^T, s_1^T|k) = \prod_{t=1}^{T} p(X_t, s_t|X_1^{t-1}, s_1^{t-1}, k) \tag{4.3}$$

where s_1^T is a time sequence of the states $s_1, ..., s_t, ..., s_T$.

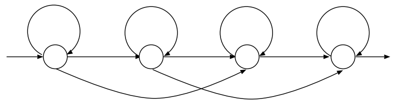
We assume now that the probability $p(X_t, s_t|X_1^{t-1}, s_1^{t-1}, k)$ depends only on the abstract states $s = s_1, ..., s_T$ of the gesture classes k (hidden: not observable). Now we can simplify (4.3) as follows:

$$p(X_t, s_t|X_1^{t-1}, s_1^{t-1}, k) = p(X_t, s_t|s_1^{t-1}, k) \tag{4.4}$$

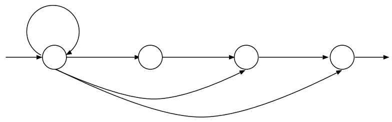
Another assumption is that the transition probabilities only depend on the predecessor state and that the emission probabilities depend on the reached state:

$$\begin{aligned} p(X_t, s_t|s_1^{t-1}, k) &= p(X_t, s_t|s_{t-1}, k) \\ &= \underbrace{p(s_t|s_{t-1}, k)}_{\text{Transition probability}} \cdot \underbrace{p(X_t|s_t, k)}_{\text{Emission probability}} \end{aligned} \tag{4.5}$$

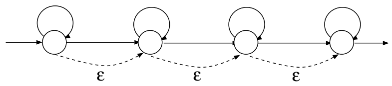
The transition probability $p(s_j|s_i, k) = a_{ij}$ is the time-independent probability of having state s_j at any time t given that the state at time $t-1$ was s_i. There is no requirement that the transition probabilities are symmetric and a particular state may be visited in succession.



(a)



(b)



(c)

Figure 4.1. Different HMM topology examples taken from [Ney 04] where the discrete states s are represented by nodes and the transition probabilities by links: (a) (0,1,2)-standard model, (b) model with long skips and (c) model with empty transitions (without observations)

We only used linear models in this work, e.g. the 0-1 model and the 0-1-2 model. The 0-1 model allows only loop and forward transitions whereas the 0-1-2 model additionally allows skip transitions. Figure 4.1 shows some examples of different HMM topologies.

In any state s_t we have a probability of emitting a particular visible state X_t, and we denote this emission probability by $p(X_t|s_t, k) = b_{ts}$.

Making these two assumptions we pass from a "First-order Markov model" to a "First-order hidden Markov model", because we have access to the visible states only, while the states s_t are unobservable and by inserting (4.5) in (4.2), we finally have:

$$
\begin{aligned}
p(X_1^T|k) &= \sum_{s_1^T} \prod_{t=1}^{T} p(s_t|s_{t-1}, k) \cdot p(X_t|s_t, k) \\
&= \sum_{s_1^T} \prod_{t=1}^{T} p(X_t, s_t|s_{t-1}, k) \qquad (4.6) \\
&\cong \max_{s_1^T} \left\{ \prod_{t=1}^{T} p(s_t|s_{t-1}, k) \cdot p(X_t|s_t, k) \right\}
\end{aligned}
$$

However, as neither $p(k)$ nor $p(X_1^T|k)$ are known in practical situations, it is necessary to choose models for the respective distributions and estimate their parameters using training data. The class conditional probabilities are modeled using Laplacian mixture densities (LMD), Gaussian mixture densities (GMD) or kernel densities (KD) in the experiments presented later.

The latter can be regarded as an extreme case of the mixture density model, where each training sample is interpreted as the center of a Gaussian distribution. A Gaussian mixture is defined as a linear combination of Gaussian component densities, which can (in principal) approximate any density function with arbitrary precision, even if only component densities with diagonal covariance matrices are used.

The observations belonging to a state s of a class k vary statistically and e.g. Gaussian or Laplacian distributions can be used to model these variations [Ney 99]. For a Gaussian distribution $N(\mu, \sigma)$ we have:

$$p(X_{td}|s_t, k) = \frac{1}{\sqrt{2\pi\sigma^2_{s_t kd}}} \cdot \exp\left(-\frac{1}{2}\left(\frac{X_{td} - \mu_{s_t d}}{\sigma_{s_t kd}}\right)^2\right) \tag{4.7}$$

where X_{td} component of the feature image X at time t, $\mu_{s_t d}$ mean and $\sigma^2_{s_t kd}$ variance.

To get the overall distribution for the image $X_t = [X_{t1}, ..., X_{td}, ..., X_{tD}]$ of dimensions D one has to multiply all components $d = 1, ..., D$:

$$p(X_t|s_t, k) = \prod_{d=1}^{D} p(X_{td}|s_t, k) \tag{4.8}$$

When assuming statistical independence of the components, the overall distribution is calculated as follows:

$$\begin{aligned} p(X_t|s_t, k) &= \prod_{d=1}^{D} p(X_{td}|s_t, k) \\ &= \frac{1}{\prod_{d=1}^{D} \sqrt{2\pi\sigma^2_{s_t kd}}} \cdot \exp\left(-\frac{1}{2}\sum_{d=1}^{D}\left(\frac{X_{td} - \mu_{s_t d}}{\sigma_{s_t kd}}\right)^2\right) \end{aligned} \tag{4.9}$$

The negative logarithm of $p(X_t|s_t, k)$ can be interpreted as a distance $d(p(X_t|s_t, k))$ and is used as emission score:

$$-\log(p(X_t|s_t, k)) = \frac{1}{2}\left(\sum_{d=1}^{D}\left(\underbrace{\left(\frac{X_{td} - \mu_{s_t d}}{\sigma_d}\right)^2}_{\text{distance}} + \underbrace{\log(2\pi\sigma^2_d)}_{\text{normalization factor}}\right)\right) \tag{4.10}$$

Often the variances are pooled over s and k, e.g. $\sigma_{s_t kd} = \sigma_d = \text{const}(s, k)$ is often used in image recognition as the image values all describe a similar measurement (of brightness). Variance thresholding, i.e. $\sigma_d \geq \sigma_0$ also is an important issue and can have high impact on the distances. This also turns out to be true for gesture recognition as shown in the results presented in Section 7.1.1. Using pooling, σ_d only depends on the vector component and is the same for all states and classes. This results in:

$$d(p(X_t|s_t,k)) = \frac{1}{2}\left(\sum_{d=1}^{D}\left(\left(\frac{X_{td}-\mu_{s_td}}{\sigma_d}\right)^2 + \text{const}(s,k)\right)\right) \quad (4.11)$$

Using a Laplacian distribution $L(\mu,\sigma)$ we have the following equations:

$$p(X_t|s_t,k) = \frac{1}{\prod_{d=1}^{D} 2\sigma_{s_tkd}} \cdot \exp\left(-\frac{1}{2}\sum_{d=1}^{D}\left|\frac{X_{td}-\mu_{s_td}}{\sigma_{s_tkd}}\right|\right) \quad (4.12)$$

$$-\log(p(X_t|s_t,k)) = \sum_{d=1}^{D}\left(\left|\frac{X_{td}-\mu_{s_td}}{\sigma_{s_tkd}}\right| + \log(2\sigma_{s_tkd})\right) \quad (4.13)$$

where X_{td} a vector component of the feature image X at time t represented as a feature vector, μ_{s_td} mean and σ_{s_tkd} variance.

Multimodal distributions are well modelled with mixture densities. In a typical case a weighted sum of Gaussian of Laplacian densities is used where each centre is identified with a unimodal density.

$$p(X_t|s_t,k) = \sum_{l=1}^{L(s_t,k)} p(X_t,l|s_t,k) \quad (4.14)$$

$$p(X_t,l|s_t,k) = p(l|s_t,k)\cdot p(X_t|s_t,k,l) \quad (4.15)$$

where $L(s_t.k)$ is the number of densities depending on state s_t and class k, $p(X_t|s_t,k)$ the multimodal distribution of state s_t from class k, $p(X_t|s_t,k,l)$ unimodal distribution for density l of state s_t from class k and $p(l|s_t,k)$ a mixture weight with normalization $\sum_l p(l|s_t,k) = 1$. The number $L(s_t,w)$ of component densities is kept constant.

Figure 4.2 shows the mean images for each state in the HMM of the gesture "Stop" obtained with difference features which shows the fast waving movement of the hand. Each image represents the trained mean values μ for one state s of a gesture class k obtained for a Gaussian kernel density with squared Euclidian distance.

Figure 4.3 shows two different alignments of the same gesture using a best score alignment of the HMM and a forced alignment to the correct model. (a) shows the alignment for the gesture "Two" which obtained the best score for gesture "Three", i.e. which was classified wrong. The upper row represents the trained mean images μ, the lower row an observation X_1^T. All images in the lower row are assigned to mean images in the upper row, but not every state was assigned to, e.g. state s_3 was not assigned to and all the observations $X_4,..,X_{14}$ were assigned to state s_4. Figure 4.4 shows their corresponding trellis diagrams.

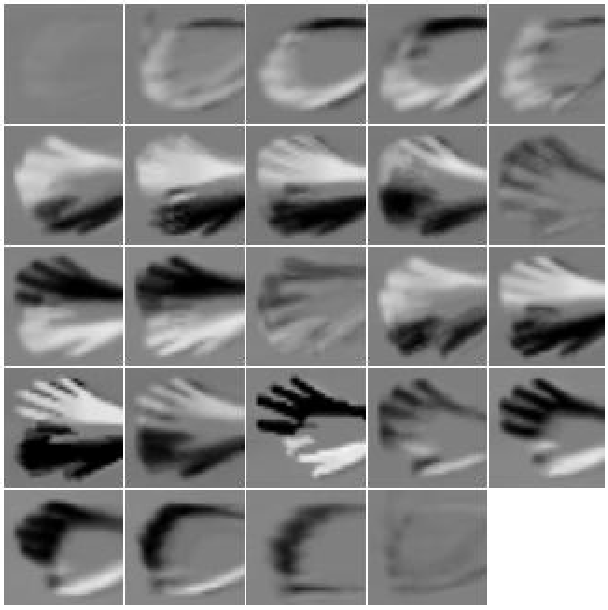


Figure 4.2. Mean images for each of the states obtained with 1^{st} difference feature in the HMM of the gesture "Stop"

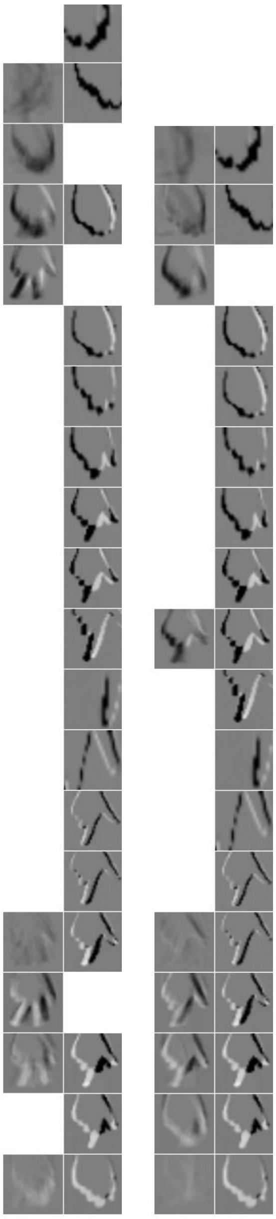
4.2 Distance Measures for the Emission Probabilities

In each state s_t of a HMM a distance has to be calculated. When working with image sequences, we have to calculate a distance between two images, e.g. we have to compare the current observation image X_t (or any transformed image $\widetilde{X}_t$) with the mean image μ_{s_t} at this state. Simply comparing the pixel values is quite often used in object recognition but different methods have been proposed to do this. An overview on methods for retrieving and comparing images can be found in [Deselaers 03].

One of the main topics in this diploma thesis is the use of different distance measures inside the HMM's emission probabilities. As in character or image recognition we want to analyze if transformation independent distance measures could improve the recognition performance.

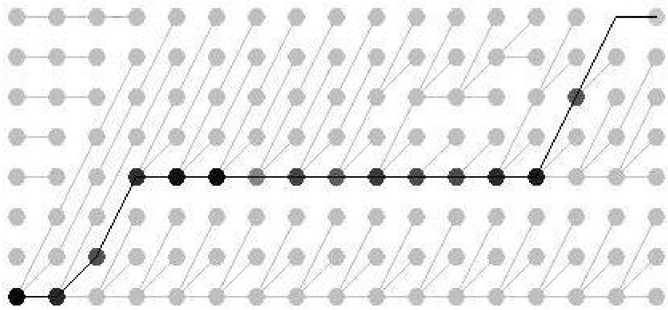
When calculating distances, thresholding the distance results by $\min(d(X_t, \mu), d_0)$ with d_0 between 5% or 10% of the maximum distance can improve recognition results [Keysers & Dahmen$^+$ 03].

Creating virtual data by rotating the images by $\pm\alpha$ degree can also improve recognition results [Dahmen & Keysers$^+$ 01]: when calculating a distance $d(X_t, \mu_{s_t})$ between an observation X_t and μ_{s_t}, the virtual data set $R(X_t) = \{X_t^{-\alpha}, X_t^0, X_t^{+\alpha}\}$ is created and the image from $R(X_t)$ with the smallest distance to the μ_{s_t} is chosen.

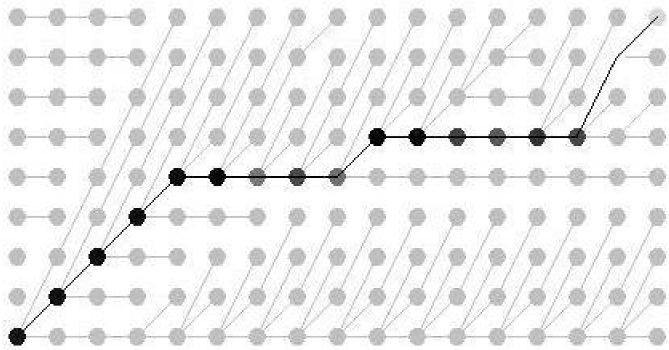


(a) **(b)**

Figure 4.3. Path alignment of the gesture "Two", where the upper row are the state images and the lower row the observation images: (a) wrong classification against the gesture "Three" (path with the best score), (b) forced alignment to the gesture "Two" (path with wors score)



(a)



(b)

Figure 4.4. Trellis diagram of the alignments, where the darkness of the points corresponds to the emission probability at this state: (a) shows the trellis of the winner alignment, (b) shows the trellis of the forced alignment

4.2.1 Minkowski Distances

If we have to compare two images $X_t = [X_{t1}, ..., X_{td}, ..., X_{tD}]$ and $\mu_{s_t} = [\mu_{s_t1}, ..., \mu_{s_td}, ..., \mu_{s_tD}]$ in a D-dimensional space, probably the most common distance measures are the ρ-Minkowski distances:

$$d_\rho(X_t, \mu_{s_t}) = \left(\sum_{d=1}^{D} |X_{td} - \mu_{s_td}|^\rho \right)^{1/\rho} \tag{4.16}$$

where $\rho = 1$ gives us the Manhattan distance and $\rho = 2$ the Euclidian distance.

Usually normalized distance measures are used, e.g. when using Laplace or Gaussian distributions, for example:

$$L_1\text{-norm} \quad := \quad d_1(X_t, \mu_{s_t}) = \sum_{d=1}^{D} \left| \frac{X_{td} - \mu_{s_t d}}{\sigma_d} \right| \tag{4.17}$$

$$\text{squared Euclidian} \quad := \quad d_2(X_t, \mu_{s_t}) = \sum_{d=1}^{D} \left(\frac{X_{td} - \mu_{s_t d}}{\sigma_d} \right)^2 \tag{4.18}$$

The Euclidian distance has been successfully used e.g. in optical character and object recognition and has been extended by different methods [Uchida & Sakoe 03].

4.2.2 Tangent Distance

As the Euclidian distance does not account for any image transformation (such as the affine transformations scaling, translation and rotation) if they are not part of the training corpus, the tangent distance (introduced by [Simard & Le Cun$^+$ 98]) as described in [Keysers & Macherey$^+$ 01] is one approach to incorporate invariance with respect to certain transformations into a classification system. Here, invariant means that image transformations that do not change the class of the image should not have a large impact on the distance between the images. A more detailed description can be found in [Keysers & Macherey$^+$ 04] and application in medical image classification can be found in [Keysers & Dahmen$^+$ 03].

Let $X \in \mathbb{R}^D$ be a pattern and $T(X, \alpha)$ denote a transformation of X that depends on a parameter L-tuple $\alpha \in \mathbb{R}^L$. We assume that T does not change class membership (for small α). The manifold of all transformed patterns $\mathcal{M}_X = \{T(X, \alpha) : \alpha \in \mathbb{R}^L\} \subset \mathbb{R}^D$ now offers new possibilities for distance calculations. The distance between two patterns X and μ can be defined as the minimum distance between the two manifolds $\mathcal{M}_X$ and $\mathcal{M}_\mu$, which is truly invariant with respect to the regarded transformations:

$$d_T(X, \mu) = \min_{\alpha, \beta \in \mathbb{R}^L} ||T(X, \alpha) - T(\mu, \beta)||^2 \tag{4.19}$$

The distance calculation between these manifolds is a hard non-linear optimization problem for which it is necessary to find faster techniques. In this case optimization is done using a tangent subspace approximation $\widetilde{\mathcal{M}}$. This subspace is spanned by a set of tangent vectors X^l that are the partial derivatives of the transformation T with respect to the parameters α_l. Thus, the transformation $T(X, \alpha)$ can be approximated using a Taylor expansion around $\alpha = 0$. The set of points consisting of all linear combinations of the tangent vectors X^l in the point X forms the tangent subspace $\widetilde{M_X}$. This is a first-order approximation of M_X.

The use of linear approximation allows to calculate the distances as a solution of a least squares problem or projections into subspaces. Both are computationally inexpensive operations.

In optical character recognition the use of six derivatives for affine transformations and one derivative accounting for line thickness yields very good results [Dahmen & Keysers+ 01]. In other domains (e.g. radiograph recognition) line thickness is replaced by brightness. In the results presented in Section 7.2.1 we used the first six derivatives for affine transformations.

Instead of calculating the distance between the current observation $\widetilde{X}_t$ and the mean image μ_{s_t}, first the mean is transformed by the function F to be invariant against affine transformations. Figure 4.5 shows a transformed mean image with respect to an observation image. The distance functions (4.17) and (4.18) are replaced by the following equations, where the distance is now calculated with the mean image $F(\mu_{s_t}, X_t)$ that is closest to X_t in the tangent subspace spanned by μ_{s_t}:

$$d_1(X_t, \mu_{s_t}) = \sum_{d=1}^{D} \left| \frac{X_{td} - F(\mu_{s_t}, X_{td})}{\sigma_d} \right| \tag{4.20}$$

$$d_2(X_t, \mu_{s_t}) = \sum_{d=1}^{D} \left(\frac{X_{td} - F(\mu_{s_t}, X_{td})}{\sigma_d} \right)^2 \tag{4.21}$$

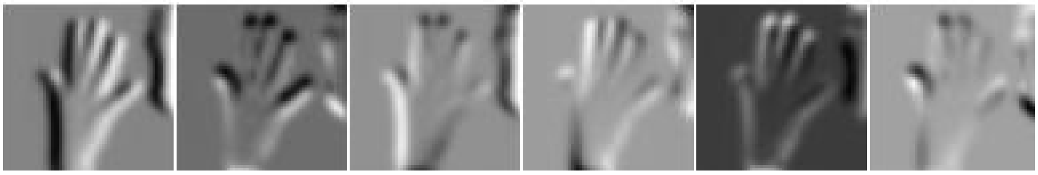
4.2.3 Image Distortion Model

The image distortion model has been examined earlier at the Lehrstuhl für Informatik VI of the RWTH Aachen [Keysers & Dahmen+ 03] and further research is presented in [Gollan 03]. The image distortion model is an easily implemented method allowing for small local deformations of an image. Each pixel is aligned to the pixel with the smallest squared distance from its neighborhood. These squared distances are summed up for the complete image to get the global distance. To compare an observation image X_t with a mean image μ_{s_t}, $d(X_t, \mu_{s_t})$ is calculated as follows:

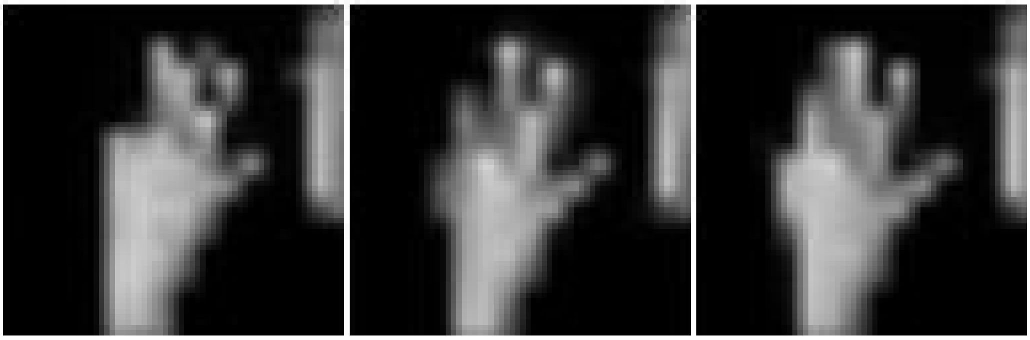
$$d_{\text{idm}}(X_t, \mu_{s_t}) = \sum_{x=1}^{N_x} \sum_{y=1}^{N_y} \min_{x'=x-w}^{x+w} \min_{y'=y-w}^{y+w} d'(X_t(x,y), \mu_{s_t}(x',y')) \tag{4.22}$$

Here, w is the warp range. That is, the radius of the neighborhood in which a pixel may be chosen for alignment and d' is a pixel distance comparing the image pixels $X_t(x,y)$ and $\mu_{s_t}(x',y')$ for example the Euclidian distance. This method can be improved by enhancing the pixel distance d' to compare sub images of size $(2v+1) \times (2v+1)$ instead of single pixels only:

$$d'(X_t(x,y), \mu_{s_t}(x',y')) = \sum_{i=-v}^{v} \sum_{j=-v}^{v} (X_t(x+i, y+j) - \mu_{s_t}(x'+i, y'+j))^2 \tag{4.23}$$



(a)



(b)

Figure 4.5. (a) shows one-sided tangents used for transformation in tangent distance: the mean image is transformed so that it is closest to X_t in the tangent subspace spanned by μ_{s_t}. The image shows the tangent vectors corresponding to the six affine transformations horizontal shift, vertical shift, first and second hyperbolic transformation, scaling and rotation of a mean image μ (from the i6-Gesture database) used to create the transformed mean image $F(\mu_{s_t}, X_t)$. (b) shows the result of such a distance calculation on the i6-Gesture database where the first image represents X_t, the second the transformed mean image $F(\mu_{s_t}, X_t)$ and the third the mean image μ.

Further improvement is achieved by using spatial derivatives instead of the pixel values directly. Intuitively, the use of derivatives makes the image distortion model align edges to edges and homogeneous areas to homogeneous areas.

In [Gollan 03] further methods for aligning images are proposed, but these are not considered here due to the high computational complexity and the low gain in classification performance in comparison to the IDM.

In our experiments in Section 7.2.2 we did not use the IDM distance directly to calculate the emission probabilities as we also wanted to analyze the effect of Laplace and Gaussian distributions. Instead we used the mean image $F(\mu_{s_t}, X_t)$ with the smallest IDM distance to X_t for respective distance function L_1-norm and Euclidian.

Figure 4.6 shows some examples of distorting mean images with respect to observations so that their pixel distance is minimal.

(a)

(b)

Figure 4.6. IDM distortion examples on I6-Gesture database, showing the effect of different patch sizes (f.l.t.r.: observation X_t, distorted mean image with the smallest distance $d'(X_t, \mu_{s_t})$, original mean image μ_{s_t}, vertical and horizontal Sobel images used for distortion): (a) with patch size 5x5 and (b) with patch size 3x3

4.3 Training and Classification

In the following section a short overview is given about the system developed in the course of this work.

For each training sequence $X_1^{T_1}, ..., X_1^{T_n}, ..., X_1^{T_N}$ of a gesture of class k with N sequences the feature images like time-derivative, spatial-derivative or skin-color are prepared and then extracted depending on a chosen tracking method, e.g. full-image extraction or extraction depending on dynamic-tracking results.

These extracted images are scaled down and used as feature vectors in the Viterbi training to train a hidden Markov model λ_k for each gesture. Usually we used the minimum sequence length of the sequences seen in training to estimate the number of states S in λ_k, .i.e $|S_{\lambda_k}| = \min_n(X_1^{T_n})$. Figure 4.7 shows this work flow.

The feature extraction of the test sequences is identical to the training process. Then for each test pattern the hidden Markov model which best describes the current observation sequence is searched. Figure 4.8 shows this work flow.

When searching the model λ_k that best explains the observation, position-synchronous pruning over all models can be applied to improve the runtime of the system. In speech recognition the system can usually be speedup by a factor 10 without degrading the error. In the experiments presented in Section 7.1.2 we could speedup the system by a factor 4 as we do not use such a large vocabulary as in speech recognition.

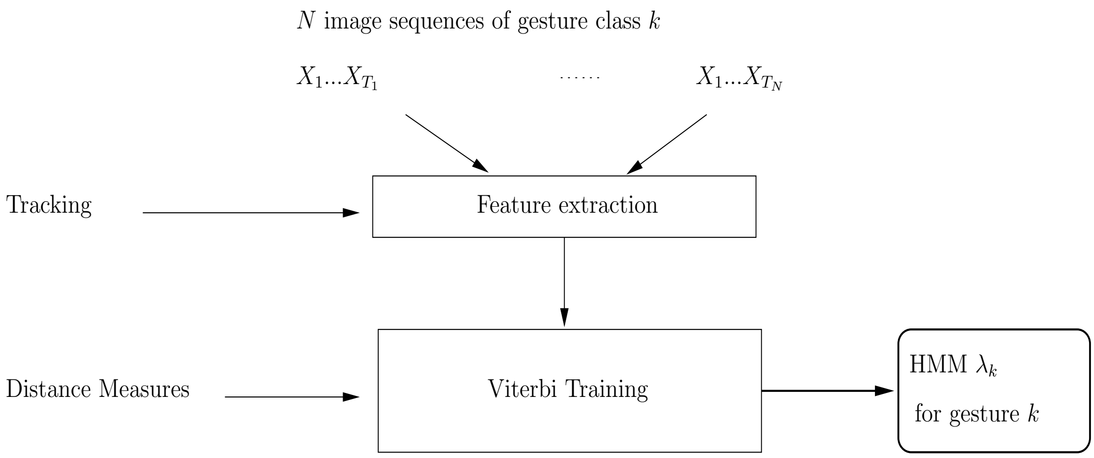


Figure 4.7. Training System

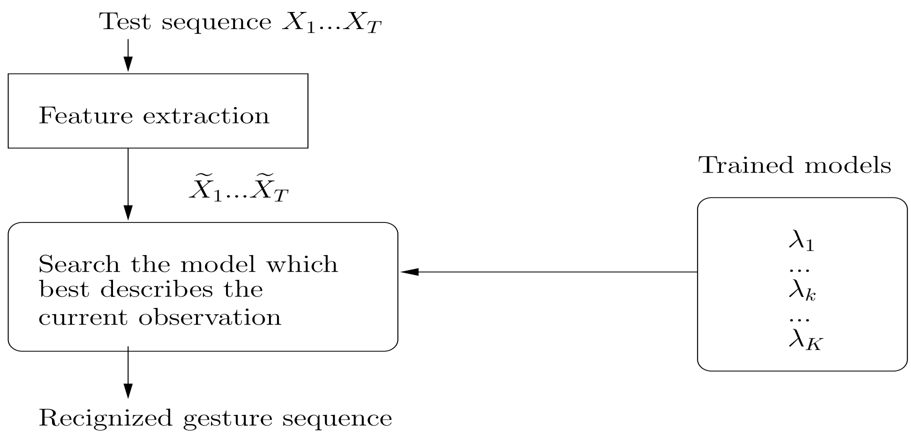


Figure 4.8. Classification System

Chapter 5

Tracking

When detailed information about moving objects in video sequences is needed, tracking comes into play. Tracking can be used e.g. to differentiate between multiple moving targets and to retrace their movements, to extract detailed image information or to detect self-occlusion from two tracked hands.

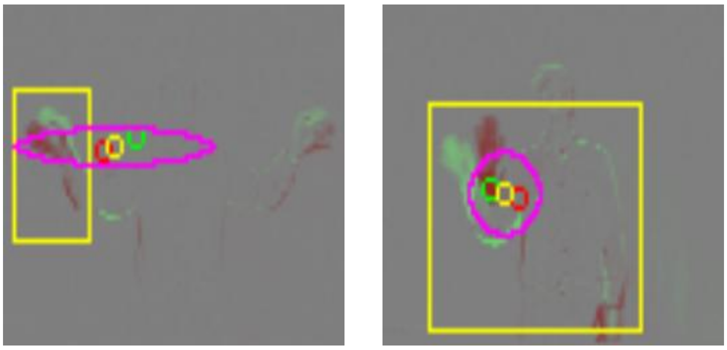
Some popular tracking methods are for example the Condensation tracking [Isard & Blake 98], Kalman filtering [Kalman 60], Meanshift tracking [Comaniciu & Ramesh+ 00] or Camshift tracking [Bradski 98].

As this work is not primarily a work on tracking we did not investigate the advantages and disadvantages of these popular tracking methods. A good overview of some tracking methods used in gesture and human movement recognition can be found in [Gavrila 99].

An introduction to Kalman filtering can be fond in [Welch & Bishop 03]. View-based location and tracking of body parts for visual interaction was analyzed by [Micilotta & Bowden 04], and [Gavrila & Giebel+ 04] shows the need of tracking in a vision-based pedestrian detection system. In [Balcells & DeMenthon+ 04] an appearance-based approach for consistently labeling people and for detecting human-object interactions using mono-camera surveillance video is presented.

In an appearance-based approach one usually uses a down scaled (transformed) image as feature vector, but when downscaling an image of size 320x240 down to 32x32 many visual details are lost due to the scaling operation. That is why we needed a method to extract only a specific part of the images, the area of interest, which will then be extracted, down scaled and used as feature vector. To achieve this we used as a first approach a simple bounding box tracking which is presented in Section 5.1.

In Section 5.3 we present a new kind of dynamic tracking algorithm which uses dynamic programming to reconstruct the path at the end of an observation sequence and thus the best tracking. We compared our results to the Camshift tracker presented in Section 5.2.



(a) (b)

Figure 5.1. Bounding-Box tracking examples for the Duisburg database in Section 6.2: (a) Tracking too small at the gesture "Hand-Waving-Both" because of a bad threshold, (b) Tracking too large at the gesture "Hand-Waving-Right" because of the moving clothing at bottom right side

5.1 Bounding-Box Tracking

One of the simplest and fastest tracking algorithms is the Bounding-Box tracking. It works on image sequences and uses difference images to track moving objects. It is very susceptible to noise or multiple moving objects (see Figure 5.1), because it tracks every pixel moving from one frame to another.

Depending on the difference calculation between frames (e.g. with or without adaptive background segmentation, difference calculation with two, three or more frames) and possible thresholds (e.g. skin color), the tracking may be controlled. This tracking method should only be used with a static background and one object to be tracked in front.

In fact, this is not a real tracking method, as no information about the object itself or any previous images are used to make a tracking decision. Nevertheless, this simple method can lead to very good results, if the assumptions about background and targets are true (cf. to results in Section 7.3.1).

5.2 Meanshift/Camshift Tracking

The Meanshift Algorithm was developed by [Comaniciu & Ramesh$^+$ 00]. They propose a real-time tracking algorithm of non-rigid objects based on visual features such as color and/or textures, whose statistical distributions characterize the object of interest.

It is robust to partial occlusions, clutters, rotation in depth and changes in camera position. The meanshift procedure earlier introduced by [Fukunaga & Hostetler 75] was used to find the target candidate that is most similar to a given target object.

The Camshift (continuously adaptive meanshift) tracking algorithm was developed by [Bradski 98]. Their computer vision color tracking algorithm was used for tracking human faces in real-time and is a modification of the mean shift algorithm, which was modified to deal with dynamically changing color probability distributions.

The implementation of the algorithm we used was taken from the LTI-Lib[1] and implemented as described in [Bradski 98].

5.3 Dynamic Tracking

This tracking algorithm prevents taking (possibly wrong) local decisions, because the tracking is done at the end of a sequence by making a traceback of the decisions to reconstruct the best path $t \to (x, y)$. This can be compared to time alignment in speech recognition.

Dynamic Tracking performs well when one wants to track an object with many occlusions, information gaps or for offline tracking. It may also be used with non-static background or multiple target objects in the foreground.

5.3.1 Dynamic Tracking with Fixed Size

Dynamic Tracking on large images is very time consuming, when making a full search of all possible tracking rectangles. In a first step we developed a simple tracking algorithm with fixed tracking rectangle size.

Since the dynamic tracking is an algorithm that can use any image score function, the minimum search window size $I \times J$ must be greater than one in order to calculate a score of at least one pixel, and in order to center the window, it should be of odd size. Thus for discrete functions or distributions, the minimum window size is set at 3x3.

In each dynamic programming algorithm, we need a recursion equation as in an HMM to calculate the best score over the whole image sequence, and to reconstruct the path which achieved this score. Our dynamic tracking recursion equation is defined as follows:

$$\begin{aligned} D(t,x,y) &= \max_{x',y' \in \mathcal{M}(x,y)} \left\{ (D(t-1,x',y') - \mathcal{J}(x',y',x,y) \right\} + d(X_t,x,y) \quad (5.1) \\ B(t,x,y) &= \operatorname*{argmax}_{x',y' \in \mathcal{M}(x,y)} \left\{ (D(t-1,x',y') - \mathcal{J}(x',y',x,y) \right\} \end{aligned}$$

[1] The LTI-Lib is an object oriented library with algorithms and data structures frequently used in image processing and computer vision. See `http://ltilib.sourceforge.net/doc/homepage/`

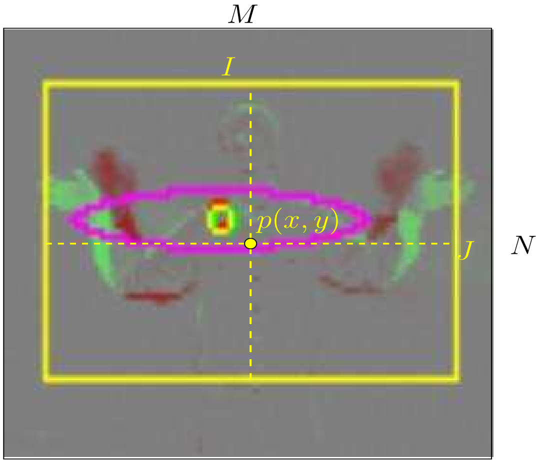


Figure 5.2. Dynamic Tracking example with first time derivative as image feature $\widetilde{X_t}$

$$
\begin{aligned}
\mathcal{M}(x,y) &= \text{Set of possible predecessors of point } (x,y) \\
d(X_t,x,y) &= \text{Tracking-score of the feature image } X_t \text{ at position } (x,y) \\
D(t,x,y) &= \text{Total-score for the best tracking until time } t \\
& \quad\, \text{which ended in position } (x,y) \\
B(t,x,y) &= \text{Backpointer at time } t \text{ of point } (x,y) \\
\mathcal{J}(x',y',x,y) &= \text{Jump-penalty from point } (x',y') \text{ to point } (x,y)
\end{aligned}
$$

Figure 5.2 shows one frame of a dynamic tracking sequence, where the current tracking window of size $I \times J$ is placed at a point (x, y) of the feature image (a first time derivative as image feature).

When using a first time derivative as image feature $\widetilde{X_t}$ the local score can be calculated by taking a weighted sum over the pixel values inside the tracking area, but any other score function may be defined here:

$$
d(\widetilde{X_t},x,y) = \sum_{\substack{x'=x-I/2\\ y'=y-J/2}}^{\substack{x'=x+I/2\\ y'=y+J/2}} w_{y'} \cdot w_{x'} \cdot |\widetilde{X_t}(x',y')| \tag{5.2}
$$

$$
w_{y'} = 1.5 - \frac{|y'-y|}{J/2}, w_{x'} = 1.5 - \frac{|x'-x|}{I/2}
$$

As jump-penalty one can use the following penalty functions where α is a relative weighting factor with respect to the score function:

- Euclidian distance:

$$\mathcal{J}(x', y', x, y) = \alpha \cdot \left(\sqrt{(x - x')^2 + (y - y')^2}\right) \tag{5.3}$$

- squared Euclidian distance:

$$\mathcal{J}(x', y', x, y) = \alpha \cdot \left((x - x')^2 + (y - y')^2\right) \tag{5.4}$$

- absolute distance:

$$\mathcal{J}(x', y', x, y) = \alpha \cdot \left(|x - x'| + |y - y'|\right) \tag{5.5}$$

One can control the tracking by the following parameters:

- tracking size I and J
- jump-width $\mathcal{M}(x, y)$ and -penalty $\mathcal{J}(x', y', x, y)$

Integrating backward pruning into the dynamic tracking algorithm can reduce the runtime as not each possible tracking center produces a high score and can thus be pruned. At time $t = 0$ each point (x,y) has to be initialized by the local score $d(X_{t=0}, x, y)$ and all points are activated as possible predecessors for time $t = 1$. From this time step until the end of the sequence, a point (x, y) will only be considered as a predecessor for time step $t+1$ if $D(t, x, y) < \max_{x,y}(D(t, x, y)) - T_0$ holds for a suitable pruning threshold T_0.

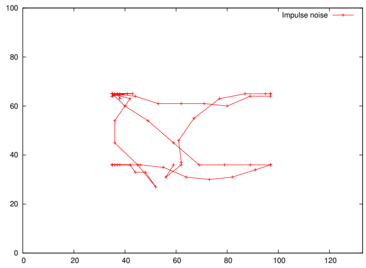
Dynamic Tracking and Camshift Tracking Under Noisy Circumstances. The advantage of considering the whole sequence before making any decision becomes apparent when tracking has to be done under noisy circumstances. To compare the dynamic tracking and camshift tracking algorithms, we added random noise in some sequences of the i6-Gesture database.

The noise was added in the sequences with the `convert`-Tool of ImageMagick[2]. This can be easily done by invoking e.g. `convert +noise Gaussian original.jpg noise.jpg` to add Gaussian noise to an image. We tested all available noise types: Uniform, Multiplicative, Gaussian, Impulse, Laplacian and Poisson[3].

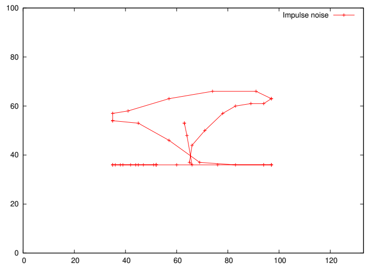
Figure 5.3 shows the dynamic tracking compared to the camshift tracker on a sequence with impulse noise. One can see that at this noise-level both trackers are able to follow the hand gesturing a "Z" of the German fingerspelling alphabet.

[2] ImageMagick is a robust collection of tools and libraries to read, write, and manipulate an image in many image formats `http://www.imagemagick.org/`

[3] A movie clip showing the tracking results with different noise levels is available from `http://www-i6.informatik.rwth-aachen.de/~dreuw/`



(a)



(b)

(c)

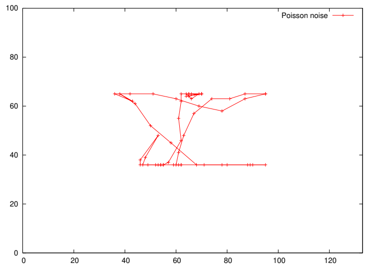
(d)

Figure 5.3. Tracking examples on impulse noise for the i6-Gesture database: (a) is a plot of the dynamic tracking window centers and (c) dynamic tracking examples at time $t = 4, 11, 18, 28, 36$. (b) and (d) are the corresponding camshift tracking examples. Both trackers are able to track the hand gesturing a "Z" of the German fingerspelling alphabet at this noise level.

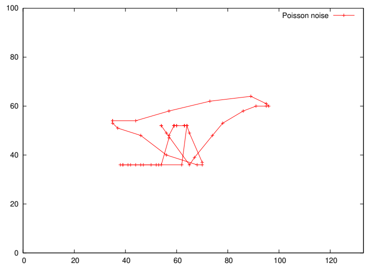
Figure 5.4 shows the dynamic tracking compared to the camshift tracker on a sequence with Poisson noise. At this noise-level The camshift is no longer able to track the hand at time $t = 28$ but still works. The dynamic tracker also looses some precision but performs still better than the camshift tracker.

Figure 5.5 shows the dynamic tracking compared to the camshift tracker on a sequence with Laplacian noise. Subfigure 5.5b shows that at this noise-level the camshift is no longer able to reasonably track the hand. Subfigure 5.5b shows that the dynamic tracker continuous to loose precision but still tracks good.

Figure 5.6 shows the dynamic tracking compared to the camshift tracker on a sequence with Gaussian noise. At this noise-level it is even very difficult for a human eye to track the movement of the hand. Subfigure 5.6b shows that the camshift does not track at all but Subfigure 5.6b shows that the dynamic tracker still tracks the hand.



(a)



(b)

(c)

(d)

Figure 5.4. Tracking examples on poisson noise for the i6-Gesture database: (a) is a plot of the dynamic tracking window centers and (c) dynamic tracking examples at time $t = 4, 11, 18, 28, 36$. (b) and (d) are the corresponding camshift tracking examples. The camshift is no longer able to track the hand around time $t = 28$ but still works.

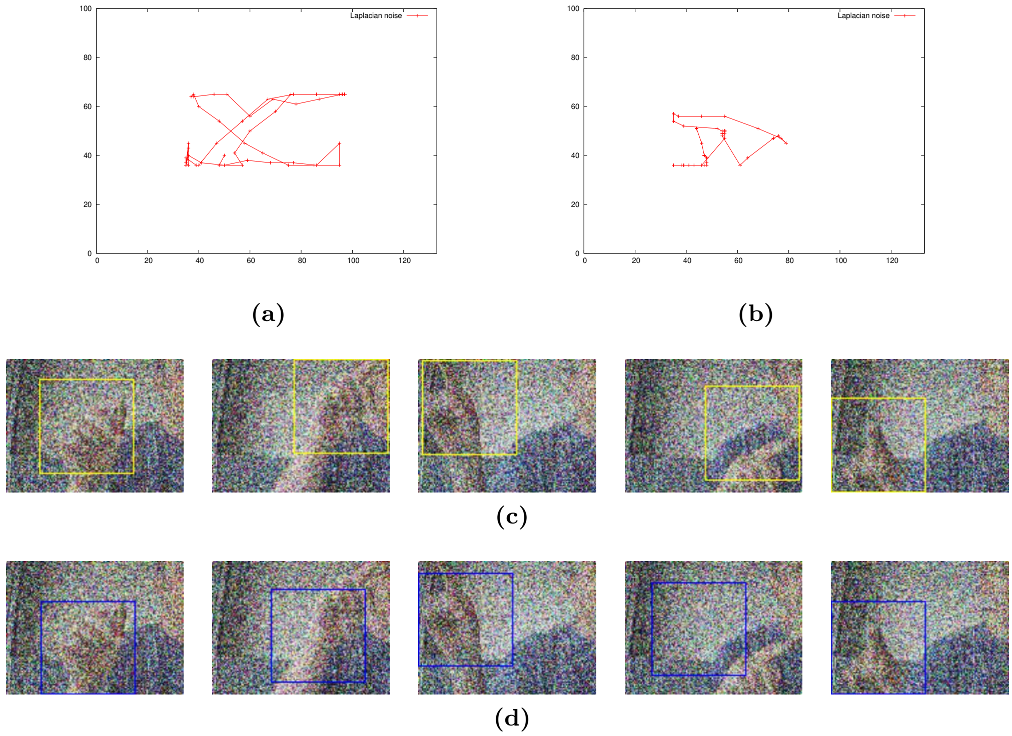
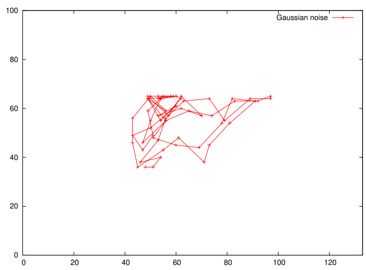
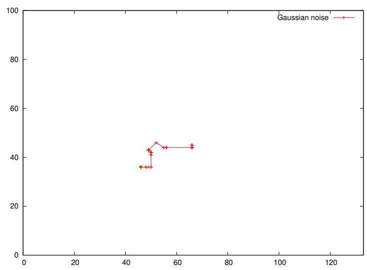


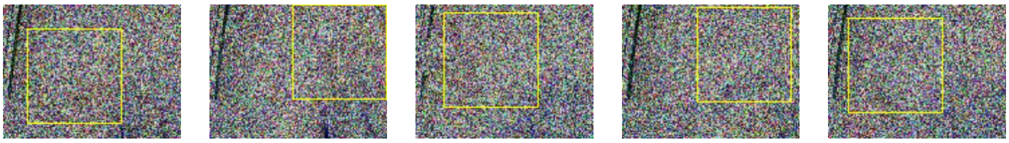
Figure 5.5. Tracking examples on Laplacian noise for the i6-Gesture database: (a) is a plot of the dynamic tracking window centers and (c) dynamic tracking examples at time $t = 4, 11, 18, 28, 36$. (b) and (d) are the corresponding camshift tracking examples. The camshift tracker is no longer able to reasonably track the hand. (a) shows that dynamic tracking still works good at this noise-level.



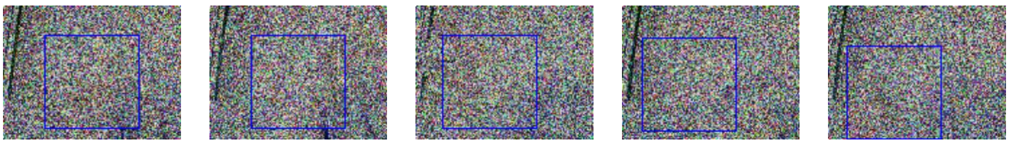
(a)



(b)



(c)



(d)

Figure 5.6. Tracking examples on Gaussian noise for the i6-Gesture database: at this noise-level it is even very difficult for a human eye to track the movement of the hand. (a) is a plot of the dynamic tracking window centers and (c) dynamic tracking examples at time $t = 4, 11, 18, 28, 36$. (b) and (d) are the corresponding camshift tracking examples. The camshift tracker does not track at all but (a) shows that dynamic tracking still works even if imprecise.

5.3.2 Dynamic Tracking with Variable Size

In many cases it might be necessary to allow a change of the tracking box size, which means a full search over all possible predecessors for the upper left and bottom right corner of the tracking rectangle. In this case we have to adapt our dynamic tracking recursion equation algorithm and jump penalty assumption as follows:

$$\begin{aligned}
D(t,x,y,r_x,r_y) &= \max_{\substack{x',y'\in\mathcal{M}(x,y)\\ r'_x,r'_y\in\mathcal{M}_\nabla(r_x,r_y)}} \{D(t-1,x',y',r'_x,r'_y) \\
&\quad -\mathcal{J}((x',y',r'_x,r'_y,x,y,r_x,r_y)\} \\
&\quad +d(\widetilde{X_t},x,y,r_x,r_y) && (5.6)\\
B(t,x,y) &= \operatorname*{argmax}_{\substack{x',y'\in\mathcal{M}(x,y)\\ r'_x,r'_y\in\mathcal{M}_\nabla(r_x,r_y)}} \{D(t-1,x',y',r'_x,r'_y) \\
&\quad -\mathcal{J}(x',y',r'_x,r'_y,x,y,r_x,r_y)\} && (5.7)\\
\mathcal{J}(x',y',r'_x,r'_y,x,y,r_x,r_y) &= \alpha\cdot\big((x-x')^2+(y-y')^2\big) \\
&\quad +\beta\cdot\big((r_x-r'_x)^2+(r_y-r'_y)^2\big) && (5.8)
\end{aligned}$$

Currently, a dynamic tracking search on the LTI-Gesture database runs on a AMD-2600XP machine approximately 1 hour with fixed size tracking and no pruning. Here we calculate what happens with variable size tracking by first making some assumptions:

- Image size: $(N_x \times N_y)$
- Number of possible tracking rectangles in one image: $(N_{r_x} \times N_{r_y})$
- Assume that
  - $(N_x \times N_y) = (N_{r_x} \times N_{r_y}) = (100 \times 100)$
  - Maximum jump width $\delta \pm 5$,
  - Maximum size change $\delta' \pm 5$

This will run into serious runtime problems with variable size. We would have the following runtime:

$$\begin{aligned}
(N_x\cdot N_y)\cdot(N_{r_x}\cdot N_{r_y})\cdot|\mathcal{M}(x,y)|\cdot|\mathcal{M}(r_x,r_y)| &= 100^2\cdot 100^2\cdot((\delta\cdot 2)+1)^2\cdot((\delta'\cdot 2)+1)^2 \\
&\cong \underbrace{1.2\cdot 10^6}_{\text{fixed size} = 1\text{ hr}} \cdot 1.2\cdot 10^6 \\
&\cong 68\text{ years!}
\end{aligned}$$

Even with using pruning as mentioned for the fixed size tracking and reducing the runtime from 1 hour down to 10 minutes, the runtime for the variable size tracking would be $0.1667\cdot 10^6 \cong 19$ years!.

5.3.3 Integration of Recognition and Dynamic Tracking

One idea when working with HMMs and tracking was the integration of the tracking inside the recognition process. Combining dynamic tracking and HMM recognition can be written as follows:

$$\begin{aligned} r(X_1^T) &= \underset{k}{\operatorname{argmax}} \left\{ p(k) \cdot p(X_1^T|k) \right\} \\ p(X_1^T|k) &= \sum_{[s_1^T, l_1^T]} p(X_1^T, s_1^T, l_1^T|k) \\ p(X_1^T, s_1^T, l_1^T|k) &= \prod_{t=1}^{T} p(X_t, s_t, l_t|X_1^{t-1}, s_1^{t-1}, l_1^{t-1}, k) \end{aligned} \tag{5.9}$$

with X_1^T sequence over time of the images $X_1, ..., X_t, ..., X_T$, s_1^T sequence over time of the states $s_1, ..., s_t, ..., s_T$ and l_1^T sequence over time of the locations $l_1, ..., l_t, ..., l_T$.

A location l_t in this notation form can be a simple point (x_t, y_t), e.g. the center of a tracking rectangle, a point with a specific range (x_t, y_t, r_t), e.g. the center of a tracking rectangle of size r_t or any other location $(x_t, y_t, r_t, ...)$.

Combining these two processes together can be very time consuming. To reduce this we have to make some additional assumptions to those from Section 4.1. We now assume that the probability $p(X_t, s_t, l_t|X_1^{t-1}, s_1^{t-1}, l_1^{t-1}, k)$ depends only on the abstract states $s = s_1, ..., s_T$ of the gesture classes k (which means "hidden" states). We can simplify (5.9) now as follows:

$$p(X_t, s_t, l_t|X_1^{t-1}, s_1^{t-1}, l_1^{t-1}, k) = p(X_t, s_t, l_t|s_1^{t-1}, l_1^{t-1}, k) \tag{5.10}$$

Another assumption is that the transition probabilities only depend on the predecessor state, and that the emission probabilities depend on the reached state as in Section 4.1:

$$\begin{aligned} p(X_t, s_t, l_t|s_1^{t-1}, l_1^{t-1}, k) &= p(X_t, s_t, l_t|s_{t-1}, l_{t-1}, k) \\ &= \underbrace{p(s_t, l_t|s_{t-1}, l_{t-1}, k)}_{\text{Transition probability}} \cdot \underbrace{p(X_t|s_t, l_t, k)}_{\text{Emission probability}} \end{aligned}$$

Additionally, we assume that the transition probability is independent from location, i.e.

$$p(s_t, l_t|s_{t-1}, l_{t-1}, k) = p(s_t|s_{t-1}, k) \cdot p(l_t|s_{t-1}, l_{t-1}, k) \tag{5.11}$$

and that the location probability only depends on the predecessor location, i.e.

$$p(s_t, l_t|s_{t-1}, l_{t-1}, k) = p(s_t|s_{t-1}, k) \cdot p(l_t|l_{t-1}, k) \tag{5.12}$$

Combining dynamic tracking and recognition into a HMM would change the emission probability to:

$$p(X_t|s_t, l_t, k) = \mathcal{N}(f(X_t, l_t)|\mu_{s_t}, \Sigma)$$

Then, the emission distribution can be modeled by the following equation:

$$N(f(X_t, l_t), l_t - l_{t-1}|\mu_{s_t}, \Sigma) = N(f(X_t, l_t), l_t - l_{t-1}|\mu^X_{s_t}, \mu^l_{s_t}, \Sigma)$$

Assuming for the transition probability that the state probability stays the same, one could assume two cases for the location probabilities:

$$-\log(p(l_t|l_{t-1}[, s_{t-1}], k)) = \begin{cases} \alpha \cdot ||l_t - l_{t-t}||^2 \\ \alpha \cdot ||(l_t - l_{t-1}) - \mu^l_{s_{t-1}}||^2 \end{cases}$$

which can be interpreted as that the distance between two locations will be learned with the HMM.

All these assumptions presented here have not been tested in this work and present only an interesting proposal for future works.

Chapter 6

Databases

In this chapter we present the three databases used to test our applications. A database with infrared images inside a car as well as a database with body centered grey-scale images to control robots were applied. A new database with color images of the German fingerspelling or manual alphabet is introduced.

Results other groups obtained on these databases are given along with our results in Chapter 7.

6.1 LTI-Gesture Database

The LTI-Gesture database was created at the Chair of Technical Computer Science[1] at the RWTH Aachen and is not freely available. It contains 14 dynamic gestures. The resolution of each video sequence is 106 x 96 grey-scale pixel. The videos were recorded with an infrared camera inside a car. In total, 364 sequences were recorded, of which 84 are used for testing and 280 are used for training. Figure 6.1 shows some examples of the different gestures.

The 280 sequences dedicated for training a continuous gesture recognition system were splitted into a train and test set. These two sets were then used to test a single gesture recognition system.

This database was also used in [Pelkmann 99] where a combination of several geometric features such as compactness, Hu-Moments and global motion as gesture features achieved an error rate of 4.5% on this database, using hidden Markov Models as statistical classifiers.

Unfortunately it was not clearly and fully indicated in [Pelkmann 99] how the data was split into a training and testing database. The database we received contains only 276 sequences, i.e. some of the 14 gestures only contain 19 sequences instead of 20.

We decided to create two setups, denoted as "1^{st} split" and "2^{nd} split" to be comparable with their results. The former consists of the first 10 (or 9) sequences of each class for training and the last 10 for testing, the latter consists of the last 10 (or 9)

[1] Visit `http://www.techinfo.rwth-aachen.de`

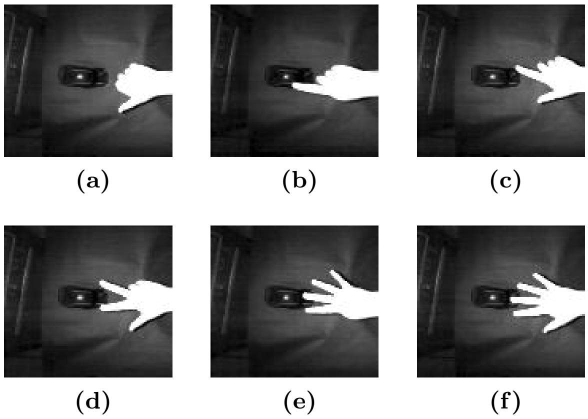


Figure 6.1. Some examples of the LTI-Gesture database showing different gestures of numbers: (a) one thumb, (b) one finger, (c) two, (d) three, (e) four, (f) five

sequences for training and the first 10 for testing. This means that we have always 140 test sequences as in [Pelkmann 99] but only 136 training data sequences.

6.2 DUISBURG-Gesture Database

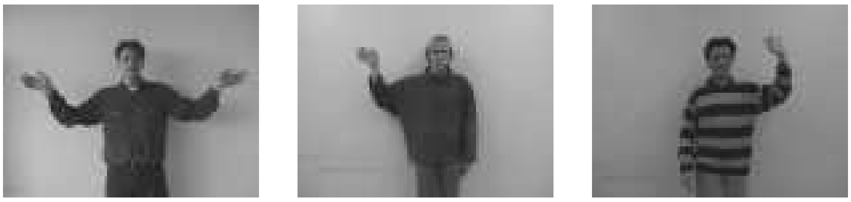
The goal of the research project described in [Rigoll & Kosmala+ 98] was the recognition of gestures to control robots. The recognition system is capable of recognizing 24 different gestures. The database consists of 336 image sequences that contain gestures of 12 different persons.

For the training and the testing of the system video sequences of 24 different gestures were recorded. The resolution of the video sequences was 96 x 72 gray-scale pixel and 16 frames per second. Each sequence consists of 50 frames, resulting in a sequence length of approximately three seconds. Figure 6.2 shows some examples of the different gestures.

6.3 i6-Gesture Database

In the course of this work, a new database of fingerspelling letters of German Sign Language (Deutsche Gebärdensprache, DGS) was created[2]. The database contains 35 gestures with video sequences for the signs A to Z and SCH, the German umlauts Ä, Ö, Ü, and for the numbers 1 to 5. Five of the gestures contain inherent motion (J, Z, Ä, Ö and Ü).

[2]Our database is freely available at http://www-i6.informatik.rwth-aachen.de/~dreuw/



(a) (b) (c)

Figure 6.2. Some examples of the DUISBURG-Gesture database showing different gestures: (a) hand waving both, (b) hand waving left, (c) hand waving right

The recording was done under non-uniform daylight lighting conditions, the background and the camera viewpoints are not constant, and the persons had no restrictions on the clothing while gesturing. Figure 6.3 shows an overview of the alphabet from A to Z.

The database consists of 1400 image sequences that contain gestures of 20 different persons. Each person had to sign each gesture twice on two different days. The gestures were recorded by two different cameras, one webcam and one camcorder, from two different points of view. Figure 6.4 shows the record setup.

The webcam recorded the sequences with a resolution of 320x240 at 25 frames per second, and the camcorder with a resolution of 352x288 at 25 frames per second. The persons were not trained to perform the signs, therefrom the gestures may differ from the standard. Figure 6.5 shows some examples of the different gestures.

For recording the gestures we programmed a shell script which gave us the possibility of recording and converting gestures for as many persons as we wanted in a flexible and easy way. All videos were recorded in MPEG-4 DivX format using the freely available software MPlayer[3]. The script offers possibilities to easily integrate new recording devices or changing the record resolution and frame rate.

Also we programmed another shell script to convert the recorded videos into single image files. For each person, session, and camera a sequence file was generated which contains all images belonging to this sequence. We chose the PNG image format with high compression factor but one may change this to any other value. These two scripts are also available online.

Before recording, the proband was asked if he agrees in making his sequence publicly available. It was clearly mentioned that he could abandon the record-session at any time. After a short explanation of the course of events he had to sign a letter of agreement. This is a very important task when recording a proband with cameras: on one hand the proband exactly knows what will happen with his records and on the other hand the proband cannot defy with hindsight to the publishing of the complete

[3] `http://www.mplayerhq.hu/`

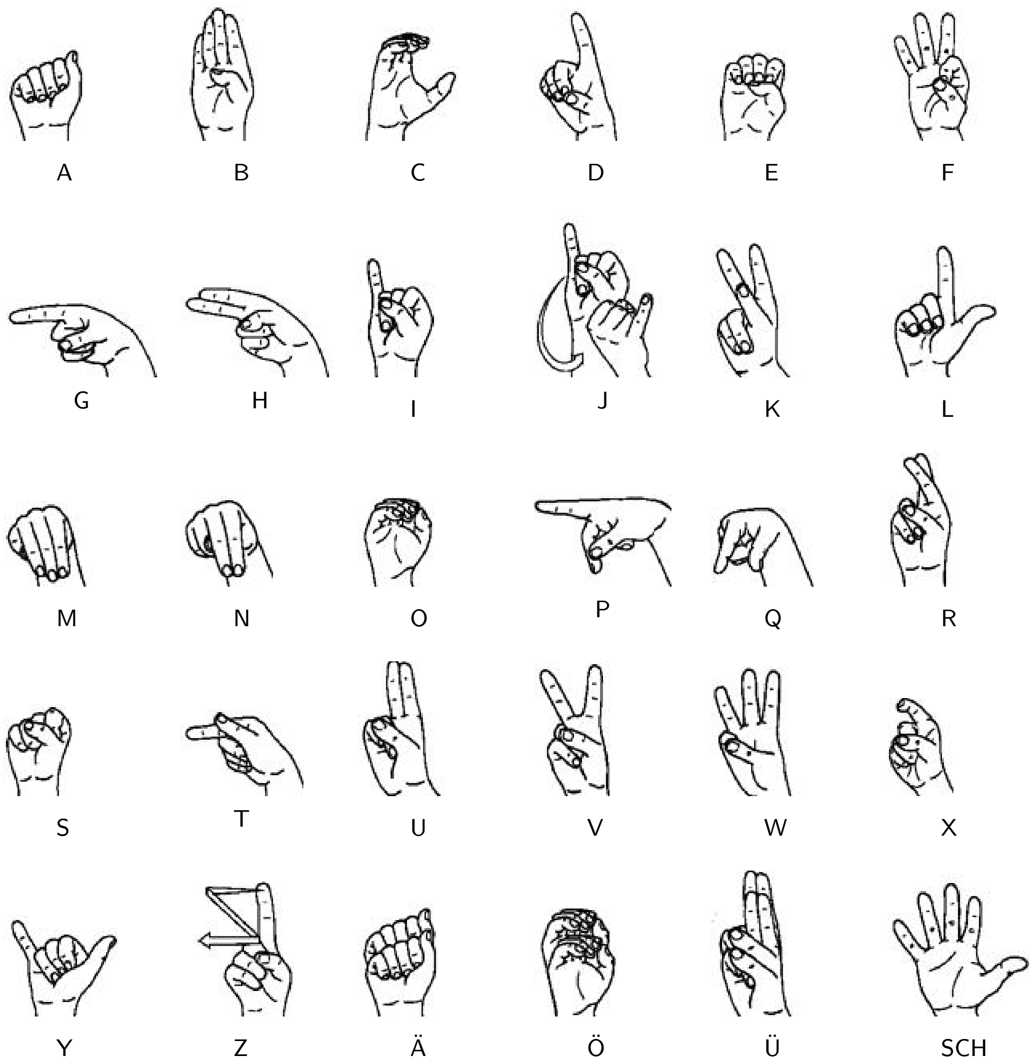


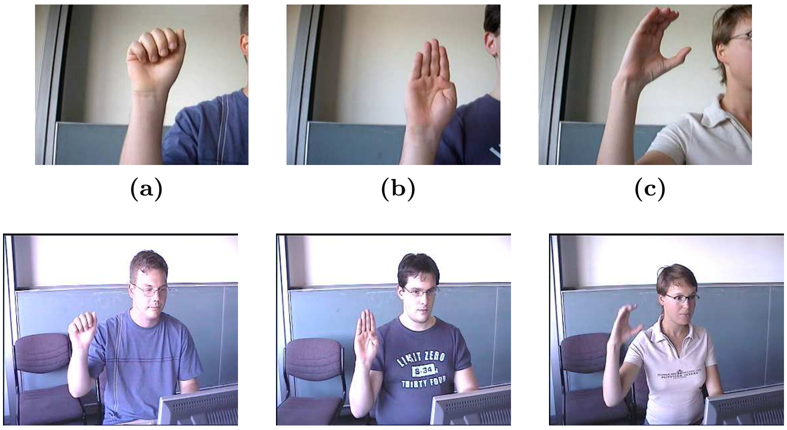
Figure 6.3. The German fingerspelling alphabet taken from
`http://www.sign-lang.uni-hamburg.de/fa/` visited 06.12.2005

database. A more detailed overview on usability evaluation and working with probands can be found in [Nielsen 00] and [Schweibenz & Thissen 02].

For each gesture an example video was shown before recording. The proband could view this video as often as he wanted. He then started the recording by hitting the RETURN-key and stopped it by hitting it again. Then his recording was displayed to be compared with the previous reference example. The proband could record his

(a) (b)

Figure 6.4. i6-Gesture database record setup: (a) record setup with the webcam on the desk and the camcorder on the chair, (a) record setup from proband sight, showing the lighting conditions



(a) (b) (c)

(d) (e) (f)

Figure 6.5. Some examples of the i6-Gesture database showing different gestures of letters: (a)-(c) the letters A-C recorded with the webcam, (d)-(f) the letters A-C recorded with the camcorder,

gesture as often as he wanted. One recording-session took between 10 and 20 minutes.

Figure 6.6 shows one example for each class of the i6-Gesture database with different persons. One can see the different lighting conditions. Also sometimes the hand is located in front of the face, which make is difficult to track and extract. No instructions concerning the clothing or jewellery like rings, bracelets or watches were given. We decided to record such a difficult database with respect to be able to build an online recognition system later which can work under no constraints.

More informations about the recording, the database, and the used hardware are available on our websites.

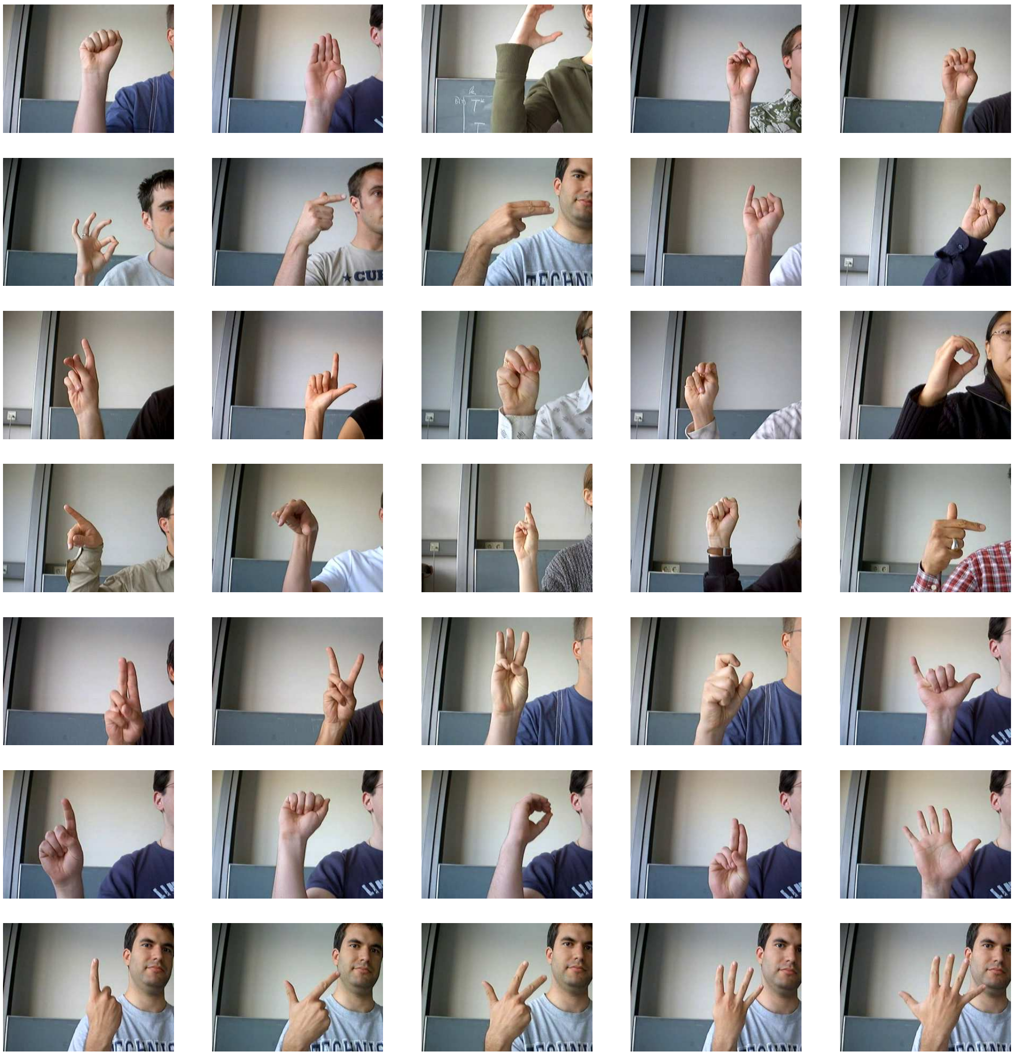


Figure 6.6. Examples of the German finger-alphabet taken from the i6-Gesture database recorded with the webcam showing the letters A-Z, Ä, Ö, Ü, SCH and the numbers 1 to 5. Note that J, Z, Ä, Ö and Ü are dynamic gestures.

Chapter 7

Experiments and Results

In this chapter the results for the HMM settings, the distance measures and the tracking methods are presented. If comparable results from other groups are available, these results are also presented here for comparison.

The promising approach of using appearance-based features in gesture and sign language recognition in combination with new distance measures is shown on different databases.

The achieved results on the LTI-Gesture database are always indicated with respect to either the 1^{st} split or the 2^{nd} split. The results on the DUISBURG-Gesture database were achieved with a leaving-one-person-out method like in [Rigoll & Kosmala^{+} 98] to be comparable. For the results on the i6-Gesture database we used the first session for training and the second session for testing. See Chapter 6 for further details.

7.1 Basic HMM Settings

The experiments presented in this section were made to build up a basic system and find out the appropriate HMM settings. All the experiments were made with the 1^{st} split dataset of the LTI-Gesture database (see Section 6.1).

7.1.1 Emission Distributions

First we had to adjust the minimum variances in the score function, e.g. we made some experiments concerning $\sigma_d \geq \sigma_0$ inside Equation 4.17 and Equation 4.18. This is a very important issue when working with appearance-based features and influences the error rate as shown in Figure 7.1. All the extracted feature values were normalized between 0.0 and 1.0. As one can see choosing $\sigma_0 = 0.1$ seems to be a good threshold for this value range and for all tests, i.e. when working with gray values between 0 and 255 one should set this value higher (e.g. to 25.5). Choosing $\sigma_0 \geq 1.0$ overestimates the variances and makes them useless while choosing $\sigma_0 \leq 0.01$ underestimates the variances of not observed images in training. We fixed $\sigma_0 = 0.1$ in all further experiments.

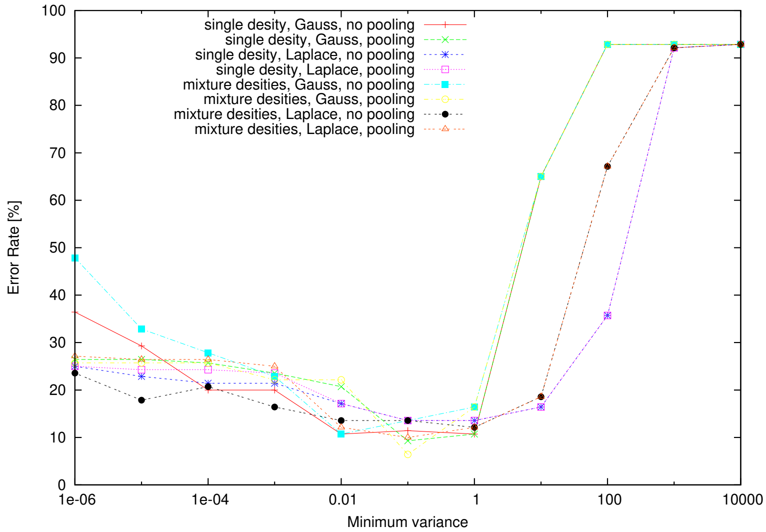


Figure 7.1. ER[%] with 32x32 original features on the 1^{st} split data from LTI-Gesture 1^{st} split against minimum variance threshold σ_0 with different distributions, densities and pooling

For these tests we used the LTI-Gesture 1^{st} split database (see Section 6.1), the minimum sequence length to determine the number of states S in each class k and estimated transitions probabilities with a 0-1-2 standard model. As features we used the whole 1^{st} time derivative images (see Section 3.1.2) downscaled to 32x32 pixels.

To have a baseline error rate for our system we first trained a simple HMM and compared it to the nearest neighbor error rate on the LTI-Gesture database. The nearest neighbor classification was done by creating a separate model for each training observation sequence, i.e. having a class k with N observations we created the classes $k_1, ..., k_n, ..., k_N$ and respective models $\lambda_{k_1}, ..., \lambda_{k_n}, ..., \lambda_{k_N}$ where a state s_t in a model λ_{k_n} represents exactly one image feature X_t of the training sequence n. Searching the model with the best score then represents a nearest neighbor classification. We used the following settings to create the models:

- original image features downscaled to 32x32
- 0-1-2 model
- single densities
- Gaussian emission densities

Table 7.1. Primary decisions and mean error rates for basic HMM settings on the LTI-Gesture 1^{st} split database using original image features

0-1-2 model [10.6% ER]	vs.	[24.0% ER] 0-1 model
Gauss [8.7% ER]	vs.	[12.6% ER] Laplace
Mixture densities [7.1% ER]	vs.	[10.3% ER] Single densities
Pooling [5.7% ER]	vs.	[8.5% ER] No pooling

Table 7.2. Error rates for basic HMM settings on the LTI-Gesture 1^{st} split database using original image features and 0-1 model

Densities	Pooling	Gaussian ER[%]	Laplacian ER[%]
Single	No	25.0	25.7
	Yes	22.8	27.8
Mixture	No	18.5	25.0
	Yes	19.2	28.5

- pooling

We achieved 5.7% ER with the nearest neighbor versus 11.4% ER with the HMM. Having better results with nearest neighbor is often the case in image recognition as the images are the most important features (and for example not necessarily the transition probabilities). Exact matching of the observations plays a big role in this task and one also can conclude that using only single densities for the HMM classification is not enough.

Then we made some experiments about the basic HMM settings concerning:

- 0-1 or 0-1-2 model?
- Which distribution should be used: Gaussian or Laplacian?
- How many densities are needed in each state?
- Should we use pooling or not for the variances?

As one can gather from the results in Table 7.2 and Table 7.3, the following settings seem to be the best in a first approach: 0-1-2 model, Gaussian distribution, mixture densities and pooling. These mean results are summarized in Table 7.1. Achieving better results with pooling was to be expected, because the single features all describe a similar object – an image pixel.

Table 7.3. Error rates for basic HMM settings on the LTI-Gesture 1^{st} split database using original image features and 0-1-2 model

Densities	Pooling	Gaussian ER[%]	Laplacian ER[%]
Single	No	11.4	13.5
	Yes	9.2	13.5
Mixture	No	8.5	13.5
	Yes	5.7	10.0

Table 7.4. Error rates for basic HMM settings on the LTI-Gesture 1^{st} split database using 1^{st} derivative difference image features and 0-1 model

Densities	Pooling	Gaussian ER[%]	Laplacian ER[%]
Single	No	29.2	30.7
	Yes	29.2	30.7
Mixture	No	21.4	29.2
	Yes	23.5	27.8

We verified these settings with another image feature, the 1^{st} time derivative difference image, on the same data set and came to the same results, as one can see in Table 7.4 and Table 7.5.

In Figure 7.2 one can see again that using the minimum sequence length of observations seen in training to estimate the number of states S in a model λ_k, the 0-1-2 model is better than the 0-1 model and allows a greater flexibility on the LTI-Gesture database. Using a 0-1 model with $|S| > 15$ it was often impossible for a testing sequence to reach an end state of a model due to the topology constraints.

We checked this on the DUISBURG-Gesture database (see Section 6.2) with COG-Features (see Section 3.2) which gave the same results as shown in Figure 7.3. One

Table 7.5. Error rates for basic HMM settings on the LTI-Gesture 1^{st} split database using 1^{st} derivative difference image features and 0-1-2 model

Densities	Pooling	Gaussian ER[%]	Laplacian ER[%]
Single	No	12.1	10.0
	Yes	8.5	12.8
Mixture	No	3.5	16.4
	Yes	5.7	13.5

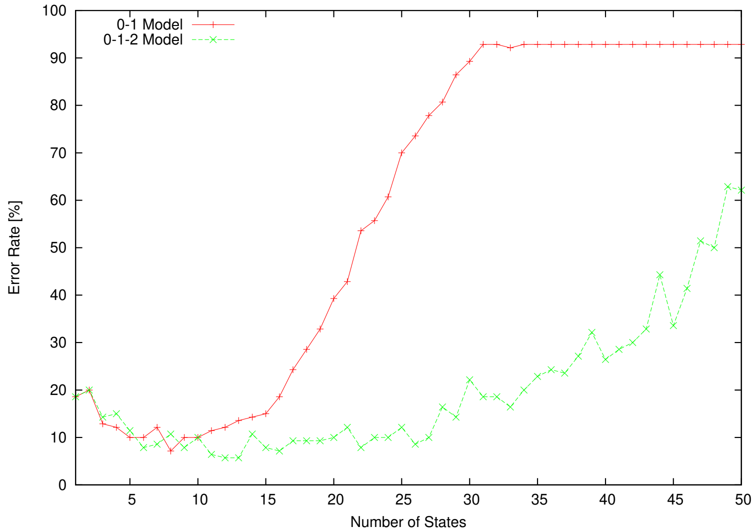


Figure 7.2. ER[%] with 32x32 original features against number of HMM states on the LTI-Gesture 1^{st} split database

can also see that the 0-1 model performs better on this database as each sequence has the same length of 50 frames.

So we can finally conclude that the following settings should be used for this gesture recognition task with appearance-based features:

- use the 0-1-2 model
- use mixture densities
- use Gaussian distributions
- use pooling

7.1.2 HMM Topology

For the experiments about the HMM topology settings concerning fixed transition and estimated probabilities we used the 1^{st} split, maximal 5 densities in each state, a Gaussian distribution, and model pooling. As features we used the whole original images (see Section 3.1.1) downscaled to 32x32.

To ensure that $\sum_{\delta=0}^{\delta_{MAX}} p_\delta = 1$, the transition probabilities were calculated as follows for a 0-1-2 standard model:

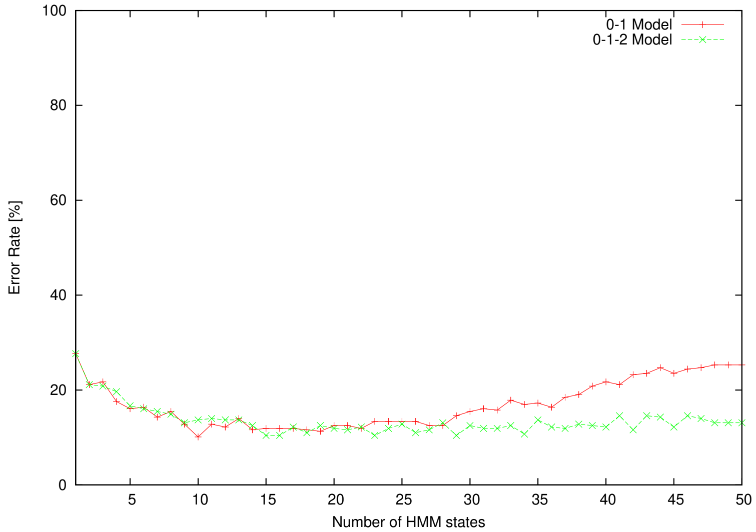


Figure 7.3. ER[%] with COG-features against number of HMM states on DUISBURG-Gesture

- Estimated:

$$p(s_t \in \{s_{t-1}, s_{t-1}+1, s_{t-1}+2\}) = \frac{\sum_{n=0}^{N_k} \sum_{s=0}^{S(N_k)} N(s, s_t)}{\sum_{n=0}^{N_k} \sum_{s=0}^{S(N_k)} \sum_{\delta'=0}^{2} N(s, s+\delta')}$$

- Fixed (with $0 < \phi < 1$):

$$\begin{aligned} p(s_t = s_{t-1}+1) &= \phi \\ p(s_t \in \{s_{t-1}, s_{t-1}+2\}) &= \frac{1-\phi}{2} \end{aligned}$$

One can conclude from the results in Figure 7.4 and Figure 7.5 that transition probabilities are not very important in the task of recognizing image sequences and that a high emission score weight is favoured. Also this coincides with the good nearest neighbor result mentioned in Section 7.1.1.

On the basis of other experiences one can say that the following options are among the best results in all cases:

- if the number of states for each gesture is chosen small (e.g. 10 states), then a 0-1 model is better, because the mean images are smoother and more often skipped in distance calculation with a 0-1-2 model.

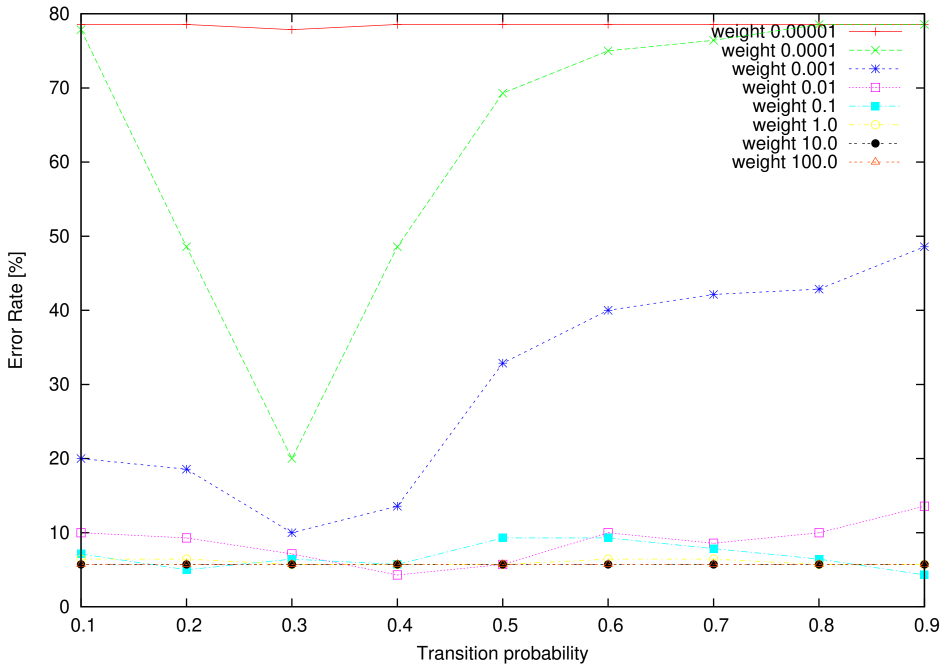


Figure 7.4. ER[%] with 32x32 original features on LTI-Gesture 1^{st} split against transition probabilities for 0-1-2 model: emission score weight with fixed transition probabilities showing the influence of transition probabilities with low emission score weight

- if the number of states for each gesture depends on the occurred minimum sequence length, then a 0-1-2 model is better

- there can be a big difference in the ER when choosing fixed transition probabilities instead of estimated ones depending on the emission score weight. Choosing fixed probabilities with a high emission score weight yields more constant results.

Experiments about pruning of hypothesis to reduce the calculations of emission probabilities were made as calculating the emission probabilities and scores in the HMM can be very time consuming. This can be reduced with beam search by introducing a pruning factor f in the score calculation function of the emission probabilities.

First we have to define the best emission score until time t denoted as follows:

$$Q_{\max}(t) := \max_{s,k}(Q(s_t, k))$$

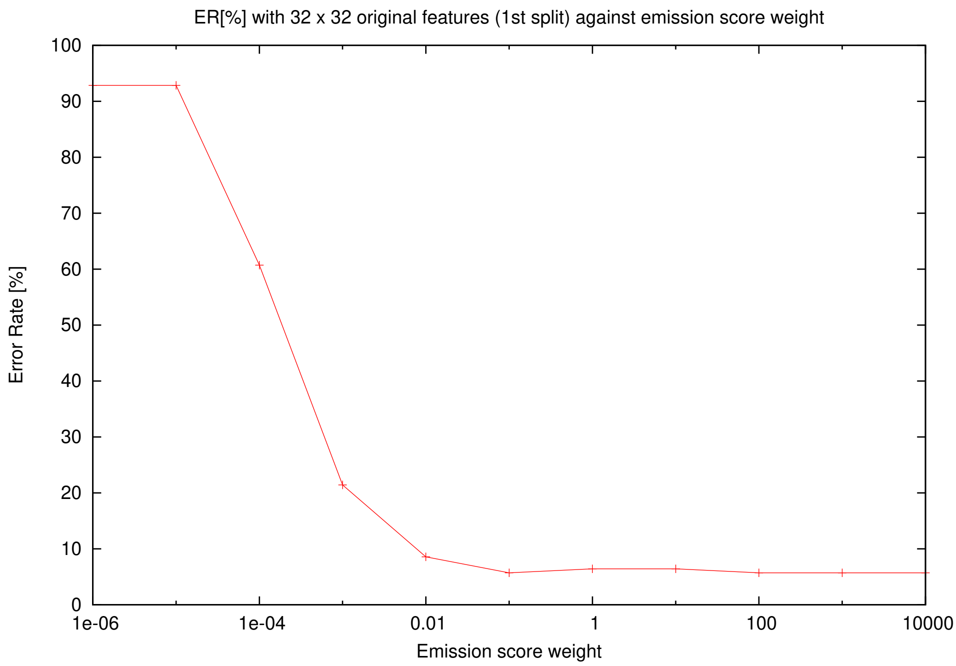


Figure 7.5. ER[%] with 32x32 original features on LTI-Gesture 1^{st} split and estimated transition probabilities for 0-1-2 model against emission score weight showing that a high emission score weight yields the best results

The emission probability $p(X_t|s_t, k)$ at time t and state s will then only be calculated, if $Q(s_t, k) \leq f \cdot Q_{\max}(t)$ holds, i.e if $-\log(Q(s_t, k)) \leq -\log(Q_{\max}(t)) + \log(f)$ holds when calculating with scores.

Figure 7.6 shows that pruning can speedup the recognition by a factor 2 without any changes in the error rate and by a factor 4 while the error rates degrades only from 5.7% to 10.0%.

7.1.3 HMM Features Using Different Databases

In this section we present the results achieved with some of the features presented in Chapter 3 on the adequate databases. Results from other groups are also presented here for comparison where available.

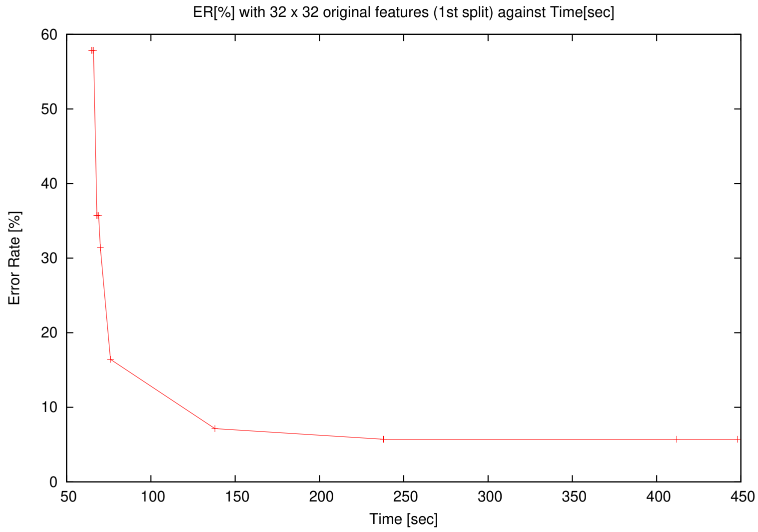


Figure 7.6. ER[%] with 32x32 original features on the LTI-Gesture 1^{st} split database against Time[sec]: the runtime can be improved by a factor 4 while degrading the error rate only to 10% and by a factor without any changes in the error rate

HMM Features on the LTI-Gesture database

As we made all the basic experiments with either the difference images or the original images we wanted to know if gray-value thresholding of the image values is helpful, i.e. reducing noise or background. For these tests we used the LTI-Gesture 1^{st} split database, maximal 5 densities, a Gaussian distribution and model pooling. As features we used the whole original or 1^{st} time derivative difference images (see Section 3.1.2) downscaled to 32x32.

Table 7.6 shows that we obtained the best results without thresholding, i.e. using as much information as possible achieves the best results as the outcome of thresholding the image features results in a loss of features, too.

In the following, we present tests concerning different appearance-based features, e.g. the original images, time derivative images, spatial derivative images (Sobel) and the COG-features. Table 7.7 shows that the difference images were better than the original ones. This was to be expected, because the difference images contain information about the shape and the movement. The 1^{st} time derivative difference images were better than the 2^{nd} as well for Sobel feature images as for original images.

The COG-features on the LTI-Gesture database achieved an error rate of only 14.2%.

Table 7.6. Error rates for feature thresholding on LTI-Gesture

Threshold	Original ER[%]	1^{st} time derivative ER[%]
0	5.7	5.0
50	5.7	10.7
100	7.1	6.4
150	7.1	9.2
200	7.8	17.1
250	9.2	48.5

This was to be expected because it is quite difficult to differentiate for example the gesture "Five" and "Two" only by the centroids and global motion.

The good error rate of 7.1% from [Rigoll & Kosmala+ 98] with these features on the DUISBURG-Gesture database (see Section 6.2) were only possible because the most discriminative characteristic of these gestures were the location and the movement direction and not the shape. We could achieve an error rate of 10.3% with the same features on the DUISBURG-Gesture database which is probably due to the missing cyclic HMM as mentioned in their work or untested feature weights.

Each gesture is always performed in a body-centered space and there are only two gestures ("Nod-No" and "Nod-Yes") which are performed in the same location with similar shapes. All other gestures can be distinguished by using their motion direction alone.

Figure 7.7 shows these problems of the COG-features on the LTI-Gesture database. These features are good to describe different motion patterns like "Thumb-Left" and "Thumb-Right" and inappropriate to distinguish between gestures with the same motion but just differing shapes like "Five" and "Two".

Using the COG-features, such gestures can only be differentiated by the global motion (see Equation 3.21) which is strongly scale dependent (i.e. person dependent) and not sufficient.

We verified these features on the LTI-Gesture 2^{nd} split database and came to similar results. Table 7.8 shows also that the vertical Sobel is better than the horizontal and that the 1^{st} time derivative is better than the 2^{nd} . One can also observe that the results on the 2^{nd} split are consequently worse than the results obtained on the 1^{st} split.

So we can finally conclude that original and time derivative images are good features and that the 1^{st} time derivative is better than 2^{nd} time derivative.

The COG-features can not achieve good results for example in gestures differing in only one finger but good results in gestures with movement like "Thumb-Left" or "Thumb-Right".

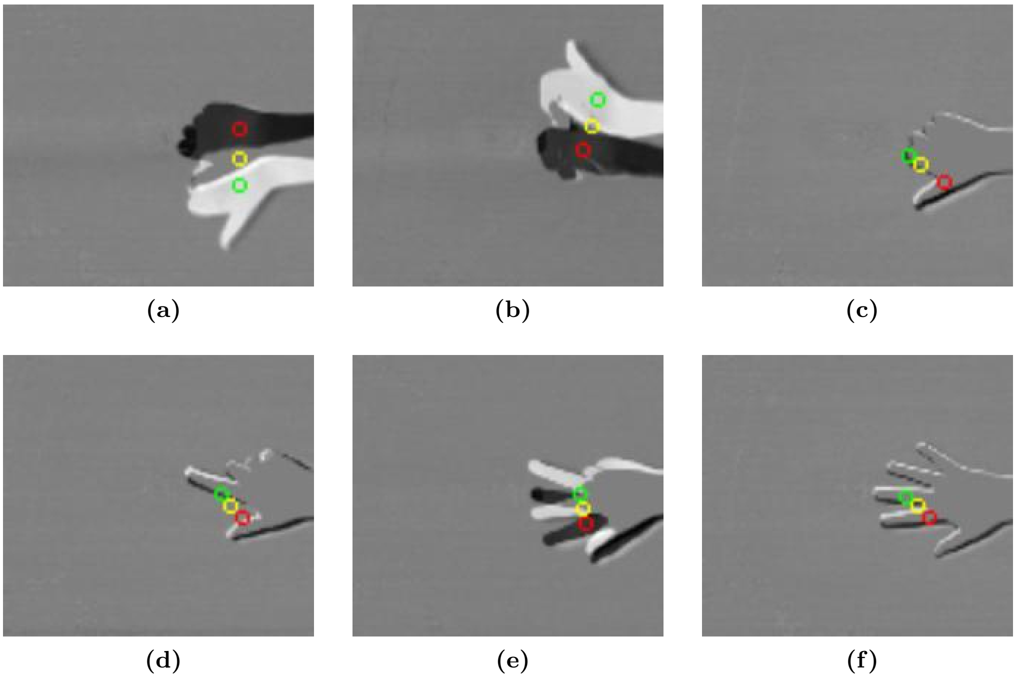


Figure 7.7. Some examples of the centroid features on the LTI-Gesture database, where a green circle means the positive COG, a yellow circle the COG and a red circle the negative COG, showing that the COG-features are insufficient to distinguish between more complex gestures (c), (d), (e) and (f) but very good to distinguish between motion intensive gestures (a) and (b)

Table 7.7. Error rates for different HMM features on the LTI-Gesture 1^{st} split database in ER[%]

Spatial derivative (Sobel)	Original	1^{st} time derivative	2^{nd} time derivative	COG-Features
no	5.7	5.0	15.7	14.2
horizontal	10.0	9.2	20.0	–
vertical	5.0	4.2	16.4	–
magnitude	7.1	5.0	7.1	–
squared magnitude	8.5	16.4	34.2	–

Using spatial derivative image features can improve the error rate for original image features once again from 5.7% to 5.0% and for time derivative features from 5.0% to 4.2% on the LTI-Gesture 1^{st} split database and from 12.1% to 6.4% for original image

Table 7.8. Error rates for different HMM features on the LTI-Gesture 2^{nd} split database in ER[%]

Spatial derivative (Sobel)	Original	1^{st} time derivative	2^{nd} time derivative	COG-Features
no	12.1	18.5	20.7	27.8
horizontal	10.7	21.4	21.4	–
vertical	6.4	16.4	28.5	–
magnitude	12.1	19.2	19.2	–
squared magnitude	15.0	25.0	40.0	–

features on the LTI-Gesture 2^{nd} split database.

Especially the vertical gradient images are good features as the signing hand in the LTI-Gesture database is presented in a horizontal way, so that the gradients of the fingertips are more meaningful than with a vertical filter.

The effect of feature augmentation was also investigated. Figure 7.8 shows that if one e.g. augments the best original image feature (vertical Sobel of original image, VSO) by the best time derivative feature (vertical Sobel of first time derivative, VSD), i.e. if one concatenates $\alpha \cdot \text{VSO} + (1-\alpha) \cdot \text{VSD}$ with $\alpha = 0.5$, one can improve the error rate from 5.0% to 2.1%.

The feature augmentation of the vertical Sobel of the original image by its horizontal Sobel also improves the error to 3.5% and is better than using the normalized Sobel, but one can also see that feature augmentation does not always improve the error rate.

With the motion-history-image (MHI) feature (see Section 3.1.4) we could achieve with a HMM template classification only an error rate of 26.4% which shows that this feature is too general for differentiating between this complex gestures. For the template classification we trained a HMM with one state for each gesture with maximal 5 densities, a Gaussian distribution and model pooling.

Choosing an appropriate value for the history size τ is important and should be estimated during training. Using the MHI as an HMM feature for the *normal classification task*, we decided to check only four different setups instead of estimating the parameter τ.

Table 7.9 shows the results obtained with this feature on the LTI-Gesture database. We achieved the best results with a maximum history size and the number of states in each HMM set to the observed minimum sequence length during training. Using this setup we could achieve a good error rate of 5.7%.

Using as much information as possible improves once again the error rate and using a HMM instead of a template matching method yields better results for this task, as the gesture movements are executed only once and contain no partial gestures. Also the more complex gestures “One” to “Five” are presented without any big rotations of

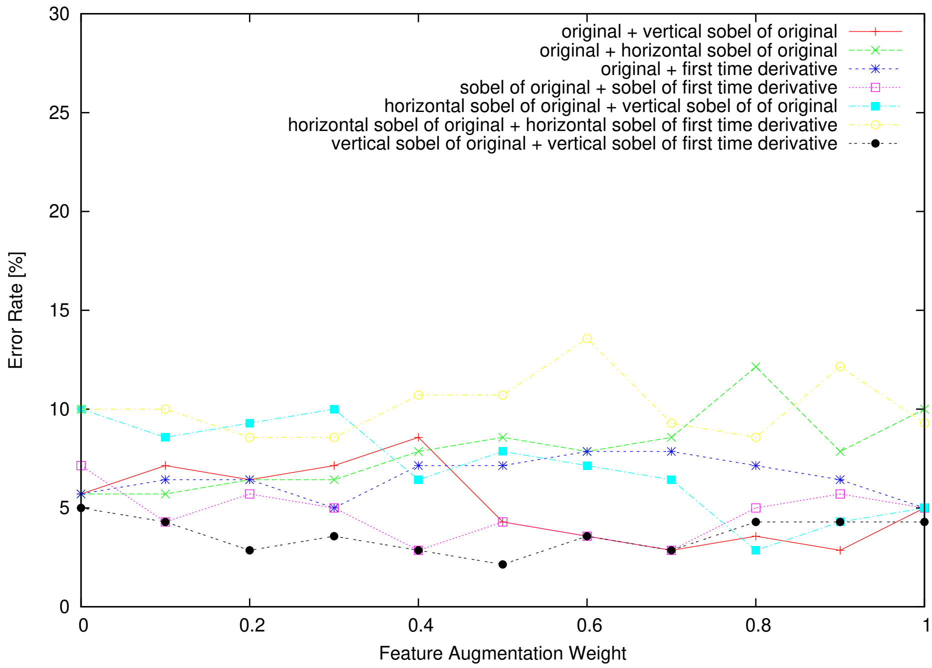


Figure 7.8. Feature augmentation on the LTI-Gesture database showing error rate versus feature weight

Table 7.9. Error rates for motion-history feature on the LTI-Gesture 1^{st} split database

History Size τ	Minimum sequence length	Number of States $S = 10$
N (Maximum)	5.7	9.2
10	12.8	14.2

the hand so that the history-images of these gestures are similar to the original images. Otherwise it is to be expected that the history-image would achieve worse results.

HMM Features on the DUISBURG-Gesture database

As mentioned before we could achieve a leaving-one-person-out error rate of 10.3% with the COG-features on the DUISBURG-Gesture database instead of 7.1% as presented in [Rigoll & Kosmala$^+$ 98]. This is probably due to the missing cyclic HMM as mentioned in their work or untested feature weights.

The original image features achieved only an error rate of 61.9% which was to be expected as the database contains 14 different persons recorded in a body-centered space.

Using the whole absolute motion features downscaled to 32x32 as described in Section 3.1.2 with the same HMM settings as for the LTI-Gesture database (maximal 5 densities, a Gaussian distribution and model pooling) we could achieve an error rate of 14.2% which is a good result and shows that the findings from the completely different LTI-Gesture database are not overfitted and also perform well on this database.

The 1^{st} time derivative image features could only achieve an error rate of 22.2% which shows that the global motion on this database is more important than the speed and the direction of the gestures when working with appearance-based features.

With the motion-history-image (MHI) feature (see Section 3.1.4) we could achieve with a HMM template classification a leaving-one-person-out error rate of 20.8%. For the template classification we used the same procedure as explained for the LTI-Gesture database. The gestures "Nod-No" and "Nod-Yes" as well the gestures "A" to "D" are very difficult to distinguish between with this feature in combination with template matching and explain the bad error rate, as the important movements and their directions are often self-overlaid.

Also we split the DUISBURG-Gesture database into two sets as the leaving-one-person-out classification is very time consuming. We considered 7 persons of the 14 for training and the other 7 persons for testing. This setup is harder than the leaving-one-person-out data setup but not as time consuming. We could achieve with the MHI feature an error rate of 22.6% for template matching. Augmenting this feature by the zero-thresholded motion-energy-image as proposed in [Bobick & Davis 01] worse the error rate to 31.5% as the gesture "To-Left" becomes more similar to "Hand-Waving-Left", "Nod-No" to "Nod-Yes", and "C" to "D". Table 7.10 and Table 7.11 show the resulting confusions when combining the MHI and MEI feature on the DUISBURG-Gesture split database.

Using the MHI as an HMM feature for the normal classification task, we decided again to check only some different setups instead of estimating the parameter τ. Table 7.12 shows the results obtained with this feature on the DUISBURG-Gesture database. We achieved the best results with a history size $\tau = 10$ and the number of states $S = 10$. Using this setup we could achieve an error rate of 18.7% which also shows that the HMM classifier is better than template matching.

Choosing $\tau \neq N$ is more important for this task as the gestures in the DUISBURG-Gesture database contain partial gestures, e.g. when signing one of the gestures "A", "B", "C" or "D" (the letters are simply drawn in front of the camera) which is not the same than signing a letter in sign language. This comes into play when using the minimum sequence length to determine the number of states where the best error rate is achieved for $\tau = 5$.

Table 7.10. Confusion matrix with MHI feature on the DUISBURG-Gesture split database, error rate 22.6% for template matching (C: correct, I: incorrect)

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	C/I	ER
Hand-Waving-Both	0	7																								7/0	0.0%
Hand-Waving-Right	1		7																							7/0	0.0%
Hand-Waving-Left	2			7																						7/0	0.0%
To-Right	3				6																	1				6/1	14.3%
To-Left	4					6									1											6/1	14.3%
To-Top	5		1				4				1														1	4/3	42.9%
To-Bottom	6							6			1															6/1	14.3%
Round-Clockwise	7								5	2																5/2	28.6%
Round-Counterclockwise	8								6	1																1/6	85.7%
Stop	9						2	1			4															4/3	42.9%
Come	10	1										6														6/1	14.3%
Nod-Yes	11												6	1												6/1	14.3%
Nod-No	12												2	5												5/2	28.6%
Clapping	13														7											7/0	0.0%
Kotow	14															7										7/0	0.0%
Spin	15																7									7/0	0.0%
Go-Left	16																	6			1					6/1	14.3%
Go-Right	17																		6	1						6/1	14.3%
Turn-Right	18																			7						7/0	0.0%
Turn-Left	19																				7					7/0	0.0%
A	20																					1	3	2	1	1/6	85.7%
B	21										1												2	1	3	2/5	71.4%
C	22																							6	1	6/1	14.3%
D	23										1												1	1	4	4/3	42.9%
	C	7	7	7	6	6	4	6	5	1	4	6	6	5	7	7	7	6	6	7	7	1	2	6	4		
	I	1	1	0	0	0	2	1	6	2	4	0	2	1	1	0	0	0	0	1	1	1	4	4	6		
	I%	3	3	0	0	0	5	3	16	5	11	0	5	3	3	0	0	0	0	3	3	3	11	11	16		

Table 7.11. Confusion matrix with MHI and MEI feature on the DUISBURG-Gesture split database, error rate 31.5% for template matching (C: correct, I: incorrect)

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	C/I	ER
Hand-Waving-Both	0	7																								7/0	0.0%
Hand-Waving-Right	1		7																							7/0	0.0%
Hand-Waving-Left	2			7																						7/0	0.0%
To-Right	3				6																	1				6/1	14.3%
To-Left	4			2		3						1			1											3/4	57.1%
To-Top	5		1				5				1															5/2	28.6%
To-Bottom	6							3			1											2	1			3/4	57.1%
Round-Clockwise	7								4	3																4/3	42.9%
Round-Counterclockwise	8								4	3																3/4	57.1%
Stop	9						1	1			4											1				4/3	42.9%
Come	10	1										6														6/1	14.3%
Nod-Yes	11												5	2												5/2	28.6%
Nod-No	12												4	2								1				2/5	71.4%
Clapping	13														5							1		1		5/2	28.6%
Kotow	14															6							1			6/1	14.3%
Spin	15																5					1	1			5/2	28.6%
Go-Left	16																	7								7/0	0.0%
Go-Right	17																		7							7/0	0.0%
Turn-Right	18																			7						7/0	0.0%
Turn-Left	19																				7					7/0	0.0%
A	20																					1	1		5	1/6	85.7%
B	21										1												2		4	2/5	71.4%
C	22										1												1	2	3	2/5	71.4%
D	23																					1	2		4	4/3	42.9%
	C	7	7	7	6	3	5	3	4	3	4	6	5	2	5	6	5	7	7	7	7	1	2	2	4		
	I	1	1	2	0	0	1	1	4	3	4	1	4	2	1	0	0	0	0	0	0	8	7	1	12		
	I%	2	2	4	0	0	2	2	8	6	8	2	8	4	2	0	0	0	0	0	0	15	13	2	23		

Table 7.12. Error rates for motion-history feature on the DUISBURG-Gesture database in ER[%]

History Size τ	Minimum sequence length	Number of States $S = 10$
N (Maximum)	33.3	25.2
5	22.9	20.2
10	25.9	18.7
15	24.7	22.3
20	26.4	23.2
25	26.4	25.0

HMM Features on the i6-Gesture database

The i6-Gesture database was unfortunately only finished at the end of the provided time for this diploma thesis that is why we can not provide as many results as for the other databases. Also it is harder to extract good features on this database and maybe one needs more know-how on segmenting or tracking the important regions which would go beyond the scope of this work.

As the background is not constant for all sequences, the signing persons wear not all the same clothing and the lighting conditions are changing, we decided to make a first test with full size skin thresholded original image features downscaled to 32x32. With this feature we achieved an error rate of 87.1%.

Using the 1^{st} time derivative of original images thresholded by their skin probability we could achieve an error rate of 72.1%.

It is obvious that this database contains gestures of much higher complexity and that one need additional methods for feature extraction or other distance measures.

7.2 Distance Measures Using Different Databases

After building up our base system with the results from Section 7.1 we investigated the effects of transformation independent distance measures. In this section, we present the achieved results for the distance measures presented in Section 4.2.2 and Section 4.2.3 on different databases.

7.2.1 Tangent Distance

In this section we present the achieved results for the tangent distance measure which is invariant against affine transformations on different databases.

Table 7.13. Error rates for tangent distance on LTI-Gesture 1^{st} split

Features	Euclidian distance ER[%]	Tangent distance One-sided ER[%]	Two-sided ER[%]
Original	5.7	2.8	2.8
vertical Sobel	5.0	2.8	4.2
magnitude Sobel	7.1	2.1	2.1
First time derivative	5.0	5.7	3.5
vertical Sobel	4.2	5.0	3.5
magnitude Sobel	5.0	3.5	5.7
	5.3	3.65	3.63

Tangent Distance on the LTI-Gesture database

The results Table 7.13 and the validation results in Table 7.14 show that the tangent distance can improve the error rate on the LTI-Gesture database.

The whole original image features downscaled to 32x32 can improve the error rate on 1^{st} split from 5.7% to 2.8% and with 1^{st} time derivative image features from 5.0% to 3.5%. On 2^{nd} split the error rate achieved with original image features can be improved with two-sided tangent distance from 12.1% to 2.8% and with 1^{st} time derivative image features from 18.5% to 17.1%.

With the spatial derivative image features obtained with Sobel filtering, the tangent distance can also improve the error rate on 1^{st} split from 7.1% to 2.1% for original images and from 5.0% to 3.5% for the 1^{st} time derivative image features. On the 2^{nd} split the error rate can be improved with two-sided tangent distance from 12.1% to 2.8% for the original images.

On both splits the tangent distance cannot improve the results achieved with the time derivative image features as much as with the original image features and their corresponding spatial derivative images.

One can see that on average the two-sided tangent distance achieves slightly better results than the one-sided tangent distance for this task but this cannot be generalized.

Table 7.15 shows the confusion matrix of the last errors on 2^{nd} split obtained with the two-sided tangent distance, i.e. we have a mean error rate of 2.8% in comparison to 4.5% presented in [Pelkmann 99]. We can observe that the remaining errors are all due to confusions between the classes “One-Thumb”, “One-Finger”, “Two” and “Three”, which means that the correct alignment of the fingers are possibly not found.

If one adds then the possibility creating virtual data by rotating the images by $\pm\alpha^{\circ}$ as explained in Section 4.2 before calculating the tangent distance, the error rate can be improved once again on the two LTI-Gesture data splits. We exemplified this by the

Table 7.14. Error rates for tangent distance on LTI-Gesture 2^{nd} split

Features	Euclidian distance ER[%]	Tangent distance One-sided ER[%]	Two-sided ER[%]
Original	12.1	5.0	2.8
vertical Sobel	6.4	5.7	5.0
magnitude Sobel	12.1	5.0	4.2
First time derivative	18.5	17.8	17.1
vertical Sobel	16.4	16.4	17.8
magnitude Sobel	19.2	19.2	20.7
	14.1	11.5	11.2

Table 7.15. Confusion matrix with two-sided tangent distance on the LTI-Gesture 2^{nd} split database, error rate 2.8% (C: correct, I: incorrect)

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	C/I	ER
DaumenLinks	0	10														10/0	0.0%
DaumenRechts	1		10													10/0	0.0%
Drei	2			8			1								1	8/2	20.0%
EinsDaumen	3				10											10/0	0.0%
EinsFinger	4					10										10/0	0.0%
Fuenf	5						10									10/0	0.0%
Hoch	6							10								10/0	0.0%
Pause	7								10							10/0	0.0%
Runter	8									10						10/0	0.0%
Stop	9										10					10/0	0.0%
Vier	10			1								9				9/1	10.0%
Vor	11												10			10/0	0.0%
Zurueck	12													10		10/0	0.0%
Zwei	13					1									9	9/1	10.0%
	C	10	10	8	10	10	10	10	10	10	10	9	10	10	9		
	I	0	0	1	0	1	1	0	0	0	0	0	0	0	1		
	I%	0	0	25	0	25	25	0	0	0	0	0	0	0	25		

original image features and their corresponding spatial derivatives on the LTI-Gesture database.

Table 7.17 shows these improvements achieved on 2^{nd} split: the error rate can be improved from 12.1% to 1.4% with tangent distance. Table 7.16 shows that the error rate can also be improved on the 1^{st} split from 5.7% to 1.4% with one-sided tangent distance. This also means that we can achieve an average error rate of 1.4% with tangent distance in comparison to 4.5% presented in [Pelkmann 99].

The runtime is extended by a factor 3 when creating this virtual data in combination with tangent distance.

Table 7.16. Error rates for tangent distance with rotation on LTI-Gesture 1^{st} split

Features	Euclidian distance ER[%]	Tangent distance One-sided ER[%]	Two-sided ER[%]
Original	5.7	1.4	2.1
vertical Sobel	5.0	2.8	4.2
magnitude Sobel	7.1	1.4	2.1

Table 7.17. Error rates for tangent distance with rotation on LTI-Gesture 2^{nd} split

Features	Euclidian distance ER[%]	Tangent distance One-sided ER[%]	Two-sided ER[%]
Original	12.1	1.4	1.4
vertical Sobel	6.4	2.8	2.1
magnitude Sobel	12.1	5.7	5.0

Tangent Distance on the DUISBURG-Gesture database

Using the whole absolute motion features downscaled to 32x32 with two-sided tangent distance we could not improve the error rate of 14.2% and achieved only an error rate of 14.8% but with the one-sided tangent distance we could improve the error rate to 13.2%.

By using the MHI feature with tangent distance to calculate the best matching HMM template we could improve the Euclidian distance error rate from 20.8% to 19.0% with one-sided tangent distance and to 19.6% with two-sided tangent distance. Also on the self-defined DUISBURG-Gesture split database we could improve the error rate from 22.6% to 16.7% which is a very good result for this template matching classifier.

Using the MHI feature with one-sided tangent distance for the normal HMM classification (with $S = 10$ and $\tau = 10$ as in Section 7.1.3) we could improve the best Euclidian distance error rate from 18.7% to 16.9% which also is a very good result. The two-sided tangent distance was also worse than the one-sided for this task and achieved only 21.1%.

This proves that using distance measures being invariant against affine transformations performs also well on history-images and template matching classifiers.

Tangent Distance on the i6-Gesture database

Due to the poor results of 87.1% and 72.1% with the full size features and runtime problems we did not check if the tangent distance could improve the error rate. We assumed that tracking must be used before to extract position independent features and thus improve the error rate. These results are presented in Section 7.3.2.

7.2.2 IDM Distance

In this section we present the achieved results for the IDM distance measure on different databases.

IDM Distance on the LTI-Gesture database

The results in Table 7.18 and Table 7.19 show that the IDM distance can improve the error rate on the LTI-Gesture database with whole original image features downscaled to 32x32 from 5.7% to 1.4%.

The 1^{st} time derivative image features cannot be improved as much as with tangent distance. Also using Sobel derivatives in IDM for distortion does not improve the error rate as much as for the original images.

This can be explained by the fact that when using Sobel images in IDM for distortion, the occurring two contours of the hand inside a time derivative image can be confused. This is also proven by the results without using Sobel in IDM for distortion which are always better for this feature than the results with Sobel.

One idea when working with time derivative images and IDM might be to distort the negative and positive part of the images separately as a form of *local motion speed adjustment*.

Also when working with time derivative images a greater patch size is appropriated which depends on the chosen difference function the time offset between to images used in this function, i.e. using $(t+1)-t$ or $(t+1)-(t-1)$ when calculating the 1^{st} time derivative (see Section 3.1.2).

All the results in Table 7.18 and Table 7.19 were obtained with warp range set to $w = 1$. Changing the local patch size and using Sobel can improve the IDM results with original image features. The results without Sobel on time derivative image features are consequently better.

We validated the achieved results also on the 2^{nd} split of the LTI-Gesture database. For these experiments we used only the original image features and their spatial derivatives. Also not all patch sizes for each feature were verified. The results presented in Table 7.20 and Table 7.21 confirmed the achieved results on the 1^{st} split of the LTI-Gesture database. Using IDM-Sobel can improve the error rate but not in all cases. Time derivative image features should be used without IDM-Sobel. The best achieved error rate on the 2^{nd} split of the LTI-Gesture database is 2.1% using the vertical Sobel-filtered original image features with a patch size of 13x13. Generally, a patch size between 7x7 and 15x15 led to the best results for this task.

Table 7.18. Error rates for IDM distance without IDM-Sobel on the LTI-Gesture 1^{st} split database in ER[%]

Features	Euclid	IDM distance with Patch Size 1x1	3x3	5x5	7x7	9x9	11x11	13x13	15x15	17x17	19x19
Original	5.7	18.5	3.5	4.2	3.5	2.8	2.8	2.8	2.1	2.1	1.4
vertical Sobel	5.0		3.5	2.8	2.8	2.8	2.8	2.8	3.5	3.5	3.5
magnitude Sobel	7.1	12.1	11.4	2.1	2.1	1.4	1.4	2.1	1.4	1.4	2.8
1^{st} time der.	5.0	7.1	14.2	15.7	4.2	5.0	5.0	2.1	2.1	1.4	
vertical Sobel	4.2	20.0	8.5	2.1	1.4	2.1	1.4	3.5	2.1	4.2	7.8
magnitude Sobel	5.0	20.0	15.7	8.5	4.2	1.4	1.4	2.1	1.4	1.4	2.8

Table 7.19. Error rates for IDM distance with IDM-Sobel on the LTI-Gesture 1^{st} split database in ER[%]

Features	Euclid	IDM distance with Patch Size 1x1	3x3	5x5	7x7	9x9	11x11	13x13	15x15	17x17	19x19
Original	5.7	9.2	6.4	1.4	1.4	3.5	3.5	3.5	2.8	2.8	
vertical Sobel	5.0		1.4	2.8	1.4	2.8	2.1	2.8	2.8	2.8	2.8
magnitude Sobel	7.1	5.7	5.7	2.1	1.4	1.4	1.4	2.8	2.1	3.5	2.1
1^{st} time der.	5.0	8.5	19.2	15.0	10.0	7.8	5.0		2.1		1.4
vertical Sobel	4.2	2.8	12.1	1.4	2.1	3.5	2.8	1.4	2.1	3.5	4.2
magnitude Sobel	5.0	11.4	7.1	10.7	2.1	3.5	1.4	3.5	2.1	3.5	4.2

IDM Distance on the DUISBURG-Gesture database

The IDM-distance calculation can be very time consuming and depends mainly on the chosen warprange and patch size. Additionally the leaving-one-person-out classification also is very time consuming that is why we investigated only one IDM setup on the whole absolute motion features downscaled to 32x32. With a patch size of 5x5 and IDM-Sobel we could not improve the error rate of 14.2% and achieved only 15.2%. We expect that choosing different patch sizes will improve this error rate.

Table 7.20. Error rates for IDM distance without IDM-Sobel on the LTI-Gesture 2^{nd} split database in ER[%]

Features	Euclid	IDM distance with Patch Size 1x1	3x3	5x5	7x7	9x9	11x11	13x13	15x15	17x17	19x19
Original	12.1		15.0			2.8	5.0	2.8	3.5	2.8	3.5
vertical Sobel	6.4					5.7	2.8	2.1	5.7	3.5	4.2
magnitude Sobel	12.1		4.2	4.2	2.8	2.1	3.5	3.5	5.7		5.0

Table 7.21. Error rates for IDM distance with IDM-Sobel on the LTI-Gesture 2^{nd} split database in ER[%]

Features	Euclid	IDM distance with Patch Size									
		1x1	3x3	5x5	7x7	9x9	11x11	13x13	15x15	17x17	19x19
Original	12.1		15.0	8.5	4.2	8.5	2.8	2.8	2.8	6.4	3.5
vertical Sobel	6.4		4.2	3.5	2.8	2.8	5.0	3.5	5.0	6.4	5.0
magnitude Sobel	12.1		10.0	5.0	4.2	5.0	7.1	6.4	5.0	3.5	2.8

Using the MHI feature with IDM distance to calculate the best matching HMM template we could improve the Euclidian distance error rate from 20.8% to 17.5% with a patch size of 9x9 and with a patch size of 15x15 to 18.1%.

On the self-defined DUISBURG-Gesture split database we could improve the error rate from 22.6% to 19.6% with a patch size of 9x9 but even to 14.8% with a patch size of 15x15.

This proves that using distance measures which consider image transformations performs also well on history images and template matching classifiers.

IDM Distance on the i6-Gesture database

Due to the longer runtime of the IDM, we did not yet check this distance function which is expected to be better than the tangent distance for this task, as the fingers of the gesturing hand can be distorted independently.

7.2.3 Feature Size Results

Scaling is a very important issue when working with appearance-based features as information is lost when downscaling an image. In all our presented results we used bilinear interpolation for downscaling the images to a quadratic size of 32x32. Smoothing the images with e.g. a Gaussian filter before downscaling improves the results in most cases.

We analyzed the effects of downscaling to different feature sizes exemplary on the original feature and its spatial derivatives. Table 7.22 shows the effects on the results obtained using Euclidian distance and Table 7.23 the effects on the results obtained using two-sided tangent distance. Using a smaller feature size, i.e. smoothing the features and using less information lead to better results for all features and both distance measures.

The error rate can be improved once again from 2.1% to 1.4% with Sobel of original image as feature and using two-sided tangent distance. These are only small improvements but show the necessity of investigating scaling methods when working with appearance-based features.

Table 7.22. Error rates for different feature sizes on the LTI-Gesture database using Euclidian distance in ER[%]

Feature size	Original	Magnitude Sobel	Vertical Sobel
4x4	25.0	30.0	53.5
8x8	10.7	11.4	15.0
16x16	3.5	5.7	3.5
32x32	5.7	7.1	5.0
64x64	11.4	10.0	7.1

Table 7.23. Error rates for different feature sizes on the LTI-Gesture database using two-sided tangent distance in ER[%]

Feature size	Original	Magnitude Sobel	Vertical Sobel
4x4	67.8	52.8	77.1
8x8	10.7	20.0	14.2
16x16	2.1	1.4	2.8
32x32	2.8	2.1	4.2
64x64	5.7	3.5	5.7

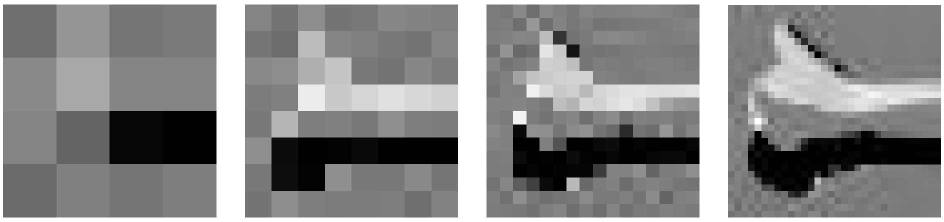
Even with a feature size of 8x8 and simply downscaling the original images we could achieve an error rate of 10.7%.

7.3 Tracking

In this section we present the best settings from Section 7.1 in combination with tracking algorithms. In contradiction to the use of whole image features only an area of interest (AOI) is extracted from the feature sequences and downscaled to a unique size.

7.3.1 Bounding-Box Tracking

The Bounding-Box tracking (Section 5.1) was only useful on the LTI-Gesture database, as the infrared-images can easily be thresholded to segment the hand from the background. The tracking was then used to crop and scale the hand part of the image as normalizing function. With this tracking method it was possible to obtain a more detailed representation of the hand, without having too large feature vectors. Even with a feature size of 4x4 pixels we can already achieve an error rate of only 15.0% (see Figure 7.9 for some scaling examples).



(a) (b) (c) (d)

Figure 7.9. Scaling examples of extracted features: from (a) scaled down to 4x4, (b) scaled down to 8x8, (c) scaled down to 16x16 and (d) scaled down to 32x32

Table 7.24. Error rates for different HMM features (downscaled to 32x32) and distance measures on the LTI-Gesture 1^{st} split database with Bounding-Box tracking

Features	Distance	ER[%]
Original	Euclidian	5.0
	Tangent	1.4
	IDM	1.4
First time derivative	Euclidian	0.7
	Tangent	0.7
	IDM	0.7

For the tests in Table 7.24 we used the 1^{st} split data and the best settings from Section 7.1. As features we used the bounding box of either the original images or the 1^{st} time derivative images (see Section 3.1.2) downscaled to 32x32. Nearly all basic results from Section 7.1.1 were confirmed and could be improved with tracking.

Table 7.25 also shows the validation results on 2^{nd} split data, depending on the different data setups, scalings and difference functions, which proves that the current settings are good and no overfitting has occurred (Note that the error rates on the LTI-Gesture 2^{nd} split database are larger on the average for all classifier setups).

7.3.2 Camshift Tracking

In this section we present some results obtained with the camshift tracker on the LTI-Gesture and i6-Gesture databases. We did not analyze the different camshift tracker options, instead we used the tracker only to extract more detailed information of the image sequences. Also we did not analyze tracking on the DUISBURG-Gesture database as all gestures are performed in a body-centered space.

Table 7.25. Error rates for the LTI-Gesture database with Bounding-Box tracking and different difference functions using Euclidian distance in ER[%]

Feature Size	1^{st} time derivative		2^{nd} time derivative	
	1^{st} split	2^{nd} split	1^{st} split	2^{nd} split
4x4	15.0	17.8	15.0	17.1
8x8	2.1	10.7	2.8	10.7
16x16	2.1	9.2	1.4	9.2
32x32	0.7	10.7	0.7	10.7
64x64	0.7	10.7	0.7	10.7

Tracking on the LTI-Gesture database

To use Camshift tracker on the LTI-Gesture database we had to initialize the tracker already on the right starting position above the signing hand on the right side of the image. We achieved an error rate of 0.0% on the 1^{st} split and 12.1% on the 2^{nd} split with 1^{st} time derivative image features and the Euclidian distance.

Tracking on the i6-Gesture database

Using a camshift tracker on the i6-Gesture database to extract the original images thresholded by their skin probability we could improve the error from 87.1% to 44.0%. With the 1^{st} time derivative image feature of original images thresholded by their skin probability in combination with tracking, the error rate could be improved from 72.1% to 46.2%. This shows the need of tracking system or a different feature extraction method to be more position and scale independent.

These two error rates were already achieved with pruning. As the runtime was still very high, we investigated the use of different pruning thresholds and the impacts on the error rate. Figure 7.10 shows the error rate versus time graph.

Using a two-sided tangent distance we could improve once again the error rate to good and currently best result of 35.7% which shows the advantage of using distance measures being invariant against affine transformations and the possibility of recognizing sign language by simple appearance-based features.

With the same features but scaled to 16x16 we achieved an error rate of 46.0% for one-sided tangent distance and 42.5% for two-sided which is even better than using 32x32 original image features without tangent-distance.

We could also improve the error rate when using the 1^{st} time derivative image feature of original images thresholded by their skin probability with two-sided tangent distance from 46.2% to 44.1%.

Table 7.26 shows the confusion matrix obtained using two-sided tangent distance on

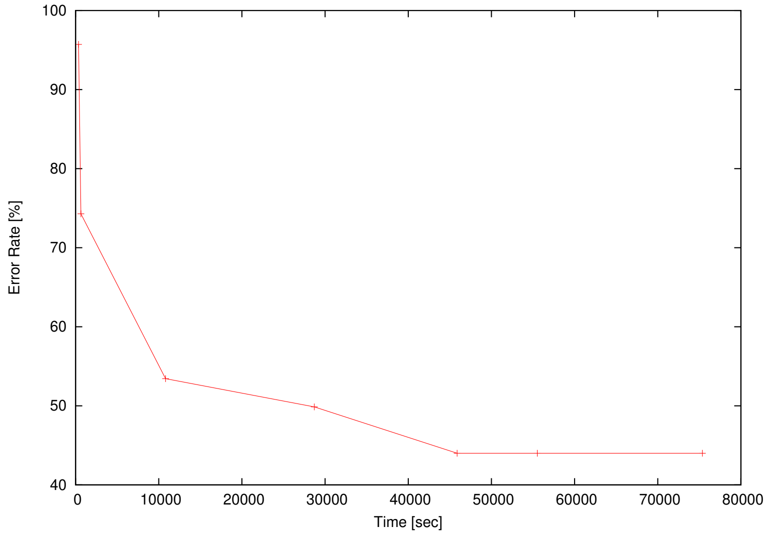


Figure 7.10. ER[%] with 32x32 original image features thresholded by their skin probability on the i6-Gesture database against Time[sec]: the biggest runtime was already achieved with pruning so we do not know the exact improvement factor. The biggest known runtime can be improved by a factor 2 without any changes in the error rate

the i6-Gesture database with original images thresholded by their skin probability as features (see Section 3.1.3). Table 7.27 shows all achieved results on this database up to now.

7.3.3 Dynamic Tracking

First, we did some experiments concerning the influence of the dynamic tracking parameters. For these tests we used the 1^{st} split of the LTI-Gesture database with 32x32 1^{st} time derivative features, mixture densities and a Gaussian distribution.

One can see from the results of Figure 7.11 that a high jump penalty or a small jump width can improve the error rate. This is due to an additional input of motion information, i.e. the short jump distances generate tracking a delay, as the tracking window moves slower than the gesturing hand and thus is not always centered in the tracking window.

Dynamic tracking can improve the error rate on the LTI-Gesture 1^{st} split database from 5.7% to 0.0% and from 18.5% to 6.4% on 2^{nd} split by using a maximum jump width of the tracking center of ± 5 pixels.

Table 7.26. Confusion matrix with two-sided tangent distance and camshift tracking on the i6-Gesture database, error rate 35.7% (C: correct, I: incorrect)

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	C/I	ER
A	0	8				4									2			1				1						3					1				8/12	60.0%
B	1		11			4								2				1	1													1					11/9	45.0%
C	2			12												3		1											3			1					12/8	40.0%
D	3		1		5	1						3	1		2			1							1		3			1		1					5/15	75.0%
E	4		1			17									1							1															17/3	15.0%
F	5			2			12			1						1									1				2		1						12/8	40.0%
G	6							17	2									1																			17/3	15.0%
H	7							3	16																			1									16/4	20.0%
I	8	2	1			2				8	5							1														1					8/12	60.0%
J	9	1				1					13			1				2		1				1													13/7	35.0%
K	10											17					1					1								1							17/3	15.0%
L	11			1					1			1	14		1			1											1								14/6	30.0%
M	12					3								10	4			1		1					1												10/10	50.0%
N	13	1				3						1		3	11							1															11/9	45.0%
O	14			2		1										9		1							1				5				1				9/11	55.0%
P	15							1									16	2									1										16/4	20.0%
Q	16	1		1					1								1	16																			16/4	20.0%
R	17										1	2						1	5			2		1			2			6							5/15	75.0%
S	18	1				5								6	3			1		3					1												3/17	85.0%
T	19							2			1			2							14											1					14/6	30.0%
U	20		1									3						1	2			5			1		2			4		1					5/15	75.0%
V	21																	1		1			17									1					17/3	15.0%
W	22		1															1					1	16											1		16/4	20.0%
X	23			1	1													1							15		1		1								15/5	25.0%
Y	24	0	1							2	1							1								15											15/5	25.0%
Z	25										1			1			1	1									16										16/4	20.0%
AE	26	2				1									1			1						1				14									14/6	30.0%
OE	27	1		2											1			2											14								14/6	30.0%
UE	28		1											1				1	2			2					2			10	1						10/10	50.0%
SCH	29											1						1													18						18/2	10.0%
Eins	30										3							1							1							15					15/5	25.0%
Zwei	31							1				1						1											1			2	14				14/6	30.0%
Drei	32							1										1														1	3	14			14/6	30.0%
Vier	33													2													1						1	1	15		15/5	25.0%
Fuenf	34							1																								1				18	18/2	10.0%
	C	8	11	12	5	17	12	17	16	8	13	17	14	10	11	9	16	16	5	3	14	5	17	16	15	15	16	14	14	10	18	15	14	14	15	18		
	I	9	7	9	1	25	0	9	4	3	12	12	1	18	15	4	3	29	5	3	0	8	1	3	7	0	12	4	13	12	2	11	6	1	1	0		
	I%	4	3	4	0	10	0	4	2	1	5	5	0	7	6	2	1	12	2	1	0	3	0	1	3	0	5	2	5	5	1	4	2	0	0	0		

We investigated also the use of different penalty functions (see Section 5.3.1 for explanations of the penalty functions). Figure 7.12 shows that the absolute penalty function achieves more constant results than the (squared) Euclidian distance function.

Table 7.27. Error rates for the i6-Gesture database with camshift tracking and different distance functions

Feature	Feature Size	Distance	ER[%]
Original thresholded by skin color prob.	32x32	Euclidan	44.0
	32x32	One-sided tangent	39.4
	32x32	Two-sided tangent	35.7
	16x16	One-sided tangent	46.0
	16x16	Two-sided tangent	42.5
1^{st} time derivative of orig. thresholded by skin color prob.	32x32	Euclidan	46.2
	32x32	Two-sided tangent	44.1

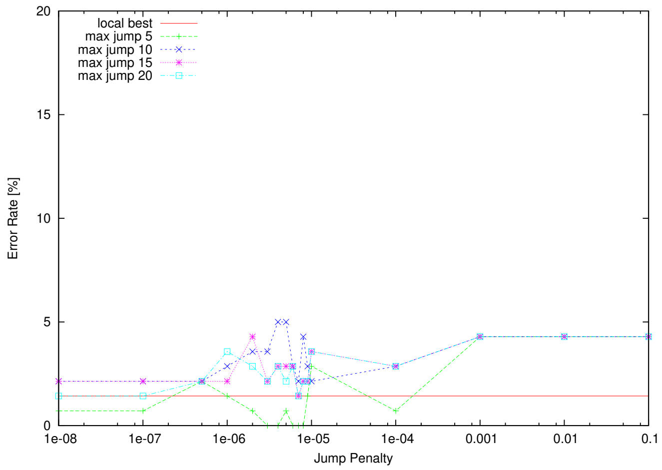


Figure 7.11. Dynamic tracking error rate on the LTI-Gesture database with Euclidian penalty function showing error rate vs. jump penalty weight

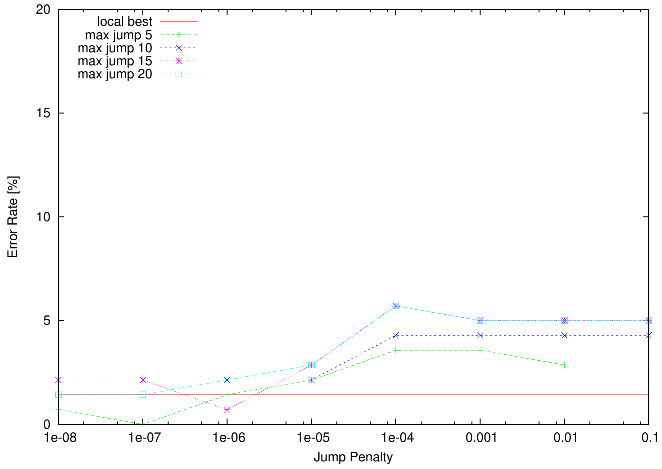


Figure 7.12. Dynamic tracking error rate on the LTI-Gesture database with absolute penalty function showing error rate vs. jump penalty weight

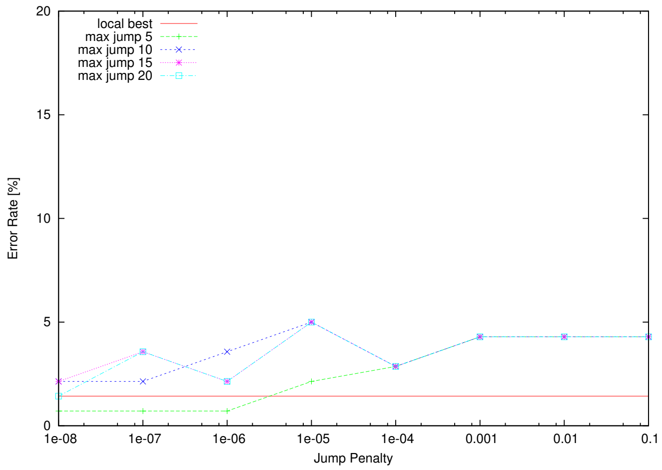


Figure 7.13. Dynamic tracking error rate on the LTI-Gesture database with squared Euclidian penalty function showing error rate vs. jump penalty weight

Chapter 8

Conclusion and Perspective

Conclusion. In this work, we showed that using simple but effective appearance-based features and appropriate models of image variability, we could obtain excellent results for various gesture recognition tasks. On the LTI-Gesture database and the DUISBURG-Gesture database we obtained very competetive results. On the newly created and much harder i6-Gesture database, we obtained very promising results. We gave a review of the different appearance-based features and the HMMs used.

Work in the field of gesture recognition usually first segmented parts of the input images — for example the hand — and then used features calculated from this segmented input. The obtained results presented in this work suggest that this intermediate segmentation step is not necessary.

The use of tangent distance and image distortion models as appropriate models of image variability in combination with appearance-based features was investigated and compared to the Euclidian distance. Using these distance measures, the error rate could be reduced on all regarded databases and shows the power of integrating these distance measures into the HMM emission probabilities for recognizing gestures.

Different tracking algorithms were analyzed and we developed a new dynamic programming tracking algorithm and showed its potential under noisy circumstances compared to camshift tracking. The use of tracking was investigated on the LTI-Gesture and i6-Gesture database and could reduce the error rate in both cases. Simple tracking algorithms like Bounding-Box tracking are very susceptible to noise but nevertheless they can lead to very good results on appropriate databases like the LTI-Gesture database.

All the features presented were used for appearance-based gesture recognition on three completely different databases:

- LTI-Gesture database: infrared images of only the gesturing hand, which can easily be segmented from the background; a constant environment and a sufficient resolution; used to control in-car devices

- DUISBURG-Gesture database: full body gestures in a body-centered space, 14 different persons gesturing 24 gestures; gray level images at a low resolution; used to control a robot

- i6-Gesture database: two views with full body gestures and gestures where only a section of the gesturing hand is visible, respectively; color images recorded under varying lighting conditions with 20 different persons gesturing 35 gestures of the German fingerspelling alphabet

These differences in the databases are also reflected in the results. We could achieve error rates between 1.4% and 2.8% without tracking and between 0.0% and 2.8% with tracking on the LTI-Gesture database comparable to the error rate of 4.5% presented in [Pelkmann 99]. On the DUISBURG-Gesture database we could achieve error rates between 13.9% and 22.2% comparable to the error rate of 7.1% presented in [Rigoll & Kosmala$^+$ 98]. These error rates, which are higher than the results obtained using COG-features, show that for this task it is sufficient to only consider global movement. The appearance-based approach here is affected by the high level of noise and the high intra-person variability. Nevertheless, the error rate is still competitive with the COG-based approach. The best achieved results up to now on the i6-Gesture database are ranging between 35.9% and 46.0% error rate and show the high complexity of this database and the need for additional preprocessing methods to normalize the images, as for example brightness and contrast normalization. Nevertheless, this result is promising because only a simple webcam without any restriction for the signer was used and some signs are visually very similar, as for example the signs for “M”, “N”, “A”, and “S”.

An important part of this work was the creation of the gesture recognition software, capable of recognizing different kinds of gestures using many different appearance-based features and distance measures. The options for the HMM architecture used to recognize the gestures were analyzed. It turned out that using variance thresholding in combination with fixed transition probabilities, a 0-1-2 model, pooling of the variances, and a high emission score weight led to the best results. Downscaling the images also has a high impact on the error rates and applying a Gaussian filter before downscaling led to the best results.

Also, a significant amount of time was dedicated to the creation of a new German fingerspelling alphabet database which will be freely available[1]. The i6-Gesture database has been recorded and is published to give all researchers working in the area of sign language and gesture recognition the possibility to compare the quality of their algorithms with ours.

Perspective. Many publications using HMM architectures were cited and show the activity in this research area. The integration of tangent distance and image distortion models as appropriate models of image variability in combination with appearance-based features is a new area of research in gesture recognition.

[1] Our database is freely available at `http://www-i6.informatik.rwth-aachen.de/~dreuw/`

At this point, still some questions remain unanswered, e.g. the missing IDM distance measure results on i6-Gesture database which are expected to improve the error rate. Also, not all distance measures were completely analyzed in combination with tracking on all databases. Further, the images of the i6-Gesture database recorded with the camcorder showing the complete body were not yet used for classification. Combining the images of the two cameras should also be investigated.

A future goal for the software created in this work is the integration of all distance measures available in the W2D library[2] and the extension to a continuous gesture recognition system (e.g. to control small desktop applications by simple gestures).

Developing applications for gesture recognition are encouraged by the growing amount of devices in household and everyday life. Systems like Konami's "Dance Dance Revolution" [Konami 02], Microsoft's investigations on "GWindows" [Wilson & Oliver 03] or the RespondDesigner's XBox game "Yourself!Fitness" [respondDesign 04], and Sony's EyeToy for Playstation [Sony 04] which has been sold more than 4 million times worldwide since its release are some examples for the commercial use of gesture recognition systems which show the interest of major enterprises in this area — and they also justify the research.

Also, an integration of our developed system in the existing Verbmobil project[3] may be an interesting step. "Verbmobil was a long-term interdisciplinary language technology project. The Verbmobil System recognizes spoken input, analyzes and translates it, and utters the translation. This speaker-independent system processes spontaneous speech. Verbmobil offers assistance in multilingual dialog situations in certain domains (e.g. scheduling appointments, travel planning and making hotel reservations). The project is a joint initiative involving information-technology companies, universities, and research centers" [Alexandersson & Buschbeck-Wolf^{+} 98].

However, because of the inherent complexity of the tasks, problems are still far from being solved (even with the use of stereo-cameras) in sign language recognition whereas simpler problems in gesture recognition can be solved under accurately defined environment constraints.

[2] W2D is a software package for appearance based image recognition supporting different pixel to pixel deformation models and was developed in [Gollan 03]. It is freely available at `http://www-i6.informatik.rwth-aachen.de/~gollan/w2d.html`

[3] `http://verbmobil.dfki.de/verbmobil/`

Appendix A

Software Documentation

In this chapter we give an overview of the software developed in the context of this work. We used many freely available programs such as Linux as operating system, gcc[1] compilers and XEmacs[2] to write programms, xv[3] , ImageMagick[4], The Gimp[5], GQView[6] and Mplayer[7] for viewing and manipulating images and videos, and finally LaTeX and xfig[8] for typesetting this thesis.

The implementation of the algorithms we used for camshift tracking was taken from the LTI-Lib[9] which is an object oriented library written in C++ with algorithms and data structures frequently used in image processing and computer vision. The camshift tracker was implemented as described in [Bradski 98].

The software developed in the context of this work is completely based on this LTI-Lib which gave us the possibility to build a quite complex software within the period fixed for this work.

GestureTooli6 (`gti6`)

The developed GestureToolI6 is the main software used to produce the results presented in Chapter 7. At the beginning this software was designed to view only the effects of applying filter or difference operations on image sequences and was then successively extended to a gesture recognition system.

Invocation of `gti6` is done as follows:

```
gti6 [OPTIONS] <filename1> [<filename2>]
```

and `gti6 --help` will show detailed help information.

[1] `http://gcc.gnu.org/`
[2] `http://www.xemacs.org/`
[3] `http://www.trilon.com/xv/`
[4] `http://www.imagemagick.org/`
[5] `http://www.gimp.org/`
[6] `http://gqview.sourceforge.net/`
[7] `http://www.mplayerhq.hu/`
[8] `http://www.xfig.org/`
[9] `http://ltilib.sourceforge.net/doc/homepage/`

Options for `gti6`

In the following the `[OPTIONS]` concerning data handling and manipulation, viewing, features and tracking are explained.

Data Options

`-st --splitTrain <id>` will split data into different datasets depending on <id>:

- 0 = train and test
- 1 = test and train
- 2 = Leaving-1-Person-Out
- 3 = NearestNeighbour (this requires a special database file)

`-lop --leaveOutPerson <uint>` will split the training data into leaving-one-person-out train- and testset and use the <uint> person as test set (implies --splitTrain(2)). This makes only sense if each person occurs only once and has the same id in each class. Can be used with the DUISBURG-Gesture database.

`-r --rotation <int>` will extend the training data by rotating each image by $\pm$ <int> angle degree (creates virtual data).

`-iY --inputScaleSizeY <int>` will scale input data to this height, x-dimension will be scaled symmetric

`-ft --fileType <id>` specifies the file format of the given filenames:

- 0 = original database file (default)
- 1 = lisp database file
- 2 = binary database file
- 3 = lisp feature file
- 4 = binary feature file

Viewing Options

If you enable the viewer, you can press the left mouse button to get more information about a pixel, or the right mouse botton to change the visualization options or to save a single image to file.

With the Left and Right keys you can change the file being displayed, with the up and down keys you can change the sequence and with the Q or X keys you can exit the viewer program.

With the keys “O”, “D”, “F”, “H”, “S” and “T” you can switch directly between the sequences (O)riginal, (D)iffernce, (F)eauture, (H)istory, (S)kin and (T)racker. Pressing “J” will save the whole currently showed sequence as JPEG images into the current folder. All files will have as prefix “viewer”. If a selected sequence or feature was not extracted a red warning image will be displayed instead.

To enable the viewer which will show each extracted sequence for a given database filename one have to select a viewpoint:

`-vo --viewOrig` will show first the original images

`-vd --viewDiff` will show first the difference images

`-vt --viewTrack` will show first the images as in tracking view

`-vs --viewSkin` will show first the skin-color probability channels

Additionally one can specify to the selected view the following parameters:

`-vb --viewBoxes` will overlay the tracking boxes on the selected view

`-vc --viewCOGs` will overlay the COGs on the selected view.

`-z --zoom <float>` will zoom each sequence by this value when displaying.

`-vy --viewOnly` is a special option is which will show only the selected features, no classification will be done. This can be useful when creating a database file with different images of different sizes only for viewing the effects of image transformations. The extracted features can still be written to file with `-wtrf` option or saved directly as images from the viewer by pressing “J”.

For example the combination `[...] -vo -vb -z 3 [...]` would display each sequence frame scaled by a factor 3 on screen, overlaid by the extraction box. The viewer would display the original images first.

Feature Options

Feature preprocessing options are:

`-fd --funcDiff <id>` set the difference function to calculate the time derivative image features:

- -1 = disabled (default)
- 0 = absolute difference $|(t+1)-(t-1)|$
- 1 = first time derivative $(t+1)-(t-1)$
- 2 = (unused)

- 3 = successor difference $(t+1) - (t)$
- 4 = second time derivative $(t-1) - 2*(t) + (t+1)$

`-N --nOfDiffImages <uint>` number of differences images to be considered for time derivative calculation (default=3)

`-X --scaleSizeX <double>` feature width for downscaled images (default=32)

`-Y --scaleSizeY <double>` feature height for downscaled images (default=32)

`-2ps --2PassScale` enables two pass downscaling of the images. This will downscale the images first to (X*2)*(Y*2) and then to X*Y.

`-ts --threshSeg` will threshold the original images with the -so values in the beginning!

`-hs --historySize <uint>` history size used for motion and skin history image features. Default is 0 which means full sequence history size

`-ht --historyTemplate` use history and energy image features for template matching with HMM, e.g. only one state will be used. If using template matching, only the features `mhi`, `mei`, `sei` and `shi` are supported.

The features are selected by `-F --Features <regexp>`. All available features can be combined by a regular expression:

`<feature>:[<option>{-<option>}]:[distance]:<weight>`
`{,<feature>:[<option>{-<option>}]:[distance]:<weight>}`.

Possible values for `<feature>` are:

`cog` or `centerOfGravity` to enable centroid features (COG-features)

`o` or `original` to enable original images as features

`d` or `difference` to enable time derivative images as features

`m` or `motion` to enable absolute motion images as features

`n` or `negative` to enable negative motion images as features

`p` or `positive` to enable positive motion images as features

`mhi` or `motionhistory` to enable motion history images as features

`mei` or `motionenergy` to enable motion energy images as features `sc` or `skincol` to enable skin color probabilitiy image features

`scd` or `skincoldiff` to enable skin color probabilitiy difference image features

`sct` or `skincolthresh` to enable original features thresholded by their skin color probability

`sctd` or `skincolthreshdiff` to enable time derivative image features of original images thresholded by their skin color probability

`shi` or `skinhistory` to enable skin probability history images as features

`sei` or `skinenergy` to enable skin probability energy images as features

All features can have multiple options. The specified options will be applied as they are ordered! Possible values for `<option>` are:

`sh` or `sobelh` will apply a horizontal sobel filter on the image feature

`sv` or `sobelv` will apply a vertical sobel filter on the image feature

`s` or `sobel` will apply a sobel filter on the image feature (magnitude)

`s2` or `sobel2` will apply a squared sobel filter on the image feature (squared magnitude)

`t` or `threshold` will thresholded the image feature by the value specified in `-tf` or `--threshFeat`

`e` or `equalize` will apply a histogram equalization on the image feature.

In the future, each feature will have its own distance function to be calculated in an HMM. Possible values for `<distance>` will be later the distance modes from the W2D library[10]. This parameter is not yet supported and can be left empty.

The parameter `<weight>` is used to weight each feature when combining and the default value is 1.0.

Example: combining the original sobel features, thresholded before applying the Sobel-filter, and the 1^{st} time derivative image features, downscaled to 32x32 with equal weights `[...] -tf 10 -fd 1 -Fo:ts-s::1.0,d:::1.0 -X 32 -Y 32 [...]`.

Also there exist some special feature selection parameters which have not been investigated in this work.

`-nf --neighborFeat` will enable neighbor features after extraction, i.e. the extracted features of X_{t-1}, X_t, and X_{t+1} will be concatenated to one feature vector.

`-spf --speechFeat` will enable special combination of original, 1^{st} and 2^{nd} time derivative image features

[10] `http://www-i6.informatik.rwth-aachen.de/~gollan/w2d.html`

Tracking Options

To extract features of an image sequence an area of interest ("aoiExtract") has to be specified. The simplest extraction area is the whole image, i.e. choosing -fb. The available tracking methods are:

-cn --centroidNorm set aoiExtract box to fixed extract box using centroid normalization of the current image

-bb --boundingBox set aoiExtract box to Bounding-Box rectangle

-eb --ellipseBox set aoiExtract box to motion-ellipse rectangle which depends on the COG-features then

-fb --fullBox set aoiExtract box to full image size rectangle (default)

-ct --camshiftTracking set aoiExtract box to the camshift tracker skin probability box

-dt --dynamicTracking enable dynamic tracking with fixed extract box to set aoiExtract box

The Bounding-Box tracking supports some additional options to control the size of the tracking box:

-bc --borderCor <uint> set border correction of Bounding-Box tracking rectangle, i.e. the rectangle will be enlarged by <uint> pixels in all directions

-ks --kernelSize <uint> size of the Gaussian filter kernel which will be applied before searching the Bounding-Box rectangle if greater than zero, otherwise disabled

-tbb --threshBBox <float> set lower threshold for area of interest in Bounding-Box search (default=1.0, i.e. one white pixel)

To control the behavior of the dynamic tracking one can specify the following options:

-dmj --dynaMaxJump <uint> maximum relative forward jump (default=5 pixels)

-dms --dynaMaxStep <uint> maximum grow/shrink step size for a tracking rectangle within local score calculation (default=5 pixels)

-di --dynaIterations <uint> number of grow/shrink iterations for a tracking rectangle within local score calculation (default=0, only fixed size rectangles)

-dpt --dynaPrunThresh <float> dynamic tracking pruning threshold for a tracking window score

-dpc --dynaPenalCenter <float> dynamic tracking penalty weight for tracking jumps

-dps --dynaPenalSize <float> dynamic tracking penalty weight for tracking grow/shrink

-dpf --dynaPenalFunc <id> dynamic tracking penalty function:

- 0 = euclidian distance (default)
- 1 = squared euclidian distance
- 2 = absolute distance

-dsw --dynaScoreWeight <float> dynamic tracking score weight relative local score (default=1.0)

-T --dynaTrackSpan <uint> number of sequence images to consider for score calculation (default=0, consider all)

-TB --dynaTraceSpan <uint> dynamic tracking nof images to look in the future when calling traceback (default=MaxInt, consider all)

The extraction size (in pixels) of the fixed size trackers centroidNorm, camshiftTracking and dynamicTracking can be set as follows:

-eX --extractSizeX <uint> set aoiExtract box width for fixed extract Box (default=70)

-eY --extractSizeY <uint> set aoiExtract box height for fixed extract Box (default=70)

HMM Options

The main HMM options are:

-tp --transProb <double> set fixed transition probability

-lsp --loopSkipProb <double> fixed loop/skip probability

-esw --emiScoreWeight <double> emission score weight. Use a value >1.0, to emphasize the emission or a value <1.0 to put more weight into the transitions (default=1.0)

`-essw --emiScoreStateWeight <double>` emission score state weight. Use a value between 1.0 and 2.0, to emphasize the emission scores at normalized position default=1.0 which means equal weights for all states

`-esst --emiScoreStateThresh <double>` emission score at this normalzed state will have the maximum weight defined by `--emiScoreStateWeight` (default=0.5, the center state)

`-S --nOfStates <int>` set the number hmm states to <int>. Special values are:

- -1 = average sequence length
- 0 = minimum sequence length (default)

`-minj --minJump <uint>` minimum relative forward jump within HMM (default=0)

`-maxj --maxJump <uint>` maximum relative forward jump within HMM (default=2)

`-md --maxDens <uint>` set maximum number of allowed densities per state (default=5)

`-msf --minScalingFactor <double>` set minimum scaling factor (variance) when calculating variances (default=0.1)

`-ls --laplaceScore` use Laplacian score function for the emission scores

`-gs --gaussScore` use Gaussian score function for the emission scores (default)

`-eml --estMaxLike` select estimator for given score function (default)

`-esd --estStdDev` use standard deviation (Gaussian maximum-likelihood)

`-emd --estMeanDev` use mean absolute deviation (Laplacian maximum-likelihood)

`-emdr --estMeanDevRoot` use root of mean absolute deviation (Laplacian maximum-likelihood)

`-pv --pruneViterbi <double>` set viterbi pruning threshold, e.g. 2e+8 (default is 1e+35, disabled)

An online recognition system was also integrated into the system but not described in this work. The online HMM has the following options:

`-pb --pruneBeam <float>` additive pruning threshold, e.g. 2e+8. For values lower than 0 no beam search is performed.

-ph --pruneHisto <uint> describes the maximum number of active hypotheses (histogram pruning). For value 0, no histogram pruning is performed (default 1000)

-B --nOfBuckets <uint> defines the number of buckets for the bucket-sort-algorithm used in histogram pruning

-atb --automaticTraceBack <uint> after the given number of timesteps, a partial trace back is performed and the calculated values are saved internally. (default is 0, disabled)

The variance pooling inside the HMM can specified by the following options:

-np --noPool disable pooling in HMM Training

-sp --statePool enable state pooling in HMM Training

-mp --modelPool enable model pooling in HMM Training (default)

-gp --globalPool enable global pooling in HMM Training

Distance options

-sd --scoreDistance use score distance (either Euclidian or L1, depending on the chosen score function) to calculate viterbi scores in classification (default)

-id --idmDistance use image distortion model distance to calculate viterbi scores in classification

-iwr --idmWarpRange <uint> set the IDM warprange to <uint> (default=1)

-ips --idmPatchSize <uint> set the IDM patchsize to <uint> (default=2)

-is --idmSobel will apply a Sobel-filter on the image features in IDM to calculate distance (default=false)

-w2d --w2dDistance <id> use W2D distance model to calculate viterbi scores in classification. This function is not yet supported but the available distance options will be as follows:

- 3 = tangentDistance2Sided
- 4 = tangentDistance1Sided
- 5 = p2DHMMDistance
- 6 = p2DHMDMDistance
- 7 = hungarianDMDistance

- 8 = w2DHMMDistance
- 9 = sA2DHMDMDistance
- 10 = iDMDistance
- 11 = iDM3Distance

`-td --tangentDistance` use tangent distance to calculate viterbi scores in classification

`-tc --tangentChoice <string>` tangent distance choice string, e.g 111111100 to enable the first 7 tangents

`-TS --tangentSides <1 | 2>` calculate (1) one-sided (default) or (2) two-sided tangent distance in classification

`-dd --distanceDev <double>` set unique variance in score functions (default=0.0, use estimated)

`-ttd --trainTestDist` enable the same distance settings for training and classification (default)

`-dr --distRotation <uint>` rotate the images by +-0 <uint> degree when calculating the distance in classification. The image with the best (shortest) distance will be used (default=0, no rotation)

`-dth --distThreshold <double>` use thresholded distance. Generally a good threshold is 5% or 10% of the maximum distance (default=0.0)

Misc

File handling and logging options:

`-dl --debuglevel <uint>` set the debugging information level to <uint> (default=10). Setting it to 0 will disable it.

`-l --log <file>` write std::clog stream to <file>

`-e --err <file>` write std::cerr stream to <file>

`-wtrf --writeTrainFeat <file>` write extracted train features in binary format to <file>

`-wtef --writeTestFeat <file>` write extracted test features in binary format to <file>

`-sfj --saveFeatJPEG` save all features as JPEG images into the folder `./feature_images/` – this can be a lot.

`-h --help` show detailed description on console

Image thresholding options:

`-to --threshOrig <float>` set lower threshold for thresholding in original image (default=0.0, disabled)

`-tf --threshFeat <float>` set lower threshold for thresholding in feature images (default=0.0, disabled)

`-tt --threshTrack <float>` set lower threshold for thresholding in tracking (default=0.0, disabled)

Examples for `gti6`

Example 1 In the following example original image features with a weight of 1.0 will be extracted with dynamic tracking on the LTI-Gesture database, using IDM distance with a patch size of 13x13 and IDM-Sobel-filter. Each feature will be downscaled to 32x32.

The tracker will allow a maximum jump width of the tracking center of 5 pixels in each frame with a penalty weight set to 0.1. Local grow/shrink will be disabled and the Euclidian distance between two tracking centers will used to calculate the penalty score. Pruning will be enabled. The tracking window will have a fixed size of 70x70.

The transition probability and the loop-skip probability of the chosen 0-1-2 model will be fixed in combination with a high emission score weight. Pruning will be used in HMM emission score calculation.

A Gaussian distribution will be used in combination with model pooling. Each HMM may have a different number of states which will depend on the minimum sequence length seen in training. Each state will have the same weight and the maximum state weight would be on the central state. Each state will have at most 5 densities.

As only one filename is given, a split option must be set to create a training and testing set. The first part of the specified database will be used for training, the last for testing.

```
gti6 --log logfile.log -dl 10 -fd 1  -F o:-::1.0 -X 32 -Y 32 -eX 70
-eY 70 -2ps -dt -dmj 5 -dpc 0.1 -di 0 -dpt 0.01 -dpf 0
-id -ips 6 -is -iwr 1  -S 0 -maxj 2 -md 5 -msf 0.1 -tp 0.4 -lsp 0.3
-esw 100000.0 -pv 2e+8 -essw 1.0 -esst 0.5 -gs -mp -eml
--splitTrain 0 lti_train.txt
```

Example 2 Example usage to obtain a result of 1.4% error rate on the LTI-Gesture 2^{nd} split with one-sided tangent distance with rotation by ±10 degree, full size extraction of the original image features, two-pass downscaled to 32x32:

```
gti6  -fd 1 -F o:-::1.0 -X 32 -Y 32 -2ps -fb  -td  -dr 10
-tc 111111000 -TS 1 -S 0 -maxj 2 -md 5 -msf 0.1 -tp 0.4 -lsp 0.3
-esw 100000.0 -pv 2e+8 -gs -mp -eml --splitTrain 1 lti_train.txt
```

Bibliography

[Alexandersson & Buschbeck-Wolf+ 98] J. Alexandersson, B. Buschbeck-Wolf, T. Fujinami, M. Kipp, S. Koch, E. Maier, N. Reithinger, B. Schmitz, M. Siegel: Dialogue Acts in VERBMOBIL-2 – Second Edition. Technical report, DFKI GmbH, July 1998.

[Balcells & DeMenthon+ 04] M. Balcells, D. DeMenthon, D. Doermann: An appearance-based approach for consistent labeling of humans and objects in video. *Pattern Analysis and Applications*, Vol., pp. in press, 2004.

[Bauer & Kraiss 02] B. Bauer, K.F. Kraiss: Video-Based Sign Recognition Using Self-Organizing Subunits. In *Proceedings of the 16th International Conference on Pattern Recognition ICPR*, Vol. 2, pp. 434–437, Québec City, Canada, August 2002.

[Black & Jepson 98] M.J. Black, A.D. Jepson: A probabilistic framework for matching temporal trajectories: Condensation-based recognition of gestures and expressions. In *European Conference on Computer Vision, ECCV-98*, Vol. 1406 of *Lecture Notes in Computer Science*, pp. 909–924, Freiburg, Germany, June 1998.

[Bobick & Davis 01] A.F. Bobick, J.W. Davis: The Recognition of Human Movement Using Temporal Templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 23, No. 3, pp. 257–267, March 2001.

[Bobick & Wilson 97] A.F. Bobick, A.D. Wilson: A State-Based Approach to the Representation and Recognition of Gesture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, No. 12, pp. 1325–1337, 1997.

[Bowden & Windridge+ 04] R. Bowden, D. Windridge, T. Kadir, A. Zisserman, M. Brady: A Linguistic Feature Vector for the Visual Interpretation of Sign Language. In J.M. Tomas Pajdla, editor, *8th European Conference on Computer Vision, ECCV04*, Vol. 1, pp. 391–401, Prague, Czech Republic, May 2004.

[Bradski 98] G.R. Bradski: Computer Vision Face Tracking For Use in a Perceptual User Interface. *Intel Technology Journal*, Vol. Q2, pp. 15–26, 1998.

[Brand & Oliver+ 97] M. Brand, N. Oliver, A. Pentland: Coupled hidden Markov models for complex action recognition. In *IEEE Conference on Computer Vision and*

Pattern Recognition, pp. 994–, Washington, DC, USA, June 1997. IEEE Computer Society.

[Bretzner & Laptev+ 01] L. Bretzner, I. Laptev, Y.S. Tony Lindeberg, Sören Lenman: A Prototype System for Computer Vision Based Human Computer Interaction. Technical report, Royal Institue of Technology Sweden, 2001. `http://www.nada.kth.se/cvap/abstracts/cvap251.html`.

[Comaniciu & Ramesh+ 00] D. Comaniciu, V. Ramesh, P. Meer: Real-Time Tracking of Non-Rigid Objects using Mean Shift. In *IEEE Conference on Computer Vision and Pattern Recognition*, Vol. 2, pp. 142–151, Hilton Head Island, South Carolina, USA, June 2000.

[Dahmen & Keysers+ 01] J. Dahmen, D. Keysers, H. Ney: Combined Classification of Handwritten Digits using the 'Virtual Test Sample Method'. In *MCS 2001, 2nd Int. Workshop on Multiple Classifier Systems*, Vol. 2096 of *Lecture Notes in Computer Science*, pp. 109–118, Cambridge, UK, May 2001. Springer.

[Darrell & Pentland 93] T. Darrell, A. Pentland: Space-time gestures. In *IEEE Computer Vision and Pattern Recognition*, pp. 335–340, New York, USA, June 1993.

[Deselaers 03] T. Deselaers: Features for Image Retrieval. Diploma thesis, RWTH Aachen University, Aachen, Germany, December 2003.

[Duda & Hart+ 01] R.O. Duda, P.E. Hart, D.G. Stork: *Pattern Classification*. New York: John Wiley & Sons, 2nd edition, 2001.

[Freeman & Anderson+ 98] W. Freeman, D. Anderson, P. Beardsley, C. Dodge, H. Kagez, K. Kyumaz, Y. Miyakez, M. Roth, K. Tanakaz, C.Weissman, W. Yerazunis: Computer Vision for Interactive Computer Graphics. *IEEE Computer Graphics and Applications*, Vol. 18, No. 3, pp. 42–53, May-June 1998.

[Fukunaga & Hostetler 75] K. Fukunaga, L. Hostetler: The Estimation of the Gradient of a Density Function, with Applications in Pattern Recognition. *IEEE Transactions in Information Theory*, Vol. 21, No. 1, pp. 32–40, January 1975.

[Gavrila 99] D.M. Gavrila: The Visual Analysis of Human Movement: A Survey. *Computer Vision and Image Understanding*, Vol. 73, No. 1, pp. 82–98, February 1999.

[Gavrila & Giebel+ 04] D.M. Gavrila, J. Giebel, S. Munder: Vision-based Pedestrian Detection: the PROTECTOR+ System. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pp. 13– 18, Parma, Italy, June 2004.

[Gollan 03] C. Gollan: Nichtlinieare Verformungsmodelle für die Bilderkennung. Diploma thesis, RWTH Aachen University, Aachen, Germany, September 2003.

[Haritaoglu & Harwood+ 98] I. Haritaoglu, D. Harwood, L.S. Davis: W^4S: A Real-Time System for Detecting and Tracking People in 2 1/2 D. In *International Conference on Automatic Face and Gesture Recognition*, pp. 222–227, Nara, Japan, April 1998.

[Hu 62] M.K. Hu: Visual pattern recognition by moment invariants. *IRE Transactions on Information Theory*, Vol. 8, No. 2, pp. 179–187, February 1962.

[Isard & Blake 98] M. Isard, A. Blake: CONDENSATION – conditional density propagation for visual tracking. *International Journal of Computer Vision*, Vol. 29, No. 1, pp. 5–28, August 1998.

[Jelinek 98] F. Jelinek: *Statistical Methods for Speech Recognition.* MIT Press, Cambridge, Massachusetts, January 1998.

[Jones & Rehg 98] M. Jones, J. Rehg: Statistical Color Models with Application to Skin Color Detection. Technical Report CRL 98/11, Compaq Cambridge Research Lab, pp. 274–280, 1998.

[Jones & Rehg 02] M.J. Jones, J.M. Rehg: Statistical color models with application to skin detection. *International Journal of Computer Vision*, Vol. 46, No. 1, pp. 81–96, January 2002.

[Kalman 60] R.E. Kalman: A New Approach to Linear Filtering and Prediction Problems. *Transaction of the ASME - Journal of Basic Engineering*, Vol. 82, pp. 35–45, March 1960.

[Keysers & Dahmen+ 03] D. Keysers, J. Dahmen, H. Ney, B. Wein, T. Lehmann: Statistical Framework for Model-based Image Retrieval in Medical Applications. *Journal of Electronic Imaging*, Vol. 12, No. 1, pp. 59–68, January 2003.

[Keysers & Macherey+ 01] D. Keysers, W. Macherey, J. Dahmen, H. Ney: Learning of Variability for Invariant Statistical Pattern Recognition. In *12th European Conf. on Machine Learning*, Vol. 2167 of *Lecture Notes in Computer Science*, pp. 263–275, Freiburg, Germany, September 2001. Springer.

[Keysers & Macherey+ 04] D. Keysers, W. Macherey, H. Ney, J. Dahmen: Adaptation in Statistical Pattern Recognition using Tangent Vectors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 26, No. 2, pp. 269–274, February 2004.

[Kim & Chalidabhongse+ 04] K. Kim, T.H. Chalidabhongse, D. Harwood, L. Davis: Background Modeling by Codebook Construction. In *IEEE International Conference on Image Processing*, pp. 24–27, Singapore, October 2004.

[Knoll & Brinkley+ 85] T. Knoll, L. Brinkley, E. Delp: Difference picture algorithms for the analysis of extracellular components of histological images. *Journal of Histochemistry and Cytochemistry*, Vol. 33, No. 4, pp. 261–267, January 1985.

[Koller & Weber+ 94] D. Koller, J. Weber, J. Malik: Robust multiple car tracking with occlusion reasoning. In *Proceedings of the third European conference on Computer vision*, Vol. 1, pp. 189–196, Stockholm, Sweden, May 1994. Springer-Verlag.

[Konami 02] Konami: Dance Dance Revolution. Website of Konami's Dance Dance Revolution, visited 10.01.2005, 2002. `http://www.konami.co.jp/am/ddr/ddr/`.

[Little & Boyd 98] J.J. Little, J.E. Boyd: Recognizing People by Their Gait: The Shape of Motion. *eJournal of Computer Vision Research: Videre*, Vol. 1, No. 2, 1998.

[Maes & Darrell+ 97] P. Maes, T. Darrell, B. Blumberg, A. Pentland: The ALIVE system: wireless, full-body interaction with autonomous agents. *Special issue on multimedia and multisensory virtual worlds*, Vol. 5, No. 2, pp. 105–112, March 1997.

[Megret & DeMenthon 02] R. Megret, D. DeMenthon: A Survey of Spatio-Temporal Grouping Techniques. Technical Report LAMP-TR-094,CS-TR-4403,UMIACS-TR-2002-83,CAR-TR-979, University of Maryland, College Park, August 2002.

[Micilotta & Bowden 04] A. Micilotta, R. Bowden: View-based Location and Tracking of Body Parts for Visual Interaction. In *BMVC*, Vol. 2, pp. 849–858, Kingston UK, September 2004.

[Moeslund & Granum 01] T.B. Moeslund, E. Granum: A Survey of Computer Vision-Based Human Motion Capture. *Computer Vision and Image Understanding*, Vol. 81, No. 3, pp. 231–268, 2001.

[Moore & Essa 02] D. Moore, I. Essa: Recognizing multitasked activities from video using stochastic context-free grammar. In *18th national conference on Artificial Intelligence*, pp. 770–776. American Association for Artificial Intelligence, July 2002.

[Morrison & McKenna 04] K. Morrison, S.J. McKenna: An Experimental Comparision of Trajectory-Based and History-Based Representation for Gesture Recognition. In *Gesture-Based Communication in Human-Computer Interaction / International Gesture Workshop*, number 1739 in Lecture Notes in Computer Science, pp. 152–163, Gif-sur-Yvette, France, March 2004. Springer-Verlag.

[Ney 99] H. Ney: Mustererkennung und Neuronale Netze. Script to the Lecture on Pattern Recognition and Neural Networks held at RWTH Aachen, 1999.

[Ney 04] H. Ney: Speech Recognition SS04. Script to the Lecture on Speech Recognition held at RWTH Aachen, 2004.

[Nguyen & Bui+ 03] N. Nguyen, H. Bui, S. Venkatesh, G. West: Recognising and monitoring highlevel behaviours in complex spatial environments. In *IEEE International Conference on Computer Vision and Pattern Recognition*, Vol. 2, pp. 620–625, Madison, Wisconsin, June 2003.

[Nielsen 00] J. Nielsen: *Designing Web Usability: The Practice of Simplicity.* New Riders Publishing, Indianapolis, 2000.

[Orio 99] N. Orio: A model for human-computer interaction based on the recognition of musical gestures. Vol. 4, pp. 333–338, October 1999.

[Pavlovic & Sharma+ 97] V. Pavlovic, R. Sharma, T.S. Huang: Visual Interpretation of Hand Gestures for Human-Computer Interaction: A Review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, No. 7, pp. 677–695, July 1997.

[Pelkmann 99] A. Pelkmann: Entwicklung eines Klassifikators zur videobasierten Erkennung von Gesten. Diploma thesis, RWTH Aachen University, Aachen, Germany, February 1999.

[Prillwitz & Leven+ 89] S. Prillwitz, R. Leven, H. Zienert, T. Hanke, J. Henning: *HamNoSys. Version 2.0; Hamburg Notation System for Sign Languages. An introductory guide*, Vol. 5 of *Internationale Arbeiten zur Gebärdensprache und Kommunikation Gehörloser*. Signum Verlag, 1989.

[Rabiner 89] L.R. Rabiner: A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*, Vol. 77, No. 2, pp. 257–286, February 1989.

[Raja & McKenna+ 98] Y. Raja, S.J. McKenna, S. Gong: Tracking and Segmenting People in Varying Lighting Conditions using Colour. In *3rd IEEE International Conference on Face and Gesture Recognition*, pp. 228–233, Nara, Japan, April 1998.

[respondDesign 04] respondDesign: yourself!fitness. Website of respondDesign's yourself!fitness, visited 10.01.2005, 2004. `http://www.yourselffitness.com/`.

[Rigoll & Kosmala+ 98] G. Rigoll, A. Kosmala, S. Eickeler: High Performance Real-Time Gesture Recognition using Hidden Markov Models. In *International Gesture Workshop Bielefeld*, Vol. 1371, pp. 69–80, Bielefeld, Germany, September 1998. Springer-Verlag.

[Schlenzig & Hunter+ 94] J. Schlenzig, E. Hunter, R. Jain: Recursive identification of gesture inputs using hidden markov models. In *Second Annual Conference on Applications of Computer Vision*, pp. 187–194, Sarasota, FL, USA, December 1994.

[Schweibenz & Thissen 02] W. Schweibenz, F. Thissen: *Qualität im Web. Benutzerfreundliche Webseiten durch Usability Evaluation (X.media.press)*. Springer-Verlag, March 2002.

[Sigal & Sclaroff+ 00] L. Sigal, S. Sclaroff, V. Athitsos: Estimation and prediction of evolving color distributions for skin segmentation under varying illumination. In *Computer Vision and Pattern Recognition*, Vol. 2, pp. 152–159, Hilton Head Island, SC, USA, June 2000.

[Simard & Le Cun+ 98] P. Simard, Y. Le Cun, J. Denker, B. Victorri: Transformation Invariance in Pattern Recognition — Tangent Distance and Tangent Propagation. Vol. 1524, pp. 239–274, Heidelberg, 1998. Springer.

[Singh 89] A. Singh: Digital change detection techniques using remotely-sensed data. *International Journal of Remote Sensing*, Vol. 10, pp. 989–1003, May 1989.

[Sony 04] Sony: Get off the couch and into the game this holiday season with EyeToy: Antigrav, exclusivley for Playstation2. Playstation Website of Sony Computer Entertainment America Inc., visited 10.01.2005, November 2004. `http://www.us.playstation.com/`.

[Starner & Leibe+ 03] T. Starner, B. Leibe, D. Minnen, TracyWestyn, A. Hurst, JustinWeeks: The perceptive workbench: Computer-vision-based gesture tracking, object tracking, and 3D reconstruction for augmented desks. *Machine Vision and Applications*, Vol. 14, No. 1, pp. 59 – 71, April 2003.

[Starner & Pentland 95] T. Starner, A. Pentland: Real-Time American Sign Language Recognition From Video Using Hidden Markov Models. In *International Symposium on Computer Vision*, pp. 265–270, Coral Gables, Florida, USA, November 1995.

[Starner & Weaver+ 98] T. Starner, J. Weaver, A. Pentland: Real-time American sign-language recognition using desk and wearable computer based video. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, Vol. 20, No. 12, pp. 1371–1375, December 1998.

[Stokoe 80] W.C. Stokoe: *Sign language structure. An Outline of the visual communication system of the American deaf*, Vol. 8 of *Studies in Linguistics. Occasional Papers.* Buffalo Press, N.Y., University of Buffalo, New York, 1980.

[Sutton 77] V. Sutton: Sutton movement shorthand writing tool for research. In J. Stokoe, William C., editor, *First National Symposium on Sign Language Research and Teaching*, pp. 267–296, Chicago,Ill, July 1977. National Association of the Deaf. `http://www.signwriting.org`.

[Triesch & von der Malsburg 02] J. Triesch, C. von der Malsburg: Classification of Hand Postures Against Complex Backgrounds Using Elastic Graph Matching. *Image and Vision Computing*, Vol. 20, No. 13-14, pp. 937–943, December 2002.

[Uchida & Sakoe 03] S. Uchida, H. Sakoe: Handwritten character recognition using elastic matching based on a class-dependent deformation model. In *7th Int. Conf. Document Analysis and Recognition*, pp. 163–167, Edinburgh, Scotland, August 2003.

[Vogler & Metaxas 01] C. Vogler, D. Metaxas: A Framework for Recognizing the Simultaneous Aspects of American Sign Language. *Computer Vision and Image Understanding*, Vol. 81, No. 3, pp. 358–384, March 2001.

[Welch & Bishop 03] G. Welch, G. Bishop: An Introduction to Kalman Filter. Technical Report TR 95-041, University of North Carolina at Chapel Hill, North Carolina, May 2003.

[Wilson & Bobick 95] A. Wilson, A. Bobick: Learning visual behavior for gesture analysis. In *IEEE International Symposium on Computer Vision*, pp. 229–234, November 1995.

[Wilson & Bobick 00] A. Wilson, A. Bobick: Realtime online adaptive gesture recognition. In *IEEE 15th International Conference on Pattern Recognition*, Vol. 1, pp. 270–275, Barcelona, Spain, September 2000.

[Wilson & Oliver 03] A. Wilson, N. Oliver: GWindows: Robust Stereo Vision for Gesture-Based Control of Windows. In *International Conference on Multimodal Interfaces*, pp. 211–218, Vancouver, Canada, November 2003.

[Wren & Azarbayejani$^+$ 97] C.R. Wren, A. Azarbayejani, T. Darrell, A. Pentland: Pfinder: Real-Time Tracking of the Human Body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, No. 7, pp. 780–785, 1997.

[Wu & Chiu$^+$ 04] C.H. Wu, Y.H. Chiu, K.W. Cheng: Error-Tolerant Sign Retrieval using Visual Features and Maximum A Posteriori Estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 26, No. 4, pp. 495–508, April 2004.

[Yamato & Ohya$^+$ 92] J. Yamato, J. Ohya, K. Ishii: Recognizing human action in time-sequential images using hidden Markov models. In *IEEE Computer Vision and Pattern Recognition*, pp. 379–385, IL, USA, June 1992.

[Yang & Levine 92] Y.H. Yang, M.D. Levine: The background primal sketch: An approach for tracking moving objects. *Machine Vision and Applications*, Vol. 5, No. 1, pp. 17–34, January 1992.

[Yang & Xu$^+$ 97] J. Yang, Y. Xu, C. Chen: Human action learning via hidden Markov model. *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 27, No. 1, pp. 33–34, January 1997.

[Zhu & Yang$^+$ 00] X. Zhu, J. Yang, A. Waibel: Segmenting Hands of Arbitrary Color. In *Automatic Face and Gesture Recognition*, pp. 446–453, Grenoble, France, March 2000.

[Zobl & Nieschulz$^+$ 04] M. Zobl, R. Nieschulz, M. Geiger, M. Lang, G. Rigoll: Gesture Components for Natural Interaction with In-Car Devices. In *Gesture-Based Communication in Human-Computer Interaction / International Gesture Workshop*, Vol. 2915 of *Lecture Notes in Artificial Intelligence*, pp. 448–459, Gif-sur-Yvette, France, March 2004. Springer-Verlag.