FAKULTA
APLIKOVANÝCH VĚD
ZÁPADOČESKÉ
UNIVERZITY
V PLZNI

Západočeská Univerzita v Plzni
Fakulta Aplikovaných Věd

Katedra Kybernetiky

Extrakce příznaků pro automatickou analýzu gest znakového jazyka

DISERTAČNÍ PRÁCE

k získání akdemického titulu doktor
v oboru **Kybernetika**

Ing. Marek Hrúz

Advisor: Doc. Ing. Luděk Müller, Ph.D.

Plzeň, 2013

FACULTY
OF APPLIED SCIENCES
UNIVERSITY
OF WEST BOHEMIA

University of West Bohemia
Faculty of Applied Sciences

Department of Cybernetics

Feature Extraction for Automatic Analysis of Sign Language Gestures

DOCTORAL THESIS

submitted in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy in the field of
Cybernetics

Ing. Marek Hrúz

Advisor: Doc. Ing. Luděk Müller, Ph.D.

Plzeň, 2013

Acknowledgments

I would like to thank to all people who supported me in my studies. My special thanks goes to my advisor Doc. Ing. Luděk Müller, Ph.D. and all the colleagues on our department for the advice and the encouragement they provided.

I would like to extend my thanks to the Ministry of Education, Youth and Sports of the Czech Republic, the Technological Agency of the Czech Republic, and the Grant Agency of Academy of Sciences of the Czech Republic for the financial support. The access to computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum, provided under the programme "Projects of Large Infrastructure for Research, Development, and Innovations" (LM2010005) was highly appreciated.

Abstract

The automatic sign language recognition is a broad problem. It has been addressed by researchers with different backgrounds. Depending on the point of view the problem can be decomposed into several areas of research: signal processing, computer vision, human computer interaction, system theory, machine learning, language processing and more. There are many publications that discuss the individual aspects of the sign language recognition but the research is still behind a related and older problem of speech recognition. It is very hard to determine the state-of-the-art in sign language recognition. We can make use of the state-of-the-art methods of the individual approaches but the results cannot be compared directly. The main reason for that is that there is no unified corpus that could be used as a benchmark for the different approaches. On the other hand the existence of such a corpus is highly questionable since the raw data describing the sign language can be very distinct (camera, data-glove, depth imaging, ...).

In this work I present several corpora that were recorded using a visual camera system for the purpose of isolated sign recognition. I have designed a system capable of tracking the hands and head in a controlled environment utilizing methods of computer vision and machine learning. The system is designed in a probability framework which enables classification from multiple sources (multi-modal approach) or to allow the application of standard adaptation techniques such as MAP (maximum a posteriori) or MLLR (maximum likelihood linear regression). Features are extracted from the tracked body parts and used for recognition using a hidden Markov model.

Keywords: sign language recognition, visual tracking, machine learning

Abstrakt

Automatické rozpoznávání znakového jazyka je problém se širokým záběrem. Byl řešen výzkumníkmi z různých oblastí. Závisle na úhlu pohledu může být tento problém rozdělen do několika výzkumných odvětví: zpracování signálu, počítačové vidění, interakce mezi člověkem a počítačem, teorie systémů, strojové učení, zpracování jazyka a další. Existuje mnoho publikací, které pojednávají o individuálních aspektech znakového jazyka, ale výzkum v této oblasti stále zaostává za příbuzným a starším problémem rozpoznávání řeči. Je velice těžké určit, které existující metody jsou pro rozpoznávání znakového jazyka nejlepší. V jednotlivých pohledech lze identifikovat nejlepší přístupy, ale porovnat výsledky je těžké. Souvisí to i s faktem, že neexistuje unifikovaný korpus, který by se používal pro srovnání výsledků. Na druhou stranu už samotná možnost existence takového korpusu je sporná, protože data, které popisují znakovou řeč, můžou mít velice rozdílný charakter (kamera, datové rukavice, snímaní hloubky).

V této práci představuju několik korpusů, které byly zaznamenány pomocí systému využívajícího kamery, se záměrem rozpoznávat izolované znaky. Navrhl jsem systém schopný sledovat ruky a hlavu v kontrolovaném prostředí, využívajícím metody počítačového vidění a strojového učení. Systém je navržen jako pravděpodobnostní, což umožňuje klasifikaci z více zdrojů, a zároveň je připravený na aplikaci standardních adaptačních metod jako MAP nebo MLLR. Ze sledovaných objektů jsou extrahovány příznaky, které jsou použity pro rozpoznávání za použití skrytých Markovových modelů.

Klíčové slova: rozpoznávání znakového jazyka, vizuální sledování objektů, strojové učení

Prohlášení

Prohlašuji, že jsem tuto disertační práci vypracoval samostatně, s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí.

V Plzni dne   Marek Hrúz

Contents

List of Tables

List of Figures

List of Abbreviations

AAM	Active Appearance Model
ASL	American Sign Language
ASM	Active Shape Model
CAMSHIFT	Continuously Adaptive Mean Shift Tracking
DOF	Degrees of Freedom
ECHO	European Cultural Heritage Online
GMM	Gaussian Mixture Model
HCI	Human-Computer Interaction
HMM	Hidden Markov Model
HOG	Histogram of Oriented Gradients
HSV	Hue Saturation Value Color Model
ICA	Independent Component Analysis
IMDI	ISLE Meta Data Initiative
LBP	Local Binary Pattern
LPC	Linear Predictive Coding
MAP	Maximal a Posteriori
NGT	Nederlandse Gebarentaal
PCA	Principal Component Analysis
RGB	Red Green Blue Color Model
SL	Sign Language
SLR	Sign Language Recognition
SMA	Specialized Mappings Architecture
UKF	Unscented Kalman Filter

Chapter 1

Introduction

1.1 Sign Language

Sign Language (SL) is defined as a language that uses a system of manual, facial, and other body movements as the means of communication. Generally every spoken language has its signed counterpart. Here we should note that natural sign languages are not a derivation of spoken languages, but they are natural languages that arose from the community of deaf people. Such languages have their own grammar. There are also artificial forms of sign languages which I will refer to as signed languages. These forms usually copy the form of spoken languages. Roughly speaking you can imagine that every word from the spoken language is signed. The methods used in this PhD thesis can be generally used for any sign/signed language, but because of the origin of the author and the available data we will now define the Czech Sign Language and other important elements according to the Czech law:

- **The Deaf** - is a person who is unable to hear since the birth or has lost hearing before the development of the spoken language, or person who suffers from full or practical deafness which occurred after the development of spoken language and a person who is hard of hearing and unable to understand spoken language.
- **Czech Sign Language** - is a basic communication system of the deaf persons in Czech Republic who them-selves consider it to be their main form of communication.

1.2 History of Sign Language

The beginnings of Sign are traced back to prehistorical era. Some researchers theorize that prehistoric humans used signs prior to vocal apparatus to communicate. These theories are based on paleontologist findings that early humans had not have the voice space to accommodate the complex speech apparatus. This theory puts the signing to a position of natural communicator as opposed to what people thought before the Renaissance era.

According to Aristoteles people learn through speech and therefore deaf people are unable to learn. This had put the deaf community to a difficult position on the edge of the society. Two thousand years later scholars were attempting to educate the first deaf people to disprove the old beliefs.

Geronimo Cardano, an Italian mathematician and physician, was probably the first scholar to identify that learning does not require hearing. He discovered, in the 1500s, that the deaf were able to be educated by using written words. He used his methods to educate his deaf son. Pedro Ponce de Leon, a Spanish monk, was very successful with his teaching methods while teaching deaf children in Spain. This was around the same time that Cardano was educating his deaf son. The written history of sign language began in the 17^{th} century in Spain. The work of Juan Pablo Bonet called *Reduction of letters and art for teaching mute people to speak* [1] is considered to be the first modern treatise of Phonetics and Logopedia. In this work a manual alphabet is presented to improve the communication of deaf people.

As technology progressed a new scientific field arose called the **Human-Computer Interaction** (HCI). This field studies the interaction between people and computers. According to the model of gesture, gestures originate as a gesturer's mental concept, possibly in conjunction with speech. They are expressed through the motion of arms and hands. From this point of view the hands are articulators. Several taxonomies have been suggested in the literature that deal with psychological aspects of gestures. From the point of view of HCI the developed taxonomy can be seen in figure 1.1 [2].

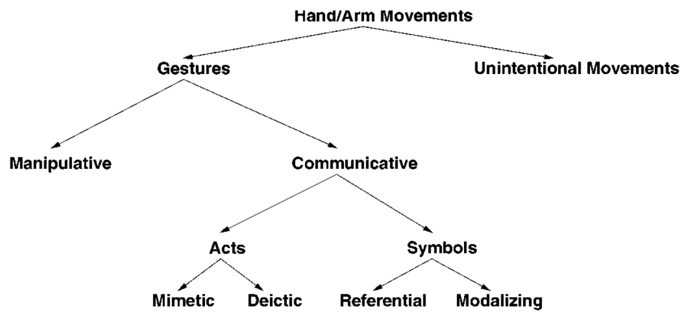

Figure 1.1: A taxonomy of hand gestures for HCI [2]

All hand/arm movements are first divided into two major classes. Gestures and unintentional movements. The unintentional movements do not convey any meaningful information. What is interesting for this work is the class of gestures. Sign language is composed of communicative gestures with symbolic (linguistic) meaning.

1.3 Motivation

This work deals with feature extraction for Automatic Sign Language Recognition (SLR) particularly the manual component of Sign Language. Feature extraction is a crucial part of a recognition system. The features should capture interesting and informative aspects of the process and represent them reasonably. The motivation is to create an automatic feature extraction system that would provide features for a recognition and categorization system using visual cues. The main reason for developing a system of SLR is to enable the communication between a deaf person and a hearing person. The SLR system is one part of the communication framework which can be seen in Figure 1.2.

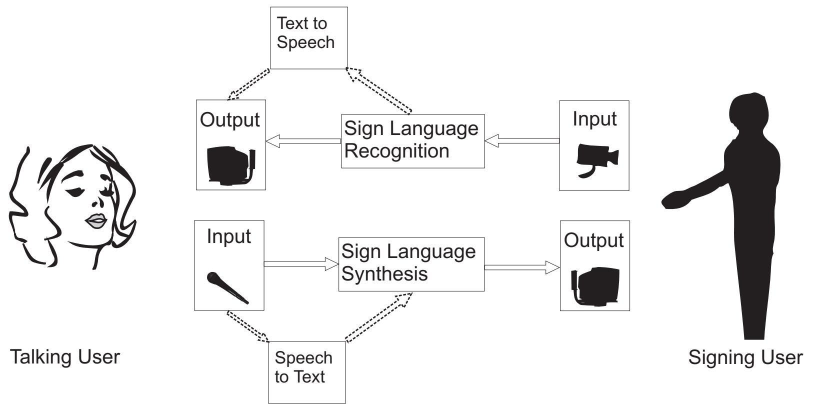

Figure 1.2: A setup enabling communication between a deaf person and a hearing person

In such a setup the deaf person is signing without the knowledge of the spoken or written form of the language. The signs have to be captured with some device enabling the automatic recognition. Such a device can be a camera or data gloves or any sensor capable of capturing important information about the signing. In this work we will discuss the visual sensors and data gloves. Both approaches have some advantages and disadvantages (see Chapter 2). Features are extracted from the captured data and used for recognition of the performed signs. Either statistical or rule based methods can be used. Depending on the form of the sign language the recognized signs have to be translated to a form of spoken language. It can be text or speech. The opposite direction requires the input from the hearing person. A sign language synthesis module generates the appropriate signs that are graphically presented to the deaf user. Such systems should be robust and with real-time capabilities. Another application with the feature of SLR can be an electronic dictionary which enables an input from camera to help search for a sign. Or a SL tutoring tools can be developed to teach SL. These are the applications that are targeted by this work. They utilize visual input. For other applications different inputs can be more suitable. If we would like to recognize signs in one place (eg. post office) a more suitable device for capturing the signs could be used. For example Kinect or data gloves. The measurements from these devices are easier to represent and recognize, but the availability of them is lower than visual cameras.

1.4 Goals of Dissertation

The main goal of this dissertation thesis is the development of a feature extraction system of the manual component of SL. The system should provide features for a recognizer capable of recognizing isolated signs or basic components of a sign gesture or for automatic sign categorization. The feature extraction system should be robust and fast and without the use of markers. One of the problems it should handle are occlusions. Usually the researchers try to identify individual body parts that form the occlusion. My assumption is that the occluded body parts carry important information about the signs. There should be no need for separating the body parts to successfully identify the sign. To summarize the goals:

- The design and implementation of a system capable of tracking body parts with explicit occlusion model
- The input will be obtained from visual system providing color information
- The tracking will not use any markers
- The tracking framework will be probabilistic
- The obtained features will be used for recognition and categorization
- Find a minimal set of features using methods of dimension reduction and decorrelation

For the purpose of testing the system of feature extraction a database of sign language is needed. In cooperation with colleagues from our department two corpora have been prepared. The UWB-06-SLR-A corpus (Section 3.1) and the UWB-07-SLR-P corpus (Section 3.2). In addition a third corpus is used which consists of recordings used in an online dictionary available at `http://znaky.zcu.cz`. The features of the corpora are:

- Non-changing light conditions
- Isolated signs
- Both long and short sleeves
- Uniform background
- Non-skin-colored clothes
- No markers
- Two different views of the scene
- Detailed view of the face

It can be seen that the conditions are laboratory-like. The system of feature extraction will be tested on these corpora. After these tests the system will be tested on entries of on-line sign language dictionary.
The efficiency of the features will be measured via a SLR system. The precision of the physical features (position, orientation, shape, etc.) will be measured using a similarity measure between the obtained features and annotated features. The annotation of features is very time consuming. For this purpose I have developed a system of semi-automatic annotation of sign language corpora that helps the annotator annotate easily detectable features automatically. The system is described in detail in Chapter 6.

The practical usage of the system is to automatically track the body parts in entries of the sign language dictionary. This is also a goal of this dissertation. The obtained features can be used in several ways; for example for automatic annotation of SL gestures or automatic processing of recordings of isolated signs.

1.5 Outline

The outline of this work is as follows. The second chapter presents systems of SLR. Next, the relevant features for SL are mentioned and put into context with the Czech Signed Language. In the third chapter the available SL corpora are introduced. In the next chapter some state-of-the-art image parametrization methods are shown. These include shape representations, shape recognition techniques, visual tracking and texture description. The next chapter (Chapter 5) concerns about the research on the manual component of SL and hand tracking. In the next chapter a tracking system is presented as a main output of this work. Lastly the experiments are presented and conclusions are made.

Chapter 2

Sign Language Recognition

As any other recognition system, **Sign Language Recognition** begins with the acquisition of data. The information that we are interested in is the movement of the hands and the head and the non-manual component of the SL represented by facial expression, pose, the speed of the movement and so on. Generally there are two methods for obtaining the data that contain the necessary information.

- Direct-measure (glove-based) devices
- Vision-based devices such as camera

A survey about the glove-based devices can be found in [3]. While interesting for the SLR, these devices are not in the scope of this work. More interesting are the vision-based devices. There are some basic assumptions that need to be fulfilled in order to capture all interesting features of a sign. The signs are performed in a space called the **basic signing space**. It is outlined by the vertex (of the head), the lower part of the body and the elbows when arms are stretched sideways. It is crucial for the recognition to capture this space with a camera. There are many works that consider only this view and try to analyze the signs captured only by one camera. Other proposed methods use more cameras or alternatively active vision can be used. These methods are usually adopted to avoid occlusions between objects, to capture the head and the hands in a higher resolution and to obtain the trajectories in 3D space. Recently the development of 3D cameras capable of capturing the depth of the scene is making progress. Some work use the 3D data for SLR [4], [5]. In general, both manual and non-manual components can by used for SLR. In this work I focus on the manual component.

2.1 Manual Component of Sign Language

As mentioned in [6] sign linguists distinguish the basic components (or phoneme subunits) of a sign gestures consisting of the handshape, hand orientation, location, and movement. The handshape is determined by the configuration of the fingers. There exist approximately 30 handshapes, although the number is different for every sign language. According to Macurová

[7] in Czech sign language there are 43 different handshapes. She has identified 12 basic handshapes which she divided into 5 groups. See Table 2.1. The other handshapes are

closed hand	bended hand
hand with fingers together (not spread)	hand with spread fingers
closed hand with lifted fingers	

Table 2.1: Groups of basic handshapes

understood as shapes derived from these basic forms and in the sense of notation they are distinguished by signs added above or/and before the letter representing the basic shape. For the recognition purposes each handshape should be interpreted by a different value of the handshape feature. It is questionable whether a high level description of the handshape is inevitable for a successful recognition.

The hand orientation refers to the direction in which the palm and the fingers are pointing relative to the body. Usually 6 different orientations are considered. See Table 2.2. A feature

up-wards	down-wards
to the body	away form the body
to the right	to the left

Table 2.2: Basic orientations of the hand

of the hand orientation can be represented by an angle formed between the palm (fingers) and the main axis of the body.

Location refers to the space where the sign is performed. This space is a subset of the basic signing space. Some signs can be performed outside of this space but they are considered as exceptions. The spaces (or places) of articulation can be seen in Table 2.3. The feature

neutral space	whole face	upper part of the head
upper part of the face, forehead	eye(s)	nose
lower part of the face, chin	under the chin	mouth and the lips
cheaks	ear(s)	neck
upper part of the body	lower part of the body	shoulders
chest	waist	left/right edge of the body
upper part of the arm	lower part of the arm	elbow
outer wrist	inner wrist	hips
thigh	from knee to the ankle	

Table 2.3: Places of articulation

of location of a hand can be represented as a 2D coordinate in a coordinate system where the head (or other identified part of the body) is the origin and a known distance is a unit distance. For example the width of the head or the distance from the neck to the shoulder.

The last feature of the manual component of the SL is the movement. It can be divided into several types (Table 2.4).

up-wards	down-wards	up and down
to the right	to the left	from one side to the other
to the body	away from the body	to and away from the body
interjection	accession	groving in distance
switch	variation	crossing
junction/hold	passage	contact
circular movement	twisting	wrist up
wrist down	oscilation	waving
bending	wiggling	circular movement of the finger tips
opening	closing	no movement
repetition	short movement	vehement movement
the ending handshape		

Table 2.4: Types of movements

The last four are not a movement themselves but they are used to describe the movement more precisely. The feature of movement can be a sequence of locations of the hand in time. That way (without a higher understanding) we are able to describe the movement in detail.

Chapter 3

Sign Language Corpora

To be able to test a SLR system and compare the results with other methods one needs a corpus. Unlike in Speech Recognition there is not a unified corpus for evaluating different methods. Many institutes generate their own corpora. This approach creates several problems. The conditions in which the corpus is made are very variable. The number of signs is different, the capturing camera parameters vary from corpus to corpus and the background can be simple or complex.

3.1 Corpus of Signed Czech *UWB-06-SLR-A*

This corpus was made at the University of West Bohemia [8] for the purpose of training and testing systems of SLR. The intention was to create laboratory conditions so that the image parametrization methods can be simple and fast. Then the researcher can focus on finding the best set of features and setting up the recognizer. The conditions were a black background, dark non-skin-colored clothing with long sleeves, constant illumination. The 15 non-native signers are performing 25 selected sings form Signed Czech Language, each sign is repeated at least 5 times. There are altogether three cameras. Two cameras capture the whole body from different points of view and the third camera is focused on the face of the signer (see Figure 3.1).

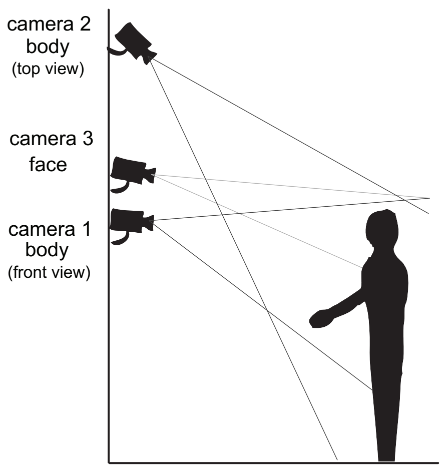

Figure 3.1: The configuration used for recording the corpora UWB-06-SLR-A and UWB-07-SLR-P

3.2 Corpus of Signed Czech *UWB-07-SLR-P*

This corpus is an extension of the corpus UWB-06-SLR-A (Section 3.1). The conditions are altered to be more realistic but they are still laboratory-like. The signers wear more colorful clothing sometimes with short sleeves. Two of the four signers are native speakers, the others are a lecturer and a student of sign language. The intention of the corpus was to record signs from the domain of train connections information [9].

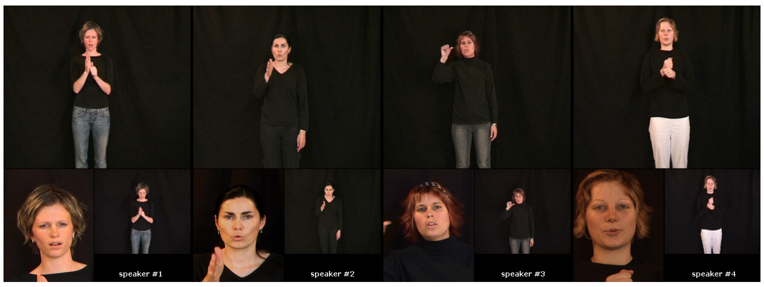

Figure 3.2: An example from corpus UWB-07-SLR-P

3.3 Corpus of Signed Czech *UWB-12-SILADON*

This corpus contains signs for educational purposes available at `znaky.zcu.cz`. An off-line version does not exist for public use and the data are used solely for this thesis. The creation of the corpus was supported by Ministry of Education, Youth and Sports of Czech Republic in the project number CZ.1.07/2.2.00/07.0189. The corpus contains one performance of each signs. The signs are selected from the subjects taught on University of West Bohemia in Pilsen. They are performed by an expert who is a lecturer of sign language.

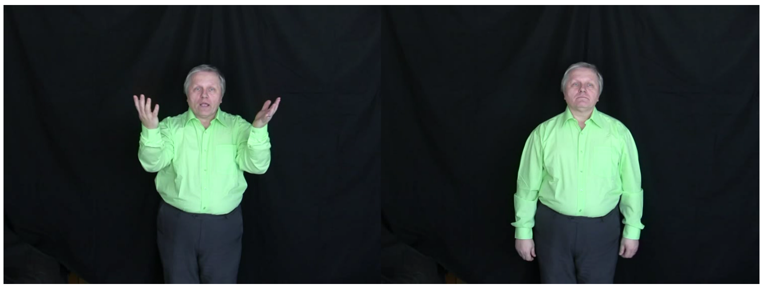

Figure 3.3: An example from corpus UWB-12-SILADON

3.4 Corpus of American Sign Language RWTH-Boston-104

This corpus made at Boston University (The National Center for Sign Language and Gesture Resources of the Boston University) contains selected sentences from American Sign Language. The primary purpose of this corpus is a linguistic research. Because of this it is not easy to use for automatic SLR. Three of the four cameras are gray-scale in resolution 312 x 242 px. There were similar corpora made in Boston. RWTH-Boston-50 (50 isolated signs, three signers), RWTH-Boston-447 (890 sentences composed of 447 signs, performed by 5 signers), and RWTH-Boston-400 (843 sentences, 406 words, four signers, divided into training, testing, and development).

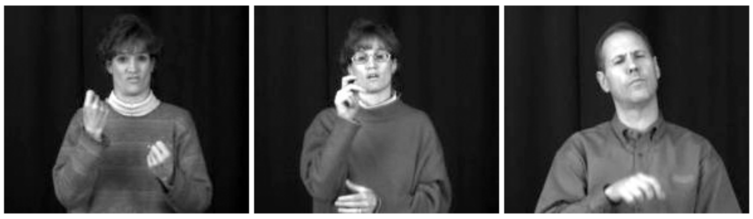

Figure 3.4: An example from corpus RWTH-BOSTON-104

3.5 Corpus of German Sign Language RWTH-Phoenix

This corpus of German Sign Language contains 1353 sentences from 11 signers from a weather forecast broadcasts. The utterances are annotated. The background is cluttered since an animation of the weather is projected there.

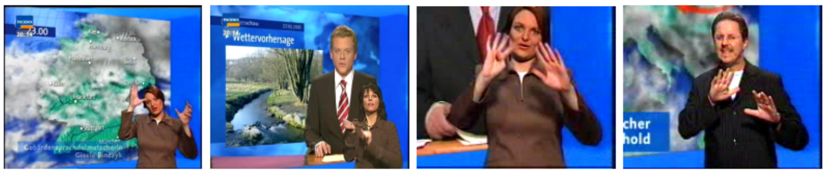

Figure 3.5: An example from corpus RWTH-Phoenix

3.6 Corpus of Sign Languages ECHO

ECHO (European Cultural Heritage Online) is a European project containing corpora of sign languages from several countries. The content is recordings of dialogs, fairy tales an so on. The corpus is being extended with new data. Up to now the corpus contains three major corpora. Sign Language of the Netherlands, British Sign Language, and Swedish Sign Language.

Figure 3.6: An example from corpus ECHO

3.7 Corpus of Chinese Sign Language

This corpus containing 5119 signs from Chinese Sign Language was developed as a part of research on SLR [10]. The corpus was created using data gloves. The obtained data can be used directly in a recognizer.

3.8 Corpus NGT

NGT (Nederlandse Gebarentaal) is a recently created corpus [11] of Netherlands Sign Language. It contains 100 native signers of different ages from all regions in Netherlands. Part of the data is annotated. The main idea is to create a large corpus available on-line.

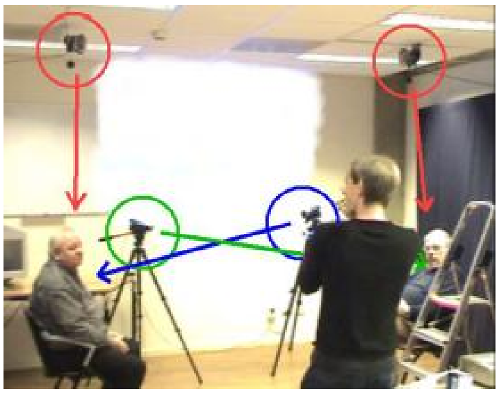
Figure 3.7: An example from corpus NGT [11]

3.9 Signs of Ireland Corpus

The corpus contains 40 signers aged between 18 and 65 from 5 locations across the Republic of Ireland. Continuous sign language was performed in a form of story telling.

Figure 3.8: An example from corpus Signs of Ireland [12]

3.10 Corpus IMDI

ISLE Meta Data Initiative (IMDI) is a standard for describing multimedia and multi modal language resources. This standard is used in Max Planck Institute for Psycholinguistics, where several different corpora are stored. For now it contains the corpus ECHO, NGT, VIDI Sign Space Project and more.

Chapter 4

Image Parametrization

In this chapter I will describe basic principles of image parametrization. These include shape representation and description, image segmentation using skin-color and texture description. A special section is dedicated to Active Shape model. This is due to special properties of the method that combines shape representation with statistical approach. Next, visual tracking is described in general and two methods are mentioned in detail. It is the color adaptive mean-shift algorithm and condensation algorithm. The first one uses color information to track the objects. The other tracks the outline of the object in the parametric space of a B-spline.

4.1 Shape Representation and Description

Shape is referred to as an external two-dimensional outline, appearance or configuration of something. From this definition we can see that shape is independent with respect to the color, matter or substance of which it is composed. It is also independent to translation, rotation and scale. That means when we observe two ellipses with different sizes and positions, we still refer to both as an ellipse (from the view of shape). After applying some other transformations (other than the mentioned above) to an object it is questionable whether its shape changed or not. Imagine a rectangle. After applying a skew transformation we observe a parallelogram. The question is: What is the shape of both objects? Are they different shapes or are they the same shape observed from different points of view? In vision based systems we have to have a higher understanding of the solved problem to overcome paradoxes such as this.

In this chapter I will present methods of shape description. Some of them have been tested in earlier versions of the tracking system and some of them are considered for future use. Although not all have not been used in the final system they are kept in this thesis as a survey.

4.1.1 Chain Codes

Chain code is a contour-based shape representation. It describes the boundary of an object as a sequence of directions of unit lines connecting the points of the contour. It can be seen as a simple tracing method when we connect the neighboring pixels with a unit vector. The neighboring pixels are determined by the choice of connectivity. Either only vertical and horizontal directions are considered (4-connectivity) or also the diagonal directions (8-connectivity). To be able to reconstruct the shape the starting point of the chain code must be specified. Chain code is also referred to as **Freeman's code** [13]. When using chain codes for recognition of or matching several shapes the description needs to be independent on rotation, choice of the starting point and scale. Rotation independence can be achieved by dividing the code modulo 4 or modulo 8, depending on the connectivity used (i.e. we consider only the change in the direction). To make the description independent on the choice of the starting point we have to find such a starting boundary pixel for which the resulting chain code is the minimal/maximal integer number. The scale independence is hard to achieve. We can represent the shape as a grammar and for an input chain code we resolve whether it can be produced by the grammar. Also, we can analyze the chain code in frequency domain to overcome the scale dependence by normalizing the magnitudes of the spectrum. The chain code is very sensitive to noise and therefore it is difficult to use in practical applications.

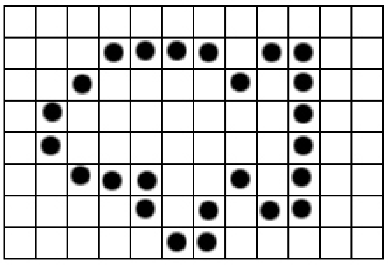

Figure 4.1: An example image of boundary pixels

For example when considering the 8-connectivity the Freeman's code of the shape in figure 4.1 would be: 1, 0, 0, 0, 7, 1, 0, 6, 6, 6, 6, 6, 4, 3, 5, 6, 4, 3, 2, 4, 4, 3, 2, 1 in the clock-wise direction with the starting point $[3, 3]$. The starting point independent code would be: 0, 0, 0, 7, 1, 0, 6, 6, 6, 6, 6, 4, 3, 5, 6, 4, 3, 2, 4, 4, 3, 2, 1, 1 with the new starting pixel $[4, 2]$. The rotation independent code would be: 7, 0, 0, 7, 2, 7, 6, 0, 0, 0, 0, 6, 7, 6, 7, 6, 7, 7, 2, 0, 7, 7, 7, 0.

4.1.2 Moments

In some cases an image (or an object in an image) can be handled as a probability density function of a 2D random variable. The properties of this random variable can be described as statistical characteristics called **moments**. A moment of order $(p+q)$ is given by

$$m_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x^p y^q f(x,y)\,dx\,dy \tag{4.1}$$

or in digitalized image of the size $N \times M$ as

$$m_{pq} = \sum_{x=1}^{N} \sum_{y=1}^{M} x^p y^q f(x,y) \tag{4.2}$$

where x, y are the coordinates in the image and $f(x,y)$ is the gray-scale value. However these moments are dependent on translation, scaling, rotation and on gray-level transformations. Usually we try to achieve an independent description for the object recognition. Using a modification in the definition of the moments we can achieve the independence. Translation invariant moments also called the **central moments** are defined as

$$\mu_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x - x_c)^p (y - y_c)^q f(x,y) dx dy \tag{4.3}$$

or in digitalized images

$$\mu_{pq} = \sum_{x=1}^{N} \sum_{y=1}^{M} (x - x_c)^p (y - y_c)^q f(x,y) \tag{4.4}$$

where x_x, y_c are the centroids (center of gravity) of the image and in the sense of moments are defined as

$$x_c = \frac{m_{10}}{m_{00}} \quad y_c = \frac{m_{01}}{m_{00}} \tag{4.5}$$

The scale normalized central moments are

$$\vartheta_{pq} = \frac{\mu_{pq}}{(\mu_{00})^\gamma} \tag{4.6}$$

where γ is a normalization factor dependent on the moment order.

$$\gamma = \frac{p+q}{2} + 1 \tag{4.7}$$

Moments as a shape description have been used for object recognition. The advantage of the image moments is that they are a statistical description. Thus, the varying appearance of similar objects results in similar moments. Of course the moments need to be independent on scale, rotation, translation and even convolution. First experiments with moment invariants were done by Hu [14] and are discussed in [15] and [16]. Hu proposed seven moments (also known as **Hu moments**) invariant to scale, rotation and translation. The seventh moment is skew invariant. As discussed in [17] a weakness of Hu's theory is that it does not provide for a possibility of any generalization. Even thought this might be seen as a flaw it does not make the Hu's moments unsuitable for feature representation. Flusser introduces a general method for a systematic derivation of affine moment invariants of any order. Furthermore he presents functionals invariant to convolution with arbitrary centrosymmetric point-spread function.

Another authors [18],[19] used the **Zerinke moments** [20] to construct rotation invariants. Zernike moments are orthogonal on a unit circle, thus they do not contain any redundant information and are suitable for image reconstruction.

4.1.3 Fourier Descriptors

Fourier descriptors are another powerful tool for shape description and are suitable features for the task of shape recognition. Fourier descriptors are generally referred to as the coefficients of the Fourier transform of a closed curve. There is a number of possibilities how to choose the shape representation that will be transformed to the fourier spectrum. For example the curve can be represented as a complex function $z(t) = x(t) + iy(t)$. Then we can write,

$$z(t) = \sum_n T_n e^{int} \tag{4.8}$$

and the coefficients T_n are the **fourier descriptors**. They can be computed as,

$$T_n = \frac{1}{L} \int_0^L z(s) e^{-i(2\pi/L)ns} ds \tag{4.9}$$

where L is the curve length and s is the curve distance.

$$s = \frac{Lt}{2\pi} \tag{4.10}$$

Another representation of the closed curve can be a chain code or a function of angle tangents versus the distance between the boundary points from which the angles were determined.

Generally the descriptors are dependent on translation and rotation. A normalization either in the time (spatial) domain or the frequency domain is needed to achieve the independence. The normalization in the time domain depends on the type of description chosen. When a complex function of two variables is used for description of the shape the coordinates of the boundary points need to centralized. That is we built a coordinate system with the origin in the mean of the function. Thus a translation normalization is achieved. To achieve the rotation invariance we consider the distance to the mean of the function instead of the boundary points' coordinates. For the chain codes normalization see Section 4.1.1. In the frequency domain the translation invariance can be achieved by setting the zeroth element T_0 to zero. The scale invariance can be achieved by dividing the fourier series with the first element ($\tilde{T}_{0..n} = \frac{T_{0..n}}{T_1}$). The information about rotation is hidden in the phase of the spectrum. By ignoring the phase we ignore the changes in rotation and thus we achieve the rotation independence. In other words $\tilde{T}_{0..n} = |T_{0..n}|$.

4.1.4 B-splines

B-splines are piecewise polynomial curves whose shape is closely related to their control polygon. A B-spline of order n has $n-1$ derivations that are continuous. For this and other reasons B-splines of third order, called cubic B-splines, are commonly used. The B-splines do not oscillate between the sampling points and they are always positioned inside a convex $n+1$ polygon for a B-spline of the n^{th} order. Another useful property of B-splines is that they are an interpolation local in character. That means when a control polygon vertex changes its position the spline changes only in the neighborhood of that vertex. Lastly, the methods of matching boundaries represented by splines to image data are based on a direct search of original image data.

Let $x_i, i = 1, ..., n$ be points of a B-spline curve $x(s)$, where $s \in R$. The curve can be represented as

$$x(s) = \sum_{i=0}^{n+1} v_i B_i(s) \tag{4.11}$$

where v_i are coefficients of the spline curve (they represent the vertices of the control polygon) and B_i are the base spline functions. The shape of the base functions is given by the order of the B-spline. For n points x_i there must be $n+2$ points v_i. The two ending points (sometimes called phantom knots) are specified by binding conditions. If we want the B-spline to have a zero curvature at the beginning and the end, then $v_0 = 2v_1 - v_2$ and $v_{n+1} = 2v_n - v_{n-1}$. For a closed curve $v_0 = v_n$ and $v_{n+1} = v_1$.

The base functions are non-negative and of local importance only. Usually the base function of a cubic spline $B_i(s)$ is defined as non-zero only for $s \in (i-2, i+2)$. That means for any i only four base functions $B_i(s)$ are non-zero. If the distance between the points x_i is constant then all base functions are of the same form and consist of four parts $C_j(t)$.

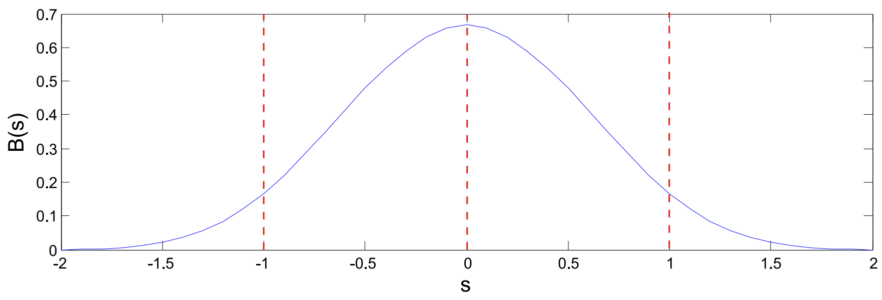

Figure 4.2: Base function of order 3. It is divided into four parts $C_0 \dots C_3$. In this case $i = 0$.

$$
\begin{aligned}
C_0(t) &= \frac{t^3}{6} \\
C_1(t) &= \frac{-3t^3 + 3t^2 + 3t + 1}{6} \\
C_2(t) &= \frac{3t^3 - 6t^2 + 4}{6} \\
C_3(t) &= \frac{-t^3 + 3t^2 - 3t + 1}{6}
\end{aligned} \tag{4.12}
$$

The equation (4.11) with respect to (4.12) becomes

$$x(s) = C_3(s-i)v_{i-1} + C_2(s-i)v_i + C_1(s-i)v_{i+1} + C_0(s-i)v_{i+2} \tag{4.13}$$

where i is the beginning of the interval $s \in [i, i+1)$ in which we compute the B-spline.

Splines are used as curve approximation and are suitable for image analysis curve representation problems. Paglieroni and Jain [21] presented a technique transforming curve samples to B-spline control polygon vertices together with a method of efficient computation of boundary curvature, shape moments, and projections from control polygon vertices. Another authors used B-splines for object matching [22], object recognition [23] and object identification [24].

4.1.5 Scalar description of shape

Scalar description of a shape is a simple numeral expression that describes some property of the shape. Alone these descriptors are not very powerful in terms of object recognition or reconstruction. But when they are used in a more sophisticated framework or when combined they can be useful. Such descriptors are for example: the perimeter of the object, area of the object, compactness of the object, curvature of the object, eccentricity of the object, elongatedness of the object and so on. Generally the can be divided into contour based and region based descriptors. In the system described in this thesis I use the following descriptors:

Let $\mathcal{O}$ be our object of interest. It can be interpreted as a set of connected pixels $\mathcal{O} = \{p_i\}_{i=1}^{N}$, where p_i is a pixel in location (x_i, y_i). The connectivity can be described in a following way; two pixels p and q are connected with respect to an object when there exist a set of pixels that form a path between p and q so that all the pixels in the path belong to the object. The contour of the object is a set of pixels that have at least one neighboring pixel that is not part of the object. Depending on the type of connectivity we choose for the neighboring pixels we obtain either 4- or 8-connectivity contours. If we choose 4-connectivity in the condition of finding neighboring pixels we obtain an 8-connectivity contour and vice versa.

Perimeter of the contour. The value of this descriptor is the number of pixels that form the contour of the object.

Area of the region. The value of this descriptor is the number of pixels that form the object. In this case it is N.

Area of the bounding box. The value of this descriptor is the product of the sizes of the sides of the bounding box of the object. If the bounding box is defined as (x_1, y_1, x_2, y_2) where (x_1, y_1) is the upper left corner and (x_2, y_2) is the lower right corner we can write: $x_1 = \min_i p_i^x, y_1 = \min_i p_i^y, x_2 = \max_i p_i^x, y_2 = \max_i p_i^y$.

Direction. The value of this descriptor is the angle between the longer side of a bounding rectangle and the image x-axis. The bounding rectangle is a rectangle that covers the whole object and has a minimum area of all such rectangles. In literature it is mentioned that direction is valid only for elongated objects. In this thesis I use it for every object since I am interested in the orientation of objects which represent hands.

4.2 Active Shape Model

Active Shape Model (ASM) introduced by Tim Cootes [25] is a statistical shape description method suitable for describing known 'general' shape whose instances can undergo variations in appearance. Due to the variation these shapes cannot be efficiently described by rigid models. This approach has many applications in different fields. For example in medicine it is used to describe the shape of bones or organs. The idea is to introduce several examples of the modeled shape to the system in the form of landmarks. Using this information a statistical model involving the shape variance is created.

4.2.1 Finding the Landmarks

One must first find suitable landmarks that describe the shape of the object. Such landmarks can be corners of objects, T junctions or other easily located features. To improve the shape description other landmarks can be chosen laying on the edge of the object between already selected landmarks. Next the connectivity of the landmarks must be recorded in order to know how the landmarks are connected to create the desirable shape. Then we have a set of n landmarks denoted as $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$. This set can be represented as a vector. For n landmarks the dimension of the vector will be $2n$.

$$\boldsymbol{x} = (x_1, \ldots, x_n, y_1, \ldots, y_n) \tag{4.14}$$

For each example of the modeled shape we can build such a vector. These training vectors have to be aligned. Generally, the shape is invariant to translation, rotation, scale and even some other transforms. For each training vector we have to find such a transformation that aligns all vectors to a referential one. Information on how to align the shape can be found in [26].

4.2.2 Statistical Models of Shape

Suppose we have s vectors $\boldsymbol{x}$ of dimension $2n$ that are aligned into a common coordinate frame. These vectors form a distribution in the $2n$-D space. The idea is to find a statistical description of this distribution that will allow us to create new shapes similar to the modeled one. First, Principal Component Analysis (PCA) is applied to the cloud of the s vectors. PCA reduces the dimensionality of data by finding its principal components. When changing the reduced amount of points we can generate new plausible shapes.

$$\boldsymbol{x} = \bar{\boldsymbol{x}} + \boldsymbol{P}\boldsymbol{b_s} \tag{4.15}$$

where $\boldsymbol{x}$ is a newly generated shape (dimension $2n$), $\bar{\boldsymbol{x}}$ is the mean of the original points, $\boldsymbol{P}$ is formed column wise by the eigenvectors of the covariance matrix of the original points, and $\boldsymbol{b_s}$ is a vector in the reduced dimension defined by

$$\boldsymbol{b_s} = \boldsymbol{P}^T(\boldsymbol{x} - \bar{\boldsymbol{x}}) \tag{4.16}$$

This vector is a set of parameters that can be changed in order to generate new plausible shapes. Lets refer to it as the shape vector.

4.2.3 Fitting the Model

Each vector $\boldsymbol{b_s}$ generates a different shape. The goal is to find such a shape vector and transformation that the modeled points fit the best to the current image data. In the coordinate system of the image this means to find the position (X_t, Y_t), the orientation θ, and the scale s of the modeled data for a given $\boldsymbol{b_s}$. On how to do that for a known set of image points refer to [26]. However usually the set of points we want to fit the model to is not known to us. In this case we have to create a cost function that tells us how good the model fits to the image. If the model is composed of landmarks that lie on edges a good idea is to find strong edges in the image. Then the cost function will be a sum of distances from the generated landmarks and the strong edges. Another approach is to train the model with texture information. For each landmark point we search along a line perpendicular to the edge of the object in that point. We store the gray scale differences and for more example shapes we create a statistical description of these differences. The cost function is then the difference between the trained values and the current values in the image. When we have the cost function available we solve an optimization problem to find the cost function's extrema. Since the parameters of the cost function are the shape parameters, we set them to the located extreme. After few iteration steps we should converge to a best fit (global extreme of the cost function). For further details refer to [26].

The ASM was later modified to also model the texture of the object. The texture does not describe the shape of the object but it can be used to detect the shape better. The new model is called Active Appearance Model described (AAM) in [27]. The extension of the model is straightforward. The texture, usually in the form of a gray scale image, is also modeled by using PCA. First the shape is warped to the mean shape using a triangulation algorithm. Then the gray level samples from this normalized image are obtained and used as examples of the texture. A PCA is performed on these samples and we obtain a liner model of the texture. Since the texture should be dependent on the shape the authors use another PCA to combine the shape and texture into one model. We can then write:

$$\boldsymbol{g} = \bar{\boldsymbol{g}} + \boldsymbol{P_g}\boldsymbol{b_g} \tag{4.17}$$

is the linear model of the gray scale values in the mean shape. The symbols correspond to the shape model 4.15 but instead of modeling landmarks the gray scale values are modeled. The combined model is

$$\boldsymbol{b} = \begin{pmatrix} \boldsymbol{W_s b_s} \\ \boldsymbol{b_g} \end{pmatrix} = \begin{pmatrix} \boldsymbol{W_s P_s^T}(\boldsymbol{x} - \bar{\boldsymbol{x}}) \\ \boldsymbol{P_g^T}(\boldsymbol{g} - \bar{\boldsymbol{g}}) \end{pmatrix} \tag{4.18}$$

where W_s is a diagonal weighting matrix that takes into account the different scale of shape and texture features. The fitting is performed as a search along the AAM parameters so that the difference in between the observed pixels and the generated ones is minimal. The search is done by training a matrix that transforms the error vector (difference of pixels values) to a vector of displacements which tells us which parameters to use in the next step. It is an iterative gradient descent method.

4.3 Shape Recognition

There are several well established approaches and methods used for recognition [28]. The basic assumptions for the task of recognition are a **set of features** representing the recognized object (or event) and the **knowledge** which enables us to identify the object as a member of a given class. In some cases we do not need the a priori information about the classes used in the classification process. Instead the classifier indicates which objects are similar. The approaches to recognition can be divided into two groups. **Rule-based** approach and **statistical** approach. The rule-based system requires a set of rules designed by an expert. The rules are then used on the input features and are combined to obtain the resulting class which the object belongs to. On the other hand, the classification process can also be seen as the testing of hypotheses. It leads to the result from the other direction. We assume the recognized object belongs to a given class and we combine the rules to verify the hypothesis. The statistical approach is based on the probability description of the feature space. The description is derived from a training set of class representatives. In the testing process, an unknown feature vector is presented to each class and its likelihood of belonging to the class is evaluated. The object is then identified as a member of the class for which the likelihood was maximal. In the task of SLR the shape recognition can be used to associate visual objects with shape classes. This information can help us to distinguish between different objects like hands, head or background. Also it provides us with information about the shape of the object (e.g. handshape).

Nowadays there exist several well established classification methods. From simple k-nearest neighbors, minimal distance classification, or naive Bayes classifier to more sophisticated neural networks, support vector machines, decision trees, decision forests, hidden Markov models, boosted classifiers and so on.

4.4 Image segmentation and Texture description

Shape is an important property of image objects. But it is not the only one. Sometimes we want to analyze the texture of the object. For example if we observe a hand in basic hand shape with extended fingers the shape is the same when the hand is observed from front and from back. The difference is in the texture of the object. Recent and widely used texture descriptors include: Local Binary Pattern (LBP) and Histogram of Oriented Gradients (HOG). In this work I present a research where we experimented with LBP. That is why I describe this method in detail later. Another problem is the detection of the object of interest in the image. This is the issue of image segmentation. We need to find such features that distinguish the objects of interest from the rest of the image - the background. The result of the segmentation is a binary image which is true for the pixels that are believed to be a part of the object and zero for the background. In HCI and other applications where humans are detected a popular way of segmentation is skin color segmentation.

4.4.1 Skin color segmentation

A good survey of skin color segmentation is given in [29] and [30]. The problems of skin color detection that need to be overcome include: illumination, camera characteristics, ethnicity, individual characteristics, and more. The first choice to make in skin color segmentation is the color space in which we want the skin color to be represented.

Basic color spaces (RGB, normalized RGB, CIE-XYZ). The advantage is that the RGB color space is the default representation of colors on a computer. That is why we can save computational time by neglecting the color space transformation. By using normalized RGB we make the model invariant on illumination intensity. Also we save one dimension in the representation since the values of normalized RGB sum up to one. The CIE (Commission Internationale de l'Eclairage) system describes color as a luminance component Y, and two additional components X and Z. CIE–XYZ values were constructed from psychophysical experiments and correspond to the color matching characteristics of human visual system.

Perceptual color spaces (HSI, HSV, HSL, TSL). These models are perceptual which means they try to describe the perception of colors by humans. The models break down the color into components easily perceived by a human. For example the HSV model has components Hue, Saturation, and Value. Hue describes the basic color from red to green, saturation is self explanatory (pink to red), and value is the intensity of the color (how black/gray/white is the color).

Orthogonal color spaces ($\mathbf{YC_bC_r}$, YIQ, YUV, YES). These color spaces represent the original RGB colors in space formed by independent components. In $\mathrm{YC_bC_r}$ it is the luminance (Y) component and chrominance components $(\mathrm{C_b}, \mathrm{C_r})$. These color spaces are very popular in skin color segmentation and many modifications have been proposed.

Perceptually uniform color spaces (CIE-Lab, CIE-Luv). These spaces model the colors as observed by a human so that they are perceptually uniform. This means that small perturbations to a component value is approximately equally perceptible across the range of that value. This is achieved by a non-linear transformation of the XYZ color space. Generally the components can by divided into luminance (L) and chroma (a, b, u, v).

The next step is the skin-color classification. It can be viewed as a two class classification problem. We observe a color of a pixel and we want to make a decision whether it belongs to skin segment or not. There are many approaches to this problem and I will mention the most popular ones briefly.

Explicit skin-color space thresholding. The straight-forward way to segment skin is to define ranges in components of the color space which delimit the skin-color cluster. This approach is suitable for controlled environments where the characteristics of skin-color are constant and known.

Histogram model with naive Bayes classifiers. This method uses a 2D or 3D histogram of the skin-colors in the chosen representation. The color space is divided into several bins. Then a probability distribution is approximated from the histogram as:

$$P(c) = \frac{count(c)}{T}, \tag{4.19}$$

where $P(c)$ is the probability distribution of colors c, $count(c)$ is the count in the histogram bin for this color and T is the total number of colors that were used for construction of the histogram. Next we can define

$$P(c|skin) = \frac{s(c)}{T_s}, \qquad P(c|non\text{-}skin) = \frac{n(c)}{T_n}, \tag{4.20}$$

where $s(c)$ is the pixel count in the bin representing the color c of the skin-color histogram, $n(c)$ is the pixel count in the bin representing the color c of the non-skin-color histogram. T_s and T_n are the total counts of pixels in the skin-color and non-skin-color histograms respectively. Using a maximum likelihood approach a simple skin-color classifier can be build. The color c is classified as skin-color if

$$\frac{P(c|skin)}{P(c|non\text{-}skin)} \geq \theta, \tag{4.21}$$

where θ is a threshold which controls the trade-off between true positives and false positives. These method is very fast when represented via n-dimensional look-up-table.

Gaussian classifier. This approach assumes that the skin-color distribution is Gaussian. Fewer data is needed to train the model and it requires less memory to represent the model. The single Gaussian models represent the distribution of skin-colors as

$$p(c) = \frac{1}{(2\pi)^{k/2}|\Sigma|^{1/2}} exp\left(-\frac{1}{2}(c-\mu)^T \Sigma^{-1} (c-\mu)\right), \tag{4.22}$$

where μ and Σ are the mean value and covariance of the training skin-colors, k is the dimension of the color representation. For each color we obtain the likelihood of it being skin-color. A threshold can be used to classify the color. Alternatively, Mahalanobis distance with threshold can be used to classify the color. If we want to achieve a more detailed description of the distribution of skin-colors we can use GMM. The training of this model is performed by using EM algorithm.

Elliptical boundary model. This model is an alternative to the computationally demanding GMM. The model is defined as

$$\Phi(c) = (c - \Psi)^T \Lambda^{-1} (c - \Psi), \tag{4.23}$$

where c is the evaluated color vector, Ψ and Λ are the model parameters defined as

$$\Psi = \frac{1}{N}\sum_{i=1}^{N} c_i, \qquad \Lambda = \frac{1}{N}\sum_{i=1}^{N} f_i (c_i - \mu)(c_i - \mu)^T, \tag{4.24}$$

where N is the total number of skin-color samples in the training set, f_i is the number of samples with the color c_i and μ is the mean of the color vectors in the training data set. The color c is classified as skin-color if $\Psi < \theta$, where θ is a threshold chosen empirically as a trade-off between the true positives and the false positives.

Other classification methods include **Multilayer perceptron, Self organizing map, Maximum entropy, or Bayesian network**. I refer the reader to the survey [29] for details.

4.4.2 Local Binary Pattern

The Local Binary Pattern (LBP), introduced by Ojala [31], serves for texture representation. The LBP is used across various computer vision fields (e.g. image synthesis, light normalization, face detection, face/expression recognition). The LBP is computed for each pixel of the image. The analyzed pixel called the center pixel is compared to the pixels in its neighborhood. If the gray-scale value of the pixel in the neighborhood is greater than the gray-scale value of the center pixel then the position of the neighboring pixel is flagged as one. Otherwise it is flagged as zero. In the original paper the neighborhood was the imminent 8-neighborhood of the center pixel. That means that 8 positions around the center pixel are evaluated which yields 8 binary values. These can be represented as decimal value in the range 0 - 255. The 8 values are handled as bits of the binary representations. Each position in the neighborhood defines a position in the binary representation. In the original paper the bits were composed from upper-left corner continuing clockwise. In my experiments I used the ordering depicted in Figure 4.3.

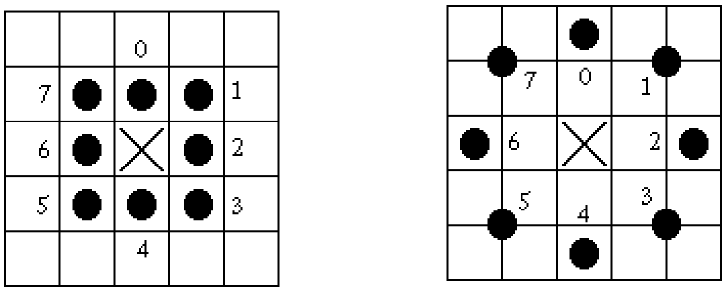

Figure 4.3: Examples of LBPs with radius one (left) and two (right). Numerical values represent the position of the patch in the binary representation of the pattern.

The feature vector is a histogram of the decimal representation of LBPs from the analyzed image or image patch. In the basic form the histogram has 256 bins. Later, modifications of the LBPs were proposed. In [32] the authors present rotation invariant patterns and uniform patterns. The uniform patterns are a subset of all possible 256 patterns. The subset is characterized by the property of the LBP that there are only two or less changes from 0 to 1 in the pattern. This subset has 59 elements. The 59^{th} bin of the histogram created from the uniform LBP represents the patterns that are not uniform. The neighborhood of the center pixel can be also modified. In Fig 4.3 on the right there is an example of LBP with radius two.

4.5 Image Understanding Control Strategies

Understanding is an abstract concept. It cannot be defined well. It can be defined as the power of abstract thought or an individual's perception or judgment of a situation. In computer vision the image understanding is the highest processing level and it's main task is to define control strategies that ensure an appropriate sequence of processing steps. In other words it is an artificial intelligence capable of understanding what is seen. In the task of SLR we need to understand where is a signing person in the image, which object is his head, his right or left hand/arm and so on.

There are two basic approaches to image understanding. One is controlled by the image data and the other by a higher-level knowledge. These different approaches can be described as [33]:

1. **Control by the image data (bottom-up process)**: Processing proceeds from the raster image to segmented image, to region (object) description, and to their recognition. In case of SLR, the processing could be: segmenting the image to skin-color regions, describing the objects using a suitable feature vector and recognizing the features (handshape, position, orientation, ...).
2. **Model-based control (top-down process)**: A set of assumptions and expected properties is constructed from applicable knowledge. The satisfaction of those properties is tested in image representations at different processing levels in a top-down direction, down to the original image data. The image understanding is an internal model verification, and the model is either accepted or rejected. ASM is an example of such an approach.

The two basic control strategies do not differ in the types of operation applied, but differ in the sequence of their application. A combination of the mentioned approaches can be used to solve more complex problems.

4.6 Tracking

In the terms of system theory the tracking process can be seen as a task of detecting a value of a system's attribute(s) changing in time. The attribute can be either observed directly or can be hidden (unobservable). In this work the attention is payed to **visual (video) trackers**. Visual tracking is the process of locating a moving object (or several ones) in time using a camera. An algorithm (tracker) analyses the video frames and outputs the location of moving targets within the video frame. The tracking process is an ill-posed problem. The movement happens in 3D space, but the observation is a 2D projection. When an occlusion is present the tracking becomes difficult as the tracked object changes its characteristics rapidly. If the movements are fast relative to the camera frame rate the detection is much harder.

There are two major components of a visual tracking system: **Target Representation and Localization** and **Filtering and Data Association**.

Target Representation and Localization is mostly a bottom-up process. Typically the computational complexity for these algorithms is low. There are some methods that are not directly used for representation of the target but are helpful when detecting the movement (e.g. optical flow) or segmenting the image into moving objects and background. The following are some common Target Representation and Localization algorithms:

- Blob tracking: Segmentation of object interior (for example blob detection, block-based correlation or optical flow)
- Kernel-based tracking (Mean-shift tracking): An iterative localization procedure based on the maximization of a similarity measure (Bhattacharyya coefficient).
- Contour tracking: Detection of object boundary (e.g. active contours or Condensation algorithm).
- Visual feature matching: Registration.

Filtering and Data Association is mostly a top-down process, which involves incorporating prior information about the scene or object, dealing with object dynamics (for example a motion model), and evaluation of different hypotheses. The computational complexity for these algorithms is usually much higher. The following are some common Filtering and Data Association algorithms:

- Kalman filter: An optimal recursive Bayesian filter for linear functions and Gaussian noise.
- Particle filter: Useful for sampling the underlying state-space distribution of non-linear and non-Gaussian processes.

4.6.1 Optical Flow

If a camera (or human eye) moves in a 3D scene, the resulting apparent motion in the sequence of images is called the **optical flow**. The optical flow describes the direction and the speed of motion of the features (or patterns) in the image. It is a very helpful tool for motion analysis or visual tracking and is often used as the first step of the processing. In a practical case we need a higher-level processing that can solve motion related problems. For example we do not need to compute the optical flow for unimportant objects such as shadows or background. Also, the optical flow should not be scattered along the image, but should create patterns that correspond to the movement of individual objects.

Optical Flow Computation

Optical flow computation is based on two assumptions:

1. The observed brightness of any object point is constant over time.
2. Neighboring pixels in the image plain move in a similar manner.

Let $f(x, y, t)$ be a dynamic image function. It refers to the gray-level of a point (x, y) at time t. Now we want to observe the changes $(\delta x, \delta y)$ in consecutive frames meaning that $t = t + \delta t$. For this reason we express the dynamic image function as a Taylor series.

$$f(x + \delta x, y + \delta y, t + \delta t) = f(x, y, t) + \frac{\partial f}{\partial x}\delta x + \frac{\partial f}{\partial y}\delta y + \frac{\partial f}{\partial t}\delta t + O(\partial^2) \tag{4.25}$$

For simplicity we denote the partial derivates of the image function as f_x, f_y and f_t. If the changes in the variables are small the higher order terms vanish and also we can approximate

$$f(x + \delta x, y + \delta y, t + \delta t) = f(x, y, t) \tag{4.26}$$

and thus the equation (4.25) becomes

$$-f_t = f_x \frac{\delta x}{\delta t} + f_y \frac{\delta y}{\delta t} \tag{4.27}$$

The goal is to compute the velocity $c = (\frac{\delta x}{\delta t}, \frac{\delta y}{\delta t}) = (u, v)$ in every point of the image function. The partial derivates of the image function can be directly approximated from the image itself. The spatial derivates f_x, f_y refer to changes in the brightness pattern, high values mean corners. The time derivate f_t describes the change of brightness in time.

To overcome the motion related problems some additional criterion or constraint has to be applied to the task. For example we do not need to compute the optical flow on the whole image but only on edges or other significant points in the image. This method has been applied for the first time by Hildreth [34]. Other methods are based on the criterion that the optical flow should be continuous throughout the image or in a local neighborhood [35]. Correlation of parts of one image with parts of the next image can be used to find the optical flow. The idea is to find the most correlated parts in the images and assume that the motion happened in this region. This approach is the most computationally expensive one. It has been used by Burt et. al. in [36] where the correlation was computed on multiple scales.

4.6.2 Continuously Adaptive Mean Shift Tracking

The **continuously adaptive mean shift (CAMSHIFT)** algorithm is based on the **mean shift** algorithm, which is a non-parametric probability distribution estimation. In other words, it is a non-parametric feature space analysis technique. It was introduced by

Fukunaga and Hostetler [37] in 1975. It is able to estimate a mode of probability distributions. In the case of tracking it has to be modified to adapt dynamically to the changing probability distribution.

This modification is referred to as the CAMSHIFT algorithm [38]. It is able to estimate the center, the size and the orientation of an object using the knowledge about the object's color. It operates on the color data represented as a probability distribution. To achieve this the color is described via histogram. First, the initial image is converted to a suitable color space. Usually the HSV color space is used. Using a pre-computed color histogram of the object a backprojection is calculated. This gives us the probability distribution we use in the next steps. Using the mean shift algorithm and a given search window the center of the object is determined. A new search window is positioned on this center. Next the size of this window needs to be computed. The new search window is then used as an initialization for the mean shift algorithm in the next step. The orientation of the object is computed from the second moments of the color probability distribution.

Algorithm Details

In this section we work with two dimensional data.

1. Choose the initial location of the search window.

   In the case that we track only one object the search window can be set to the whole image. But still the color of the object needs to differ from the background. If it does not then we need to make sure that at the initialization step the tracked object is the only object visible. If we track more objects we need to know their approximate locations.

2. Mean Shift initialized with the search window.

   When the mean shift converges we need to store the zeroth moment of the distribution (the area) as it is used for computing the size of the search window in the next step. The zeroth moment is computed as,

$$M_{00} = \sum_x \sum_y I(x, y) \tag{4.28}$$

   where $I(x, y)$ is the probability of the color at position x, y.

3. Compute the orientation and size of the object.

   The orientation and size of the object can be computed from the second moments of the probability distribution.

$$M_{20} = \sum_x \sum_y x^2 I(x, y) \quad M_{02} = \sum_x \sum_y y^2 I(x, y) \tag{4.29}$$

   Then using the substitutions

$$a = \frac{M_{20}}{M_{00}} - x_c^2, \quad b = \frac{M_{02}}{M_{00}} - y_c^2, \quad c = 2\frac{M_{11}}{M_{00}} - x_c y_c \tag{4.30}$$

where x_c, y_c is the center of the object (blob), the orientation can be computed as

$$\theta = \frac{1}{2} arctan \frac{c}{a-b} \tag{4.31}$$

and the size can be computed as

$$l = \sqrt{\frac{(a+b) + \sqrt{c^2 + (a-b)^2}}{2}} \quad w = \sqrt{\frac{(a+b) - \sqrt{c^2 + (a-b)^2}}{2}} \tag{4.32}$$

where l is the length and w is the width of the object.

4.6.3 Condensation Algorithm

Condensation stands for **Con**ditional **Dens**ity Propag**ation** and it is used for visual tracking. It was introduced by Michael Isard and Andrew Blake [39]. Its main goal is a robust real-time tracking of object's outline in a dense visual clutter. The approach is rooted in ideas from statistics, control theory and computer vision. In order to apply the algorithm, specific probability densities must be established for dynamics of the object and for the observation process. The dynamics of the object refer to the changing appearance of the object and the observation process refers to the probability of the appearance given an image. From this information we are able to determine the likelihood of an appearance of an object in consecutive frames.

State Space Model

In Condensation algorithm the shape is modeled as a B-spline 4.1.4. Let us denote the curve as $r(s,t)$.

$$\boldsymbol{r}(s,t) = (\boldsymbol{B}(s)\boldsymbol{Q}^x(t),\ \boldsymbol{B}(s)\boldsymbol{Q}^y(t)) \tag{4.33}$$

for $0 \le s \le L$, $\boldsymbol{B}(s)$ is a vector of B-spline basis functions, $\boldsymbol{Q}^x$ and $\boldsymbol{Q}^y$ are vectors of B-spline control points (coefficients, vertices, knots) divided into x and y components. L is the number of spans. Next, the configuration of the spline is described as a shape-space of vectors $\boldsymbol{X}$ defined by

$$\begin{pmatrix} \boldsymbol{Q^x} \\ \boldsymbol{Q^y} \end{pmatrix} = \boldsymbol{W}\boldsymbol{X} + \begin{pmatrix} \bar{\boldsymbol{Q}}^x \\ \bar{\boldsymbol{Q}}^y \end{pmatrix} \tag{4.34}$$

where $\boldsymbol{W}$ is a matrix of rank N_X considerably lower then the $2N_B$ degrees of freedom of the unconstrained spline. Thus, this description of the spline is restricted to some plausible instances of the curve. $\bar{\boldsymbol{Q}}$ is a template shape from which the other shapes are generated. The space is constructed by applying an appropriate combination of three methods to build a $\boldsymbol{W}$-matrix:

1. determining analytically combinations of contours derived from one or more views [40], [41], [42]
2. capturing sequences of key frames of the object in different poses [43]
3. performing principal component analysis on a set of outlines of the deforming object [44], [45]

Dynamical Model

In condensation algorithm the dynamical model is a second order discrete process represented by a linear difference equation

$$\boldsymbol{x}_t - \bar{\boldsymbol{x}} = \boldsymbol{A}(\boldsymbol{x}_{t-1} - \bar{\boldsymbol{x}}) + \boldsymbol{B}\boldsymbol{w}_t \tag{4.35}$$

where $\boldsymbol{w}_t$ are independent vectors of independent standard normal variables. The state vector $\boldsymbol{x}_t$ is represented as

$$\boldsymbol{x}_t = \begin{pmatrix} \boldsymbol{X}_{t-1} \\ \boldsymbol{X}_t \end{pmatrix} \tag{4.36}$$

$\bar{\boldsymbol{x}}$ is the mean value of the state and $\boldsymbol{A}, \boldsymbol{B}$ are matrices of deterministic and stochastic components of the dynamical model, respectively. The parameters of the process equation can be set either manually by an expert, or better, estimated from the input data while the object preforms typical motions. Methods for doing this via Maximum Likelihood Estimation are essential to the condensation algorithm and can be found in [43] and [46]. The dynamical model can be re-expressed in such a way as to make clear that it is a temporal Markov chain:

$$p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) \propto exp(-\frac{1}{2}\left\|\boldsymbol{B}^{-1}((\boldsymbol{x}_t - \bar{\boldsymbol{x}}) - \boldsymbol{A}(\boldsymbol{x}_{t-1} - \bar{x}))\right\|^2) \tag{4.37}$$

Observation Model

The observation process defined by $p(\boldsymbol{z}_t|\boldsymbol{x}_t)$ is assumed to be stationary. Thus we obtain a static function $p(\boldsymbol{z}|\boldsymbol{x})$ that needs to be specified. The authors in [39] first derive the observation in one-dimension and then expand it to two dimensional observations. In one dimension when certain assumptions are made the observation model leads to

$$p(\boldsymbol{z}|x) \propto 1 + \frac{1}{\sqrt{2\pi}\sigma\alpha}\sum_m exp\left(-\frac{v_m^2}{2\sigma^2}\right) \tag{4.38}$$

where $\alpha = q\lambda$ (λ is the spatial density of a Poisson process describing the clutter, q is the probability with which the object is not visible) and $v_m = z_m - x$ (z_m is the $m-th$ observation). A two dimensional representation is then

$$p(\boldsymbol{z}|\boldsymbol{x}) = Zexp(-\frac{1}{2r}\int_L^0 f(\boldsymbol{z}_1(s) - \boldsymbol{r}(s); \mu)ds) \tag{4.39}$$

where Z is a constant, $\mu = \sqrt{2}\sigma log(1/\sqrt{2\pi}\alpha\sigma)$ is a spatial scale constant, $\boldsymbol{r}$ is a variance constant and $\boldsymbol{z}_1(s)$ is the closest associated feature to $\boldsymbol{r}(s)$:

$$\boldsymbol{z}_1(s) = \boldsymbol{z}(s'), \text{ where } s' = argmin\,|\boldsymbol{r}(s) - \boldsymbol{z}(s')|.$$

Propagation

Given a continuous Markov chain with independent observations, the conditional state density p_t at time t is defined by

$$p_t(\boldsymbol{x}_t) \equiv p(\boldsymbol{x}_t|Z_t) \tag{4.40}$$

where Z_t is the observation history up to time t. The rule for propagation of state density over time is

$$p(\boldsymbol{x}_t|Z_t) = k_t p(z_t|x_t)p(x_t|Z_{t-1}) \tag{4.41}$$

where

$$p(\boldsymbol{x}_t|Z_{t-1}) = \int p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})p(\boldsymbol{x}_{t-1}|Z_{t-1})d\boldsymbol{x}_{t-1} \tag{4.42}$$

and k_t is a normalization constant. Given a suitable state space model, dynamics model and observation model (which can be generally non-Gaussian) we are able to propagate the effective prior $p(x_t|Z_{t-1})$ from the posterior $p(\boldsymbol{x}_{t-1}|Z_{t-1})$.

Chapter 5

State-of-the-Art

In this chapter I describe state-of-the-art methods dealing with the manual component of SL. As mentioned in Chapter 2.1 the manual component of SL is composed of the trajectory of hands, orientation of hands and shape of the hands.

5.1 Manual Component Analysis for Sign Language Recognition

In [47] Ding and Martinez state that models of American Sign Language (ASL) require that handshapes and motion patterns are described independently. From a research on ASL they derive that only a set of linguistically significant fingers is necessary for the full understanding of each word. Being able to recover the 3D shape of these fingers from 2D video sequences is a necessary milestone to develop efficient HCI systems. Similarly, to properly represent ASL motions, one needs to describe the 3D trajectory of the dominant hand with respect to time. As observed by the researchers many signs in ASL start with one handshape and end with the same handshape. Therefore they are using an affine structure from motion algorithm to estimate the 3D shape of the fingers. For this purpose they use linear fitting of low rank matrices with missing data. The fingers are represented as connected knuckles (joints).

Next, they present the motion estimation problem as the three point perspective pose problem. They robustified the algorithm to be insensitive to spatial noise. The authors report the percentage of error of depth estimation of the knuckles to be less than 5%. Avarage error of shape reconstruction is below 0.1%. It is important to mention that the authors tracked the joints manualy and then synthetically added noise to the data and removed some data in order to simulate occlusions. They do not present a method for extracting the features from the image.

In [48] Fillbrandt, Akyol, and Kraiss define the requirements of a SLR system. They focus

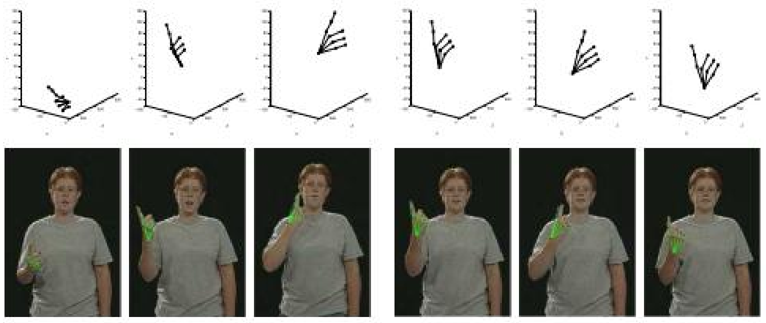

Figure 5.1: Reconstruction of the 3D handshape [47].

on handshape parametrization. In their opinion such a system should fulfill the following.

- Since the hands can move very quickly in SL, there are huge differences between subsequent frames. Therefore, the method has to cope with coarse initialization in each frame.
- The image sequence shows at least the upper part of the body (Chapter 2), meaning the hands are observed in low resolution. Additionally, motion blur impairs the image quality. Thus, the method should work stably even when little image information is available.
- Because occlusion of the hands does appear in SL, the method should provide the possibility to deal with partly missing image information.
- The method should have real-time capabilities.

To handle all of these problems Active Appearance Model (AAM) is adopted. To train a single appearance model one handshape with a small variation in pose and finger constellation is used. As model parameters they chose equidistantly distributed points on the silhouette of the hand. These points are then semi-automatically matched to the hand to create a more precise 3D mesh.

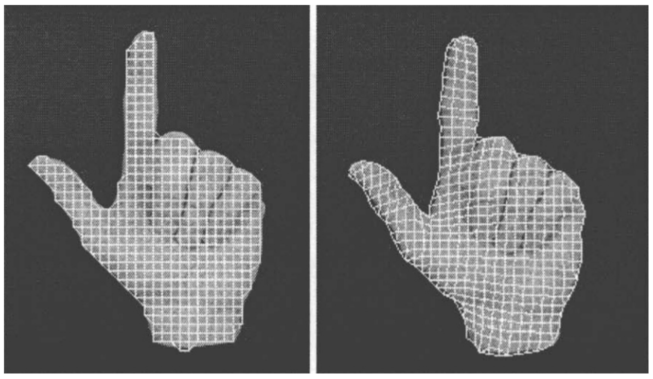

Figure 5.2: Creation and adaptation of the model graph [48].

The model fitting is realized as linear regression - a learned relation between the error vector and the shape parameters. The error vector represents the difference between the image data and the modeled data. The shape parameters are divided into two groups. Geometrical parameters describing position and orientation and shape parameters describing the mesh and gray-scale values on it. The model for hand had to be modified, because AAMs were originally developed for faces that have distinguishable landmarks inside the model. Therefore, the authors add additional layer across the border of the hand to the model. This helps them distinguish between hand shapes. At last a transition network between the hand shapes is built. This fastens the search of the next model, as the transition network delimits the choice of the model only to possible models (see Figure 5.3).

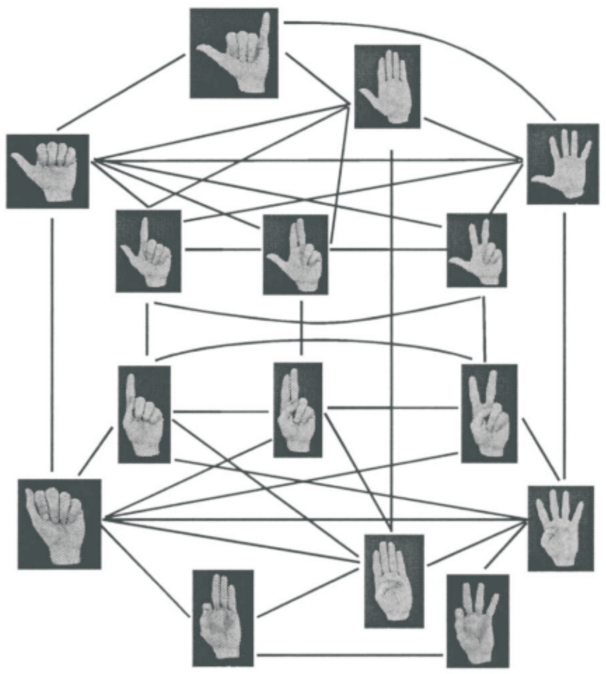
Figure 5.3: Transition network of the front view [48].

This approach is able to recognize which fingers are extended, whether they are spread, and the geometrical pose of the hand. Authors report a 90% recognition success, which should be enought for the purpose of SLR. The method cannot handle occlusions yet and does not run at high frame rates.

Human hand has 15 joints and one joint has one to four degrees of freedom (DOF). Thus, a hand is parametrized by a high dimensional vector. In [49] Kato, Chen, and Xu represent the hand motion in reduced PCA/ICA space. In this reduced space they track the hand motion using a particle filter (Section 4.6.3), which cannot handle high dimensional searches efficiently. They choose silhouette and edge information as the observation model. They show that the ICA reduction improves performance compared to just PCA. They are able to track the hand just with first five ICA parameters corresponding to finger motion since the joints on one finger move more dependently then joints on different fingers.
Similar assumptions are made in [50]. They use only PCA to reduce the dimensionality. In the reduced space they build transformation subspaces. The subspaces are grouped and organized into a hierarchical decision tree. The method is supposed to be invariant to transformations. It is tested on images that are rotated, translated and scaled. The recognition rate is from 85% to 100% depending on the rate of deformation.

In [51] Rosales, Athitsos, Sigal, and Sclaroff recover 3D hand pose from monocular color sequences. The system employs a non-linear supervised learning framework, the specialized mappings architecture (SMA), to map image features to likely 3D hand poses. First, they acquire a set of training data using a data glove. The output of the data glove are the angles of the fingers' joints. Together there are 22 degrees of freedom for the hand configuration plus two DOFs for the pose. These 24 parameters are then mapped to the image features. The authors chose to use image moments (Section 4.1.2, particularly Hu moments) to describe the visual appearance of the hand. The mapping is non-linear and many to many. Using the SMA they split the mapping to many mappings which are then combined. The SMA should also provide an inverse mapping from the image features to the hand configuration model. To segment the hand the authors use skin color model. Tracking of the hands is provided by a second-order Markov process. The Markov process models the evolution of color histogram of each hand. Currently this approach cannot handle occlusions. The experimental results show that mean square error of the hand reconstruction is between 1% and 3% depending on the angle of view.

In [52] Athitsos and Sclaroff estimate the 3D pose of a hand from one cluttered image. They formulate the hand pose estimation as an image database indexing problem, where the closest matches for an input hand image are retrieved from a large database of synthetic hand images. Estimating the pose from one image can be used to initialize a hand tracker. They model the hand as an articulated object, consisting of 16 links: the palm and 15 links corresponding to finger parts (Figure 5.4).

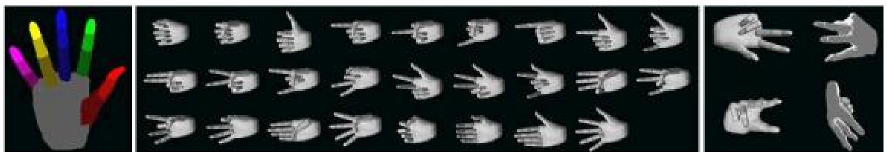

Figure 5.4: Synthetic images of hands [52].

First the dataset is ranked using a modified Chamfer Distance. The modification makes the computation of the distance less expensive. After the first ranking a second ranking is applied. Lines in the dataset images are found. A method (probabilistic line matching) to compute the similarity between the lines from the dataset images and the lines from the testing image is proposed. According to this measure the ranked dataset images are re-ranked. The efficiency of the algorithm is measured as the rank of the dataset image that is equal to the ground truth pose. The right pose was ranked as first in 13.6% of training images. In other cases 84% of the correct poses were ranked between 1 and 256.
Similar method of estimating the pose of the hand is shown in [53]. The main difference is that the authors in this work use multiple cameras. The results are then combined using a maximum a posteriori (MAP) framework. Hand contour extraction is provided by a skin color model in RG space and active contour algorithm. The main output of this work is a conclusion that two cameras improve the recognition rate dramatically (more than two times).

In [54] Chen, Fujiki, Arita, and Taniguchi also use multiple cameras. From all the views, two views with the most information are selected. The authors show that such views are those that contain the largest hand region. The proposed algorithm relyes on a robust feature extraction from the image. The features are the center of wrist, center of the palm, and finger feature referring to arc points projected outermost on the hand contour.

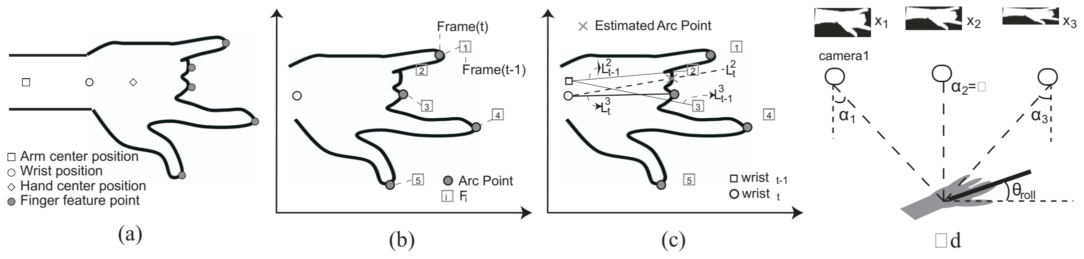

Figure 5.5: (a) Shape feature points (b) Correspondence between arcs and fingers (c) Estimation of undetected arc points (d) Camera layout [54]

Then the global pose parameters of the hand are estimated. It is the yaw and pitch rotation for the wrist and translation and roll rotation of the elbow. Next the hand configuration constraints [55] are applied and combined with inverse kinematics to estimate the pose of the fingers. The problem of the approach is that the hand features are supposed to be visible. The approach should work with cluttered background but without skin colored regions. The authors do not report the recognition rate only the real-time capabilities.

In [56] El-Sawah, Joslin, Georganas, and Petriu present a framework for tracking dynamic gestures. The constraints and conditions for their system are: generic non-restricted environment, generic not-specific application, and marker gesture recognition. Although the goal is a markerless tracking they use a dark glove with markers to detect the fingertips and 2 cm square markers to derive the palm's position and orientation with respect to the camera. The palm marker is distinguished by color. They track the corners of the marker to estimate position and orientation. The markers on fingertips are also segmented using a color model. Hand silhouette is derived by calculating the image gradient in the direction perpendicular to the hand model links. The 3D model of the hand consists of linked joints with 26-DOF. The model provides forward and inverse kinematics transformation utilities. The detected 2D marker features are used to generate hand posture hypothesis. The posture hypothesis is verified using an observation model. The hand posture hypothesis is applied on the hand model using forward kinematics, and the resulting 3D model is projected on the image plane. From this projection several measurements are made which resolve into the probability of the pose. Dynamic gestures are recognized using Dynamic Bayesian Network models consisting of an initial and terminal posture probability vectors and a sequence of posture transition stochastic matrices. The recognition rate was evaluated on a set of six gestures. The system showed recognition improvement from 25% using a single posture per sample to around 70% recognition using 40 postures per sample. The framework could be generalized to be used without markers. In that case a robust feature extraction system in unconstrained conditions is needed.

In [57] Tanibata, Shimada, and Shirai propose a method for extracting features from sequences of images in complex background. First, the person region is extracted using a background image captured by a stationary camera. They initialize the positions of the face, hands and elbows by matching the initial pose template to the person's region. From the detected face and hands a skin-color model is build. From the other colors a model of clothes is trained. The models are realized in HSV color space. From the parameters of the models a model of middle colors is build (Figure 5.6). These are the colors on the edges between the skin-color segments and the clothes. This way a more precise hand region is obtained (Figure 5.8).

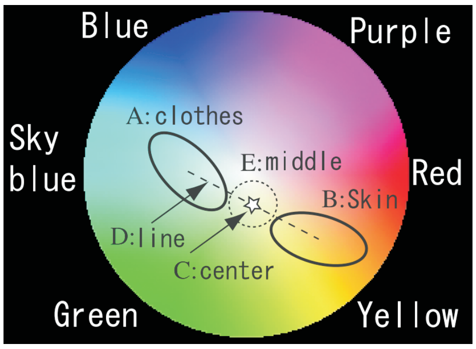

Figure 5.6: The ranges of colors in H-S space [57].

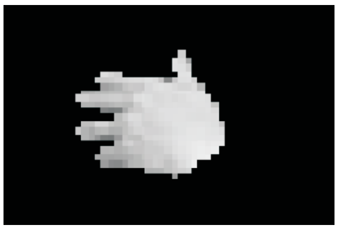

Figure 5.7: Hand region only with the range of skin color [57].

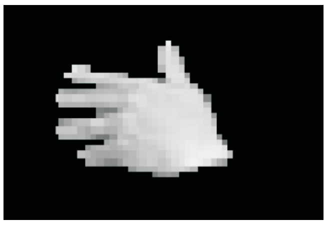

Figure 5.8: With the range of skin and middle color.

When the head and hands are initialized they are tracked in the next frames. The position of the hands is predicted using the position of the hand in the last frame, its speed and acceleration. From the predicted position the nearest skin-color region is selected as the object in the next frame. Finally, the textures of head and hands are registered to be used in the case of overlap. The occlusion is detected using the information of the predicted position and the area of the skin region. If the region becomes more than a half larger an occlusion is detected. In this case the position and orientation of the head is estimated using the registered template. Then using the template of the hand on the non-face region the position of the hand is estimated. The same principle is applied when two hands are overlapped. The elbow is tracked as an arc. The parameters of the arc are estimated from the person's region. The extracted features are then:

- r : The flatness of hand region
- (x_{hand}, y_{hand}) : The gravity center position of the hand region relative to that of face region
- A : The area of the hand region
- θ_{motion} : The direction of the hand motion in the image coordinate
- θ_{hand} : The direction of the hand region in the image coordinate
- N_P : The number of protrusions

The signs are then modeled as HMMs. The tests were done on a small database consisting of 65 sings. 64 of them were recognized correctly.

In [58] Liu and Lovell extract hand gestures by ASMs. First, they extract the hand contour in real-time. Using a look-up-table in a color-space composed of Hue component from HSV color model and UV components from YUV color model they segment the image into hand region and background. Then they track the hand with CAMshift algorithm (see section 4.6.2) and Kalman filter. From the retrieved contours they train an ASM. The landmarks are chosen semi-automatically. First, the points on the hand contour with high curvature are obtained. These points refer to fingertips and junctions of the fingers. A median filter is used to obtain a precise number of landmarks. Other landmarks are added manually on the curve connecting the already detected landmarks. Following the idea of ASM (Chapter 4.2) the landmarks are aligned into a common coordinate system, PCA is performed on the data which results into a vector of shape parameters. The error function that is minimized to estimate these parameters when a new instance of a hand is presented is the distance from a landmark to the nearest strong edge perpendicular to the model edge. The system has real-time capabilities, but the authors do not present any success rate in finding the hand.

In [59] Hamada, Shimada, and Shirai present a method of hand posture estimation from silhouette images taken by two cameras. The conditions are set so that the hand extraction problem is minimal (dark background, dark clothes). The contour is described by 256 values. The values are the distances of the contour points to the center of gravity of the hand region. They are further normalized by dividing with the hand region area and realigned so that the most significant peak or valley is the beginning. Next, a database of the handshapes is built. Similar handshapes are analyzed using PCA and k eigenvectors are chosen to describe the handshape. When evaluating a hand contour it is described with the normalized features, then they are transformed to the eigenspace and then euclidean distances between the sampled hand and the models are computed. The hand is recognized as the handshape for which the distance was minimal. Because the matching is often ambiguous, they use a pair of stereo images. To determine which view is more informative they compute the complexity of the shapes. The complexity is defined as:

$$c = \sum_{i=1}^{256} \frac{r_{i+k} - r_i}{k} \tag{5.1}$$

where r are the normalized features representing the hand contour. More complex shapes are supposed to be more informative. They are weighted with bigger weights. When the dimensionality of the eigenspace was set to 12 the recognition rate was 95.9%.
When recognizing gestures, the changes in handshape are limited. The authors propose a method for automatically creating a transition network of handshapes. They cluster the eigenspace using the distance function and a pre-set threshold. When recognizing the transition network is used to find the next handshape candidates. This way the method is much faster (only 11% of all handshapes are considered in one step).

In [60] Schreer and Ngongang present a real-time solution for gesture recognition. To extract the hands from the image, skin-color model in YUV color space is adopted. Using a region growing approach on a sub-sampled image they achieve real-time capabilities with good results. As features describing the handshape the authors chose the distance of a contour point to the opposite side of the contour, whereas the direction is defined by the normal of the tangent along the contour (Figure 5.9). This yields into a signature of the hand or

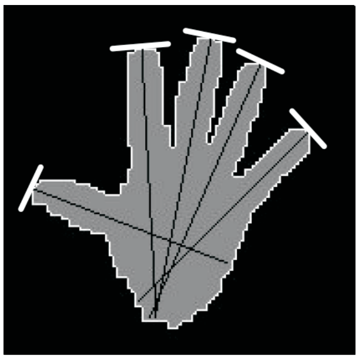

Figure 5.9: Figure of the features proposed by [60]

as authors propose a distance function. Next, they find the fingertips in the signature, the distance between the fingertips and the length of the fingers. The handshapes are then analyzed over a set of consecutive frames. When a handshape is recognized on a certain number of frames it is considered as truth.

In [61] Holden and Owens recognize moving hand gestures using methods from speech recognition. The hand region is detected using a skin-color model. The RGB color space is transformed as $(R, G, B) = ((R + B + G)/3, (R - B), (2G - R - B)/2)$ and then analyzed using PCA. In order to eliminate the noisy skin-colored patterns the condensation algorithm (section 4.6.3) is employed to track the largest blob. The observation model is represented as a gaussian

$$\pi_l^n = \frac{1}{\sqrt{(2\pi)\Phi}} e^{v^2/2\Phi^2} \tag{5.2}$$

where v is a pattern matching result that is the number of non-skin pixels within a M x M neighborhood, $\Phi = M^2/10$. Next a set of features is extracted from the hand image. The image is converted to polar image and normalized. The outer and inner contour is retrieved and the finger region is detected on the contour. The finger-only contours are analyzed to produce an eight dimensional feature set representing the hand shape using cepstral coefficients. These are the Linear Predictive Coding (LPC) parameters. The handshape is classified using Mahalanobis distance between the unknown shape and a model shape. The reported recognition rate is 96% on a set of 14 videos.

In [62] the authors employ so called pictorial structures to estimate the pose of the body and arms. The chosen pictorial structure consists of scalable rectangles connected with joints as seen in Figure 5.10. The proposed method tries to find the best parameters of the model given an image. Each part (rectangle) of the model is specified by scale s and angle α. The probability of a given hypothesis of limb configuration L is given as

$$p(L|I) \propto p(L) \prod_{i=1}^{N} p(c_i|\lambda_i) \prod_{j \in \{LL, LR\}} p(h_j|l_j) \tag{5.3}$$

The formulation incorporates two appearance terms modeling the agreement between the

]

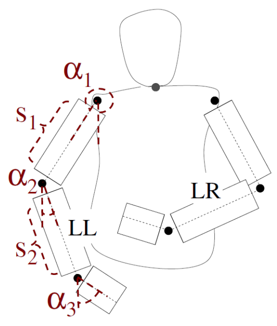

Figure 5.10: Pictorial structure modeling upper body [62]

image I and the configuration L. The term $p(c_i|\lambda_i)$ models the likelihood of observed pixel colors. The term $p\left(h_j|l_j\right)$ models the likelihood of observed gradients based on Histogram of Oriented Gradient (HOG). The term $p(L)$ models the prior probability of the configuration thus enforcing the kinematic chain. The model is relatively high dimensional and with the chosen discretization there are approximately 10^{13} possible configurations of the limbs. The authors present a computationally efficient model fitting. They use the approach published in [63] which makes use of the tree-like structures of the pictorial models and modify it so that it is in compliance with their occlusion model. After tracking videos of British weather forecast the authors are trying to learn the signs automatically using text cues which are displayed in the videos alongside the signing. After the learning phase they achieve 65% success rate of finding the correct words. Next, they present a framework for sign recognition from the learned signs on unseen signers. They use SVM classification and achieve 67% recognition rate on 15 signs. Since this is signer-independent recognition the results are quite good. The tracking is robust and relatively fast.

5.2 Hand tracking

The systems of SLR are complicated. The state-of-the-art is extensive and there is a lot of partial issues that need to be handled. I would like to mention a part of SLR which is very close to my thesis. That is hand detection, tracking and hand pose estimation. Hand detection is used for static images where candidate patches are evaluated and a decision about the patch is made. Such detectors work very similarly to face detectors. Hand tracking in 3D and hand pose estimation are very closely related. Either you can find a mapping between the observed data and pose parameters. Or you can generate 3D hand models given the pose and verify how well this model fits the observed data. Although this thesis does not use all of these approaches which are originally not used for SLR, I find it fitting to mention them in this chapter.

In [64] the authors present a method for representing the model of articulated hand by a quadratic. This framework allows them to generate 2D contours of the hand from 3D model. The 3D model is composed from geometric primitives, namely truncated cylinder for the palm, a cone for the phalanx of the finger connected by hemispheres (see Figure 5.11). The authors show how to compute the contours of the 3D model efficiently even when seen

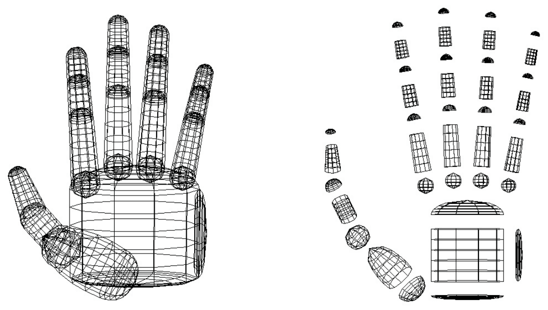

Figure 5.11: The 27 DOF hand model composed from the quadrics [64].

from different viewpoints. This allows to use this approach in a framework of more cameras. For the tracking the authors use an Unscented Kalman Filter (UKF) to estimate 7 DOFs of the 27 DOF model. The observation consists of edge points in the image that are the closest to the modeled edges generated by the 3D model which is updated according to the observation and UKF equations. The algorithm needs to be manually initialized and the authors demonstrate the functionality of the algorithm on a set of images, but without giving any error measurements.

In [65] the authors use hierarchical Bayesian Filter to estimate the hand shape in an observed image. First, they present the tree-based filtering for evaluating the Bayesian inference equations and compare the approach to other types of filtering. The most notable approaches are grid filtering [66] with modifications on how to efficiently compute the estimations. It is for example modeling each mode of the distribution by a separate adapting grid while creating and deleting the local grids on-line. Another approach is to use fixed grid, but avoid the evaluation at grid points where the probability mass is below a threshold. Another type of filtering mentioned in the work is Particle Filtering. The idea of tree filtering lies in the partitioning of feature space into pyramidal structure. At each level the centroid of the partition is evaluated. If it is probable enough then in the next level the centroids of the child partitions are evaluated. This way only probable partitions are evaluated. The leaf nodes represent the finest partitioning of the feature space. For better understanding refer to Figure 5.12.

The second part of the work is describing what features were used and how the probability of them is evaluated. It is edge and color information. The chamfer distance function is used to evaluate the distance between a template image (generated by the features in given partition of the feature space) and an observed image. The color information is obtained by a GMM of the skin color distribution generated at the position of the template. The template can be a pre-generated 2D image or a 3D model of hand. Also models of dynamics are presented. The authors train the probability between the last leaf state and the next states in every level of the pyramid. The experiments are conducted on three short sequences of moving hand. The authors report approximately error of 7 pixels between the ground truth and the tracked position of fingertips. The image size was 320x240 pixels.

There are also less complex solutions for hand tracking. In [67] the authors present a real-time tracking of multiple skin-colored objects. The method relies on a robust on-

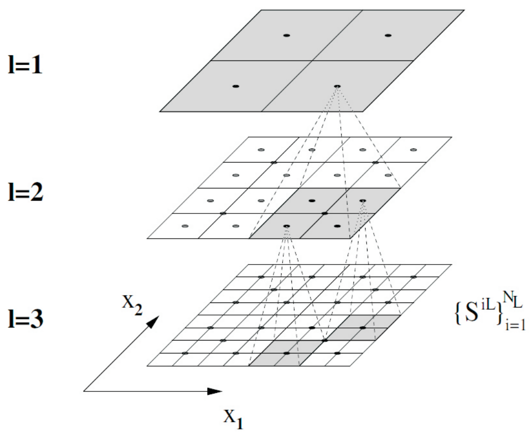

Figure 5.12: Hierarchical partitioning of the state space [65].

line skincolor segmentation. First an off-line non-parametric skincolor model is trained in YUV color space using only U and V channels. When an image is segmented a hysteresis thresholding is performed. The first thresholding finds very probable skincolored segments and the second threshold is used to find less probable segments which are connected to the ”strong” segments. Also an adaptation framework is presented so that the resulting model is a combination of the off-line model and the skincolored segments detected recently. Then the tracking is performed by using very simple computation based on a distance of a skincolored pixel from an ellipse which represents the tracked object. The ellipses can be generated if a new skincolored object appears or can be removed if the object supporting the ellipse disappears. A simple first order motion model is used to predict the position of the ellipse in the next frame. Then the distance between each skincolored pixel and each ellipse is computed. This yields a label for each pixels telling us to which ellipse it belongs. After this the new ellipses are estimated from the pixels which have the same label. Surprisingly the method can cope with many motions and handles occlusions well. The authors present the results on video sequence that they prepared themselves. It seems the motions in the video reflect the first order motion model very well and that is why the tracking is perfect. In SL the motions are not in compliance with the first order model. This means that the prediction of the ellipses movement would have to be modeled more complexly. If such a model would be found the tracking could work for SL. This hypothesis will be tested in the experiments section of this theses.

Since the release of Microsoft Kinect there is a lot of research being conducted on the topic of tracking from the depth map. In [68] the authors propose a solution to the problem of recovering and tracking the 3D position, orientation and full articulation of a human hand from markerless visual observations obtained by a Kinect sensor. First, the hand is segmented as the largest skin-colored blob. All pixels within this blob with depth difference less than 25 cm are kept. This observed 3D pixels are compared to a synthetic hand model similar to the one used in [64]. The rendering of the synthetic hand yields a depth map r_d. A discrepancy between the observed skin and depth maps and the rendered skin and depth maps is computed using the Equation 5.4.

$$E(h, O) = D(O, h, C) + \lambda_k \cdot kc(h) \tag{5.4}$$

where λ_k is a normalization factor, kc is a function that asserts the kinematically plausible hand configurations, and the function D is defined as

$$D(O,h,c) = \frac{\sum \min\left(|o_d - r_d|, d_M\right)}{\sum (o_s \vee r_m) + \epsilon} + \lambda \left(1 - \frac{2\sum (o_s \wedge r_m)}{\sum (o_s \wedge r_m) + \sum (o_s \vee r_m)}\right) \tag{5.5}$$

where o_d is the observed depth, r_d is the rendered depth, d_M is the distance threshold, o_s is the observed skin color, r_m is a label defining whether the skin-colored pixel is informative (the depth difference of the pixel is less than a threshold),ϵ is a small constant to avoid division by zero, and λ is a normalization factor. When observing the function one can see that the minimal value will be achieved if the observed and rendered depth signals are the same and furthermore all pixels have maximal skin-color probability. Next step is the generation of hypotheses of the synthetic hand configuration. Particle Swarm Optimization (PSO) is used with manual initialization on a known hand pose and position. The hand model has 27 parameters. The result of optimization from one frame is used as an initial step (particle population) to the next frame. The authors present experiments on how to set the PSO parameters, and what is the effect of noise and distance from the sensor. The system operates near-real time (15Hz) and the results are satisfactory.

5.3 Conclusion

In this chapter, there have been several methods introduced that deal with the manual component of the SL. It is an ill-posed problem. One has to make a compromise between the robustness and the speed of the recognition. One of the issues is that both properties are required from a SLR system. It can be seen that there are many approaches to solve this problem and they usually take very different paths. On the other hand, there is an outline that is followed by almost every researcher (see Figure 5.13).

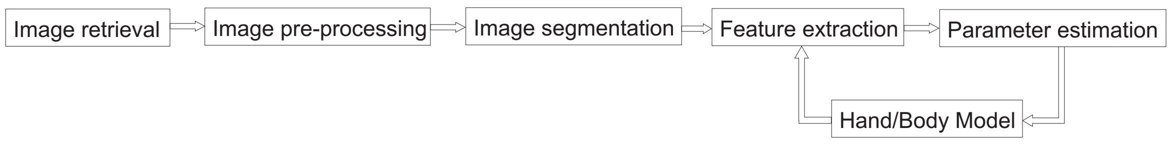

Figure 5.13: General framework of hand/body parameter estimation.

When you look at the diagram, the first block is image retrieval. It is not that significant and can be provided by standard methods. Image pre-processing is used to clean the image of noise and to emphasize important features in the image. The goal of image segmentation is to segment the image into objects of interest and background. In the case of manual component of SL the objects of interest are the hands and the head. Usually information about the skin-color is adopted or motion cues when only the objects of interest are moving. As mentioned in section 4.5 there are two major approaches to understand what can be seen in the image. One approach is to extract features from the image and from these features directly estimate the parameters of the objects (position, rotation, shape, etc.). The second approach is trying to fit a model to the image data and iteratively find the best fit. When we are estimating the parameters from visual features several problems occur. The features are extracted from the two dimensional appearance of the object, while the object itself is three dimensional. Additionally, hands have many degrees of freedom, one hand can mimic the other and more configurations of the hand can result in similar appearances. Moreover, some movements in SL are very fast and that leads to another degradation of the image data. When we are using the top-down approach the biggest problem is the high number of degrees of freedom of the model. This means that many appearances can be synthesized from a model of a hand. Combination of bottom-up and top-down approaches can be also used. With the introduction of Kinect sensor some of the problems created by monocular image acquisition disappear. This may be a new way how to approach the problem. Although 3D image acquisition existed before, Kinect is produced massively and therefore is available for broader masses.

It is hard to say which of the mentioned methods is the best. All have some advantages but also drawbacks. It is difficult to compare the results of the different approaches. The main reason for that is that every method is tested on different data. The amount of data is usually small and therefore the results are not reliable. Other question is how detailed the description of the hand configuration should be. Many researchers try to achieve a detailed description while others conclude that only several features are important. This conclusion can be misleading since there is not enough data to test the hypothesis.

Regardless of what was mentioned before, in my opinion the best method is a combination of top-down and bottom-up approach. In this case we can extract visual features that will be used to choose a set of possible hand configurations. This should reduce the time needed when fitting different models as their number is lowered. From the fitting of the model we can derive another set of features that can be used to describe the hand configuration more detailed and so on. The idea of reducing the dimensionality of the motion of the hand should be also adopted, since the fingers usually move dependently. Moreover, the model of hand should have statistical properties in order to cover more appearances from different people.

Chapter 6

Tracking System

In this chapter I present the system for tracking the hands and the head in a SL scenario. The system utilizes known principles from the fields of computer vision and machine learning and as such is one of the output of this thesis. The system is unique in the way how it handles occlusions of body parts. The occlusions are modeled explicitly and thus no special effort is needed to maintain the non-occluded body parts. This means that when two or more objects become occluded the system analyses this situation and tracks the new object as a composition of more objects. There is thus no need to approximate the unseen pixels so that the tracking can go on after the occlusion. To my knowledge such system was not used for tracking of body parts before. Because the problem of tracking the body parts in an unconstrained environment is extremely difficult some assumptions are made. These assumptions are in compliance with the practical use of the system which is the tracking the body parts in recordings of isolated signs for the use in a dictionary.

The tracking system is an extended version of the baseline system which is described in [69]. All the features were kept and enhanced to improve the tracking results and quality. Also some rules and pre-processing was added so that we can cope with more variate data.

The baseline system was a semi-automatic annotation tool capable of tracking hands and head in video streams with the option to interfere with the tracking in case it fails. The final tracking system is improved with a Tracker Manager that takes the role of a human operator and uses the low level measurements of trackers to interpret them in scenarios defined by rules. The system was implemented in C++ using object oriented programming.

6.1 System Overview

Let $\mathcal{O}_t$ be a set of detected objects in an image I in frame t.

$$\mathcal{O}_t = [o_1^t, o_2^t, \dots, o_n^t] \tag{6.1}$$

where n is the number of objects. The object is represented as an image via matrix. See

Section 6.3.

Let $\mathcal{T}$ be a set of trackers.

$$\mathcal{T} = [t_1, t_2, \ldots, t_m] \tag{6.2}$$

where m is the number of trackers. The number of trackers depends on how many objects we want to track. In our case we track three objects: hands and head.

Let $\mathcal{C}$ be a set of all possible mappings $\mathcal{T} \to \mathcal{O}$. Each mapping $C_k \in \mathcal{C}$ fully describes which tracker tracks which object. The mapping should be imagined as a listing and not as a function and as such is a result of the tracking algorithm. In other words the tracking algorithm finds the optimal map C^* according to a defined criterion. The resulting map can be defined as a set, for example $\{(t_1, o_1), (t_2, o_2), (t_3, o_1)\}$. This tells us that tracker 1 and tracker 3 track the object 1 and tracker 2 tracks the object 2. Note that two objects in occlusion form just one object. That is why it is possible for the two trackers to track the same object.

6.1.1 Flow of the System

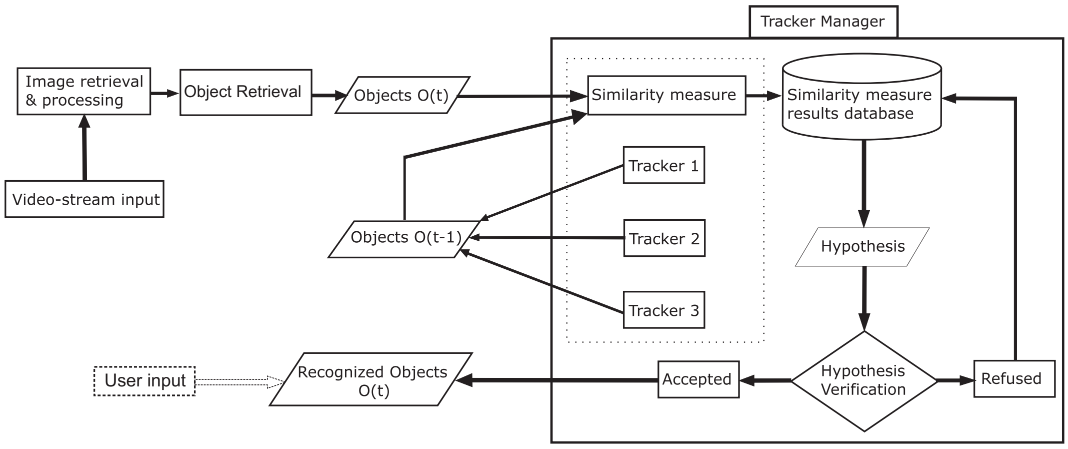

Figure 6.1: The flow of the system.

In Figure 6.1 you can see the flow of the system. In the described scenario we assume that there are three trackers. Two trackers track hands and one tracker tracks the head. At first the image is segmented and the objects are retrieved (time = t). Next, the objects are compared with the objects from the last frame (time = $t-1$) in a similarity measure. If no objects exist in the last frame the trackers are initialized according to defined rules. The results from the similarity measure are stored in a database. The database can be imagined as a three-dimensional matrix. The rows refer to the trackers (t_1, t_2, t_3), the columns refer to the objects $(o_1, o_2, \ldots, o_n)$, and the depth refers to time. See Figure 6.2.

TrackerManager now analyses the situation that occurred.

1. Based on the number of detected body parts select a proper $C_k \in \mathcal{C}$

2. Compute the log-likelihood of the selected configuration

$$L_k = \sum_{i=1}^{3} \log p_m(t_i|C_k(t_i)) \tag{6.3}$$

   This can be interpreted also in the terms of similarity measures stored in the database as

$$L_k = \sum_{i=1}^{3} S_{iC_k(t_i)} = (S_{1C_k(t_1)} + S_{2C_k(t_2)} + S_{3C_k(t_3)}) \tag{6.4}$$

3. If this is the maximum likelihood seen so far, store it as a new maximum

4. If there are no more configurations to test, select the C_k with maximum L_k as the recognized configuration, else go to point 1

After the system has chosen the final hypothesis the user can interfere with the flow and set the configuration of the objects manually. This can be useful when the models for similarity measure are not known or were trained on different data. Also a problem arises because one hand can mimic the other. They are very similar in appearance and it can be confusing for the tracker.

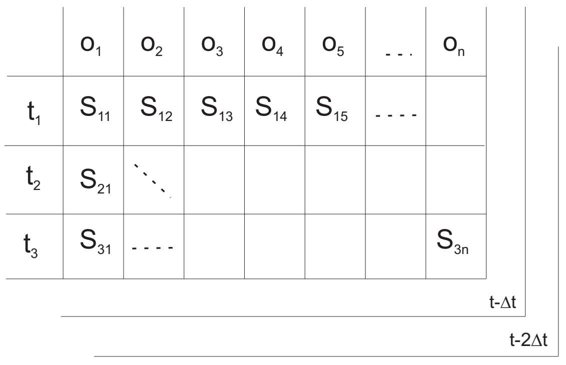

Figure 6.2: The representation of similarity measure database.

6.2 Selection of Features

There are many options for choosing the features for automatic sign language recognition. The following features were selected:

- trajectory - a set of 2D points representing the mean of the contour of an object (or center of mass) for every frame
- hand shape - seven Hu moments [14]
- hand orientation - angle between the image x-axis and a minimal bounding rectangle of the hand

From this set of features the annotation of the image data has been derived. It can be seen that all the features can be computed when the contours of objects in the scene are known. Detecting the contour manually can be very time expensive for a human but there are many methods for extracting the contour automatically. Next step is to decide which object the contour represents. It is a very easy task for a human but again the annotation can be time consuming. Imagine that for every frame a person would need to identify each object. That is why a tracker has been developed for this purpose (see Section 6.4).

6.3 Object Retrieval

The image is segmented using a skin color model [70]. If no skin color model is available or the skin color model gives poor segmentation results it is created automatically. The algorithm that rates the quality of skin color segmentation is described in Algorithm 1. To train the model we need samples of skin color in the video. A robust method for detecting faces [71] is used to obtain the first set of samples. The problem is that not all pixels in the face image are skin colored. Mean-shift algorithm is used to segment the colors into various clusters. The Luv color space was used for the clustering. Then the biggest cluster is used as a training set. The model is in the form of a look-up-table. Two approaches were used to build the final model. First the RGB values of the training pixels are used directly. This leaves us with a sparse look-up-table of RGB values. The data in the look-up-table are morphologically dilated. This will make the sparse skin color regions more dense. The second approach is to model the skin color as a Gaussian. The training data are used to compute the mean and covariance in RGB color space. The training of the Gaussian may take a while depending on the number of training data. Next, we have to transform this Gaussian to the form of a look-up-table. Since for every RGB value we can compute the probability of being a skin color we need to find a threshold of the probability so that the points in the look-up-table are true skin colors. This was achieved by knowing the approximate skin color volume ratio in the RGB color space. From the conducted experiments the ratio of 4% was derived. Note that this value is dependent on the nature of the data. In this case the 4% are meaningful for Caucasian race captured in constant conditions. The ratio would grow with adding more races and more recording conditions. I find such a threshold for which the ratio of skin colored pixels is the desired 4%. The resulting models can be seen in Figure 6.3. Note that both models are trained on the same video file.

In the figure it can be seen that the look-up-table made of the dilated samples describes the colors in the training set very precisely. No additional colors are visible. The drawback is that there could be holes in the model. This can be addressed by changing the size of the dilatation kernel. In my experiments the size of two worked good enough to segment and track the body parts. The Gaussian model is convex meaning that no holes will be present in the model. But we can observe that less colors that refer to the skin are present and some very dark colors are included in the model. This depends on the skin ratio parameter of the training method.

The model is applied to the image and the segmentation quality algorithm is used to see the effect of the training samples. If the quality is not good a second face color cluster which is the closest to the first cluster in the color space is added and so on until a defined threshold of the distance. If the segmentation is still not good then another image from the video is used and the process is repeated until a good segmentation is obtained. This

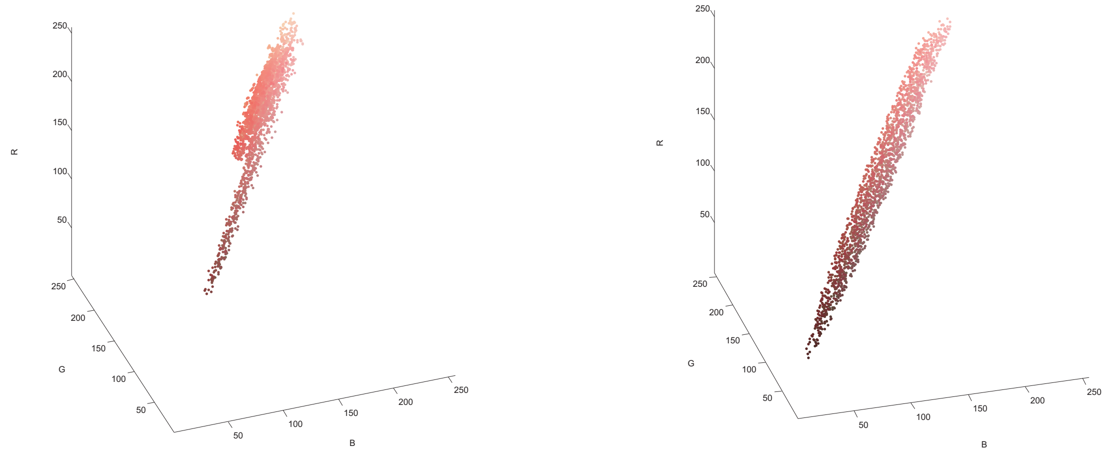

Figure 6.3: Results of skin color training. Left - dilated samples, dilatation was performed two times. Right - Gaussian of the training samples.

algorithm is described in Algorithm 2. The results of automatic skin color detection can be seen in Figure 6.4.

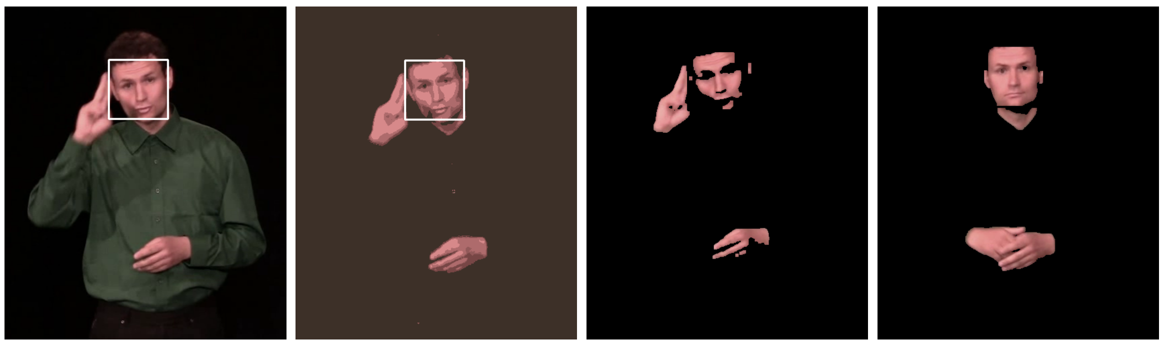

Figure 6.4: Process of automatic skin color detection. First image: the detected face, second image: mean-shift clustering result, third image: chosen face color clusters, forth image: final segmentation.

After the segmentation the background is labeled as zero and the foreground is composed of various blobs of skin-colored segments. These blobs are represented as contours using the algorithm developed in [72]. These blobs are filtered. The idea of the filtration is to eliminate improbable blobs that are not part of human body but rather clothes or background. That is why small blobs and large blobs are filtered out. The words *small* and *large* are relative and dependent on the approximate size of hands and head in the image. This way we retrieve n objects $o_1, o_2, \dots, o_n$ as candidates of hands and head. The objects are described as a set of coordinates representing the outer contour of the blob. Since the Tracker Manager takes into account the number of detected objects we need to ensure that we pass only the objects that are truly the head and hands to it. The problem occurs when there are more than three objects detected. In this case we want to merge the objects, particularly the ones that form a single object which was split due to shadows or occlusion or due to imperfect skin-color segmentation. We pick two candidate objects to merge. These candidates are the two objects that are the closest to each other. The merge is performed by drawing a line between the centroids of the objects and the entropy of the sizes of the new objects is computed.

$$H(S) = -\sum_{i=1}^{N} p(s_i) \log p(s_i) \tag{6.5}$$

where s_i is the size (area) of object o_i in pixels. The objects should have similar sizes which implies a large entropy. Then the second closest objects are selected and so on until a distance threshold (we do not want to merge far away objects). Then the merging that resulted in the maximal entropy of the sizes is selected and performed to obtain the result. We still can obtain more than three objects. In such case the process is repeated until three objects remain.

6.4 Tracking Process

The tracking is based on a similarity measure of scalar description of the objects. The objects are described by:

- seven Hu moments of the contour
- a gray scale image (template)
- position
- velocity
- perimeter of the contour
- area of the bounding box
- area of the contour.

The assumption is that these features will change in time very slowly for each object. Such assumptions are very common in visual tracking, for example when computing optical flow. Furthermore these features are easily computable and expressible as scalars. For every new frame all objects in the image are detected and filtered. Every tracker instance computes the similarity of the tracked object and the evaluated object. First, distance functions are computed. The distance between the features from the last frame and from the present frame should be very small. If the object would not change in appearance and position, the features should be the same, yielding a zero distance. But since the metric of the features is different in every dimension we need to represent the distance in some other way. That is why a similarity measure in the form of probability models is derived from the distance functions. The probability density functions are estimated from a set of training data. This will be explained later. The distance functions are defined as

$$D_{Hu} = \sum_{i=1}^{7} \frac{1}{m_i^A} - \frac{1}{m_i^B} \tag{6.6}$$

where A denotes the first shape (tracked object in the last frame), B denotes the second shape (object in the actual frame),

$$m_i^A = sign(h_i^A) \cdot log(h_i^A) \tag{6.7}$$

$$m_i^B = sign(h_i^B) \cdot log(h_i^B) \tag{6.8}$$

where h_i^A is the i^{th} Hu moment of the shape A and accordingly for h_i^B. D_{Hu} then denotes the shape (contour) distance. Next, the distance function of templates is presented. For this purpose the correlation between the template of the tracked object and the evaluated object has to be computed.

$$R(x,y) = \frac{\sum_{x'} \sum_{y'} T'(x',y') \cdot I'(x+x',y+y')}{\sqrt{\sum_{x'} \sum_{y'} T'(x',y')^2 \sum_{x'} \sum_{y'} I'(x+x',y+y')^2}} \tag{6.9}$$

where

$$T'(x',y') = T(x',y') - \frac{1}{(w \cdot h) \cdot \sum_{x''} \sum_{y''} T(x'',y'')} \tag{6.10}$$

$$I' = I - \frac{1}{(w \cdot h) \cdot \sum_{x''} \sum_{y''} I(x+x'',y+y'')} \tag{6.11}$$

where I is the image we search in, T is the template that we search for, w and h are the width and height of the template respectively. Because the function is zero for non-correlated images and rises up to one as the images are more correlated, it is a similarity measure rather than a distance measure. To convert it to a distance measure it has to be subtracted from one. Then

$$D_T = 1 - \max_{x,y} R(x,y) \tag{6.12}$$

is the template distance. If the evaluated object is in occlusion the template is rotated several times by different angles and the template distance is computed for every rotation. The idea is to allow in-plain rotation of the object which the template matching method does not take into account. This way the template distance is small if one hand rotates on top of the other. The angles are chosen so that the longer the object is in occlusion the larger the range of tested angles is. The other distance functions are an absolute difference between the values in last frame and the values in the present frame.

$$D_P = \sqrt{(x_t - x_{t-1})^2 + (y_t - y_{t-1})^2} \tag{6.13}$$

is the distance of position, where $[x,y]^T$ is the center of mass of the object.

$$D_V = \sqrt{(v_x^t - v_x^{t-1})^2 + (v_y^t - v_y^{t-1})^2} \tag{6.14}$$

is the distance of velocity, where $[v_x, v_y]^T$ is the velocity of the object. The velocity can be approximated as

$$\vec{v} = \begin{bmatrix} x_t - x_{t-1} \\ y_t - y_{t-1} \end{bmatrix} \tag{6.15}$$

thus the equation 6.14 becomes

$$D_V = \sqrt{(x_t - 2x_{t-1} + x_{t-2})^2 + (y_t - 2y_{t-1} + y_{t-2})^2} \tag{6.16}$$

$$D_{PC} = |p_t - p_{t-1}| \tag{6.17}$$

is the distance of the perimeter of the object, where p is the perimeter of the object.

$$D_{ABB} = |abb_t - abb_{t-1}| \tag{6.18}$$

is the distance of the area of the bounding box, where *abb* is the area of the bounding box of the object. A bounding box is a non-rotated rectangle that fits the whole object and has minimum area.

$$D_{AC} = |ac_t - ac_{t-1}| \tag{6.19}$$

is the distance of the area of the object, where *ac* is the area of the object.
It can be seen from the equations that some of the features depend on the metric of the image. For example Equation 6.13 represents the distance of the centroids of the objects in pixels. This value is dependent on the distance of the object from the camera or the resolution of the camera. The same conclusions can be made on the features described in Equations 6.14 - 6.18. That is why I normalize the features before using them in the tracking process.

Feature Normalization

The shape distance (Equation 6.6) and the template distance (Equation 6.12) are normalized by the definition. All the other features are normalized by the scale factor proportional to the size of the objects. In Figures 6.5 - 6.6 the impact of the normalization is shown.

It can be seen that the original features are directly dependent on the size of the image, hence on the size of the objects. When the normalization is applied the features become rather similar, although not exactly the same. The features are normalized in the following way:

$$D_{Hu} = D_{Hu} \tag{6.20}$$

$$D_T = D_T \tag{6.21}$$

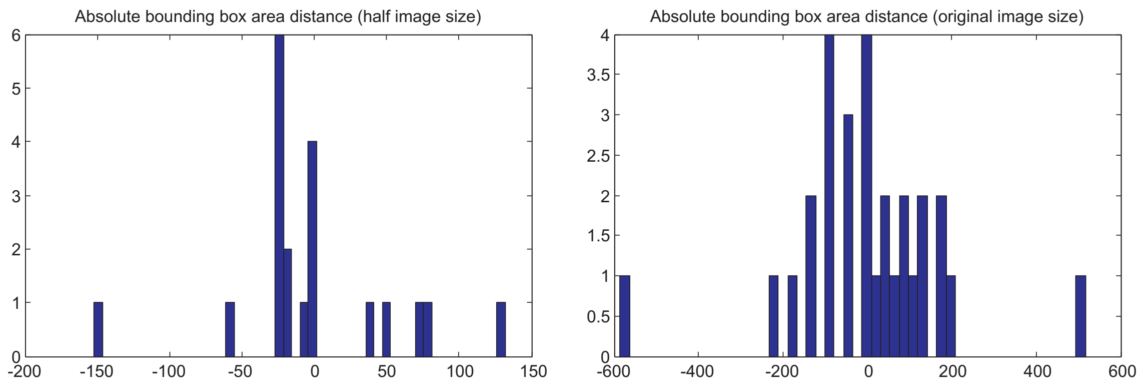

Figure 6.5: Area distance WITHOUT normalization of one video. Left image - half size of the original image, right image - original size of the image

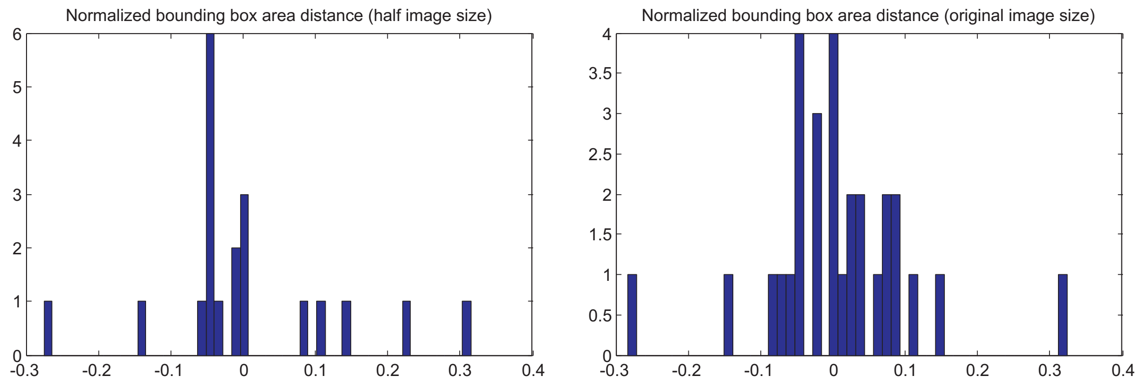

Figure 6.6: Area distance WITH normalization of one video. Left image - half size of the original image, right image - original size of the image

$$D_P = \frac{D_P}{ObjectArea} \tag{6.22}$$

$$D_V = \frac{D_V}{ObjectArea} \tag{6.23}$$

$$D_{PC} = \frac{D_{PC}}{ObjectPerimeter} \tag{6.24}$$

$$D_{ABB} = \frac{D_{ABB}}{ObjectArea} \tag{6.25}$$

$$D_{AC} = \frac{D_{AC}}{ObjectArea} \tag{6.26}$$

The expression $ObjectArea$ is the area of the tracked object from the last frame in pixels. The $ObjectPerimeter$ is the length of the outline of the tracked object from the last frame in pixels.

Similarity models

Based on the values of the distance functions the tracker has to determine the likelihood (or certainty) with which the object is the tracked object. The likelihood function can be build in many ways. I have chosen a trained Gaussian Mixture Model (GMM) to determine the likelihood. Every distance function responds to one dimension. There are seven distance functions. That means that they form a 7D feature space. The training samples are collected during annotation with an untrained tracker. For each object the tracker saves the evaluation to a file. The evaluation is a vector of the values of the distance functions. The data are split into several files depending on the classification of the object. The classes are:

- The object is the tracked object - **Good Model**
- The object is not the tracked object - **Wrong Model**
- The object is in the first frame of occlusion - **First Occlusion Model**
- The object is in the first frame after occlusion - **After Occlusion Model**
- The object is in occlusion (but not the first frame of occlusion) - **Occlusion Model**

For each of the classes a different GMM is trained using the algorithm mentioned in [73]. The object changes dramatically in appearance when it becomes occluded. Some of the distance measures become less important (i.e. the shape of the object). That is why the probability density should be different for every class. The Tracker Manager is able to automatically determine whether the object is in occlusion, whether it entered occlusion or left the occlusion (see section 6.4.3). Based on this knowledge the tracker uses the appropriate GMM to compute the likelihood. Then we can write

$$S_{ij} = p(\boldsymbol{d}; \lambda) = \sum_{i=1}^{M} w_i g(\boldsymbol{d}|\mu_i, \Sigma_i) = p_m(t_i|o_j) \tag{6.27}$$

where S_{ij} is the similarity measure, p is the GMM with parameters λ and $\boldsymbol{d}$ is the vector composed of the values from particular distance measures. The parameters λ determine which model will be used. Alternatively we can use the last term of the equation where m determines the model type, t_i is the tracker that is used (every tracker has its own models) and o_j is the evaluated object. Finally, g is a Gaussian weighted by factor w_i and represented as:

$$g\left(\boldsymbol{d}|\mu_i, \Sigma_i\right) = \frac{1}{(2\pi)^{D/2}\left|\Sigma_i\right|^{1/2}} exp\left\{-\frac{1}{2}\left(\boldsymbol{d}-\mu_i\right)^T \Sigma_i^{-1}\left(\boldsymbol{d}-\mu_i\right)\right\} \tag{6.28}$$

6.4.1 Model training

The untrained tracker does not provide good results and that is why the user has to manually annotate almost every frame. This problem can be resolved by manually setting the tracking parameters. In this case the overall similarity function is a linear combination of the distance functions. That is

$$S = \vec{w}^T \begin{bmatrix} D_{Hu} \\ D_T \\ D_P \\ D_V \\ D_{PC} \\ D_{ABB} \\ D_{AC} \end{bmatrix} \tag{6.29}$$

where $\vec{w}$ is the weighting vector. An expert can set the weights for better tracking performance. Then the weights can be iteratively recomputed based on the data from the annotation using a least squares method. After few iterations the data can be used to train the GMM.

Next, I will present the process of computing the decorrelation and dimension reduction methods that were used in this thesis. They can be divided into two groups - generative and discriminative. The generative models include PCA, ICA, and FA. The discriminative models include LDA and HLDA. The task of the methods is to find a representation of the data in a lower dimension. The discriminative models also take the between class covariance into account.

For every object of interest and every class I train one transformation. That means that 15 models are trained in total. The transformations are used to reduce the dimensionality of the features. The results of the tracking with the employment of dimension reduction can be found in Section 7. The necessity of training each class separately can be observed in Figures 6.7 - 6.9.

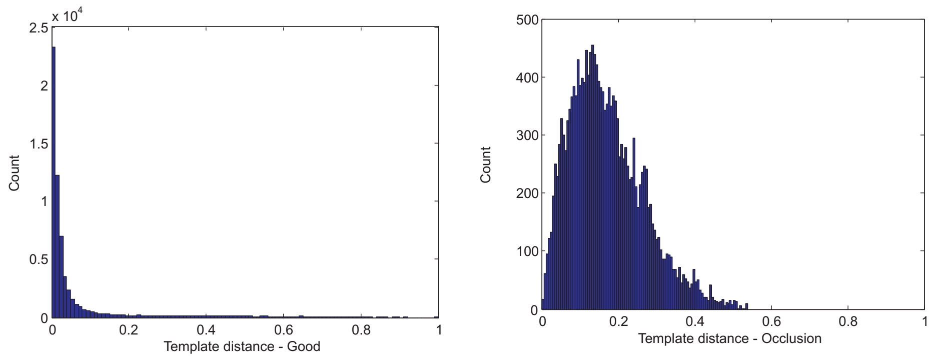

Figure 6.7: Template distance, counts of training values of two different classes (Good, Occlusion) of the right hand.

PCA, ICA, FA transforms. To compute the transformations we need data belonging to one of the five possible classes. Each class is handled separately. This means that for each class we obtain a separate transformation. Each of the methods tries to find a transformation matrix based on different criterion. PCA will align the space according to the largest variance of the data. ICA will find such transformation that the components (axes) of the transformed space are statistically independent. FA finds so called factors which generate the observed

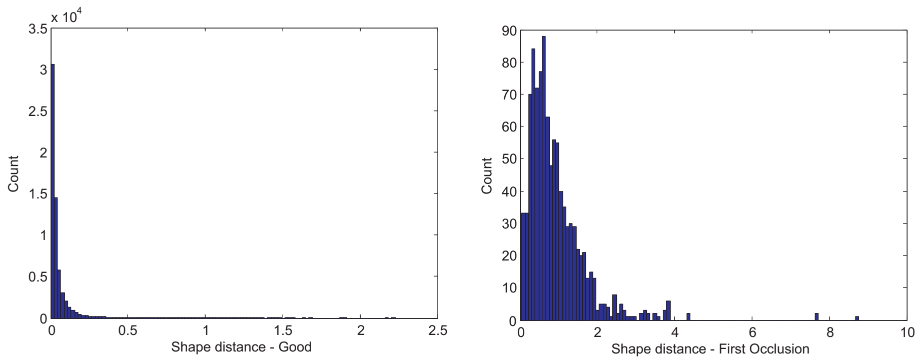

Figure 6.8: Shape distance, counts of training values of two different classes (Good, First Occlusion) of the right hand.

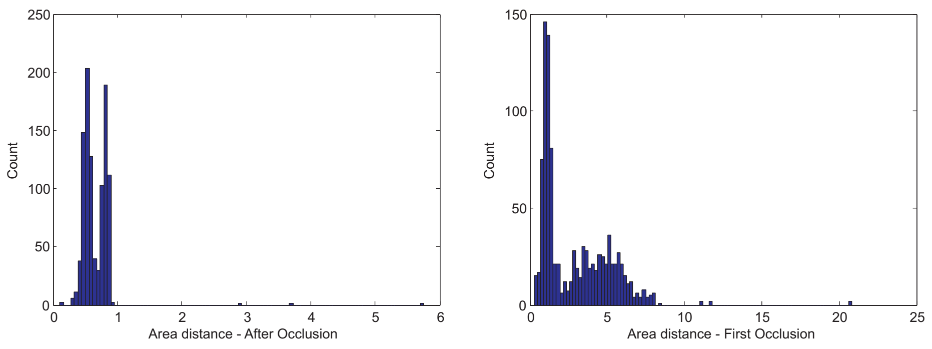

Figure 6.9: Area of object distance, counts of training values of two different classes (After Occlusion, First Occlusion) of the right hand.

data when put into linear combination and combined with an error term. The training data are transformed using the proper transformation and a GMM is computed on the transformed data for each class. PCA was implemented by myself in Matlab. The implementation of ICA was used from the Matlab toolbox FastICA developed by members of Helsinki University of Technology available at `http://research.ics.aalto.fi/ica/fastica/`. The FA was implemented by my colleague Lukas Machlica and the implementation is available at our department.

LDA, HLDA transforms. To train the transformations we need data belonging to all of the five possible classes at once. The idea is to maximize the between class covariance of the data. This should help to separate the feature space so that the classes are far away from each other. Then the transformation is the same for each class (unlike when training generative models). The training data are transformed and for each class a different GMM is trained. The difference between LDA and HLDA is that LDA assumes the class covariance is the same for every class and HLDA does not assume this. The HLDA training was implemented in Matlab by Lukas Burget who followed the work [74]. The LDA training was also implemented in Matlab by Michael Kiefte as a part of Discriminant Analysis Toolbox. He followed the work [75].

The models are then used for the training of the tracking models and then in the tracking process itself. Each of the models can be represented as a transformation matrix. This is due to the linear nature of the models. The matrix is either square or rectangular depending whether any dimension reduction takes place. The transformation is then applied as a product of the transformation matrix and the original feature vector.

$$d_{red} = Td \tag{6.30}$$

where d_{red} is the decorrelated/reduced vector, T is the trained matrix of the decorrelation method, and d is the original feature vector. The translation is not taken into account since it is handled by the GMM. The Equation 6.27 then becomes

$$S_{ij} = p(\boldsymbol{d_{red}}; \lambda) = p(\boldsymbol{T_i d}; \lambda) = p_m(t_i|o_j) \tag{6.31}$$

Note that each tracker is using its own transformation matrix. In Figure 6.10 you can see an example of trained GMM for the right hand tracker. Training data are obtained from the dataset **UWB-06-SLR-A**.

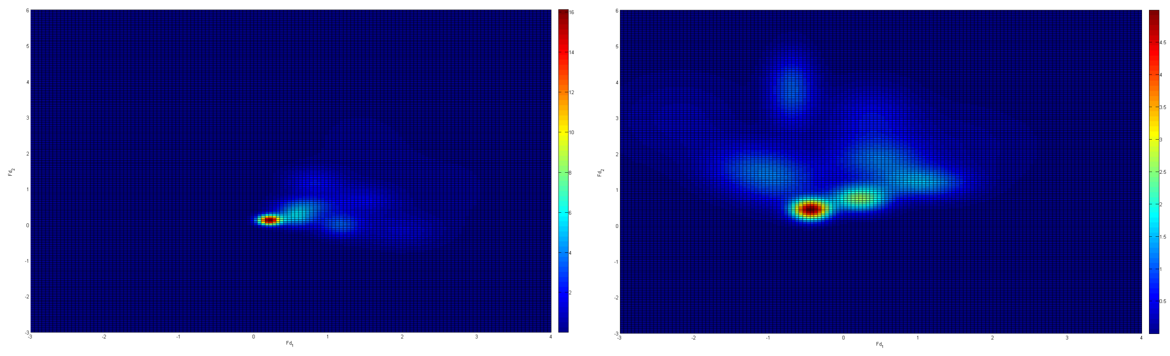

Figure 6.10: Trained GMM for the RIGHT hand. On the left there is the model trained from GOOD features. On the right the model is trained from the WRONG features.

6.4.2 Detection of Occlusion

If a perfect tracker was available the annotation could be created automatically. But the trackers usually fail when an occlusion of objects occurs. Because of this problem the system must be able to detect occlusions of objects and ask the user to verify the result of tracking. In my system I assume that the bounding box of an overlapped object becomes relatively bigger in the first frame of occlusion and relatively smaller in the first frame after occlusion. The area of the bounding box is considered as a feature which determines the occlusion. In Figure 6.11 you can see the progress of the area of the bounding box of the right hand through the video stream of a sign *Brno*. Figure 6.12 is the difference of the area computed as

$$\Delta a = a_t - a_{t-1} \tag{6.32}$$

where a_t is the area of the bounding box in time (frame) t. Figure 6.13 shows the relative difference and thresholds. Other examples can be seen in Figure 6.18 and 6.20.

$$\Delta a = \frac{a_t - a_{t-1}}{a_{t-1}} \tag{6.33}$$

The upper threshold set to 0.5 is used for the detection of first occlusion. The lower threshold set to -0.4 is used for the detection of the first frame after occlusion. The experiments were done on database UWB-06-SLR-A (section 3.1).

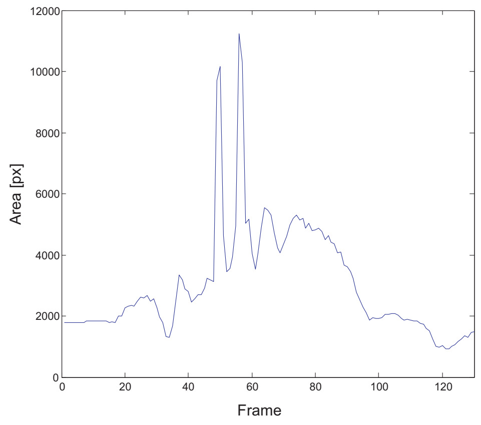

Figure 6.11: Area of the bounding box of the right hand in pixels.

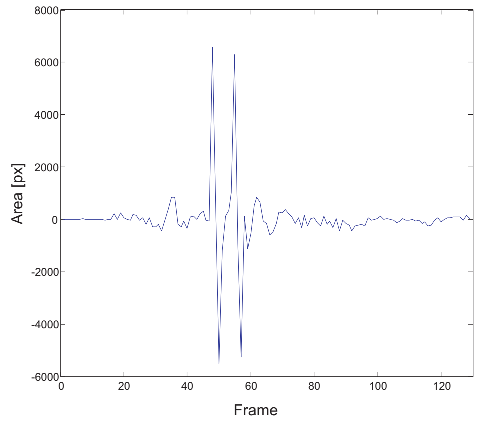

Figure 6.12: Difference of the area of the bounding box of the right hand in pixels.

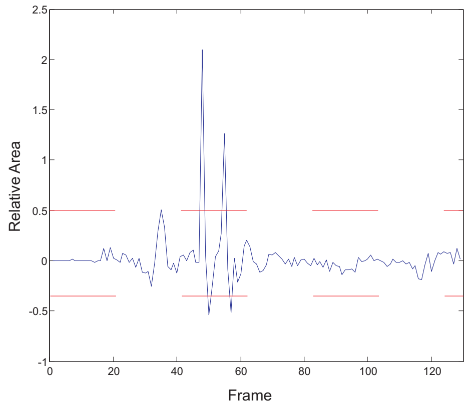

Figure 6.13: Relative difference of area of the bounding box of the right hand in pixels. The dashed lines are upper and lower thresholds for occlusion detection.

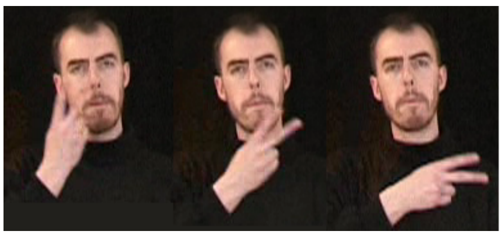

Figure 6.14: Selected frames (48, 49, 50) from the video stream of a sign Brno. Notice that in the frame 48 the relative difference of area is over the upper threshold and in frame 50 is below the lower threshold.

This analysis is useful for the annotation process. If we want to track the objects automatically without any user interference we make the decision about what models to use from the hypothesized configuration. The configuration itself bears the information about occlusion. Therefore the likelihood of the considered hypothesis is computed with proper models. This approach assumes that the tracking models are trained on a large dataset or on a specific signer that is analyzed. In the next section I will describe the configuration determination process.

6.4.3 Configuration Determination

As mentioned in Section 6.1 I am trying to find the optimal map C^* that tells us which tracker tracks which object. Each mapping falls into one of five possible cases of hand/head mutual relations. In this context I will refer to the mutual relation as *configuration*. All possible configurations are depicted in Figure 6.15.

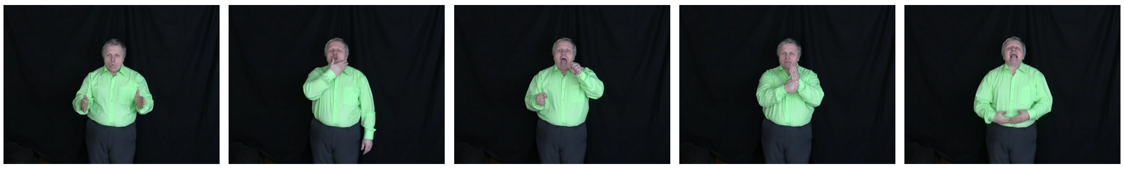

Figure 6.15: Five possible cases of hand/head mutual relation.

The possible configurations are:

1. Nothing is occluded
2. Right hand and head are occluded
3. Left hand and head are occluded
4. Everything is occluded
5. Hands are occluded

Based on the number of detected objects the Tracker Manager chooses only the possible configurations that need to be evaluated. Furthermore the Tracker Manager remembers the last configuration which gives it the information about occlusion changes. For example if the last detected configuration was number 2 (right hand and head are occluded) and the evaluated configuration is number 3 (left hand and head are occluded) the Tracker Manager determines that the right hand will be evaluated by the **After Occlusion** model, the left hand will be evaluated by the **First Occlusion** model and the head will be evaluated by the **Occlusion Model**. The configuration based on the number of detected objects are

- One object - configuration 4.
- Two objects - configuration 2., 3., 5.
- Three objects - configuration 1.

After the best configuration is found, the trackers update their tracked object status. If the object is in occlusion the last non-occluded object shape and appearance is remembered.

This means that for the shape and the template distance the last observed non-occluded objects are used. This approach addresses the fact that during a performance of a sign the shape of the hand does not change too often. If the object leaves occlusion the appearance and shape is updated.

6.4.4 Apriori knowledge

To help the tracker to make a decision in ambiguous situations a model of relative hand/head position is used. As the hands move closer to each other with similar hand shapes the feature vectors become very similar. The norm of the vectors is closing to zero. This may lead to wrong decision about the hands' positions. Also when both hands are in front of the face occluding it for a long time and then leave the occlusion the tracker is confused as the newly observed hand shape may have changed dramatically from the last remembered hand shape. The a priori model is trained from annotated positions of hands and head. First, the positions need to be normalized. The normalization needs to address the translation and metric of the coordinate system. The translation problem comes from a different position of the performer in the recording. Also the body size of the performer creates translations of the objects' positions. The metric problem means that a vector with a norm of one pixel is not a constant distance in the real world. This can be due to different properties of the recordings namely the resolution. The distance of the speaker from the camera also plays its role.

Normalization The translation problem is handled by setting the origin of the coordinate system to a known point in the image. This point should be the same for all recordings. One way to do it is to put a calibration rig into the scene. But this way the data may be corrupted and not fit for publication for example in an on-line dictionary. That is why I use the center of mass of the head as the center of the coordinate system. The head will be in every recording since it makes the sign language more intelligible. We can also use the head to define the metric of the new coordinate system that is not dependent on the properties of the recordings. The metric is set so that one unit is the width of the head. This means the normalization is defined as

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} (x - c_x)/w \\ (y - c_y)/w \end{bmatrix} \tag{6.34}$$

where x', y' are the normalized coordinates, x, y are the original coordinates, c_x, c_y is the center of mass of the head and w is the width of the head in pixels. In practice the position of an object is represented by its centroid. When a configuration is evaluated the object that is considered to be the head is used for the computation of c_x, c_y, and w.

Apriori knowledge representation We need to incorporate the a priori knowledge into the probabilistic framework of the tracking system. To do this I train a GMM over the training samples of relative positions of the head and hands. The relative positions are concatenated to create a feature space. The concatenation is performed in the order left hand, right hand. The head has always coordinates of $[0, 0]$ and thus does not need to be incorporated into the feature vector. Such concatenation yields a four dimensional feature space. A four mixture GMM is trained to represent the a priori knowledge as a probabilistic model. The resulting total likelihood of the tracker t_i tracking the object o_j is then

$$l_{ij} = p_m(t_i|o_j)p_a(x') \tag{6.35}$$

where $p_m(t_i|o_j)$ is the term defined in Equation 6.27, x' is a vector of the normalized positions of the objects, and $p_a(x')$ is the probability of the relative positions of the left and right hand. This shows that the computation of the probability that the object o_j is tracked by the tracker t_i is dependent on the positions of the other objects. This means that this probability can be pre-computed for a given configuration. This changes the computation of the log-likelihood in Equation 6.3 to

$$L_k = \sum_{i=1}^{3} \log p_m(t_i|C_k(t_i)) + log(p_a(x')) \tag{6.36}$$

Since the configuration C_k tells us which tracker tracks which object we are able to compute the relative positions for the configuration easily. Lets assume that tracker 1 is designed to track the left hand, tracker 2 tracks the right hand and tracker 3 tracks the head. Then the vector of relative positions is computed as

$$x' = \begin{bmatrix} \left[CoM(C_k(t_1)) - CoM(C_k(t_3))\right]/Width(C_k(t_3)) \\ \left[CoM(C_k(t_2)) - CoM(C_k(t_3))\right]/Width(C_k(t_3)) \end{bmatrix} \tag{6.37}$$

where CoM is a function that computes the center of mass of an object, and $Width$ is a function that computes the width of an object. Every row of the right side of the equation is in fact a two dimensional vector. This means the resulting x' will have the desired four dimensions. One way how to compute the function CoM is

$$CoM(o) = \frac{1}{N}\sum_{i\in o} o^i = \frac{1}{N}\sum_{(x,y)\in o} \begin{bmatrix} x \\ y \end{bmatrix} \tag{6.38}$$

where N is the number of pixels in the object, and o^i is the i^th pixel in the object. The $Width$ can be computed as

$$Width(o) = \arg\max_{x\in o} x - \arg\min_{x\in o} x \tag{6.39}$$

In other words the function $Width$ is equal to the width of a bounding box of the object o. Since the head is not rotating a lot this approximation of the width of the object is sufficient. Examples of the training data x' from the Equation 6.37 are depicted in Figure 6.16.

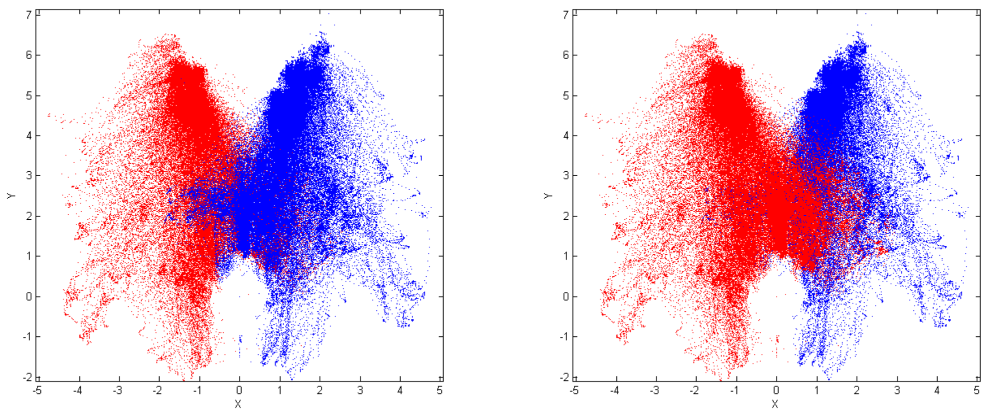

Figure 6.16: Training data for a priori knowledge model of the dataset **UWB-06-SLR-A**. On the left side the data of the LEFT hand are drawn over the data of the RIGHT hand. On the right side vice versa.

6.5 Annotation Process

The annotation begins with loading the video file. In the first frame the trackers are initialized. There is one tracker for each object. In the case of SL the objects are head, left and right hand. Thus, there are three trackers in this scenario. The initialization process is as follows. The image is segmented using a skin color model. All the small objects and the very large objects are filtered out. Every tracker is created with a search window. If an object is found in this window, the tracker is initialized by the object. The result of the initialization is presented to the user. He has to decide whether the trackers are initialized correctly and if not, he has to initialize them manually. The trackers identify themselves by a green contour of the tracked object, a blue bounding box of the object and a string with the name of the object (left hand, right hand, head). See Figure 6.17. After the initialization the above mentioned tracking process begins. The human operator can pause the video stream in any frame and with a key strike he is able to view the stream frame-by-frame. If the area of the bounding box changes rapidly, the system pauses the stream automatically. Usually this is a sign that two or more objects collided with each other or were separated from each other. This state can create confusion for the tracker and the user has to verify the correctness of the automatic annotation. If the annotation does not seem to be right, the user modifies it. In this case all the detected objects are presented to the user one by one and he can annotate (assign a tracker to) the object. This way the user does not need to annotate every frame which means he saves time. The annotation of the object is performed by pressing the corresponding number of the object(s) (1 - Left Hand, 2 - Right Hand, 3 - Head, 4 - LH & RH, 5 - LH & H, 6 - RH & H, 7 - LH & RH & H). Each frame the tracking features are saved to a proper file defined by the configuration of the body parts. These features are then used for the training of GMMs.

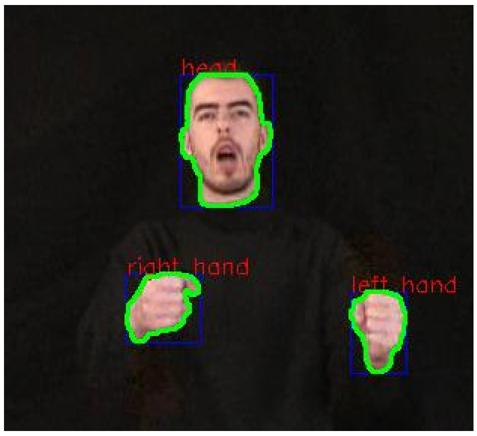

Figure 6.17: Example image from the annotation process.

6.6 Verification Process

After the annotation is done the user can verify it. The system loads the saved features of the video stream and presents them to the user. In every frame the system draws the detected contours and bounding boxes along with the string identifier into the image from the video stream. See Figure 6.22. This way the user is able to tell whether the annotation was successful or not. Some additional information can be seen in the verification mode. It is a line connecting the center of mass of the object in the last frame and in the present frame. The length of the line is also written on the screen. This may be helpful when an expert is setting the tracking parameters. Again, the user can pause the stream at any time and view the video frame-by-frame.

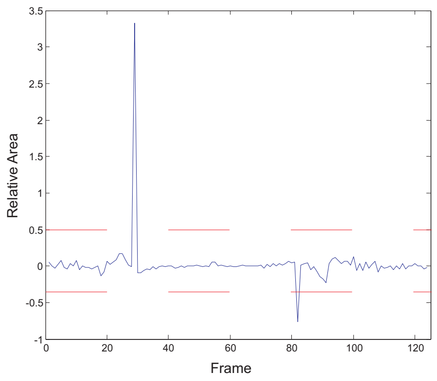

Figure 6.18: Relative difference of area of the bounding box of the right hand in pixels of the sign divka (girl).

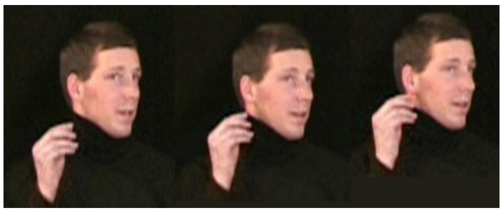

Figure 6.19: Selected frames from the video stream of a sign divka. You can observe a frame just before occlusion (29), the first frame of occlusion (30) and the consequent frame (31).

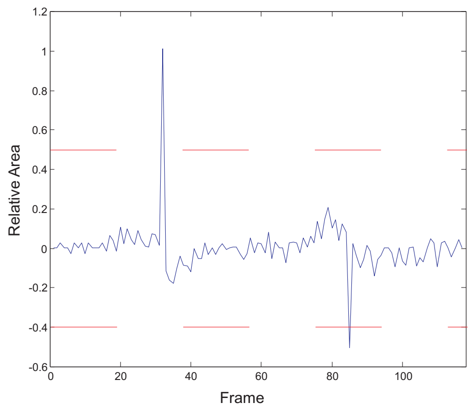

Figure 6.20: Relative difference of area of the bounding box of the right hand in pixels of the sign loucit se (farewell).

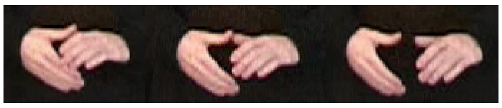

Figure 6.21: Selected frames from the video stream of a sign loucit se. You can observe the last frame of occlusion (83), the first frame after occlusion (84) and the consequent frame (85).

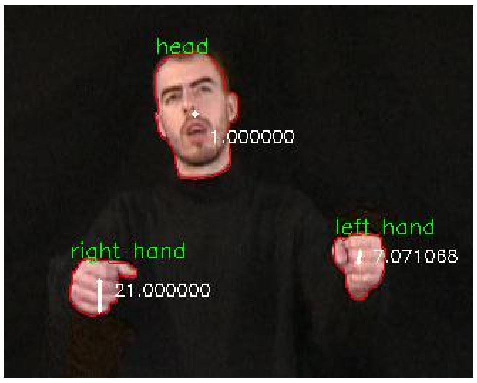

Figure 6.22: Example image from the verification process.

6.7 Conclusion and Application

A system capable of tracking the head and hands in video sequences is presented. The system can help experts to annotate SL video streams without any major time consumption. The annotation is used for computation of the features. This way a system of recognition can be developed independently from the feature extracting system. New algorithms for feature extraction can be compared with the baseline system, not only in the domain of recognition but also in the correctness of the extracted features. The system can be combined with a lip reading system which is available [76] on our department to achieve better recognition results. The verification mode is a fast way to verify the annotation and it helps experts to set the tracker parameters manually.

The features were tested using a SLR system. In paper [77], in cooperation with other colleagues, we presented experimental results. The selected features were

- x, y - the center of mass of the object
- 7 Hu moments describing the shape of the object

p	PCA	ICA	LDA	HLDA	rHLDA
6	77.7	76.6	76.4	80.4	78.6
7	80.4	80.4	75.4	84.3	81.9
8	84.3	83.6	75.6	87.7	82.0
9	84.9	86.8	77.6	86.1	85.2
10	86.9	89.2	79.9	88.9	88.0
11	*88.1*	*89.6*	*82.6*	*89.4*	***89.8***

Table 6.1: Recognition accuracy for the dimension p of a feature vector

- the angle of the object relative to the x-axis of the image

- a Boolean value representing whether the object is in occlusion

and they were manually proofread so that we use only proper features. The occlusion flag is used in post-processing. If an occlusion is detected, the Hu moments and the angle are linear interpolated between the last values before occlusion and the first values after occlusion. Center of mass (x, y) is not interpolated as it tells us how the occluded objects move. The final step of the post-processing is a normalization in the spatial domain. The mean position of the head is considered as the origin of the coordinate system and the mean width of the head is considered as one unit. The normalized features (excluding the occlusion flag) are concatenated in the following order: left hand, right hand, and head. For every object, 10 features are obtained, which makes a total count of 30 features for every frame.

Next, we tested several methods to reduce the dimensionality of the data (the dimension $p \in \{6, \dots 11\}$). For each of these setups, several HMM models were trained and evaluated (from $z \in \{4, \dots, 9\}$ states and $c \in \{1, \dots 50\}$ mixes in each state). This evaluates to 180×15 experiments and $180 \times 50 \times 15$ results (see Table 6.1).

In another research [78] we tested the Local Binary Patterns. The system described above was used to obtain the positions and contours of the head and hands. This information was used to obtain a window containing the texture of the objects. We chose a window of the size 16x16 pixels centered on the center of mass of the object (Figure 6.23).

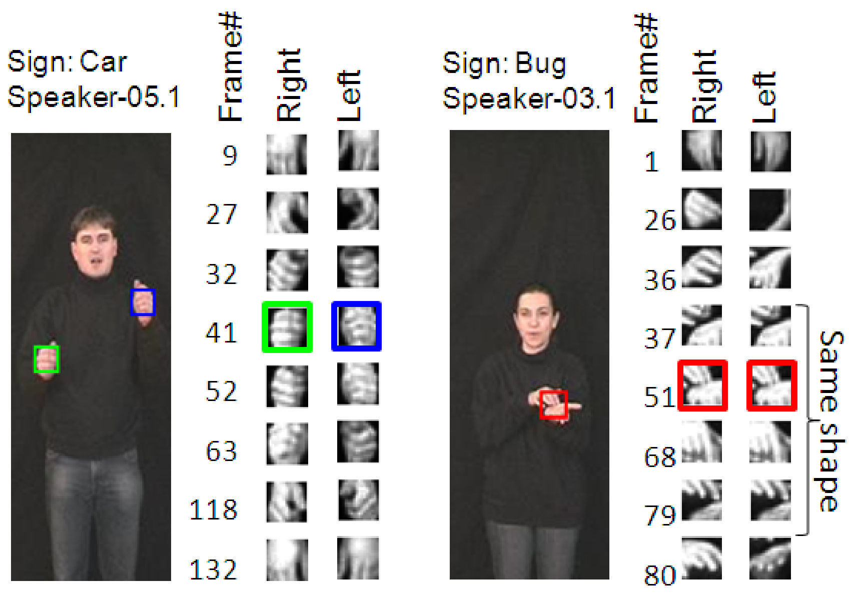

Figure 6.23: Examples of segmented hands for LBP extraction.

In one of the experiments the window was divided into 16 equal sub-windows and each sub-window was handled independently. The texture in a sub-window was analyzed using multi-scale LBP operator with circular neighborhood of two pixel radius. Each sub-window produced a histograms of LBPs represented as decimals and these histograms were concatenated into one feature vector (Figure 6.24). In other experiment we analyzed the whole 16x16 window.

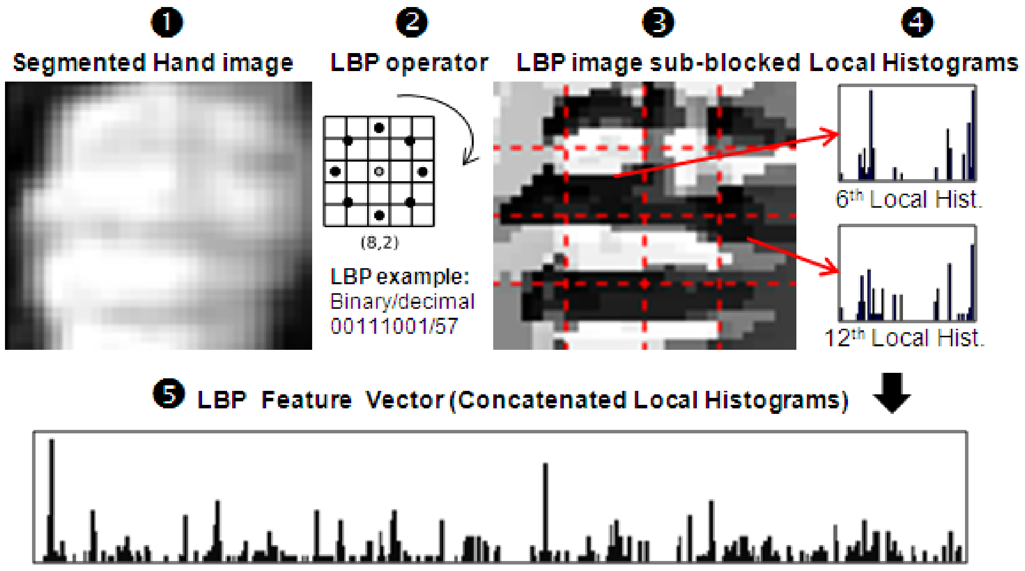

Figure 6.24: LBP feature vector forming.

To evaluate the discriminative power of LBPs we designed several experiments. They were realized for signer-semi-dependent (SSD) and signer-independent(SI) recognition test.

Features Abbreviation	Features Description	Dim
GM	Hu moments, location, orientation	30
$GM + \Delta GM$	GM + delta coefficients between 2 frames	60
LBP	Cropped hand 16x16, hist. from 4 subblocks	472
LBP_2	Cropped hand 16x16, hist. from whole image	118
ASM_1	Inner + outer lip, eyes, eyebrows	96
ASM_2	Inner + outer lip, eyebrows	62
ASM_3	Outer lip, eyebrows (best non-manual)	48
ASM_{3_norm}	ASM_3 centered by head in first frame	48
ΔASM_3	Delta coefficients between 2 frames	48
$ASM_3 + \Delta ASM_3$	ASM_3 + its delta coefficients	96
ASM_4	Outer lip	24
ASM_5	Inner lip	14
ASM_6	Eyebrow	24
$GM + LBP_2$	Manual-geometric + Manual-appearance	148
$GM + ASM_3$	Manual-geometric + best non-manual	84
$LBP_2 + ASM_3$	Manual-appearance + best non-manual	166
$Comb - 1$	$GM + LBP_2 + ASM_3$	196
$Comb - 2$	$GM + LBP_2 + ASM_3 + \Delta GM$	226
$Comb - 2_{PCA}$	$(GM + LBP_2 + ASM_3 + \Delta GM)_{PCA}$	40

Table 6.2: List of features used for experiments. First part lists the manual features, second part lists the non-manual features, and third part lists their combination. Abbreviation: Dim - dimension of feature vector, LBP - Local Binary pattern, GM - geometric moments, ASM - active shape model, PCA - principal component analysis

Data used for experiments consist of 23 signs performed by 11 signers with 3-6 repetitions, in total 1065 video files. Training data for the SSD test contains 60% of repetitions for each signer and every sign, remaining data is used for testing. Training data for the SI test contains all signs from 60% of signers, testing is done for unseen signers. We also added features representing the non-manual component. Active shape model was implemented to track the facial features. The extracted features are summarized in Table 6.2.

We summarize the results of the experiments for various combinations in Tables 6.3 and 6.4. The recognition was realized via Hidden Markov Model using the HTK toolkit. Following the work [77] the left-right Hidden Markov Model (HMM) is used. In every experiment the first and the last state is non-emitting. To perform optimal recognition each feature set requires a different number of HMM states and different number of Gaussians in a mixture for state modeling (in experiment denoted as *# of state mixtures*). To obtain the optimal number of these two parameters a brute force test was applied with no initial guess.

Features	# of HMM states/ # of state mixtures	Signer-Semi Dependent	# of HMM states/ # of state mixture	Signer Independent
GM	9 / 10	83.09%	10 / 10	27.90%
$GM + \Delta GM$	9 / 30	87.58%	-	-
LBP	6 / 10	94.36%	6 / 10	58.18%
LBP_2	8 / 10	**94.61%**	5 / 10	**68.42%**

Table 6.3: Recognition rate for individual manual features.

Features	# of HMM states/ # of state mix.	Signer-Semi Dependent	# of HMM states/ # of state mix.	Signer Independent
$GM + LBP_2$	8 / 20	92.72%	5 / 10	**57.54%**
$GM + ASM_3$	8 / 40	83.09%	-	-
$LBP_2 + ASM_3$	9 / 10	92.89%	9 / 30	20.10%
$Comb - 1$	9 / 10	90.69%	-	-
$Comb - 2$	10 / 5	91.67%	10 / 10	35.17%
$(Comb - 2)_{PCA}$	10 / 5	**99.75%**	9 / 10	37.93%

Table 6.4: Recognition rate for combined features.

The experiment results showed that feature representing hand shape via appearance are more discriminative than geometric moment containing information about the trajectory and parametric representation of hand (orientation, area size, etc.). On the other hand the appearance of the hand is strongly dependent on the position and orientation relative to the camera. The main message of our experiments with non-manual features is that eyebrows and outer lip contour are discriminative enough to capture the difference among the signs. The combination of features $comb - 2$ has for the SSD test the best recognition rate among all experiments. This information is valuable when we want to recognize a database with low number of known signers (for example in a dictionary). The SI test achieved relatively low recognition rate. This can be due to relatively high dimension of the data versus small number of samples.

The features can be used for analysis of the signs that is different than recognition. In [79] we used the features for automatic categorization of lexical signs. We chose several categories mostly derived from the manual component summarized in Table 6.5.

All the categories can be determined from the trajectory of the hands.
Hand movement. To determine whether the sign is one handed or two handed the sum of variance of x and y components of the trajectory as in Equation 6.40.

$$V = \sum_i (x_i - \mu_x)^2 + \sum_i (y_i - \mu_y)^2 \tag{6.40}$$

Table 6.5: The sign categories chosen for the experiment.

Hand movement	*Body contact*	*Hand location*	*Head*
one handed	no contact	at waist	mouth wide open
two handed	contact of head and right hand	at chest	mouth wide closed
symmetric	contact of head and left hand	at head	lip pressed together
non-symmetric	contact of hands	above head	lip pucker
	contact of everything	above head	closed eyes

where (x_i, y_i) is the relative position of the hand in the i^{th} frame. The relative position is computed by using the Equation 6.34. If the value of V is above a threshold the hand is considered as moving. Then the decision about one/two handed sign is straightforward. If for both hands V is above the threshold the sign is considered two handed else one handed. The symmetry of hands in a sign is defined for example in [80]. The definition is not simple but can be summarized as when the trajectories of the hands are mirrored or there is a phase shift in the movements then the hand movement is symmetric. The later can be seen as a sort of anti-symmetry (one hand is up moving down, the other down moving up). If the signer is recorded so that he/she is facing the camera and the normal of the body is perpendicular to the camera image plane then correlation of the trajectories of the hands is directly dependent on the symmetry. To utilize this knowledge the Pearsons' correlation coefficients are computed using the trajectories of both hands. Each component has to be computed separately and then the results need to be analyzed to decide about symmetry. For each component the symmetry coefficients is computed as

$$S = \frac{cov(X_1, X_2)}{\sigma_{X_1}\sigma_{X_2}} = \frac{E[(X_1 - \mu_{X_1})(X_2 - \mu_{X_2})]}{\sigma_{X_1}\sigma_{X_2}} \tag{6.41}$$

where X_1 is the x respectively th y component of the trajectory of one hand and X_2 is the trajectory of the other hand. For each component we obtain a scalar value representing the symmetry. The value is put into absolute value in order to reflect the antisymmetry of the trajectories. If both components are above threshold of 0.89 then the sign is declared to be symmetric.
Hand location. The location of a hand symbolizes what space relative to the location of the head has the hand occupied the most. We compute a histogram of relative y positions of hands consisting of 5 bins. The bins are chosen so that they correlate with the categories. Then the category connected to the most occupied bin is chosen. This approach can fail if the sign duration is relatively small to the video duration. That is why we consider only the segment of the video where the hands are moving and are out of starting position.
Body segment contact. The last category is body segment contact. For now we can only tell whether the objects occlude each other relative to the camera or touch each other. This is a necessary condition for the body parts contact, but not sufficient. Further experiments are needed. This condition is met when two trackers report the same object as the tracked one.

The categorization has been tested on 426 recordings of entries in the SL dictionary.

Category	Recognition rate [%]
One/two handed	1.0
Contact	0.54
Occlusion	1.0
Left hand location	1.0
Right hand location	1.0
Symmetry	0.86

Table 6.6: Recognition accuracy of the categorization

These were automatically tracked by the system presented in this work and the features from the tracking were analyzed as described above. Among the linguists there are discussions about the categories and how to determine them. Therefore, I have manually annotated the 426 videos by my best judgment. The results are presented in Table 6.6.

The most problematic category is the contact. The system is able to detect occlusion of the body parts which does not necessarily mean that a contact occurred. This phenomena needs to be investigated further.

Chapter 7

Experiments

To evaluate the performance of the system several experiments were conducted on corpora **UWB-06-SLR-A** (Section 3.1) and **UWB-12-SILADON** containing signs that are available on `znaky.zcu.cz` (Section 3.3). First, the video sequences were semi-automatically annotated using the system as described in Section 6.5. The annotation was used as a training set for the models and also as ground truth for the system evaluation.

The corpus **UWB-06-SLR-A** contains data from 15 performers, each performer is signing 25 signs with 4 to 7 repetitions. For my experiments I left out two performers which performed the signs poorly and as such they were not suitable for these experiments. This way there were 1642 video sequences in total. With the low number of signs and relatively high number of repetitions and performers this dataset is suitable for testing signer independent tracking or sign independent tracking.

The corpus **UWB-12-SILADON** contains one professional performer signing 1142 signs with no repetitions. Hence the corpus is suitable for testing signer dependent tracking with high variation of movements and hand shapes.

Furthermore the features were analyzed to obtain decorrelated features with the possibility to reduce the dimensionality. Several such methods were tested - Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA, sometimes called Fisher's linear discriminant), Heteroscedastic LDA (HLDA), Factor Analysis (FA), and Independent Component Analysis (ICA).

The tracker's model is a GMM. For each number of dimensions and decorrelation method I find the best number of components for the GMM.

Also the effect of incorporating a priori knowledge about the relative hand positions is tested. The models trained on one corpus are evaluated on the data from a different corpus. These experiments show the generality and robustness of the tracking models. This way the normalization of the features and a priori knowledge model's training data is tested. Finally the models are trained with lower number of training examples and the results are compared.

The evaluation of the tracking is similar to [81]. First, the corpora are semi-automatically annotated as described in Section 6.5. Then I compute the Euclidean distance between the tracked position of an object and the annotated position. I consider the centroid as the position indicator. If the distance is less than a threshold τ for the given frame I label the frame as being tracked correctly. Otherwise I count it as a tracking error. For the given threshold I compute the relative count of the faulty frames versus total frames. I evaluate every object separately and also jointly. When evaluating the objects jointly it means that if at least one object is tracked faulty the whole frame is considered to be faulty. Thus the joint or combined error rate will be always larger than the individual objects' tracking errors. Note that for the dataset **UWB-12-SILADON** there is 506 730 total frames and for the dataset **UWB-06-SLR-A** there is 1 368 750 frames.

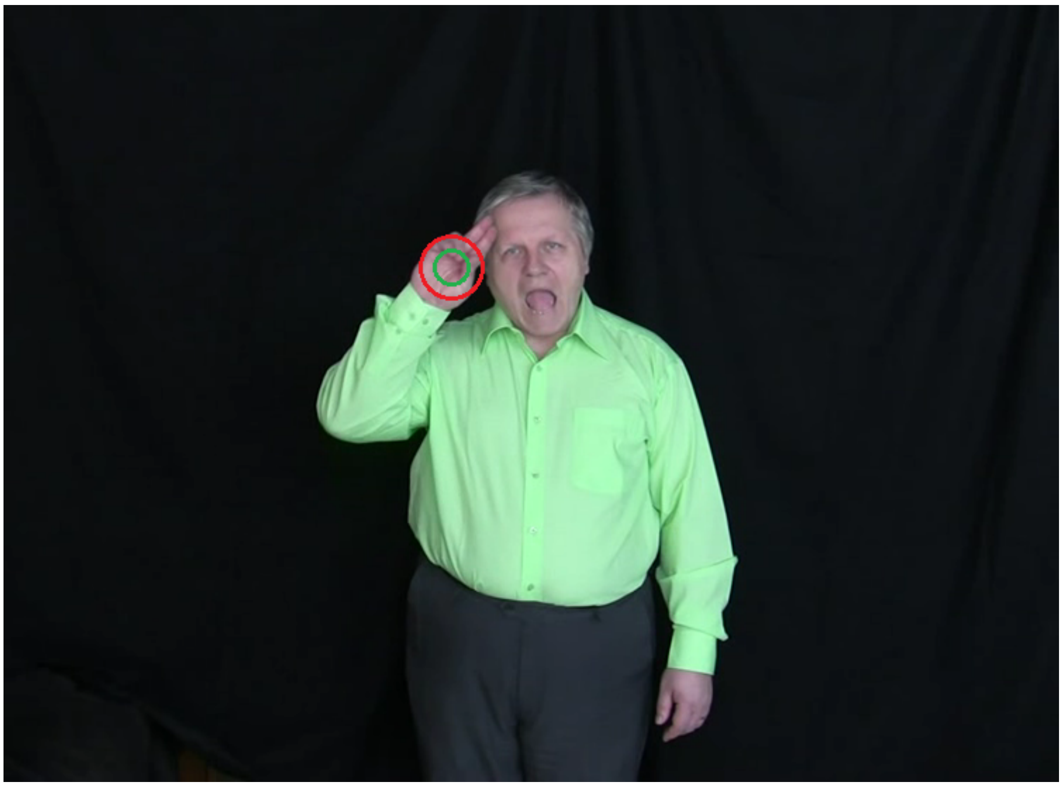

Figure 7.1: Visualisation of the tracking thresholds - radius 25px - green, 45px - red

The value of τ has been chosen with respect to the properties of the video. The different values can be categorized as perfect tracking, good tracking, and mediocre tracking. Perfect tracking means that the detected centroids of the objects are very close to the annotated value. The distance is a small portion of the size of the objects of interest. Good tracking means that the tracked centroid is in the vicinity of the annotated object. In other words the tracked centroid lays inside of the annotated object. Mediocre tracking means that the tracked object was not lost or confused with other object. For both corpora **UWB-12-SILADON** and **UWB-06-SLR-A** the thresholds were chosen as 5 pixels, 25 pixels and 45 pixels. The thresholds are visualized in Figures 7.1 and 7.2.

The method described in the paper [67] and mentioned in Section 5.2 is tested on the same data to measure the success rate of the tracker developed in this thesis. The results can be seen in Tables 7.1 and 7.2.

The naïve approach to track the hands seems to fail. This is due to the complex latent motion model of the hands which is theorized to be non-linear. The hands are often joint into

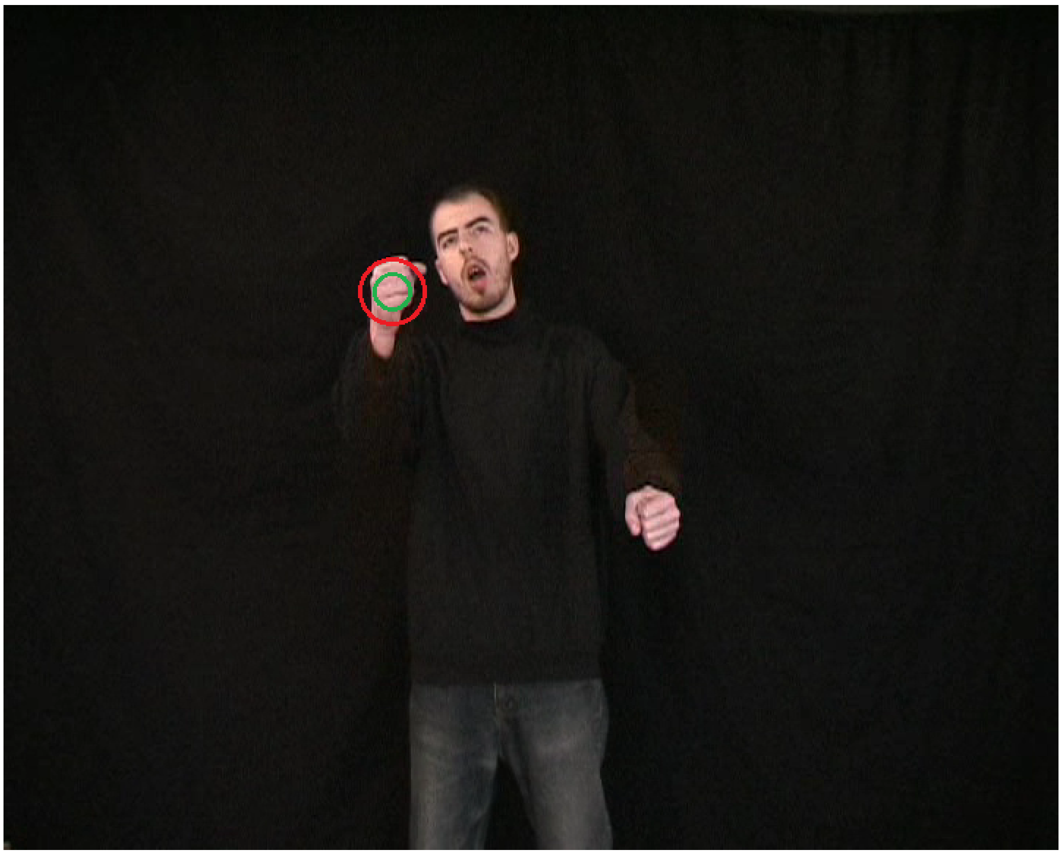

Figure 7.2: Visualisation of the tracking thresholds - radius 25px - green, 45px - red

one object when the degradation of the smaller ellipse begins. If the hands do not separate quickly the smaller ellipse will degrade completely leaving it to be of size one pixel. We can observe that the system works much better with the corpus **UWB-06-SLR-A**. This may be due to the small number of signs from which a lot of signs are simple without occlusion for long times. These signs may be considered as natural movements of hands which the system addresses. These kind of gestures are handled by the system quite well. The corpus **UWB-12-SILADON** is tracked much worse. There is a large variety of movements performed by a professional signer. The movements are in nature non-linear with varying acceleration. The first order movement model does not cope with such movements well. With some modifications the system would perform better but in this work its role is to experimentally show that such simple approaches fail in SL scenario. The system was implemented by myself following the paper [67] in C++ and OpenCV.

Pixel tolerance τ	Left hand	Right hand	Head	Combined
5 px	31.04%	47.25%	13.17%	51.24%
25 px	18.80%	33.27%	2.52%	36.70%
45 px	14.14%	22.53%	0.53%	25.42%

Table 7.1: Error rates for the corpus **UWB-06-SLR-A**. Tracked with the system presented in [67] by Argyros et al.

Pixel tolerance τ	Left hand	Right hand	Head	Combined
5 px	55.58%	74.02%	62.74%	76.95%
25 px	45.06%	67.92%	17.18%	69.71%
45 px	43.22%	63.67%	15.28%	65.30%

Table 7.2: Error rates for the corpus **UWB-12-SILADON**. Tracked with the system presented in [67] by Argyros et al.

7.1 Experiment 01: The basic functionality

Now I will present and discuss the results of my tracking system. The first experiment was focused on the basic functionality of the tracking system. During the annotation phase the trackers seemed to perform well. The annotator needed to make one or two corrections during a single sign. This was the first clue that the system will work as intended. The problem occurred when the features were not normalized and the tracking models trained from them were tested on different data. The tracking failure was extreme even in simple cases. This has made the work of the annotator very difficult. Almost every frame had to be annotated manually. After some attempts to normalize the features the best choice was found which is explained in Section 6.4 under paragraph Feature Normalization. The whole corpus **UWB-12-SILADON** was annotated. All 1142 videos were labeled semi-automatically. The annotator mistakes were corrected by me to obtain flawless features for training. From the corpus **UWB-06-SLR-A** not all recordings were annotated. For every sign three repetitions were annotated a model was trained and then the rest of the video files was tracked automatically. The results were checked and corrected where needed. In the first experiment the models were trained on a portion of the individual corpora and tested on the whole dataset. This means that the training data were also included in the testing data. In the tables there will be results for threshold $\tau = 25$ pixels and the combined error rate of all body parts. The individual dimension reduction methods are evaluated. The a priori knowledge is used in these experiments. The results can be seen in Tables 7.3 - 7.8. The log-likelihood is given by

$$L_k = \sum_{i=1}^{3} \log p_m(t_i|C_k(t_i)) + log(p_a(x')) = \sum_{i=1}^{3} \log p(\boldsymbol{T_i d_i}; \lambda) + log(p_a(x')) \tag{7.1}$$

where T_i is the transformation matrix used by the i^{th} tracker, d_i is the feature vector created by the i^{th} tracker when the object given by the configuration $C_k(t_i)$ is observed. In case that there is no transformation applied the matrix T_i is the identity matrix.

dim / mixtures	1	2	3	4	5	6	8	10
7	7.54%	6.56%	6.14%	6.69%	**4.90%**	5.41%	4.98%	5.24%

Table 7.3: Error rates for the corpus **UWB-12-SILADON**. Tracked with the system presented in this thesis. Transformation: **NONE**, training files: 1032, a priori knowledge: true

dim / mixtures	1	2	3	4	5	6	8	10
2	7.84%	8.19%	8.30%	8.14%	7.52%	7.63%	7.45%	**7.40%**
3	7.73%	8.20%	8.36%	8.32%	7.61%	7.60%	7.50%	7.50%
4	8.72%	8.18%	8.77%	8.52%	8.54%	8.37%	8.41%	8.48%
5	24.28%	9.46%	9.18%	7.51%	8.12%	44.46%	33.94%	27.65%
6	62.05%	33.48%	25.89%	43.60%	47.60%	19.22%	33.00%	60.71%
7	14.28%	57.46%	44.58%	50.28%	59.08%	48.60%	52.38%	80.54%

Table 7.4: Error rates for the corpus **UWB-12-SILADON**. Tracked with the system presented in this thesis. Transformation: **FA**, training files: 1032, a priori knowledge: true

dim / mixtures	1	2	3	4	5	6	8	10
2	7.67%	8.00%	8.17%	8.09%	8.19%	8.20%	8.38%	8.62%
3	7.67%	8.16%	7.98%	8.23%	7.97%	7.88%	7.88%	7.93%
4	7.99%	8.10%	8.06%	8.08%	8.06%	7.87%	7.91%	7.78%
5	8.26%	8.25%	8.25%	8.32%	8.03%	7.98%	8.15%	8.18%
6	8.27%	8.01%	8.23%	8.30%	8.40%	8.37%	8.31%	8.25%
7	8.00%	8.07%	8.20%	7.85%	7.86%	7.93%	7.60%	**7.62%**

Table 7.5: Error rates for the corpus **UWB-12-SILADON**. Tracked with the system presented in this thesis. Transformation: **HLDA**, training files: 1032, a priori knowledge: true

dim / mixtures	1	2	3	4	5	6	8	10
2	7.94%	7.80%	7.64%	7.76%	7.59%	7.55%	7.63%	7.62%
3	7.96%	7.72%	7.77%	7.59%	7.69%	7.67%	7.54%	7.97%
4	7.99%	7.77%	8.04%	7.85%	7.81%	7.88%	7.57%	**7.48%**
5	8.22%	8.26%	7.98%	7.79%	7.97%	7.81%	7.98%	7.79%
6	8.11%	8.11%	7.92%	7.66%	8.16%	8.05%	7.73%	8.17%
7	8.00%	8.26%	7.86%	7.67%	8.28%	8.50%	7.83%	8.12%

Table 7.6: Error rates for the corpus **UWB-12-SILADON**. Tracked with the system presented in this thesis. Transformation: **LDA**, training files: 1032, a priori knowledge: true

dim / mixtures	1	2	3	4	5	6	8	10
2	7.74%	7.95%	7.93%	7.89%	7.74%	7.76%	7.75%	7.72%
3	10.27%	9.22%	8.90%	8.74%	7.85%	7.72%	7.94%	7.88%
4	9.82%	8.49%	8.46%	7.83%	7.70%	8.18%	7.91%	7.43%
5	7.98%	7.42%	7.60%	**7.35%**	7.44%	7.79%	7.68%	7.61%
6	8.23%	7.90%	7.92%	7.58%	7.77%	7.93%	7.97%	7.75%
7	8.00%	7.60%	7.88%	7.66%	7.72%	8.03%	7.86%	7.86%

Table 7.7: Error rates for the corpus **UWB-12-SILADON**. Tracked with the system presented in this thesis. Transformation: **PCA**, training files: 1032, a priori knowledge: true

dim / mixtures	1	2	3	4	5	6	8	10
2	7.50%	7.49%	7.68%	7.83%	7.75%	7.74%	7.74%	7.73%
3	11.24%	9.23%	9.02%	8.63%	7.86%	7.81%	7.82%	7.60%
4	9.33%	**7.29%**	7.85%	7.69%	7.66%	7.41%	7.50%	7.40%
5	8.02%	7.49%	7.99%	7.81%	7.66%	7.51%	7.56%	7.65%
6	8.05%	7.42%	7.87%	7.84%	7.56%	7.57%	7.37%	7.34%

Table 7.8: Error rates for the corpus **UWB-12-SILADON**. Tracked with the system presented in this thesis. Transformation: **ICA**, training files: 1032, a priori knowledge: true

The extensive tests showed that with the dataset **UWB-12-SILADON** the tracking error rate is definitely lower then the relatively simple approach of Argyros et al. When there is no transformation applied the results are better. There is some possibility to reduce the dimension of the feature vector and still obtain very reasonable error rates. Moreover when the features are transformed the tracking error seems more stable with respect to the choice of number of mixtures in the GMM. When using FA the system performed the best in two dimensions. On the other hand when 5 and more factors are used the system performs very poorly. When the factors were analyzed it was clear that only four factors bear the information relevant for tracking. This was revealed when computing the covariance matrix of the transformation matrix and only the four eigenvalues were significant. The rest were several magnitudes smaller. This resulted into chaotic identification of body parts without any consistency. The other transformations were comparable in the means of tracking error. The fact that the system works the best with no transformation of the features may be in the characteristics of the dataset. When only one signer performs the signs the original feature vectors describe him sufficiently well since they are compact. Furthermore these experiments show that the system can handle unseen signs well. This is because the sign is broken down into consecutive frames. If the training data cover a wide variety of the consecutive changes of hand shapes and positions then the models generalize these changes to be able to describe any movement of hands. Now I will present the results for the dataset **UWB-06-SLR-A**.

dim / mixtures	1	2	3	4	5	6	8	10
7	8.09%	6.62%	5.86%	5.72%	5.98%	6.03%	5.07%	**4.99%**

Table 7.9: Error rates for the corpus **UWB-06-SLR-A**. Tracked with the system presented in this thesis. Transformation: **NONE**, training files: 1246, a priori knowledge: true

dim / mixtures	1	2	3	4	5	6	8	10
2	4.87%	4.89%	4.33%	4.18%	4.17%	3.96%	3.99%	**3.89%**
3	6.89%	6.51%	7.74%	7.60%	7.88%	8.60%	10.67%	11.12%
4	7.82%	6.89%	6.98%	7.09%	7.18%	7.20%	7.05%	7.14%
5	8.38%	7.85%	5.29%	4.72%	5.12%	4.29%	4.22%	4.22%
6	10.70%	9.80%	8.26%	7.35%	7.30%	7.46%	7.82%	7.87%
7	8.09%	8.09%	5.31%	4.75%	5.20%	6.45%	5.53%	5.51%

Table 7.10: Error rates for the corpus **UWB-06-SLR-A**. Tracked with the system presented in this thesis. Transformation: **PCA**, training files: 1246, a priori knowledge: true

dim / mixtures	1	2	3	4	5	6	8	10
2	5.36%	4.28%	3.90%	**3.78%**	3.95%	4.09%	4.22%	4.18%
3	5.28%	5.23%	4.64%	5.23%	4.52%	4.62%	4.21%	4.49%
4	6.16%	6.52%	5.27%	5.81%	5.25%	5.23%	4.82%	4.57%
5	6.47%	6.36%	4.70%	5.11%	6.53%	6.52%	4.75%	4.98%
6	8.13%	7.26%	4.45%	5.28%	4.99%	6.16%	6.25%	6.15%
7	8.09%	8.64%	6.06%	5.14%	6.36%	5.72%	5.24%	4.97%

Table 7.11: Error rates for the corpus **UWB-06-SLR-A**. Tracked with the system presented in this thesis. Transformation: **HLDA**, training files: 1246, a priori knowledge: true

dim / mixtures	1	2	3	4	5	6	8	10
2	6.07%	7.47%	6.16%	6.02%	5.03%	5.59%	5.28%	8.74%
3	7.49%	8.54%	6.07%	5.51%	5.58%	5.15%	5.00%	5.27%
4	7.26%	7.53%	5.37%	5.06%	5.12%	5.35%	4.79%	5.65%
5	8.11%	7.63%	5.59%	4.99%	5.19%	5.06%	**4.18%**	4.89%
6	8.90%	7.20%	6.25%	5.59%	5.47%	5.05%	4.85%	5.04%

Table 7.12: Error rates for the corpus **UWB-06-SLR-A**. Tracked with the system presented in this thesis. Transformation: **ICA**, training files: 1246, a priori knowledge: true

dim / mixtures	1	2	3	4	5	6	8	10
2	6.17%	5.03%	7.61%	6.86%	6.60%	5.67%	5.56%	5.47%
3	5.25%	6.18%	5.82%	5.52%	5.26%	**4.69%**	5.00%	5.66%
4	6.09%	6.20%	5.78%	5.36%	5.11%	4.94%	5.07%	4.92%

Table 7.13: Error rates for the corpus **UWB-06-SLR-A**. Tracked with the system presented in this thesis. Transformation: **FA**, training files: 1246, a priori knowledge: true

dim / mixtures	1	2	3	4	5	6	8	10
2	6.58%	5.65%	5.77%	5.04%	5.13%	4.98%	4.92%	4.91%
3	5.42%	4.95%	4.92%	4.26%	4.39%	4.31%	**4.13%**	4.16%
4	5.94%	5.69%	5.60%	5.10%	4.85%	4.53%	4.73%	4.58%
5	7.90%	7.07%	6.85%	6.10%	6.75%	5.51%	5.47%	5.27%
6	7.79%	6.37%	7.28%	5.69%	6.34%	5.17%	4.75%	5.05%
7	8.08%	7.09%	7.63%	7.28%	7.30%	5.61%	5.51%	5.57%

Table 7.14: Error rates for the corpus **UWB-06-SLR-A**. Tracked with the system presented in this thesis. Transformation: **LDA**, training files: 1246, a priori knowledge: true

The impact of dimension reduction and decorrelation methods is much bigger with dataset **UWB-06-SLR-A**. This may be due to the fact that much more performers are present in this corpus which results in a greater variability of the features. This leads to a conclusion that when we are interested in tracking just one signer we should neglect the transformation of features. This will happen when we want to annotate a corpus of sign dictionary entries where the number of performers is low. We can train the models for each of the signer individually. If we want to make an application for general hand/head tracking the transformation should be applied. From this point of view the HLDA transformation into 2 dimensions performed the best. The overall better results on this dataset can be explained by the fact that all recognized signs were present in the training set. Although the testing signs were performed differently the system is able to generalize well.

Transform	Best Dim	Best no. Comps	Error Rate
NONE	7	5	4.90%
FA	2	10	7.40%
HLDA	7	10	7.62%
LDA	4	10	7.48%
PCA	5	4	7.35%
ICA	4	2	7.29%

Table 7.15: Summary of the best results for the corpus **UWB-12-SILADON**

Transform	Best Dim	Best no. Comps	Error Rate
NONE	7	5	4.99%
FA	3	6	4.69%
HLDA	2	4	3.78%
LDA	3	8	4.13%
PCA	2	10	3.89%
ICA	5	8	4.18%

Table 7.16: Summary of the best results for the corpus **UWB-06-SLR-A**

When looking at the summary of the best results in Tables 7.15 and 7.16 it can be seen that when no transformation is applied the error rates are pretty much the same. For the dataset **UWB-12-SILADON** the number of dimensions in which the system performs the best is higher than for the dataset **UWB-06-SLR-A**. This leads to the conclusion that less signs and more signers are generalized better than less signers and more signs. Generally, the more components are used the better results we obtain. This would mean that the distribution of the features is non-Gaussian and spreads across the feature space non-uniformly. This can be seen in Appendix B.

7.2 Experiment 02: Neglecting the a priori knowledge

Theoretically, the a priori knowledge should have a positive impact on the tracking success rate. To test this assumption the experiments conducted in the first trial are repeated but the a priori knowledge is not taken into account. Each test is performed only for a subset of dimensions. Usually no dimension reduction is applied to preserve the information contained in the feature vectors. The likelihood formula for these experiments is thus

$$L_k = \sum_{i=1}^{3} \log p_m(t_i|C_k(t_i)) = \sum_{i=1}^{3} \log p(\boldsymbol{T_i d_i}; \lambda) \tag{7.2}$$

Transform	dim/mixs	1	2	3	4	5	6	8	10
NONE	7	12.76%	10.97%	10.40%	11.02%	9.83%	9.52%	**9.19%**	9.37%
FA	4	15.99%	13.96%	14.25%	13.46%	11.82%	11.62%	12.05%	**11.53%**
HLDA	7	12.76%	11.69%	11.32%	10.02%	10.38%	10.59%	10.06%	**9.52%**
LDA	7	12.76%	11.06%	9.62%	**9.31%**	13.52%	12.95%	11.49%	12.14%
ICA	6	15.63%	12.24%	11.87%	11.10%	10.97%	10.85%	10.23%	**9.89%**
PCA	7	12.76%	11.29%	10.22%	10.48%	10.10%	9.47%	**9.35%**	9.45%

Table 7.17: Error rates for the corpus **UWB-12-SILADON**. Tracked with the system presented in this thesis. Training files: 1032, a priori knowledge: false

The tests show that the recognition rate dropped when no a priori knowledge is considered by approximately 4%. These tests were performed on the dataset **UWB-12-SILADON**. There is only one signer but a larger variety of signs. That means that the metric of the relative positions should be fairly consistent but the positions vary more. The next tests were performed on the dataset **UWB-06-SLR-A**.

Transform	dim/mixs	1	2	3	4	5	6	8	10
NONE	7	11.00%	10.14%	7.88%	7.63%	8.48%	8.37%	7.43%	**6.98%**
FA	4	10.89%	11.38%	9.44%	9.13%	9.19%	9.23%	8.88%	**8.53%**
HLDA	7	11.00%	11.60%	8.54%	8.29%	9.24%	8.62%	**7.88%**	7.97%
LDA	7	10.94%	10.55%	10.82%	9.65%	9.42%	**7.86%**	7.90%	8.16%
ICA	6	12.04%	10.57%	10.21%	8.79%	8.85%	8.16%	**7.17%**	7.67%
PCA	7	11.00%	10.89%	7.77%	**7.44%**	7.78%	8.50%	7.79%	7.87%

Table 7.18: Error rates for the corpus **UWB-06-SLR-A**. Tracked with the system presented in this thesis. Training files: 1246, a priori knowledge: false

The test on **UWB-06-SLR-A** show that the drop of recognition rate was smaller than on the dataset **UWB-12-SILADON**. Around 2.5%. This shows that the normalization of the data for a priori knowledge representation is signer independent. The results indicate that the variability of the trajectories in the sense of metric reduces the effect of the a priori knowledge. Adding new signers and signs will introduce more and more possible combinations of relative positions of hands. The question is whether there is a finite number of signs that would make the a priori knowledge neglectable. This would need to be experimented with huge amount of data. For datasets of reasonable size the a priori knowledge is important.

7.3 Experiment 03: Signer Independent tests

In this experiment the signer independence is tested. This test has a meaning for the dataset **UWB-06-SLR-A** only. The training data were obtained from 9 signers and were tested on 4 signers. The training and testing set were disjunct. In total 1083 video files were used for training and 487 for testing.

Transform	dim/mixs	1	2	3	4	5	6	8	10
NONE	7	6.29%	4.87%	4.35%	4.36%	4.45%	4.56%	**4.15%**	4.37%
FA	4	4.25%	5.01%	6.30%	**3.57%**	3.70%	4.17%	3.99%	3.86%
HLDA	7	6.40%	7.70%	4.48%	4.91%	4.96%	**3.91%**	4.06%	4.18%
LDA	7	6.40%	5.79%	5.68%	4.78%	4.80%	**4.77%**	5.11%	5.19%
ICA	6	6.26%	6.78%	**3.23%**	3.72%	3.49%	3.47%	3.64%	3.81%

Table 7.19: Error rates for the corpus **UWB-06-SLR-A**. Tracked with the system presented in this thesis. Training files: 1083, a priori knowledge: true, signer independent

The results show, that the system is optimized for tracking a person who has not been seen before. Provided that the sign was included in the training data although performed by another person. This shows that the normalization of features and a priori knowledge is working as intended.

7.4 Experiment 04: Model switching

In this experiment the tracking models will be trained on one dataset and used on the other. The purpose of this experiment is twofold. When the dataset **UWB-12-SILADON** is used for training there is only one performer and many signs. Also the performer is a professional which changes the dynamics of the signs. When the models are used to track the signers from **UWB-06-SLR-A** the question is whether the models will be able to handle a large number of unseen signers. This has seemed to work in the previous experiment but the signs were the same. Only the performers changed. Also the signs are performed with different dynamics. The experiment should show whether the breakdown into frame-by-frame dynamics will be able to handle the different style of signing. Also the a priori knowledge models are switched to simulate the situation when we want to track totally different data.

Transform	dim / mixs	1	2	3	4	5	6	8	10
NONE	7	8.04%	7.62%	7.11%	7.76%	**6.92%**	7.31%	7.21%	7.41%
HLDA	7	8.04%	7.95%	7.74%	6.79%	6.48%	7.08%	**6.30%**	6.76%
FA	4	7.44%	8.74%	7.92%	6.65%	7.39%	7.10%	7.15%	**6.19%**
PCA	7	7.97%	**6.26%**	6.79%	6.28%	6.91%	7.17%	6.82%	6.93%
LDA	7	8.33%	8.92%	7.48%	**6.83%**	7.23%	7.72%	7.66%	7.35%
ICA	6	8.71%	6.85%	6.97%	6.25%	**6.10%**	6.12%	6.29%	5.98%

Table 7.20: Error rates for the corpus **UWB-06-SLR-A**. Tracked with the system presented in this thesis. Training files: 1032 from dataset UWB-12-SILADON, a priori knowledge: from dataset UWB-12-SILADON

When comparing these results with the results from previous experiments we can see a drop in recognition rate. The drop is around 1% - 2%. The error rates are reasonable and the system could be used for tracking and recognition. The different transformations are comparable in the sense of error rate. This experiment shows that the frame-by-frame dynamics describe the signs well. Even though the dataset did not contain the recognized performer neither the signs he is performing the tracking still works acceptable.

Transform	dim/mixs	1	2	3	4	5	6	8	10
NONE	7	14.24%	11.84%	9.34%	8.23%	8.08%	8.19%	7.27%	**7.10%**
HLDA	7	14.24%	14.89%	10.91%	9.80%	10.84%	9.26%	**7.75%**	7.92%
FA	4	11.26%	10.64%	11.31%	10.05%	**8.09%**	9.16%	8.89%	8.64%
PCA	7	14.24%	13.15%	11.01%	9.91%	10.08%	9.42%	8.79%	**8.67%**
LDA	7	14.24%	11.48%	10.26%	**9.13%**	10.14%	9.41%	9.78%	9.30%
ICA	6	16.23%	14.34%	13.94%	10.03%	11.08%	11.15%	**8.94%**	9.85%

Table 7.21: Error rates for the corpus **UWB-12-SILADON**. Tracked with the system presented in this thesis. Training files: 1246 from dataset UWB-06-SLR-A, a priori knowledge: from dataset UWB-06-SLR-A

When the models and recognition dataset is switched the drop in recognition rate is approximately the same. The tracking of the dataset **UWB-12-SILADON** is still worse than with the other dataset. This can be expected since the signs are generally more complex in UWB-12-SILADON. The a priori knowledge seems to work again as intended. This again shows that the normalization is generalizing well.

7.5 Experiment 05: Lower number of training data

In this experiment the tracking models will be trained using a lower number of training data. The test was be performed for the dataset **UWB-06-SLR-A**. In the first experiment 1246 of the 1641 video files were used for the training. That makes approximately 76% of training data and 24% of unseen data in the testing set. In this experiment 653 files were used for training which accounts for approximately 39% of training data and 61% of unseen data. The expectation is that the error rate will be bigger. The question is how much worse will be the recognition rate?

Transform	dim/mixs	1	2	3	4	5	6	8	10
NONE	7	7.73%	6.39%	6.16%	6.01%	5.89%	5.94%	**5.28%**	5.30%
HLDA	7	7.73%	7.72%	6.76%	6.53%	6.23%	6.14%	5.40%	**5.11%**
FA	4	10.89%	11.38%	9.44%	9.13%	9.19%	9.23%	8.88%	**8.53%**
PCA	7	7.73%	7.21%	**5.35%**	5.85%	6.08%	6.37%	5.61%	5.62%
ICA	6	7.76%	5.80%	6.76%	5.74%	**5.06%**	5.21%	5.23%	5.46%

Table 7.22: Error rates for the corpus **UWB-06-SLR-A**. Tracked with the system presented in this thesis. Training files: 653, a priori knowledge: true

The drop in the recognition rate is present as expected. It is around 2% - 5% depending on the feature transformation used. Still the results are very reasonable although the amount of data is reduced almost two times. FA gives the worst results. It seems that FA will be affected by the number of training data the most since in the other experiments it performed well in some cases outperforming other transformations. The results indicate that the mode components are used in the GMMs the better.

Chapter 8

Conclusion

This thesis handles the problem of body parts tracking in a SL scenario. Although the problem is being addressed by many researchers it is still not conclusively resolved. Partial problems are handled and usually some limitations are assumed to help the researcher create a useful application to help the community of deaf or make a progress in the scientific field of human computer interaction.

I will recap the goals of the dissertation and discuss them individually.

- The design and implementation of a system capable of tracking body parts with explicit occlusion model

The main goal of this thesis was to create a tracking system capable of tracking body parts in a defined scenario. The system is extensively described in Chapter 6. The system uses a distance measure represented by a probability model. The distance is computed between features extracted from a known body part in the previous frame and an unidentified body part in the present frame. The assumption is that the features will change slowly for one body part which results in small distances. The system is unique in the way how it handles occlusions of body parts. The occlusions are modeled explicitly and thus no special effort is needed to maintain the non-occluded body parts. This means that when two or more objects become occluded the system analyses this situation and tracks the new object as a composition of more objects. There is thus no need to approximate the unseen pixels so that the tracking can go on after the occlusion. To my knowledge such system was not used for tracking of body parts before.

- The input will be obtained from visual system providing color information
- The tracking will not use any markers

The system does not use any kind of markers. The performance was tested on two datasets. One is composed of one signer performing each sign one time (**UWB-12-SILADON**) the other is composed of many signers signing the signs with repetitions (**UWB-06-SLR-A**).

- The tracking framework will be probabilistic

The system is defined in a probabilistic framework. The small distances of features extracted from the same object in consecutive frames are represented by high probability. This approach has its the benefits. The models can be trained on a large set of performers to yield a background model. Then this model can be adopted to a new signer using standard adaptation methods. Another advantage is the possibility to fuse the tracking probabilities with other probability models, such as a predetermined movement model or language model in the case of continuous sign language. In this thesis the tracking probabilities are combined with a priori knowledge represented as a probability model.

- The obtained features will be used for recognition and categorization

At first the problem was approached in general but soon the need for the limitations emerged. The main issue was to define the problem. A useful application for the purpose of education was designed to help people learn the sign language. In cooperation with my colleagues a sign language on-line dictionary was developed available at `znaky.zcu.cz`. For the purpose of the on-line dictionary thousands of isolated signs were recorded. An application was developed to extract only the signs segments from a long recording automatically. The application takes the user through the whole process of video editing. A very short summary of the application is published in [82]. My work was on skin-color modeling and segmentation when provided with example skin-colors, and detecting a starting and ending pose of a sign to make assumptions about the signs boundaries. When the isolated signs are prepared and published a means of searching among them is necessary. The tracked body parts are used to create the categorization automatically. The categories can then be used as a searching cue. For these purposes the problem of body parts tracking was defined in controlled environment utilizing constant lightning, and non-skin-colored clothes with long sleeves. The results of categorization are available in Section 6.7 in Table 6.6. The features were also tested in a SLR system. The results are shown again in Section 6.7. Two approaches were taken. One used the tracking features directly (Table 6.1) and the other used LBP for texture description combine with ASM parameters (Table 6.4).

- Find a minimal set of features using methods of dimension reduction and decorrelation

Several methods for reducing the dimension of the feature vector were tested. Namely PCA, ICA, FA, LDA, and HLDA. Also the best parameters of the method are looked for. The performance is measured in the domain of the features rather than recognition rate. This is due to the usage of the features for automatic categorization. A recognizer can recognize the sign successfully even if the body parts are not tracked correctly during the recording. This may occur if only a temporary error is present (tracking fails in one frame). Most recognizers can handle such an isolated error as noise. The performance measured in the domain of features does not make this problem disappear.

8.1 Experiments discussion

Several experiments were performed on the two datasets **UWB-12-SILADON** and **UWB-06-SLR-A**. In the first case the system performed the best when no dimension reduction was used. The optimal number of mixtures in GMM was found to be 5. This can be explained so that the model is describing the one performer in the original feature space the best way. The number of mixtures is not that important if we use sufficient mixtures. When comparing the results with the later dataset the PCA transformation yielded the best results. This can be explained by the large number of signers. PCA will generalize the models to be able to cope with the larger variety. If we neglect the a priori knowledge we obtain lower accuracy as expected. If we switch the models as in Experiment 7.4 the performance is still good even though not as good as when the model is trained on a sample of the recognized signers and signs. This leads to the conclusion that for the best results the training of the models should be application dependent. If we have a problem of a closed set recognition the training data should be obtained from this set. Otherwise we should use as many training examples as possible with a following decorrelation of features. If we want to recognize just one performer we should use the features directly without the application of any transformation.

Overall the success rate is acceptable; somewhere between 90% - 95%. Since this is measured over the total number of frames it does not necessarily mean that one faulty tracked frame makes the whole tracked sequence unusable. This can be seen in the experiment on categorization where several categories resulted in 100% success rate.

8.2 Future Work

Despite a great effort some of the issues discussed in this thesis are left unresolved. There are notable boundaries that could be perceived as flaws. I am referring mainly to the limited conditions in which the tracking works. It is highly dependent on the segmentation of body parts. If the segmentation fails then the tracking will be faulty. This is an unwelcome feature in a tracker. Also the tracking requires that one object does not change much during the recording. This means that the segmentation needs to be very robust. The dependency on the segmentation is an issue that should be handled in the feature.

The possibility of adapting the probability models is present but needs to be handled. The question is whether there will be enough adaptation data from the signer and how to obtain them. Then different adaptation methods need to be tested.

Appendix A

Algorithm

The application loop is as follows:

1. Resolve the program arguments
2. Load proper models and video file
3. Prepare the proper output format
4. Initialize trackers
5. Obtain an image from the video stream
6. Segment the image using the provided skincolor model and compute the quality of the segmentation using Algorithm 1

```
input  : Segmented image
output: Quality of segmentation
objects ← DetectObjects(image);
$c_1$ ← number of objects;
objects ← FilterObjects(objects);
$c_2$ ← number of filtered objects;
object_ratio ← $c_1$ / $c_2$;
if object_ratio $< 10$ or $c_2 > 3$ or $c_2 < 2$ then
    quality ← low;
else
    quality ← good;
end
```

Algorithm 1: Skin color model quality determination

```
input  : Processed video
output: Skincolor model

Detect faces in the video;
if number of faces in an image== 1 then
    bestFace ← face ;                          // store the face as a template
end
for every frame do
    Determine which detected object is a face by comparing it to the bestFace;
    Convert the image of the video to Luv color space;
    Use mean-shift segmentation on the colors in the Luv image;
    Segment the Luv image to color regions according to the resulting clusters from
    mean-shift;
    // Analyze which clusters (colors) occupy the face region.
    Make a histogram of the color clusters in the face region;
    The cluster with the biggest count is chosen as the representative color of the face;
    for each cluster do
        if the color that the cluster represents is close to the representative color then
            Add the color to the color list
        end
    end
    for each color in color list do
        Increase the value in the skincolor model for the color;
        // The value represents the likelihood of that color to be
           skincolor
    end
end
```

Algorithm 2: Skin color model training

7. If **quality** is good, go to point 8, else compute a new skincolor model using the Algorithm 2

8. We have a skincolor model ready and begin to process the video using Algorithm 3

```
input  : Processed video
output: Tracked objects
for each image in video do
    // Segment the image using the skincolor model
    segmented image ← SkinColorDetection(image);
    Use median blur and morphological operations on the segmented image;
    objects ← DetectObjects(segmented image);
    objects ← FilterObjects(objects);
    // Use trackers to compute the likelihood of each object being the
       tracked one
    EvaluateTrackers(objects);
    // Based on the reported likelihoods assign an object to each tracker
    AssignObjects;
    Save the features for this frame;
end
```

Algorithm 3: Tracking process

9. Clean up! Exit

We will also introduce algorithms for evaluating the object using the trackers (Alg. 4) and for assigning objects (Alg. 5).

```
input  : Object
output: Tracker Evaluation for given Object
for each tracker do
    for each object do
        Allocate a new trackerEvaluation structure;
        // Evaluate the object using the tracker and fill trackerEvaluation
        if tracker is in initialization mode then
            if object lies in the tracker window then
                tracker ← object;
            end
        else
            // Compute the likelihood of the object being tracked by the
               tracker
            features ← $\sqrt{(\text{objectCenter} - \text{trackerCenter})^2}$ ;        // 2 dimensions
            features ← templateMatching(object) ;                    // 1 dimension
            features ← matchShapes(object, tracker) ;                // 1 dimension
            features ← boundingBoxAreaDifference(object, tracker) ;          // 1
            dimension
            features ← objectVelocity − trackerVelocity ;            // 2 dimensions
            trackerEvaluation ← computeLogLike(features, reductionMethod, model)
        end
        Save the trackerEvaluation;
    end
end
```

Algorithm 4: Evaluating the objects by tracker

```
input  : Trackers' evaluations
output: Assigned trackers
for each tracker do
    if tracker is in initialization then
        assignObject(tracker, object) // Where the object is the one from
            initialization
        continue;
    else
        Save the relevant information about the tracker - relative bounding box area,
        log-likelihood for each object, ...
    end
end
for all possible configurations do
    // The possible configurations are based on number of objects
    Compare the investigated configuration with observed data based on the relative
    change of the bounding box area. The code for this is very easy, but takes a lot of
    space. Please refer to the text for details.
    if Investigated configuration is in compliance with observed data then
        Compute the total log-likelihood of the configuration based on tracker
        evaluation.
        if The total log-likelihood is the best then
            Store it as the result.
        end
    end
end
```

Algorithm 5: Deciding which object is tracked by which tracker

Appendix B

GMM evolution

These Figures show the evolution of the GMM based on the number of components for different transformations. The visualization is done utilizing the two dimensional models. The axis are the directions of the transformed feature space and are a linear combination of the original 7 dimensional feature space summarized in Equations 6.20 - 6.26.

B.1 PCA - Good models for the right hand

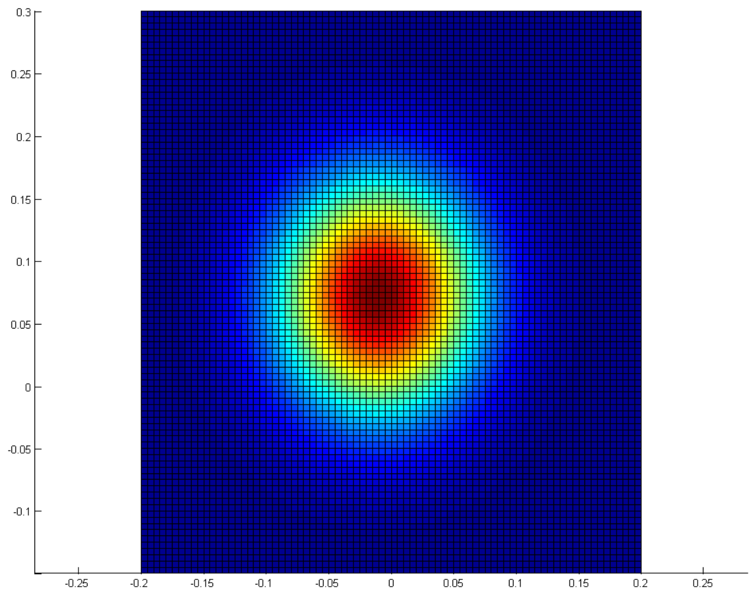

Figure B.1: Trained GMM for the RIGHT hand, trained with GOOD features and PCA transformation. One mixture in GMM

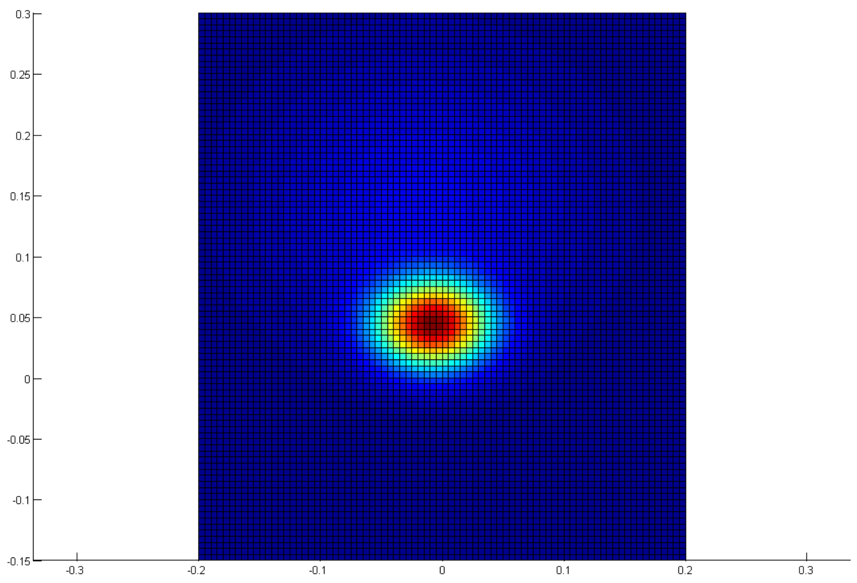

Figure B.2: Trained GMM for the RIGHT hand, trained with GOOD features and PCA transformation. Two mixtures in GMM

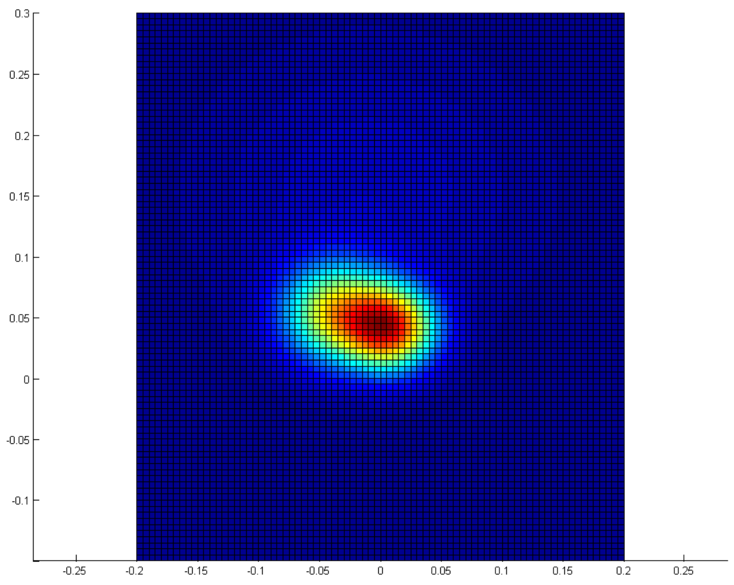

Figure B.3: Trained GMM for the RIGHT hand, trained with GOOD features and PCA transformation. Three mixtures in GMM

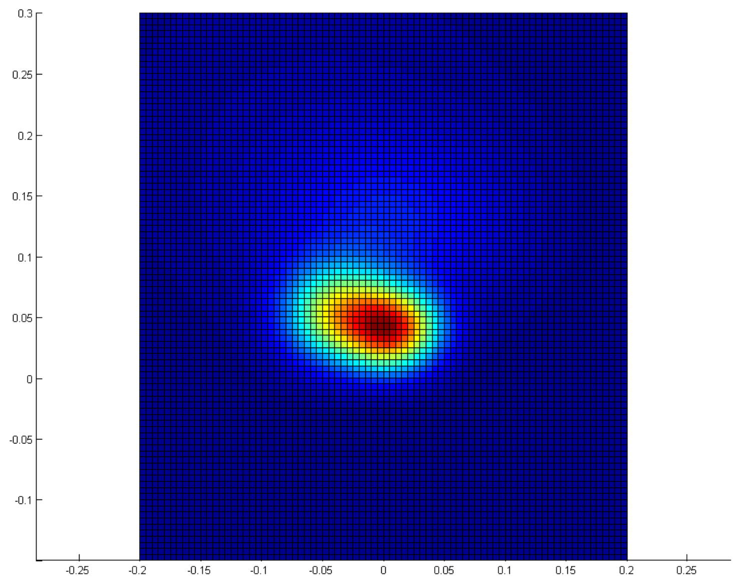

Figure B.4: Trained GMM for the RIGHT hand, trained with GOOD features and PCA transformation. Four mixtures in GMM

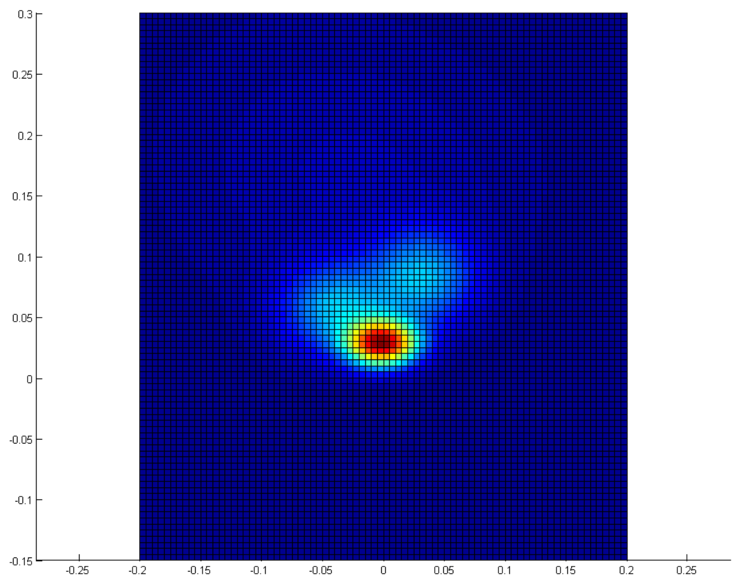

Figure B.5: Trained GMM for the RIGHT hand, trained with GOOD features and PCA transformation. Five mixtures in GMM

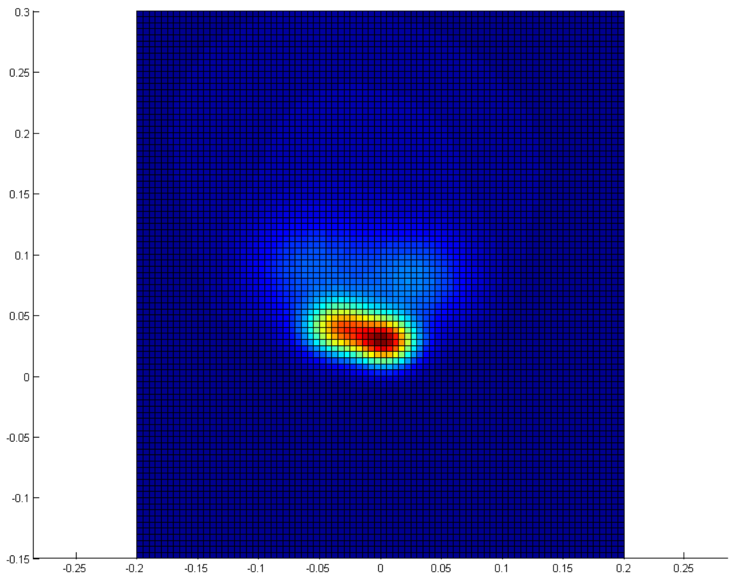

Figure B.6: Trained GMM for the RIGHT hand, trained with GOOD features and PCA transformation. Six mixtures in GMM

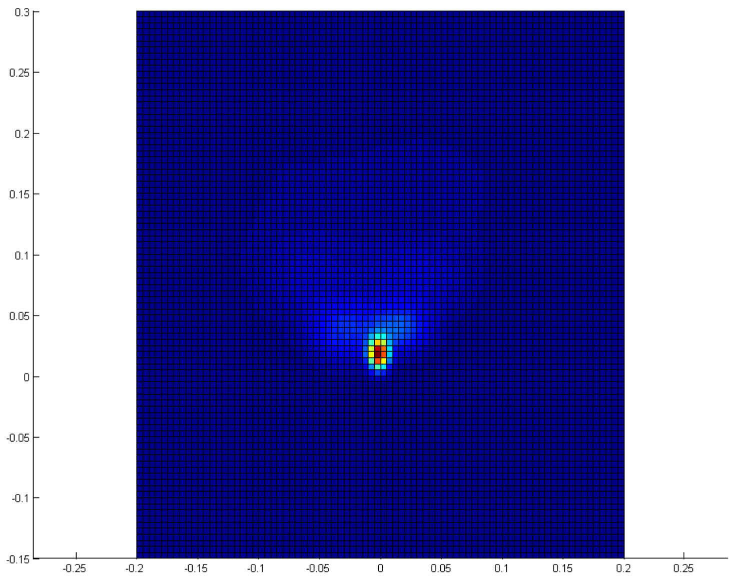

Figure B.7: Trained GMM for the RIGHT hand, trained with GOOD features and PCA transformation. Eight mixtures in GMM

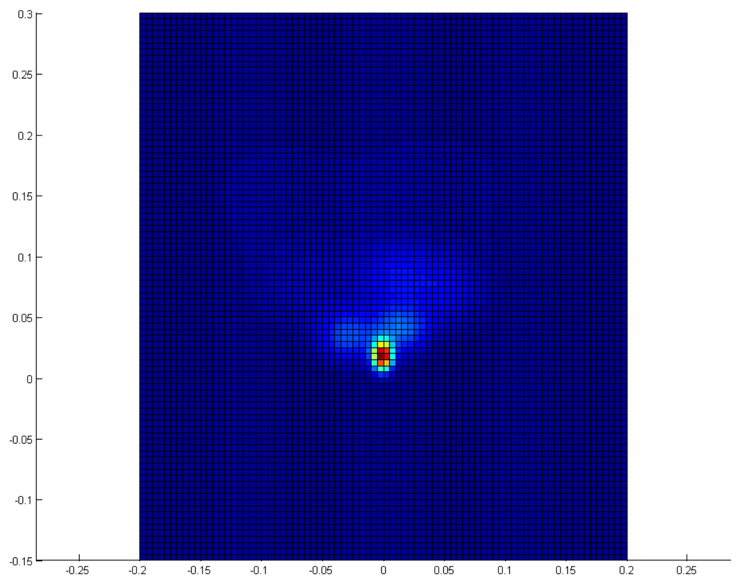

Figure B.8: Trained GMM for the RIGHT hand, trained with GOOD features and PCA transformation. Ten mixtures in GMM

It can be observed that majority of the mass of the GMM is close to the origin of the coordinate system. Since the features represent relative differences of the features it can be expected. As the number of mixtures grows the largest peak is slowly moving towards the origin of the system and the rest of the modes describe the lightly populated areas of the feature space.

B.2 PCA - After occlusion models for the right hand

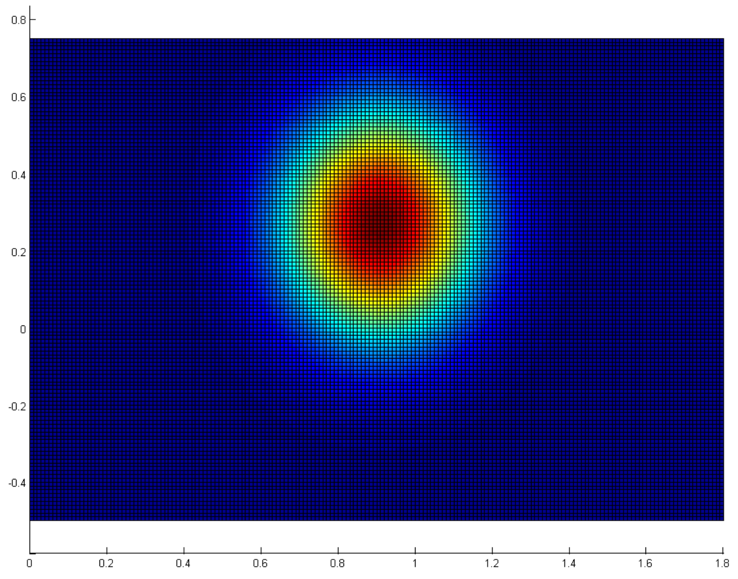

Figure B.9: Trained GMM for the RIGHT hand, trained with AFTER OCCLUSION features and PCA transformation. One mixture in GMM

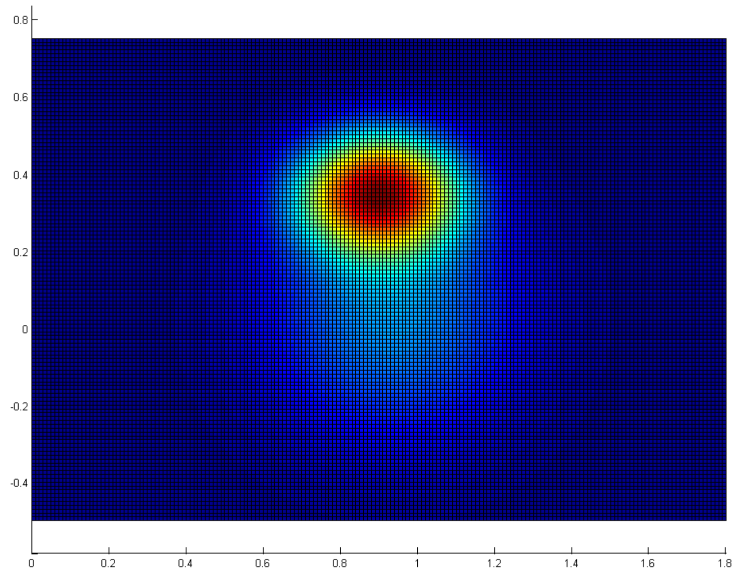

Figure B.10: Trained GMM for the RIGHT hand, trained with AFTER OCCLUSION features and PCA transformation. Two mixtures in GMM

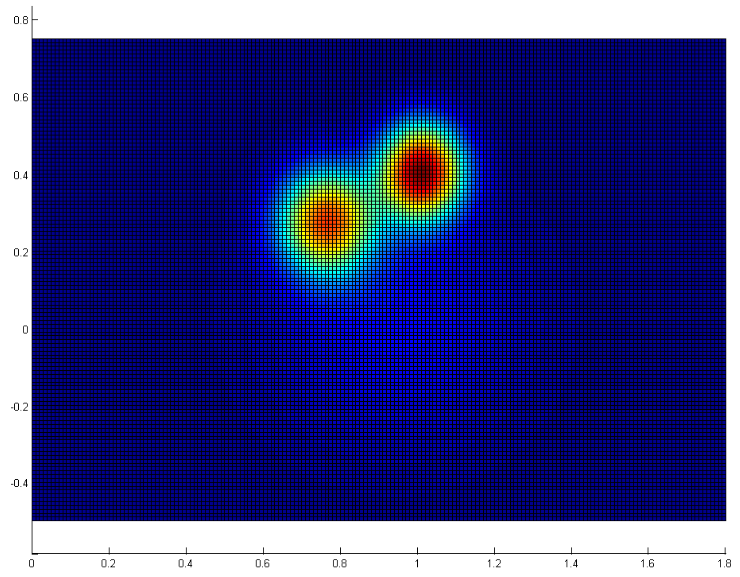

Figure B.11: Trained GMM for the RIGHT hand, trained with AFTER OCCLUSION features and PCA transformation. Three mixtures in GMM

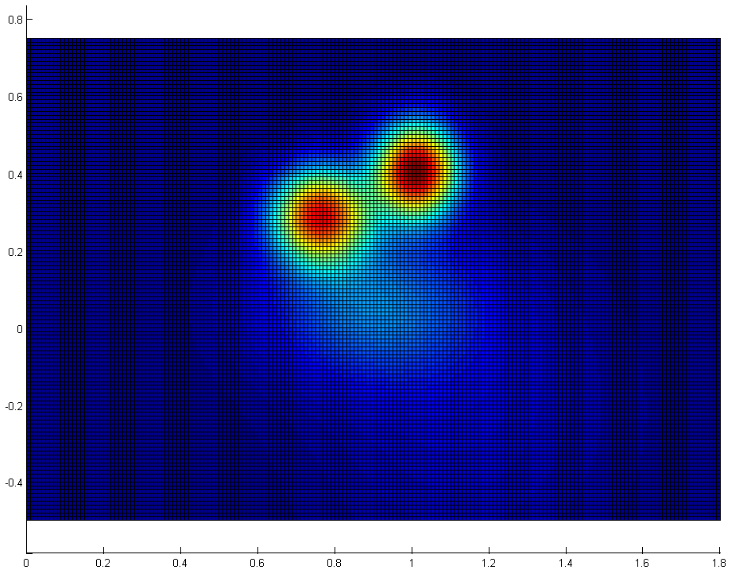

Figure B.12: Trained GMM for the RIGHT hand, trained with AFTER OCCLUSION features and PCA transformation. Four mixtures in GMM

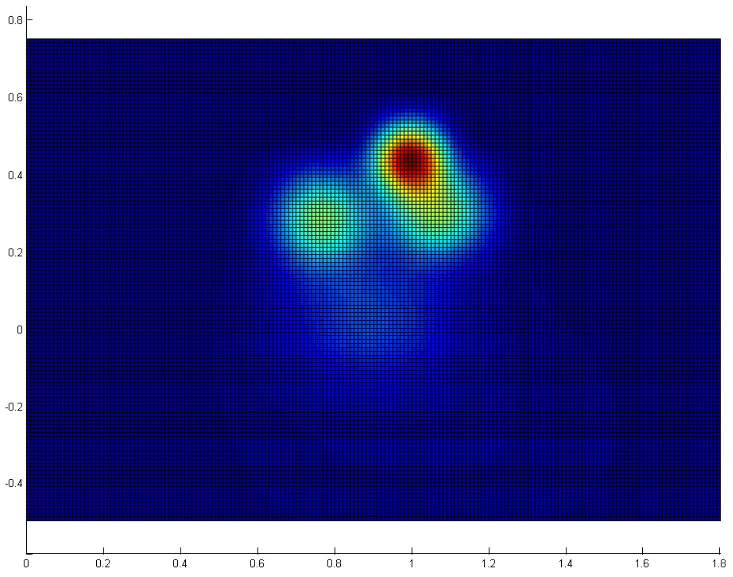

Figure B.13: Trained GMM for the RIGHT hand, trained with AFTER OCCLUSION features and PCA transformation. Five mixtures in GMM

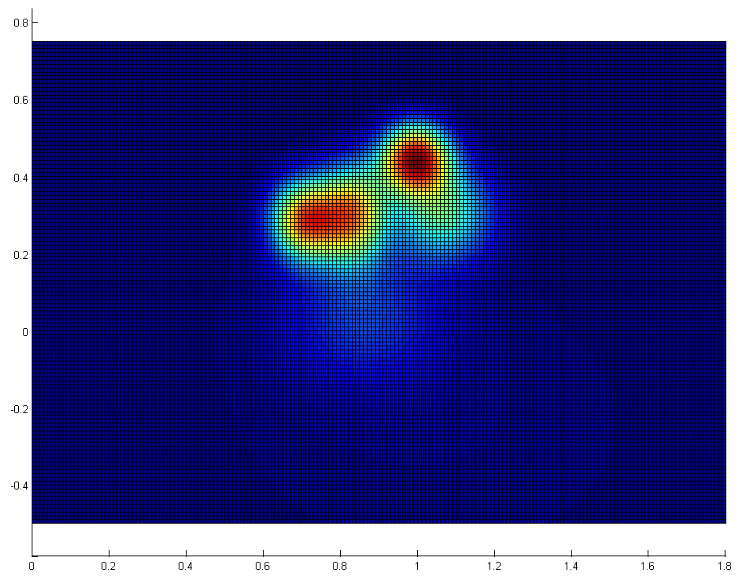

Figure B.14: Trained GMM for the RIGHT hand, trained with AFTER OCCLUSION features and PCA transformation. Six mixtures in GMM

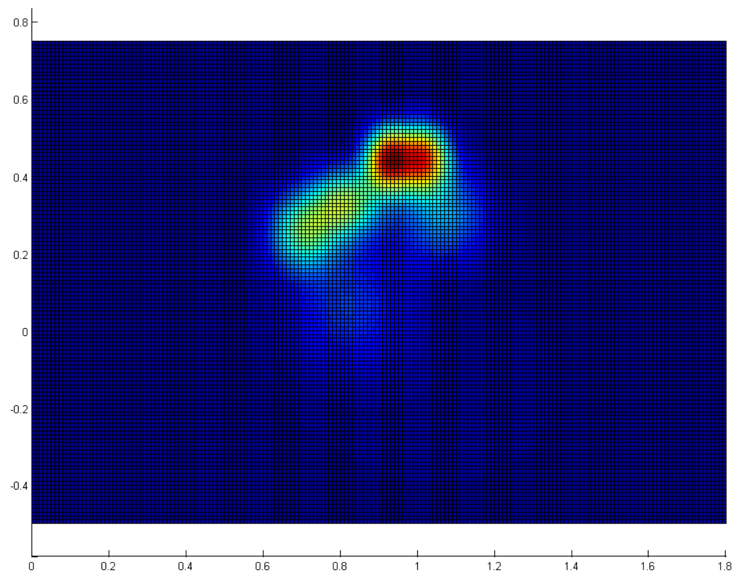

Figure B.15: Trained GMM for the RIGHT hand, trained with AFTER OCCLUSION features and PCA transformation. Eight mixtures in GMM

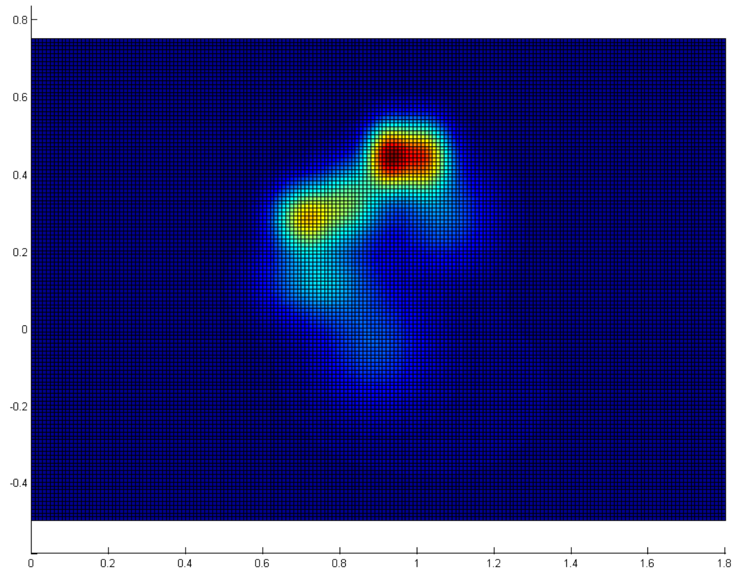

Figure B.16: Trained GMM for the RIGHT hand, trained with AFTER OCCLUSION features and PCA transformation. Ten mixtures in GMM

The after occlusion model behaves differently than the good model. The mean of the mixture is located further away from the origin of the system. This is again as expected since after occlusion the objects change dramatically. Also it seems that there are more modes present in the mixture. With higher number of components the feature space is described more precisely. In the case when 10 components are used a third and forth mode start to appear although with lower probability values.

Bibliography

[1] J. Bonet, J. Garrido, and L. Portero, *Reducción de las letras y arte para enseñar a hablar los mudos*, ser. Biblioteca moderna de filosofía y ciencias sociales. Francisco Beltran, Libreria Española y Extranjera, 1930.

[2] V. I. Pavlovič, R. Sharma, and T. S. Huang, "Visual interpretation of hand gestures for human-computer interaction: A review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, pp. 677–695, 1997.

[3] D. Sturman and D. Zetlzer, "A survey of glove-based input." *IEEE Computer Graphics and Applications*, vol. 14, pp. 30–39, 1994.

[4] S. Malassiotis, N. Aifanti, and M. G. Strintzis, "A gesture recognition system using 3d data," *3D Data Processing Visualization and Transmission, International Symposium on*, vol. 0, p. 190, 2002.

[5] K. Fujimura and X. Liu, "Sign Recognition using Depth Image Streams," in *FGR06*, 2006, pp. 381–386.

[6] S. Ong and S. Ranganath, "Automatic sign language analysis: A survey and the future beyond lexical meaning." *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 27, pp. 873–891, 2005.

[7] A. Macurová, "Proč a jak zapisovat znaky českého znakového jazyka." *Speciální pedagogika*, vol. 6, pp. 5–20, 1996.

[8] P. Campr, M. Hrúz, and M. Železný, "Design and recording of czech sign language corpus for automatic sign language recognition." *Proc. of Interspeech 2007*, pp. 678–681, 2007.

[9] M. Železný, P. Campr, Z. Krňoul, and M. Hrúz, "Design of a multi-modal information kiosk for aurally handicapped people." *Proc. of SPECOM 2007*, pp. 751–755, 2007.

[10] C. Wang, S. Shan, and W. Gao, "An approach based on phonemes to large vocabulary chinese sign language recognition." *Proceedings of the Fifth IEEE International Conference on Automatic Face and Gesture Recognition*, p. 411, 2002.

[11] O. Crasborn and I. Zwitserlood, "The corpus ngt: an online corpus for professionals and laymen." *Proceedings of the sixth Conference on Language Resources and Evaluation (LREC 2008)*, pp. 44–49, 2008.

[12] L. Leeson and B. Nolan, "Digital deployment of the signs of ireland corpus in elearning." *Proceedings of the sixth Conference on Language Resources and Evaluation (LREC 2008)*, pp. 112–122, 2008.

[13] H. Freeman, “On the encoding of arbitrary geometric configuration.” *IRE Transactions on Electronic Computers.*, vol. EC-10, pp. 260–268, 1961.

[14] M. K. Hu, “Visual pattern recognition by moment invariants.” *IRE Trans. Information Theory*, vol. 8, pp. 179–187, 1962.

[15] A. K. Jain, “Fundamentals of digital image processing,” *Prentice-Hall, Englewood Cliffs, NJ*, 1989.

[16] W. K. Prat, “Digital image processing,” *Wiley, New York, 2nd edition*, 1991.

[17] J. Flusser, “Moment invariants in image analysis,” *Poceedings of World Academy of Science, Engineering and Technology*, vol. 11, pp. 196–201, 2006.

[18] M. R. Teague, “Image analysis via the general theory of moments,” *J. of Optical Soc. of America*, vol. 70, pp. 920–930, 1980.

[19] A. Wallin and O. Kubler, “Complete sets of complex zernike moment invariants and the role of the pseudoinvariants,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 17, pp. 1106–1110, 1995.

[20] F. Zernike, “Beugungstheorie des schneidenverfahrens und seiner verbesserten form, der phasenkontrastmethode,” *Physica*, vol. 1, pp. 689–704, 1934.

[21] D. W. Paglieroni and A. K. Jain, “Control point transforms for shape representation and measurement.” *Computer Vision, Graphics, and Image Processing.*, vol. 42, pp. 87–111, 1988.

[22] Y. Wang and E. K. Teoh, “2d affine-invariant contour matching using b-spline model.” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 29, pp. 1853–1858, 2007.

[23] Y. Ma, F. Pollick, and W. T. Hewitt, “Using b-spline curves for hand recognition.” *Proc. of the 17th International Conference on Pattern Recognition (ICPR’04)*, vol. 3, pp. 274–277, 2004.

[24] F. Y. Shih, Y. F. Camel, and K. Zhang, “Multi-view face identification and pose estimation using b-spline interpolation.” *Information Sciences*, vol. 169, pp. 189–204, 2005.

[25] T. F. Cootes, T. C. J., and A. Lanitis, “Active shape models: Evaluation of a multi-resolution method for improving image search.” *Proc. of the British Machine Vision Conference.*, vol. 1, pp. 327–336, 1994.

[26] T. F. Cootes, *An introduction to Active Shape Models.* Ed.R.Baldock and J. Graham, Oxford University Press, 2000, pp. 223–248.

[27] T. F. Cootes, G. J. Edwards, and C. J. Taylor, “Active appearance models,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, pp. 681–685, 2001.

[28] S. Theodoridis and K. Koutroumbas, *Pattern Recognition, Third Edition.* Academic Press, February 2006. [Online]. Available: http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20&amp;path=ASIN/0123695317

[29] P. Kakumanu, S. Makrogiannis, and N. Bourbakis, “A survey of skin-color modeling and detection methods,” *Pattern Recogn.*, vol. 40, no. 3, pp. 1106–1122, Mar. 2007. [Online]. Available: http://dx.doi.org/10.1016/j.patcog.2006.06.010

[30] V. Vezhnevets, V. Sazonov, and A. Andreeva, “A survey on pixel-based skin color detection techniques,” in *IN PROC. GRAPHICON-2003*, 2003, pp. 85–92.

[31] T. Ojala, M. Pietikäinen, and D. Harwood, “A comparative study of texture measures with classification based on featured distributions,” *Pattern Recognition*, vol. 29, no. 1, pp. 51–59, Jan. 1996.

[32] T. Ojala, M. Pietikainen, and T. Maenpaa, “Multiresolution gray-scale and rotation invariant texture classification with local binary patterns,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 7, pp. 971 –987, jul 2002.

[33] M. Sonka, V. Hlavac, and R. Boyle, *Image Processing, Analysis, and Machine Vision.* Thomson-Engineering, 2007.

[34] E. C. Hildreth, “Computation underlying the measurement of visual motion.” *Artificial Intelligence*, vol. 23, pp. 309–354, 1984.

[35] B. K. P. Horn and S. B. G., “Determining optical flow.” *Artificial Intelligence*, vol. 17, pp. 185–203, 1981.

[36] P. J. Burt, J. R. Bergen, R. Hingorani, R. J. Kolczynski, W. A. Lee, A. Leung, J. Lubin, and J. Shvaytser, “Object tracking with a moving camera; an application of dynamic motion analysis.” *IEEE Workshop on Visual Motion*, 1989.

[37] K. Fukunaga and L. Hostetler, “The estimation of the gradient of a density function, with applications in pattern recognition.” *Information Theory, IEEE Transactions*, vol. 21, pp. 32–40, 1975.

[38] G. R. Bradski, “Computer vision face tracking as a component of a perceptual user interface.” *In Workshop on Applications of Computer Vision*, pp. 214–219, 1998.

[39] M. Isard and A. Blake, “Condensation–conditional density propagation for visual tracking.” *International Journal of Computer Vision*, vol. 29, pp. 5–28, 1998.

[40] S. Ulman and R. Basri, “Recognition by linear combinations of models.” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 13, pp. 922–1006, 1991.

[41] J. Koenderink and A. Van Doorn, “Affine structure from motion.” *Journal of Optical Soc. of America A.*, vol. 8, pp. 337–386, 1991.

[42] A. Blake, R. Curwen, and A. Zisserman, “A framework for spatio-temporal control in the tracking of visual contours.” *Int. Journal of Computer Vision*, vol. 11, pp. 127–145, 1993.

[43] A. Blake, M. Isard, and D. Reynard, “Learning to track the visual motion of contours.” *Artificial Intelligence*, vol. 78, pp. 101–134, 1995.

[44] T. Cootes, C. Taylor, A. Lantis, D. Cooper, and J. Graham, “Building and using flexible models incorporating grey-level information.” *Proc. 4th Int. Conf. on Computer Vision*, pp. 242–246, 1993.

[45] A. Baumberg and D. Hogg, “Learning flexible models from image sequences.” *Proc. 3rd European Conference on Computer Vision*, pp. 299–308, 1994.

[46] D. Reynard, A. Wildenberg, A. Blake, and J. Marchant, "Learning dynamics of complex motions from image sequences." *Proc. 4th European Conference on Computer Vision*, pp. 357–368, 1996.

[47] L. Ding and A. M. Martinez, "Three-dimensional shape and motion reconstruction for the analysis of american sign language," in *CVPRW '06: Proceedings of the 2006 Conference on Computer Vision and Pattern Recognition Workshop.* Washington, DC, USA: IEEE Computer Society, 2006, pp. 146–151.

[48] H. Fillbrandt, S. Akyol, and K. Kraiss, "Extraction of 3d hand shape and posture from image sequences for sign language recognition," in *AMFG '03: Proceedings of the IEEE International Workshop on Analysis and Modeling of Faces and Gestures.* Washington, DC, USA: IEEE Computer Society, 2003, pp. 181–186.

[49] M. Kato, Y.-W. Chen, and G. Xu, "Articulated hand tracking by pca-ica approach." *Proceedings of the seventh Conference on Automatic Face and Gesture Recognition.*, pp. 329–334, 2006.

[50] T. Coogan and A. Sutherland, "Transformation invariance in hand shape recognition," in *ICPR '06: Proceedings of the 18th International Conference on Pattern Recognition.* Washington, DC, USA: IEEE Computer Society, 2006, pp. 485–488.

[51] R. Rosales, V. Athitsos, L. Sigal, and S. Sclaroff, "3d hand pose reconstruction using specialized mappings," in *In ICCV*, 2001, pp. 378–385.

[52] V. Athitsos and S. Sclaroff, "Estimating 3d hand pose from a cluttered image," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2003, pp. 432–439.

[53] H. Guan, J. S. Chang, L. Chen, R. S. Feris, and M. Turk, "Multi-view appearance-based 3d hand pose estimation," in *CVPRW '06: Proceedings of the 2006 Conference on Computer Vision and Pattern Recognition Workshop.* Washington, DC, USA: IEEE Computer Society, 2006, p. 154.

[54] W. Chen, R. Fujiki, D. Arita, and R. ichiro Taniguchi, "Real-time 3d hand shape estimation based on image feature analysis and inverse kinematics," in *ICIAP '07: Proceedings of the 14th International Conference on Image Analysis and Processing.* Washington, DC, USA: IEEE Computer Society, 2007, pp. 247–252.

[55] Y. Wu, J. Y. Lin, and T. S. Huang, "Capturing natural hand articulation," in *In ICCV*, 2001, pp. 426–432.

[56] A. El-Sawah, C. Joslin, N. D. Georganas, and E. M. Petriu, "A framework for 3d hand tracking and gesture recognition using elements of genetic programming," *Computer and Robot Vision, Canadian Conference*, vol. 0, pp. 495–502, 2007.

[57] N. Tanibata, N. Shimada, and Y. Shirai, "Extraction of hand features for recognition of sign language words," in *In International Conference on Vision Interface*, 2002, pp. 391–398.

[58] N. Liu and B. C. Lovell, "Hand gesture extraction by active shape models," in *DICTA '05: Proceedings of the Digital Image Computing on Techniques and Applications.* Washington, DC, USA: IEEE Computer Society, 2005, p. 10.

[59] Y. Hamada, N. Shimada, and Y. Shirai, “Hand shape estimation using image transition network,” in *HUMO ’00: Proceedings of the Workshop on Human Motion (HUMO’00).* Washington, DC, USA: IEEE Computer Society, 2000, p. 161.

[60] S. Schreer, O.; Ngongang, “Real-time gesture recognition in advanced videocommunication services,” in *14th International Conference on Image Analysis and Processing*, 2007.

[61] E.-J. Holden and R. Owens, “Recognising moving hand shapes,” in *ICIAP ’03: Proceedings of the 12th International Conference on Image Analysis and Processing.* Washington, DC, USA: IEEE Computer Society, 2003, p. 14.

[62] P. Buehler, M. Everingham, and A. Zisserman, “Employing signed tv broadcasts for automated learning of british sign language,” in *4th Workshop on the Representation and Processing of Sign Languages: Corpora and Sign Language Technologies*, 2010, pp. 33–40.

[63] P. F. Felzenszwalb and D. P. Huttenlocher, “Pictorial structures for object recognition,” *Int. J. Comput. Vision*, vol. 61, no. 1, pp. 55–79, Jan. 2005. [Online]. Available: http://dx.doi.org/10.1023/B:VISI.0000042934.15159.49

[64] B. Stenger, P. Mendonca, and R. Cipolla, “Model-based 3d tracking of an articulated hand,” in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 2, 2001, pp. II–310 – II–315 vol.2.

[65] B. Stenger, A. Thayananthan, P. H. Torr, and R. Cipolla, “Model-based hand tracking using a hierarchical bayesian filter,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, pp. 1372–1384, 2006.

[66] R. S. Bucy and K. D. Senne, “Digital synthesis of non-linear filters,” *Automatica*, vol. 7, no. 3, pp. 287–298, May 1971. [Online]. Available: http://dx.doi.org/10.1016/0005-1098(71)90121-X

[67] A. A. Argyros and M. I. A. Lourakis, “Real-time tracking of multiple skin-colored objects with a possibly moving camera.” in *ECCV (3)*, ser. Lecture Notes in Computer Science, T. Pajdla and J. Matas, Eds., vol. 3023. Springer, 2004, pp. 368–379. [Online]. Available: http://dblp.uni-trier.de/db/conf/eccv/eccv2004-3.html#ArgyrosL04

[68] N. K. Iason Oikonomidis and A. Argyros, “Efficient model-based 3d tracking of hand articulations using kinect,” in *Proceedings of the British Machine Vision Conference.* BMVA Press, 2011, pp. 101.1–101.11, http://dx.doi.org/10.5244/C.25.101.

[69] M. Hrúz, P. Campr, and M. Železný, “Semi-automatic annotation of sign language corpora,” in *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC’08)*, 2008.

[70] O. Aran, I. Ari, P. Campr, M. Hrúz, D. Kahramaner, and S. Parlak, “Speech and sliding text aided sign retrieval from hearing impaired sign news videos.” Louvain-la-Neuve: TELE, Universite catholique de Louvain, 2007, pp. 37–49. [Online]. Available: http://www.kky.zcu.cz/en/publications/AranO_2007_Speechandsliding

[71] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1, 2001, pp. I–511 – I–518 vol.1.

[72] S. Suzuki and K. Be, "Topological structural analysis of digitized binary images by border following," *Computer Vision, Graphics, and Image Processing*, vol. 30, no. 1, pp. 32–46, Apr. 1985. [Online]. Available: http://dx.doi.org/10.1016/0734-189X(85)90016-7

[73] A. Padrta and J. Vaněk, "Introduction of improved uwb speaker verification system," *Lecture Notes in Artificial Intelligence*, pp. 364–370, 2005. [Online]. Available: http://www.kky.zcu.cz/en/publications/PadrtaA_2005_Introductionof

[74] M. J. F. Gales, "Semi-tied covariance matrices for hidden markov models," *IEEE Transactions on Speech and Audio Processing*, vol. 7, no. 3, pp. 272–281, 1999.

[75] B. D. Ripley, *Pattern Recognition and Neural Networks.* Cambridge University Press, 1996. [Online]. Available: http://books.google.cz/books?id=2SzT2p8vP1oC

[76] J. Trojanová, P. Císař, and M. Železný, "Combined visual parametrization for automatic lip-reading." *Proceeding of the 4th Joint Workshop on Multimodal Interaction and Related Machine Learning Algorithm MLMI 07.*, 2007.

[77] J. Trmal, M. Hrúz, J. Zelinka, P. Campr, and L. Müller, "Feature space transforms for czech sign-language recognition," in *In ICSLP*, 2008.

[78] M. Hrúz, J. Trojanová, and M. Železný, "Local binary pattern based features for sign language recognition," *Pattern Recognition and Image Analysis*, pp. 398–401, 2011. [Online]. Available: http://www.kky.zcu.cz/en/publications/HruzMarek_2011_LocalBinaryPattern

[79] M. Hrúz, Z. Krňoul, P. Campr, and L. Müller, "Towards automatic annotation of sign language dictionary corpora," *Lecture Notes in Computer Science*, pp. 331–339, 2011. [Online]. Available: http://www.kky.zcu.cz/en/publications/HruzMarek_2011_TowardsAutomatic

[80] H. van der Hulst and A. Mills, "On the other hand," *Issues in the phonology of sign language*, vol. 98, pp. 121–144.

[81] P. Dreuw, D. Stein, and H. Ney, "Enhancing a sign language translation system with vision-based features," *Gesture-Based Human-Computer Interaction and Simulation*, vol. 5085, no. 1, pp. 108–113, Jan. 2009.

[82] J. Kanis, P. Peňáz, P. Campr, and M. Hrúz, "A methodology for automatic sign language dictionary creation," Brno, Czech Republic, 2011. [Online]. Available: http://www.kky.zcu.cz/en/publications/KanisJakub_2011_AMethodologyfor

Authored and Co-authored Works

[1] Marek Hrúz, Jana Trojanová, and Miloš Železný. **Local binary pattern based features for sign language recognition**. *Pattern Recognition and Image Analysis*, pp. 398–401, 2011.

[2] Jakub Kanis, Petr Peňáz, Pavel Campr, and Marek Hrúz. **A Methodology for Automatic Sign Language Dictionary Creation**. *Universal Learning Design*, Brno, Czech Republic. 2011.

[3] Marek Hrúz, Pavel Campr, Erinç Dikici, Ahmet Alp Kındıroglu, Zdeněk Krňoul, Alexander Ronzhin, Haşim Sak, Daniel Schorno, Hülya Yalçın, Lale Akarun, Oya Aran, Alexey Karpov, Murat Saraçlar, and Milos Železný. **Automatic Fingersign-to-speech Translation System**. *Journal on Multimodal User Interfaces*, pp 61–79, 2011.

[4] Jakub Kanis, Marek Hrúz, and Pavel Campr. **Metodika pro automatizovanou tvorbu slovníku znakového jazyka**. *INSPO*, 2011.

[5] Ahmet Alp Kındıroglu, Hülya Yalcin, Oya Aran, Marek Hrúz, Pavel Campr, Lale Akarun, and Alexey Karpov. **Multi-lingual Fingerspelling Recognition for Handicapped Kiosk**. *Pattern Recognition and Image Analysis*, pp 402–406, 2011.

[6] Marek Hrúz, Pavel Campr, Zdenek Krňoul, Milos Železný, Oya Aran, and Pinar Santemiz. **Multi-modal Dialogue System with Sign Language Capabilities**. *The proceedings of the 13th international ACM SIGACCESS conference on Computers and accessibility - ASSETS '11*, New York, USA. ACM Press. 2011.

[7] Marek Hrúz, Zdeněk Krňoul, Pavel Campr, and Luděk Müller. **Towards Automatic Annotation of Sign Language Dictionary Corpora**. *Lecture Notes in Computer Science. Text, Speech and Dialogue*, volume 6836 of *Lecture Notes in Artificial Intelligence*, Springer, Berlin-Heidelberg, Germany, pp 331–339, 2011.

[8] Marek Hrúz and Jan Široký and David Maňas. **Knoop hardness measurement using computer vision**. *Proceedings of the 21st International DAAAM Symposium, Intelligent Manufacturing & Automation: Focus on Interdisciplinary Solutions*, pp 537–538, 2010.

[9] Jan Trmal and Marek Hrúz. **Evaluation of Feature Space Transforms for Czech Sign-Language Recognition**. *MetaCentrum Yearbook 2010*, pp 145–150, 2010.

[10] Pavel Campr, Erinç Dikici, Marek Hrúz, Alp Kindiroglu, Zdeněk Krňoul, Alexander Ronzhin, Hasim Sak, Daniel Schorno, Lale Akarun, Oya Aran, Alexey Karpov, Murat Saraclar, and Miloš Železný. **Automatic Fingersign to Speech Translator**. *eNTERFACE'10 The Summer Workshop on Multimodal Interfaces*, 2010.

[11] Zdeněk Krňoul, Marek Hrúz, and Pavel Campr. **Correlation Analysis of Facial Features and Sign Gestures**. *IEEE 10th International conference on signal processing proceedings*, Beijing, pp 732–735, 2010.

[12] Pavel Campr, Marek Hrúz, Jiří Langer, Jakub Kanis, Miloš Železný, and Luděk Müller. **Towards Czech On-line Sign Language Dictionary - Technological Overview and Data Collection**. *LREC 2010, Seventh international conference on language resources and evaluation; 4th workshop on the representation and processing of sign languages: corpora and sign language technologies*, Valletta, Malta, pp 41–44, 2010.

[13] Marek Hrúz, Pavel Campr, Alexey Karpov, Pınar Santemiz, Oya Aran, and Miloš Železný. **Input and Output Modalities Used in a Sign-language-enabled Information Kiosk**. *Proceedings of SPECOM'2009*, pp 113–116, 2009.

[14] Pavel Campr, Marek Hrúz, Alexey Karpov, Pınar Santemiz, Miloš Železný, and Oya Aran. **Sign-language-enabled Information Kiosk**. *eNTERFACE'08 The Summer Workshop on Multimodal Interfaces.* 2009.

[15] Pavel Campr, Marek Hrúz, and Jana Trojanová. **Collection and Preprocessing of Czech Sign Language Corpus for Sign Language Recognition**. *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco. 2008.

[16] Jana Trojanová, Marek Hrúz, Pavel Campr, and Milos Železný. **Design and Recording of Czech Audio-Visual Database with Impaired Conditions for Continuous Speech Recognition**. *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08).* 2008.

[17] Jan Trmal, Marek Hrúz, Jan Zelinka, Pavel Campr, and Luděk Müller. **Feature Space Transforms for Czech Sign-Language Recognition**. *Proceedings of the 9th Annual Conference of the International Speech Communication Association (Interspeech 2008).* Causal Production Pty ltd, pp 2036–2039, 2008.

[18] Marek Hrúz, Pavel Campr, and Miloš Železný. **Semi-automatic Annotation of Sign Language Corpora**. *LREC 3rd Workshop on the Representation and Processing of Sign Languages Construction and Exploitation of Sign Language Corpora*, Marrakech, Morocco, pp 78–81, 2008.

[19] Oya Aran, Ismail Ari, Lale Akarun, Erinc Dikici, Siddika Parlak, Murat Saraclar, Pavel Campr, and Marek Hruz. **Speech and Sliding Text Aided Sign Retrieval from Hearing Impaired Sign News Videos**. *Journal on Multimodal User Interfaces.* 2008.

[20] Marek Hrúz and Pavel Campr. **An Overview of Features for a Sign Language Recognition System from the Database UWB-06-SLR-A**. *The 1st Young Researchers Conference on Applied Sciences (YRCAS 2007)*, Pilsen, Czech Republic. University of West Bohemia, pp 189–191, 2007.

[21] Pavel Campr, Marek Hrúz, and Miloš Železný. **Design and Recording of Signed Czech Language Corpus for Automatic Sign Language Recognition**. *Interspeech*, pp 678–681, 2007.

[22] Miloš Železný, Pavel Campr, Zdeněk Krňoul, and Marek Hrúz. **Design of a Multimodal Information Kiosk for Aurally Handicapped People**. *SPECOM 2007 proceedings*, pp 751–755, 2007.

[23] Oya Aran, Ismail Ari, Pavel Campr, Erinç Dikici, Marek Hrúz, Deniz Kahramaner, Siddika Parlak, Lale Akarun, and Murat Saraclar. **Speech and Sliding Text Aided Sign Retrieval from Hearing Impaired Sign News Videos**. *eNTERFACE'07 The Summer Workshop on Multimodal Interfaces*, Louvain-la-Neuve. TELE, Universite catholique de Louvain, pp 37–49, 2007.