# COMPUTER VISION BASED MULTI-LINGUAL FINGERSPELLING RECOGNITION

by

AHMET ALP KINDIROĞLU

B.S, Computer Sciences & Engineering, Sabancı University, 2008

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering

Boğaziçi University

2011

# ACKNOWLEDGEMENTS

I would not have been able to complete this work without the encouragement and support of many people. My first greatest gratitude and appreciation goes to my advisor, Prof. Lale Akarun, for her direction, dedication and invaluable advice during this study.

I would like to thank my committee members, Assoc. Prof. Murat Saraçlar for his helpful and constructive feedback over the course of this thesis and Dr.Hülya Yalçın for her devotion of time and effort in providing me with guidance and assistance in my studies.

I would like to extend gratitude to my fellow colleagues and friends Ismail Ari and Mehmet Gönen for their invaluable philosophic, academic and technical guidance. I would also like to specially thank Umut Şimşekli, Furkan Kıraç, Yunus Emre Kara, Pınar Santemiz, Aycan Yüksel, Oya Aran and all other friends and colleagues who have collaborated with me in my studies.

I would like to thank Erinç Dikici, Haşim Sak, Alexey Karpov, Milos Zlezny, Marek Hruz; the members of our project group at the eNTERFACE'10 workshop, whose contributions have shaped parts of this thesis.

I would like to thank Tubitak, for the sum of research fund they made available to conduct this research. The research described in this thesis has been supported by the bilateral TÜBİTAK project 108E113.

And finally, I extend my special and profound thanks to my family for their love and support.

# ABSTRACT

## COMPUTER VISION BASED MULTI-LINGUAL FINGERSPELLING RECOGNITION

In this thesis, we focus on the problem of computer vision based automatic sign-language recognition and its related subtasks. The study focuses on the recognition of fingerspelling gestures, which are a subset of sign languages that provide manual representation for spoken alphabet letters. Fingerspelling gestures make use of hand shapes, orientation, location and movements. We perform the task of fingerspelling recognition of Turkish, Czech and Russian manual alphabets with the purpose of integrating these sign alphabets to multi-modal and multilingual deployable applications. In the thesis, we divide the automatic fingerspelling recognition task into sub-challenges and design methodologies to improve overall sign recognition performance. We describe an approach to tracking of hands and a face in an image sequence containing the frontal pose of a signing person. A classical Camshift algorithm is extended in this study to contain automatic skin color model initialization, hand re-detection and collision handling. The algorithm performs robust, close to real-time hand tracking. Secondly, we focus on hand gesture representation. We evaluate the usage of appearance based features for describing the manual component of Sign Languages; in particular Elliptic Fourier Descriptors, Hu Moments, Radial Distance Function and Local Binary Patterns. We test the recognition performance of individual features and their combinations. Local Binary Patterns show the best recognition performance on isolated gestures with a recognition rate of up to 92 per cent. We explore the usage of features such as hand motion and motion blur in the problem of temporal segmentation to separate gesture start and end locations in continuous gesture videos. We investigate the fusion of temporal and appearance features using sequence voting, discrete HMMs and continuous HMMs. We test the fingerspelling recognition accuracy of our system on a self collected multilingual fingerspelling dataset consisting of Turkish, Czech and

Russian manual alphabets from multiple signers with multiple repetitions. Finally, we have demonstrated the applicability of our system in a prototype application that functions as a multi-lingual fingerspelling to speech translator.

# ÖZET

## BİLGİSAYARLA GÖRME TABANLI ÇOK DİLLİ PARMAK ALFABESİ TANIMA

Bu tezde bilgisayarla görme tabanlı otomatik işaret dili tanıma ve ilgili alt konular üzerine yoğunlaşmış çalışmalar yaptık. Çalışmada üzerinde yoğunlaşılan el alfabeleri, işaret dillerinin, işaret dilinde karşılığı olmayan kelimelerin sadece parmak hareketlerini kullanarak temsilini sağlayan bir alt kümesidir. El alfabeleri, kavramları ellerin şekillerini, yönelimlerini, konumlandırmasını ve hareketlerini kullanarak temsil eder. Bu çalışmada, çok kipli ve çok dilli sistemlerde kullanılabilecek, Türk, Çek ve Rus el alfabelerinde yarı gerçek zamanlı el alfabesi tanıma yapan bir sistem geliştirdik. Otomatik işaret dili tanıma problemi üzerine yaptığımız çalışmalarda, el izleme ve bölütleme, el özniteliklerinin temsili, sınıflandırılması ve zamansal bölütlenmesi gibi alt konularda yoğunlaşarak geliştirdiğimiz ve kullandığımız metotların karşılaştırmalı analizlerini yaptık. Geliştirdiğimiz el ve yüz izleme yöntemiyle kamera karşısında işaret dili icra eden bir kullanıcının ellerini dayanıklı ve verimli bir şekilde takip edebiliyoruz. Klasik Camshift algoritmasına yaptığımız çoklu obje izleme, otomatik renk modeli oluşturma, otomatik el bulma ve kesişip ayrışan objeleri işaretleme yöntemleriyle kesintisiz videolarda dayanıklı el işareti tanıması yapılmasına olanak sağladık. El hareketi temsil metotlarımızda Eliptik Fourier betimleyicileri, Hu momentleri, ışınsal uzaklık fonksiyonu, yerel ikili örüntüler gibi iki boyutlu imgelerden elde edilen görüntü kipli özniteliklere ağırlık verdik. Bu özniteliklerin tanıma performanslarını tek tek ve birlikte inceleyerek sistemin detaylı bir analizini gerçekleştirdik. İmge dizileriyle yaptığımız testlerde, izole el hareketleri için en iyi tanıma başarımını yüzde 92 ile yerel ikili örüntü betimleyicileri verdi. Kesintisiz el işareti dizilerinde, işaretlerin başlama ve bitiş zamanlarını bulmak için hareket ve harekete bağlı bulanıklığı bir öznitelik olarak kullandık. Son olarak zamansal ve görsel özniteliklerin, el işareti dizilerini tanıma için kaynaşımını gerçekleştirerek dizilerde ağırlıklı oylama, ayrık Saklı Markov Modelleri ve kesintisiz

Saklı Markov Modelleriyle el hareketlerini modelledik. İşaret dili tanıma başarımını ölçmek için yaptığımız testlerde, kendi topladığımız Türk, Çek ve Rus el alfabelerinden oluşan çok dilli veritabanını kullandık. Bu tez kapsamında, çalışmalarda geliştirilen yöntemleri kullanan bir parmak alfabesinden sese tercüme uygulaması geliştirdik.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF SYMBOLS/ABBREVIATIONS

| | |
|---|---|
| $A$ | Transition Matrix |
| $a_{ij}$ | Transition probablity from state i to j |
| $a_k$ | Elliptic Fourier coefficient for point k on a curve |
| $B$ | Observation Matrix |
| $b_j(m)$ | Probablity of seeing observation m at state j |
| $b_k$ | Elliptic Fourier coefficient for point k on a curve |
| $c_x$ | X coordinate of center of mass for a binary shape |
| $c_y$ | Y coordinate of center of mass for a binary shape |
| $c_k$ | Elliptic Fourier coefficient for point k on a curve |
| $d_k$ | Elliptic Fourier coefficient for point k on a curve |
| $H(h, s)$ | 2D Histogram bin corresponding to values h and s |
| $I(x, y)$ | Pixel of image I at coordinates x and y |
| $I_x$ | Image derivative mask in x direction |
| $I_y$ | Image derivative mask in y direction |
| $M_{ij}$ | Central Image moment of order i + j |
| $PI(x, y)$ | Pixel of probablity distrubition image PI at coordinates x and y |
| $T$ | Perimeter of a closed curve |
| | |
| $\alpha(i)$ | Forward Variable in HMM |
| $\beta_t(i)$ | Backward Variable in HMM |
| $\gamma_t(i)$ | Probablity of being in state i at time t |
| $\mu_{ij}$ | Normalized Central Image moment of order i + j |
| $\lambda(A, B, \pi)$ | Hidden Markov Model |
| $\pi(i)$ | HMM prior probabilities |
| $\pi$ | The number 3.14159265 |
| $xi_t(i, j)$ | Probablity of moving from state i at time t to state j at time t+1 |

| | |
|---|---|
| CSA | Czech Sign Alphabet |
| CV | Cross Validation |
| DA | Description of abbreviation |
| DCT | Discrete Cosine Transform |
| DOF | Degrees of Freedom |
| DD | Double Differencing |
| EFD | Elliptic Fourier Descriptor |
| F2S | Fingersign to Speech |
| FRR | Fingerspelling Recognition Rate |
| FSR | Fingerspelling Recognition |
| HCI | Human Computer Interactions |
| HOG | Histogram of Oriented Gradients |
| HMM | Hidden Markov Model |
| HSV | Hue, Saturation, Value |
| HU | Invariat Moments of Hu |
| GMM | Gaussian Mixture Model |
| KNN | K-Nearest Neighbours |
| LBP | Local Binary Patterns |
| PCA | Princple Component Analysis |
| RDF | Radial Distance Function |
| RGB | Red Green Blue Colorspace |
| RSA | Russian Sign Alphabet |
| SLR | Sign Language Recognition |
| SVM | Support Vector Machines |
| SURF | Speeded Up Robust Features |
| TSA | Turkish Sign Alphabet |

# 1. INTRODUCTION

## 1.1. Motivation

In todays computers, keyboards, mice and touchpads are still the most popular and dominant Human Computer Interaction devices. However, they are inconvenient and unnatural. In recent years, the use of new modalities such as speech and human movements, especially hand gestures, has become an important part of Human Computer Interactions. Compared to speech commands, hand gestures are advantageous in noisy environments, in situations where speech commands would be disturbing, as well as for communicating quantitative information and spatial relationships. This serves as a motivating force for research in modeling, analyzing and recognition of hand gestures. Many techniques developed in HCI can be extended to other areas such as sign language recognition, surveillance, gaming, robot control and teleconferencing. Recognizing gestures is a complex task which involves many aspects such as motion modeling, shape analysis, pattern recognition, natural language processing and machine learning.

Sign Languages are a form of manual communication, which have developed as an alternative to speech among the deaf and speaking impaired. Sign languages make use of different modalities such as hand shape, movement, gestures and facial expressions to convey meaning. A gesture is a form of non-verbal communication made with a part of the body and used instead of verbal communication. Although sign languages are the universal communication medium of deaf people, languages of different communities show a large amount of variance. As sign languages emerged and developed locally within hearing communities, the content of each sign language differs from each other [1]. In addition, as sign languages aim to convey meaning clearly and rapidly with as few signs as possible they also syntactically and grammatically differ from spoken languages.

Sign languages contain unique gestures or signs for every different concept. Therefore most sign language corpora do not contain a sign for every possible word in the

spoken languages. For this purpose sign languages make use of fingerspelling gestures, which they use to directly represent a word in the spoken languages. Fingerspelling is a subset of sign languages that is used mainly for representing out of vocabulary words. It uses static and dynamic gestures composed of differing hand and finger combinations. For example, to sign the word Alp, the signer performs the gestures corresponding to 'a','l' and 'p' sequentially. In terms of grammar, fingerspelling preserves the grammar properties of written languages instead of the properties of sign languages. While fingerspelling signs vary from language to language like sign languages, they all share the common communication medium of being performed by one or two hands.

Developing sign language applications for deaf people can be very important, as many of them are also not able to read or write a spoken language. The aim of automatic sign language recognition is to make the life of deaf people easier. In daily life situations where a deaf person tries to communicate with a hearing person signers either need their correspondent to understand sign language or use a hearing interpreter to help him communicate. If they are successfully employed, by providing an easy, efficient and accurate mechanism to transform sign language into text or speech; sign language recognition systems will enhance the capabilities of the hearing impaired by reducing their dependence on interpreters. Sign language does not have a general written form and documents are therefore produced using live video and video-tapes. For instance, deaf students and instructors make their exams, tests and homeworks on videos. The flexibility of this medium is much smaller than that of paper as only sequential access to the material is possible. In order to access a certain part of a sign language video recording, one must traverse the video manually trying to interpret signs until he can find what he is looking for.

With the improvements of image processing, machine learning and natural language processing techniques in the last 20 years, research on SLR systems have become an active topic. Although most sign languages contain similar modalities and signs, due to the sheer number of different sign languages in the field, research efforts in the field mostly converge on the sign languages of researchers such as American or British sign languages. While performing SLR with different sign languages, the difficulty

does not lie with methodologies and algorithms as they do not need to change. The main problem is the lack of training data, which needs to be recorded under certain restrictions to train different systems.

## 1.2. Research Methodology and Contributions

This thesis presents a computer vision based, close to real-time fingerspelling recognition system. The system is trained to perform recognition in three sign alphabets: the Turkish Sign Alphabet, the Czech Sign Alphabet and the Russian Sign Alphabet. The overall aim of the study is the establishment of a fingerspelling input modality, that can later be used in various interactive modules such as "finger-sign to word" or "finger-sign to speech" translators, interactive environments and information systems.

In sign language recognition, researchers tend to distinguish the task of recognition with regard to the systems purpose as isolated and continuous SLR. In isolated sign language recognition systems the user performs signs one by one clearly indicating the beginning and end of each sign. A large amount of research on sign language recognition is done on isolated gestures, as it simplifies the task of SLR to differentiating different hand poses. However, in online systems where the user performs gestures one after another, continuous SLR is required to segment isolated signs from continuous sequences of signs. Thus, we have chosen to perform our research on SLR methodologies using isolated sign gestures, while exploring the usage of continuous gestures through temporal segmentation methodologies in our deployable applications.

In this thesis we have preferred to perform fingerspelling recognition over sign language recognition as a modality for the hearing impaired. Although the usage of fingerspelling is not as popular as sign languages among deaf communities, its usage in our system simplifies our task and improves system usability. Using fingerspelling, it is possible to represent the entire spoken language lexicon using a small, limited number of signs. In addition, as fingerspelled words directly correspond to words of spoken languages, the semantic gap between spoken and sign languages can be ignored;

removing the need for the application of natural language processing methods. In a context where interpreted signs are translated from one spoken language to another; (for example from Turkish to Russian) usage of fingerspelling allows the use of simple word level dictionaries for translations. Although the focus of this study is on the methodologies of sign language recognition, the techniques developed are being applied to different problems.

The main contributions of this thesis can be summarized as follows:

- Joint Cam-shift based tracking of hands and face: Accurate tracking of signing hands and faces are a necessity for sign language recognition applications. The accuracy of hand descriptor representation and modeling depends on hands being localized and perfectly segmented. We describe an approach to tracking of hands and a face in an image sequence containing the frontal pose of a signing human. A classical Camshift algorithm is expanded in this study to contain automatic skin color model initialization, hand re-detection and collision handling. The algorithm performs robust, close to real-time hand tracking.
- Multilingual fingerspelling recognition method: We have designed, analyzed and realized a fingerspelling recognition methodology for interpreting isolated and continuous hand gesture videos. Using a two tiered classification method, we attempt to recognize the meaning that sequences of images containing hand gestures try to convey. We make use of appearance based methods to extract features of a hand shape by analyzing 2D hand images. We prefer the usage of these features due to their simplicity, effectiveness and low computation times, as we aim to generate real time applications. Through detection of motion blur and hand displacement, we separate frames that contain hand gestures with meaning from those that are not. Finally we combine these features using classifiers in different combinations to model hand gestures and fuse the results to make sequence level final decisions.
- Usage of motion blur as a keyframe selection method: An important challenge in continuous fingerspelling recognition is the problem of designating which frames

the signer actually performs a gesture(keyframes) and which frames the signer does not(transition frames). Through detecting motion blur on segmented hand images, we attempt to use motion blur as a feature in keyframe detection. Since the presence of motion blur prevents accurate segmentation in a sequence of gestures, it is desirable to choose the frame with the least blur to maximize segmentation performance and therefore recognition accuracy. For detecting motion blur, we employ a method that makes use of the gradient strengths of hand edges. Compared to unblurred images, a major characteristic of blurred images is that edges tend to be smoother and contain smaller gradient values. By focusing on the distribution of gradient values in certain image patches, we can obtain an idea on the presence of partial motion blur.

- Establishing a multi-lingual fingerspelling dataset: Training sign language recognition systems requires large datasets containing multiple examples of each gesture to be recognized. Although various datasets belonging to each of our target fingerspelling alphabets exist in different sources, they all seem to vary in different critical aspects such as lighting, background constrictions or body pose. Therefore, in order to train a compatible multilingual system, we have collected a video database of fingerspelling sings. The dataset contains the fingerspelling alphabets of the Czech, Russian and Turkish sign languages and is approximately 400 minutes in length.
- FingerSign to Speech translator: Being the first prototype multimodal application of this study, the FingerSign to Speech translator translates fingerspelling gestures to speech and vice versa. It was developed in the eNTERFACE'10 Summer Workshop on Multimodal Interfaces in Amsterdam with the contributions of our Czech and Russian research partners. It provides a communication modality, where people with hearing and visual disabilities from different nationalities can communicate with each other. In addition to our fingerspelling recognition module, it includes a fingerspelling synthesis, speech recognition and a text to speech module.
- Information Kiosk For the Disabled: This information terminal, which aims to serve people with different disabilities, is being developed as a part of the bilateral

Tubitak project with the same name. It aims to service people with different disabilities in a multimodal fashion.

## 1.3. Literature Review

Since the beginning of 1990's hand gesture recognition has become an active research topic. In[2, 3] reviews on recent research in sign language and gesture recognition are presented. Most studies on sign language recognition can be grouped into three subcategories as the studies on hand tracking and segmentation, the studies on hand gesture representation and the studies on gesture modeling and recognition. In section 1.3.1, we give a biref overview of hand gestures and present the challenges in recognition. Then in sections 1.3.2 to 1.3.4, we present a review on different methodologies used in various sign language recognition themed studies.

### 1.3.1. Hand Gestures

Hands are the chief physical organs of humans for manipulating the environment. Using hands, humans can both perform tasks that either require gross motor skills such as weight lifting or require fine motor skills such as writing. As can be seen in Figure 1.3.1, the human hand has 27 bones and roughly 30 degreed of freedom.

In the literature, there have been numerous studies on the linguistic usage of the human hand. [3] categorizes hand gestures according to their purpose as conversational gestures, controlling gestures, manipulative gestures and communicative gestures. In this study, we are interested in the communicative gestures, which include pantomimes and sign languages as their most common examples.

In sign languages, each represented concept needs to have a separate sign that differentiates it from other gestures. For that reason sign languages fully exploit the highly flexible nature of the human hand by trying to generate numerous different gestures using modalities like hand finger combinations, hand position, hand orientation

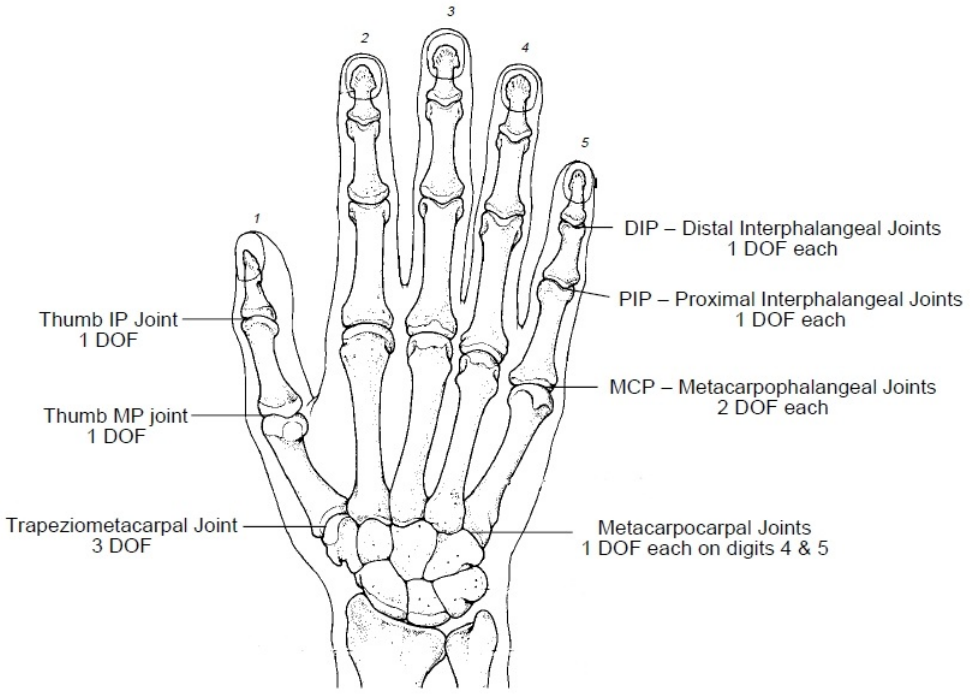



Figure 1.1. Hand Skeleton and associated Degrees of Freedom [4]

and hand movement. Therefore, when dealing with the academic problem of hand gesture recognition, sign languages constitute an excellent test case. In addition, dealing with sign language recognition also addresses the communication problems of the hearing impaired with their environment.

Fingerspelling, which is a subset of sign languages makes use of only manual hand gestures to represent out of vocabulary sign language words. Automatic fingerspelling recognition is a task that differs from sign language recognition in certain aspects. While fingersigns contain fewer signs and arm movements than sign languages, the meaning presented by them is conveyed through combinations of fingers and hands; thus, negating possible features like relative displacement of hands. In addition, the rapid movement of fingers and hands causes automatic FSR systems to require a higher amount of precision than automatic SLR systems.

Table 1.1. Survey of computer vision based sign language recognition Studies

| | Year | Detection Method | Segmentation Method | Features | Classification | Continious | Gesture # | Accuracy |
|---|---|---|---|---|---|---|---|---|
| [5] | 1995 | Colored glove | Color segmentation | Geometric moments | HMM | no | 31 | 83 per cent |
| [6] | 1997 | Skin color | | PCA | Template Matching | yes | 25 | 99 per cent |
| [7] | 1997 | Active shape models | Shape cues | Hand contour | Gaussian | no | 5 | |
| [8] | 1998 | 7 markers on hand | 3D model fitting | Model Parameters | | yes | | |
| [9] | 1999 | Skin color | | Moment and edges | Neural Network | no | 25 | 96 per cent |
| [10] | 2000 | Skin color | Color and edge cues | Shape cues and keyframes | dHMM | yes | | |
| [11] | 2002 | Skin color and bracelet | Color segmentation | Clustered pixel values | NN boosting | yes | 46 | 99 per cent |
| [12] | 2002 | Background subtraction | Skin color | Shape cues | HMM | no | 65 | |
| [13] | 2002 | Bird-eye skin color | Color,motion,shape cues | Geometric cues | Template Matching | no | 8 | 83 per cent |
| [14] | 2002 | Motion based | Region matching | Color, shape , motion cues | Template Matching | no | 25 | 98 per cent |
| [15] | 2004 | Multiflash camera | Meanshift and edges | LBP | Template Matching | yes | 25 | 96 per cent |
| [16] | 2005 | Skin color | Skin color | Motion, moments | HMM | yes | 20 | 93 per cent |
| [17] | 2006 | Skin color | Skin color | Moments, RDF | kNN,SVM | no | 29 | 99 per cent |
| [18] | 2006 | Skin color and motion | Camshift | Moments, image difference | HMM | yes | 24 | 97 per cent |
| [19] | 2007 | Stereo images | 2D model fitting | Model Parameters | Template Matching | no | 23 | 87 per cent |
| [1] | 2008 | Colored glove | Particle filter | Shape,motion,position | HMM | yes | 29 | 79 per cent |
| [20] | 2008 | Segmented images | | DCT | HMM | no | 23 | 94 per cent |
| [21] | 2009 | Skin color | Skin color based | Hog | HMM | no | 26 | 98 per cent |
| [22] | 2010 | Skin color | Region growing | Hog,dct,lbp,moments | HMM | yes | 40 | 98 per cent |

Table 1.1 presents an overview of the fingerspelling and sign language recognition systems according to their methods of detection, segmentation, representation and classification. Each system is evaluated according to its input type, dataset diversity and recognition accuracy.

### 1.3.2. Hand Tracking and Segmentation

Accurate tracking of signing hands and faces is a necessity for sign language recognition applications. The accuracy of hand descriptor representation and modeling depends on hands being localized and accurately segmented. Although hand tracking and segmentation has been an active research topic for many years, challenges of the human hand such as its highly deformable and self occluding nature; has not yet enabled the development of methods that can robustly and swiftly segment all hands in unconstrained environmental conditions. For this reason, in order to recognize human hands, researchers have made use of specially customized glove devices or computer vision techniques. Some earlier studies on hand gesture and sign language detection have focused on the usage of customized data collection gloves [23–27]. Although these methods provided highly accurate data that allowed the reconstruction of three dimensional hand models, the high degree of freedoms of the hand caused the data gloves to contain between 30 to 80 degree of freedoms. Likewise, using the customized data collection gloves is a cumbersome task for the user that affects his/her natural behaviour. In addition, the high dimensionality of the input data on top of the change in the size and shape of user hand limbs prevents researchers from developing cheap plug and play systems capable of universal usage with this method.

On the other hand, hand tracking and segmentation with computer vision techniques promise more robust and less intrusive methods. In computer vision based hand recognition systems, the hand is recorded using a camera or a set of cameras. The task of hand segmentation can then be divided into two separate parts as the segmentation of hands from images and the tracking of hands in continuous image sequences by making use of prior knowledge on hand locations. Some studies that focus on single image based hand recognition focus on hand segmentation[17, 28], while others making

use of image sequences make use of tracking techniques[7, 1, 29, 30, 6, 31]. In continuous image sequences, the most commonly used methods for hand segmentation can be generalized as color (skin or glove color), motion, depth, edge and shape cues. Among these methods, skin color based hand extraction seems to be the most focused and common method [11, 16, 17, 21, 33–35]. Skin-color information as a cue has gained much attention as skin-color provides computationally effective yet, robust information against rotations, scaling and partial occlusions. Skin detection using color information can be a challenging task as the skin appearance in images is affected by various factors such as illumination, background, camera characteristics, and ethnicity. A good survey paper that presents the recent trends in skin color based segmentation is [36]. A viable alternative to the usage of skin color is to use skin colored gloves, removing the need for skin color training [37, 1].

Due to technical considerations, most recording equipments use RGB representation. However, previous studies such as [36] show that this is not necessarily the best color representation for characterizing skin-color. The human eye can adapt to different illumination conditions. Likewise, by using illumination preserving color spaces such as Hue and Saturation of HSV or normalized RG, it is possible to remove brightness from skin-color representation without losing descriptive color information.

In performing skin color segmentation, some methods prefer the usage of Gaussian skin color models[22], while others, using Gaussian Mixture Models with multiple mixture components obtain better segmentation results[38]. As predictions with Gaussian Mixtures are computationally more expensive than histogram lookup operations, the use of color histograms as our skin color models is widely preferred in real-time applications [39, 40]. Using these models, segmentation approaches like histogram-backpropogation [39, 40] or region growing algorithms are employed[22, 41]. In addition to these methodologies, [42] makes use of graph-cutting algorithms to aid segmentation without previously training a skin color model. Some studies make use of motion cues to improve skin color based segmentation results[21, 35, 43].

Although several studies with different color spaces such as infrared cameras tend to yield better results [18], they too bring additional constraints such as a dark recording environment. In addition to color based segmentation strategies, the constant motion of hands around the body and the self occlusion caused by their high degree of freedom in 3D space makes the use of depth information a viable method for segmentation[44, 19]. While using stereo cameras to capture depth was considered to be too expensive until recently, Microsoft's Xbox Kinect gaming console presented a very cheap and usable depth capture hardware [45]. Therefore, we expect a rapid increase in hand gesture detection studies using depth cues in the close feature.

Furthermore, the studies on the usage of Haar like shape based hand recognition methods, although not as successful as their application in face recognition, have also produced promising results [46]. To incorporate the information of segmentation results of past images in image sequences, various tracking approaches have been applied in hand tracking. The Camshift algorithm tracks hands by climbing the peaks of the skin color probability distribution image [39–41]. Particle filters have also been applied in numerous studies, by selecting particle parameters such as the width, the height and the angle of an ellipse surrounding the hand. Particle filters provide robust tracking, labeling and occlusion handling[22, 1].

### 1.3.3. Feature Extraction

Following the removal of the hands from the non-hand background information, each frame in the image sequence is converted into a feature space where the features can be used to mathematically distinguish gestures in a generalizable fashion. In the literature, numerous feature descriptors with different characteristics have been applied to describe hand gestures. These descriptors can be classified based on their acquisition method, their invariancies or what they represent. In order to distinguish fingerspelling signs shape and motion of hand gestures must be uniquely represented. The ultimate aim of a fingerspelling recognition system is to distinguish hand gestures in a sequence of images. Therefore, factors such as varying signer hand shapes, signer performances and camera conditions should have a minimal effect on hand gesture descriptors. In hand

gesture recognition with sensory input gloves, the data is generally directly used with a nearest neighbor classifier [23–27]. These approaches also make use of dimensionality reduction techniques to improve the recognition accuracy of their data. In [27] a heuristic called Range of Motion is used to calculate the amount of variance each hand joint angle can tolerate to make the hand glove data more user independent.

Popular color based descriptors such as color histograms and correlelograms are not directly applicable as human hands with differing shapes cannot be distinguished from one another using color cues. Instead, many hand gesture recognition methods make excessive use of two dimensional appearance based features to represent signer hands. In [47], an excellent survey on shape based descriptors is presented. In the studies [6, 48], normalized hand images are directly used with PCA to use them as hand features. Likewise [18] used normalized integral hand images. However, since the normalization of hands with numerous possible shapes is not a feasible task, the usage of these kinds of wholistic features are not widespread.

There are numerous studies that make use of hand gradients and oriented gradient histograms[22, 21]. Other methods make use of local descriptors such as local binary patterns[49] and discrete cosine transform [22, 50]. In [51] Elliptic Fourier descriptors are employed to represent object outer contour, while [50] makes use of the radial distance function to calculate distance of the seedpoints to hand contours. Studies like [52] make use of local Surf descriptors with keypoints using the bag of visual words method.As motion descriptors, the studies in [20, 43] make use of hand displacement along with dynamic time warping as features.

There are also 3D recognition approaches that attempt to fit 3D hand models to stereo 2D pairs[44, 19]. These features employ depth information to fit models to images.

### 1.3.4. Classification and Temporal Segmentation

In sign language recognition, the aim is to classify gestures into different classes using the descriptors described in Section 1.3.3. In order to perform the classification of continuous image sequences, three tasks must be handled. These tasks are classification of hand features, temporal segmentation and alignment.

The first of these tasks is the task of classification which compares hand descriptors obtained from single frames containing hand images. In the literature, numerous classifiers for classifying samples with identical feature descriptors exist. Studies such as [53, 54, 17] make use of parametric Naive Bayes and non parametric KNN classifiers to classify hand images. [55] makes use of support vector machines to classify isolated hand gesture images. Similarly, [11, 15, 48, 19] makes use of template matching approaches to compare hand gesture images. Earlier studies in the field also contain examples of Neural Network classifications[6, 23].

However, unlike the studies shown above, most hand gesture recognition systems contain sequences of images which needs alignment and temporal segmentation over time. For that reason, numerous studies have made use of Hidden Markov Models for the alignment and classification of hand gestures [18, 16, 20, 35, 21, 33, 22, 1]. Likewise [56] explores fusion of generative classifier approaches with weighted voting to obtain combined decisions.

## 1.4. Thesis Outline

This chapter has made an introduction by describing the problem of recognizing fingerspelling gestures in multiple languages. The contributions and outline of this thesis along with a review of state-of-the-art hand gesture and sign language recognition systems are also presented in this chapter. Chapter 2 presents the Camshift based joint hand and face tracking algorithm. In this chapter, methods we employed to track and segment hand images from background pixels are detailed. In Chapter 3, appearance based feature descriptors that we employ to represent hands are detailed.

In chapter, Chapter 4 we present our methodologies for hand gesture and fingerspelling classification using generative models. Chapter 5 introduces the fingersign to speech application that was developed in conjunction with this thesis and presents the experimental results of our integrated system. In Chapter 6, we present an overall conclusion and discussion of the contributions of the thesis.

# 2. HAND TRACKING AND SEGMENTATION

Hand gestures are the primary component of sign language communication . Therefore, in order to distinguish, recognize and understand fingerspelling gestures, accurate extraction and representation of hand gestures is a necessity. Being an active research topic in Human Computer interaction, tracking and segmentation of human hands is still considered as a challenging problem. Due to the high degree of freedoms and self occluding nature of human hands; an unconstrained, fast and accurate hand gesture recognition system is still far from being realized. Many of the proposed solutions addressing this issue constrain the users actions one way or the other. These restrictions vary from making the user wear markers or position his hands between the camera and his body to preventing hand and face interactions. However, while performing sign language gestures, most of these restrictions violate the users sign space, causing him to perform gestures differently than he would in his natural signing environment. Therefore, sign language recognition systems that constrain the hand movements of signers are undesirable as they reduce the systems overall universal usability.

In this chapter, we focus on the continuous tracking of hands and face of sign language performers from a frontal pose. Our system tries to achieve a balance between three primary concerns: robustness, speed and reliability. The presence of highly mobile gestures requires the hand gesture recognition to operate at a close to real-time speed. Therefore, in the task of online SLR, appearance based simplistic approaches may be preferred over complex 3-D modeling of hands. However, aside from runtime speed, the diversity of possible hand gestures in sign and fingerspelling languages imposes tough to achieve robustness and reliability requirements. In order to handle issues like hand-hand, hand-face and hand-self occlusions using appearance based methods, simple constraints on the user are deemed necessary.

## 2.1. Joint Hand and Face Tracking

Accurate tracking of hands and face of a natural signer is a challenging task that is crucial for a FSR applications. The accuracy of hand descriptor representation and modeling depends on hands being localized and perfectly segmented. As the signing human hand often tends to make fast and non-linear movements, modeling and tracking the hand becomes a greater challenge. Likewise, the occlusions of the hands with each other and with the face is very frequent. To handle this task of hand tracking we have explored the usage of different algorithms. In the end we have decided to use a customized Camshift algorithm that would be equipped with the necessary capabilities to handle the tracking of signing hands. The tracking of hands and face with our methodology can be broken down to three seperate subtasks, which are visualized in Figure 2.1.


Figure 2.1. Flowchart of joint hand and face tracking

The continuously adaptive Mean Shift Algorithm, also known as Camshift is a moving object tracking algorithm. It is an extension over the Mean-shift[57] tracking algorithm and performs tracking function over continuous sequences of images. It is a non parametric method that makes use of a color probability distribution image to track hands.

### 2.1.1. Initailzation of Adaptive Skin Color Histograms

Camshift is a semi-automatic tracking algorithm that requires user to define the tracked object prior to execution. Common implementations of the algorithm [40, 39] request the user to manually designate the object that is to be tracked. The first frame

of the video sequence is displayed and the user is asked to draw a box around the object to be tracked. This box is then used for two separate purposes. Color information, obtained from the box is used to create a color histogram that is later used to create probability maps to identify the possible locations of the tracked object. In addition, the box is used by the Camshift algorithm as an initial region of interest to estimate of the location of the object.

Our tracking method accomplishes tracking by using the color properties of tracked objects. Therefore the accuracy of the algorithm in distinguishing tracked objects solely depends on the accuracy of the histogram built during initialization. For accurate initialization, the users' hand needs to be flawlessly segmented from the background. In common implementations of the algorithm, the initialization process is implemented in a manual fashion, asking the user to draw a box on the object to track. While this type of a training method is satisfactory for experimental studies, it is an undesirable burden for an interactive system. For that reason, we have explored several techniques to automatically train skin color models to make a system that is adaptive to user and environmental illumination changes.

One method we have considered for this task is to display the live feed, draw two boxes on the screen and ask the user to move his/her hands inside these boxes to commence recognition. This method makes use of motion detection to understand when a foreground object is moving into the box and when it is waiting stably inside the boundaries of the box. Color histograms are then calculated from the image segments inside the patches. Although this method was theoretically simple, practice showed that the accuracy of the skin color model greatly depended on the success of the user in orienting his hand and finding the box. From a HCI point of view, the task of trying to find an inexistent box in three dimensional space while looking at a two dimensional mirror image is not an easy task. For that reason we resumed our search for alternative, more reliable hand detection methods.

Another method we employed in training skin color histograms was the use of face detection. Since the human face contains more easily distinguishable features from

the human hand, generic detection tools for frontal face detection are widely available. Using shape cues through the Viola-Jones face detection algorithm [58], we detect the boundaries of the signers' face. A skin-color model is then extracted from the face of the signer. Although more reliable than using static hand boxes for initialization, this method fails detection in two exceptional cases. Self occluded hand parts such as finger edges often do not receive direct illumination from point light sources and appear darker than other skin colored pixels. Therefore a skin color model trained from relatively flat faces fails to detect parts of hand images. In addition, while the Viola-Jones algorithm has a high detection rate, it also seldomly mistakes background shapes for faces due to its high false positive detection rate, resulting in the generation of erroneous skin color models.

Finally we have used a skin color initialization method that makes use of motion cues to distinguish hands from the background. By asking the user to stand still while waving his/her hands at the camera, an easy to perform skin color model initialization can be achieved [37].

In order to perform hand detection using motion cues, we make use of the frame differencing technique. The absolute value of the difference of pixel values between two successive image sequence frames are used to obtain the frame difference image. Although the obtained image marks all moving objects, the amount of background noise still prevents us from extracting any viable foreground object information. A solution to this problem is the Double Differencing method proposed by Xia [59].

By calculating the absolute differences of three successive frames, the displacement of moving objects from times t-1 to t and t to t+1 are captured (Figure 2.1.1). Pixel displacements caused by the slow movement of smaller background objects and minor illumination changes hardly repeat themselves in successive images. Therefore, the combination of successive frame difference images are usually much more tolerant to noise. After combining the two successive frame difference images with a pixel-wise logical *AND* operation, we obtain a binary image that only contains movements that were sustained over time. After a Morphological opening operation to remove

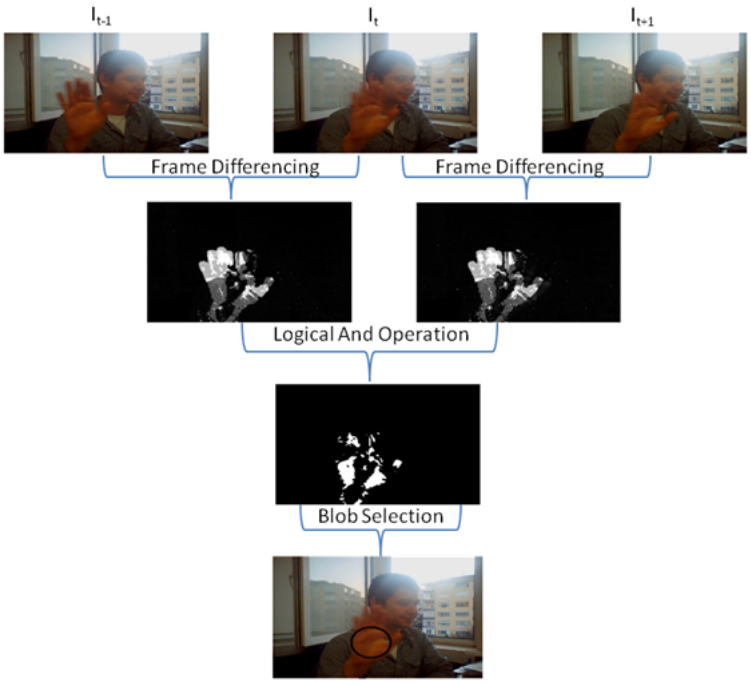



Figure 2.2. Detection of waving hand using double differencing method

any remaining background noise, the resulting mask can be used to detect the largest groups of movement in the image. Due to the irregular nature of hand movement and extensive noise reduction, the double-difference image usually consists of smaller blobs rather than complete hand images. For this reason, the location of the hand is usually occupied by a multiple number of blobs that are in close proximity to one another. Starting from the largest blob in the image-mask, all blobs within an expanding radius proportional to total blob area are used to establish a hand location estimate. After estimating the region radius, all the blob pixels falling in this estimated region are used to build a skin color histogram.

### 2.1.2. Tracking in continuous sign language videos

The Camshift algorithm makes use of color information to track objects. As long as the object sustains its color, it is resilient against deformations in size and shape. It is highly sensitive towards changes in environmental illumination. Thus

many common implementations [39] perform the tracking task in HSV color-space with a heavy emphasis on hue and saturation channels.

To commence tracking on a sequence of images, the Camshift algorithm first needs the input of a region of interest containing the approximate coordinates of the object to be tracked. Since we chose to automate this step instead of demanding manual user input, details of our initialization and skin color histogram generation can be found in Section 5.2. After converting each incoming image to HSV color space, we obtain a 16x8 Hue-Saturation histogram representing the skin color of the user to be tracked. Using the Histogram Back-projection Method explained in Section 2.1.3.1, we obtain a color probability distribution image. Each pixel in the color probability distribution image is assigned the probability value that the pixel belongs to the target object. Tracking with the Camshift algorithm can be summarized as in Figure 2.3 [39]:

1. Given a region of interest for the object calculate the probability distribution image around the target region.
2. Iterate Meanshift Algorithm on the expanded region of interest in the probability distribution image. Calculate the new centroid of the region using equations 2.1 and 2.2.
3. Move the region centroid to the new centroid. If movement is larger than a threshold or maximum iteration count has not yet been reached, jump to step 2.
4. Calculate the size of the tracking box using the zeroth moment (distribution area) using equations 2.1 and 2.2. Set the tracking box as the region of interest for the next frame and jump to step 1.

Figure 2.3. Camshift Algorithm

$$\mathbf{M}_{00} = \sum_x \sum_y \mathbf{I}(x,y); \mathbf{M}_{10} = \sum_x \sum_y \mathbf{xI}(x,y); \mathbf{M}_{01} = \sum_x \sum_y \mathbf{yI}(x,y) \qquad (2.1)$$

$$\mathbf{c}_x = \frac{\mathbf{M}_{10}}{\mathbf{M}_{00}}; \mathbf{c}_y = \frac{\mathbf{M}_{01}}{\mathbf{M}_{00}}; \tag{2.2}$$

The algorithm described in Figure 2.3, performs object tracking using a single tracking box and it is only capable of tracking at most one object of the same color any given time. As our task of fingerspelling tracking involves tracking a head and two hands(figure 2.4), the increased number of object brings new challenges.Thus we extend the Camshift algorithm presented above to detect, track and handle three targets with similar color distributions. The algorithm assumes that a signer comes in front of the camera and starts signing with one or two of his hands. Cases such as hands leaving and entering the sign space, hands merging with each other, hands merging with the face and hands seperating are handled using a heuristic algorithm described in Figure 2.1.2.

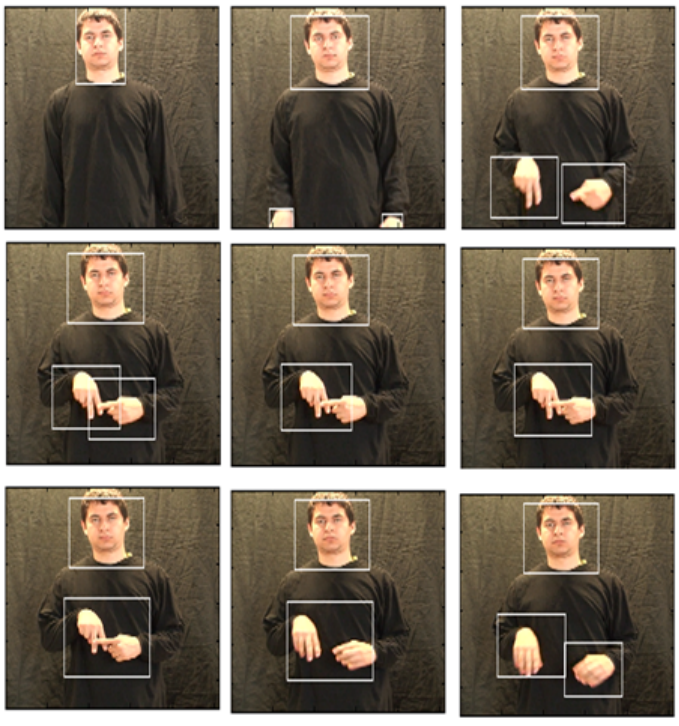


Figure 2.4. Multi-Object Head and Hand Tracking

1. If an adaptive color histogram for the object to be tracked is not present, run Automatic Adaptive Color Histogram Generator.
2. Using the Viola-Jones head detection algorithm try to detect a head. If detection is not successful go to Step 2.
3. If the right hand is not being tracked, run Hierarchical Redetection algorithm on the right side of face.
4. If the left hand is not being tracked, run Hierarchical Redetection algorithm on the left side of face.
5. If tracking boxes centroids are closer than a third of the sum of their radii, remove one of the boxes preventing convergence.
6. If the right hand is not being tracked and the left hand is, designate the tracked hand as right hand.
7. If the hands are being tracked and the face is lost, remove all boxes and go to step 2.

Figure 2.5. Hand Labeling Algorithm

The method involves the use of a hierarchical re-detection module, which hierarchically divides the probability distribution image into smaller sub-images. The largest object whose center is closer than a certain threshold on two consecutive image levels is chosen as the found object. The algorithm is described in Figure 2.6

**HierarchicalRedetect**$(I[m,n], Box[x,y,w,h], P_{thresh}, W_{thresh})$

**Require:** I[m,n] <- Grayscale m x n color probablity distribution image)

**Require:** Box[x,y,w,h] <- Rectangle containing possible tracking box of object

1. $If(w < W_{thresh} || h < W_{thresh})$
   *return* empty tracking box
2. $If$(ZerothImageMoment$(I, Box) < P_{thresh}$)
   *return* empty tracking box
3. Divide Box into 4 equal subregions $Box_{Sub}$
4. $Box_{Sub}$ = Meanshift($Box_{Sub}$) localize objects in boxes
5. for all $Box_{Sub}$, call HierarchicalRedetect(I(m,n),$Box_{Sub}$)
6. for all $Box_{Sub}$ starting from the largest
7. $If$ (center of $Box_{Sub}$ is closer to center of $Box$ then $W_{thresh}$)
   *return* $Box_{Sub}$ as object tracking box

Figure 2.6. Hierarchical Redetection Algorithm

### 2.1.3. Skin Color Segmentation

While the Camshift algorithm described in Section 2.1 is useful for locating the hands and its approximate borders, it does not segment the hands from the background. The task of hand segmentation therefore needs to be performed on the tracked hand images.

As the human hand has many degrees of freedom, it has no definite shape or movement patterns which can be used to stably recognize it. Therefore, color cues remain the most frequently applied feature for hand segmentation. In color based

segmentation, usage of different color-spaces and modeling methods makes optimizing skin color segmentation possible. While we know that there can be no magical skin color model to differentiate the skin color of all humans from that of every possible background; the usage of effective models with some minor background restrictions allows for successful segmentation. Using illumination preserving color spaces such as Hue and Saturation channels of HSV or normalized RG, the effect of brightness in skin-color representation can be reduced without losing descriptive color information.

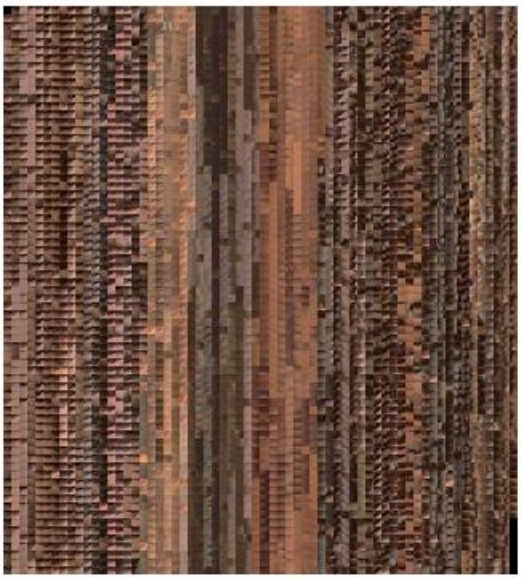


Figure 2.7. Skin-Color Sample Image

In performing skin color segmentation, one common method is to generate skin color models from sample images and use them to calculate probability distribution images. Figure 2.7 shows a sample skin color sample we collected from one of our dataset subjects. In our experiments with various models, we saw that single Gaussian distributions were usually insufficient to represent the skin color of more than one user. On the other hand, using Gaussian Mixture Models with as many mixtures as users provided good segmentation results. However, as predictions with Gaussian Mixtures were computationally more expensive than histogram lookup operations, we preferred the use of color histograms as our skin color models.

In our online fingerspelling recognition system, we have used two segmentation methods with different methods. In the first method, we used a $32x8$ hue-saturation skin color histogram to threshold hand shapes. In the second method, we segment hand shapes using region growing. We then perform a logical and operation on the resulting images to obtain the fused segmentation image.

2.1.3.1. Histogram Back-propagation. Color histograms are used to represent images by grouping the number of pixels which fall in fixed color ranges. By quantizing similar colors into the same histogram bins reduces computational and space complexity. In the Histogram Back-propagation method [39], a histogram representing the color distribution of an object is used to create a color probability image. Then, each pixel in the image is associated with the probability value in the corresponding histogram bin. Equation 2.3 demonstrates the calculation of the probability distribution Image $PI(x, y)$, using the histogram $H(hue, sat)$ and image $I(x, y)$.

$$\mathbf{PI}(\mathbf{x}, \mathbf{y}) = \mathbf{H}(\mathbf{I}(\mathbf{x}, \mathbf{y}).\mathbf{hue}, \mathbf{I}(\mathbf{x}, \mathbf{y}).\mathbf{sat}); \tag{2.3}$$

Normalizing the histogram using Equation 2.4 makes sure that the image is in the scale [0-255], yielding a gray-scale probability image. After obtaining a probability distribution image, we threshold it with a fixed threshold of 30 to segment foreground and background pixels.

$$\mathbf{PI_{normalized}}(\mathbf{x}, \mathbf{y}) = min(\frac{\mathbf{PI}(\mathbf{x}, \mathbf{y})}{\mathbf{max}(\mathbf{PI})} * 3, 255); \tag{2.4}$$

2.1.3.2. Region Growing. The region growing algorithm is a skin color based segmentation algorithm that attempts to detect the binary shape masks of connected components.As the histogram back-propagation method is greatly dependent on the

skin color model, a failure in histogram generation directly affects segmentation accuracy. Therefore, to improve our shape probability images, we have made use of region growing segmentation.

In this method, given a bounding box and a probability image containing probable hand locations and some seed points, we attempt to segment hand images from background pixels. The region growing operation, like the skin color model, is performed in the Hue and Saturation channels of the HSV color-space. The seed locations are chosen using the probability image. Since all seed-points are not guaranteed to be ideal, we attempt to eliminate background seed-points. By comparing the HS color values of seed-points to the mean HS, we decide if a seed point should be used in expanding the shape mask.

Then, having selected our initial key-points, we start expanding the shape masks. From each seed point, we attempt to expand the shape mask in the direction of its eight neighbors. If a neighbor of the seed-point has closer Hue and Saturation values to the center, we accept the neighbor as a foreground pixel. Although the threshold that is used to compare the neighboring pixels to grow the foreground region depends on the distribution of skin color pixels, we have chosen to assign a static value and present it to the user as a parameter to allow the fine tuning of segmentation quality. One drawback of Region Growing is that it often tends to fail when hands occlude other skin colored objects. Therefore, to prevent the shape-masks from expanding too much, we have added a control condition. We have added a cutoff threshold that stops growing the region when the search area grows to %60 of the entire tracking box.

# 3. FEATURE EXTRACTION

In this chapter, we describe the feature extraction methods we have employed to mathematically represent hand gestures. In order to distinguish fingerspelling signs shape and motion of hand gestures must be uniquely represented. The ultimate aim of a fingerspelling recognition system is to distinguish hand gestures in a sequence of images. Therefore, factors such as varying signer hand shapes, signer performances and camera conditions should have a minimal effect on hand gesture descriptors.

As our fingerspelling recognition systems aim to be realtime, we did not employ accurate 3D hand model fitting techniques. Instead we have focused on the usage of different appearance based features, that can be acquired from 2D frontal images. We chose our appearance based features based on their strengths on representing different properties of hand shapes, such as shape, texture and external contours. In this study, we used Shape based features such as image moments, texture representation based features such as Local Binary Patterns and external contour based features such as Elliptic Fourier Descriptors and Radial Distance function.

## 3.1. Image Moments

Used in numerous computer vision applications as shape descriptors, the invariant moments of Hu are computed out of binary shape masks belonging to segmented shapes[60]. They are derived from Central Image Moments, which can be derived from raw image moments. For a two dimensional function $f(x, y)$ the moment of order $(p+q)$ can be calculated using equation 3.1.

$$\mathbf{M_{pq}} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x^p y^q f(x, y)dxdy \tag{3.1}$$

In a two dimensional grayscale image $I(x, y)$ these raw moments can be calculated as in equation 3.2:

$$\mathbf{M_{ij}} = \sum_{x} \sum_{y} x^i y^j I(x, y) \tag{3.2}$$

Using these moments, central moments for an image can be calculated using Equation3.3. These moments provide the raw moments translation invariance[22].

$$\mu_{\mathbf{pq}} = \sum_{x} \sum_{y} (x - \hat{x})^p (y - \hat{y})^q f(x, y) \tag{3.3}$$

To ensure scale invariance the centralized moments can be normalized by dividing each by a factor of the zeroth moment as seen in Equation 3.4:

$$\mu_{\mathbf{ij}} = \frac{\mu_{ij}}{\mu_{00}^{1+\frac{i+j}{2}}} \tag{3.4}$$

From the central normalized moments, Hu derived seven invariant moments. The first six of Hu's Moments are rotation, scale and translation invariant, whereas the seventh moment is skew orthogonal invariant, a rather desired attribute to distinguish mirror images of identical objects. Hu's Moments are calculated in Equations 3.5-3.11:

$$\mathbf{Hu}_1 = \mu_{20} + \mu_{02} \tag{3.5}$$

$$\mathbf{Hu}_2 = (\mu_{20} - \mu_{02})^2 + (2\mu_{11})^2 \tag{3.6}$$

$$\mathbf{Hu}_3 = (\mu_{30} - 3\mu_{12})^2 + (3\mu_{21} - \mu_{03})^2 \tag{3.7}$$

$$\mathbf{Hu}_4 = (\mu_{30} + \mu_{12})^2 + (\mu_{21} + \mu_{03})^2 \tag{3.8}$$

$$\begin{aligned}\mathbf{Hu}_5 = (\mu_{30} - 3\mu_{12})(\mu_{30} + \mu_{12})[(\mu_{30} + \mu_{12})^2 - 3(\mu_{21} + \mu_{03})^2] + \\ (3\mu_{21} - \mu_{03})(\mu_{21} + \mu_{03})[3(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2]\end{aligned} \tag{3.9}$$

$$\mathbf{Hu}_6 = (\mu_{20} - \mu_{02})[(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2] + 4\mu_{11}(\mu_{30} + \mu_{12})(\mu_{21} + \mu_{03}) \tag{3.10}$$

$$\begin{aligned}\mathbf{Hu}_7 = (3\mu_{21} - \mu_{03})(\mu_{30} + \mu_{12})[(\mu_{30} + \mu_{12})^2 - 3(\mu_{21} + \mu_{03})^2] - \\ (\mu_{30} - 3\mu_{12})(\mu_{21} + \mu_{03})[3(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2]\end{aligned} \tag{3.11}$$

In addition to these descriptors, we have experimented adding two more features to this feature vector as hand orientations relative to each other and the face. However, due to the unique properties of fingerspelling gestures allowing the user to perform the same gesture on different parts of the sign space, a consistency of hand orientations was not observed even during the performance of a single user, thus leaving us no option but to omit this feature.

## 3.2. Elliptic Fourier Descriptors

Calculated from the external shape contour of an object, these descriptors are used to represent an object in the frequency domain. Using a set of ellipses, the closed external outline of a shape can be transformed to yield a shape spectrum [61, 51]. The lower frequency descriptors contain information about the general features of the shape, and the higher frequency descriptors contain information about finer details of the shape. Although the number of coefficients generated from the transform is usually large, a subset of the coefficients is enough to capture the overall features of the shape.

For an n-harmonic elliptic Fourier descriptor representation of a 2-D closed shape is given in Equation 3.12. In the equation, the center of the curve is $(a_0, c_0)$. $(a_k, b_k, c_k, d_k)$ for k=1....n are elliptic Fourier coefficients of the curve up to n Fourier harmonics. T is the perimeter of the closed curve.

$$\begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} a_0 \\ c_0 \end{bmatrix} + \sum_{k=1}^{N} \begin{bmatrix} a_k \cos(\frac{2k\pi t}{T}) + b_k \sin(\frac{2k\pi t}{T}) \\ c_k \cos(\frac{2k\pi t}{T}) + d_k \sin(\frac{2k\pi t}{T}) \end{bmatrix} \tag{3.12}$$

The elliptic fourier descriptors $(a_k, b_k, c_k, d_k)$ are calculcated for each point k of the curve as seen in the Equations 3.13 - 3.16.

$$a_k = \frac{1}{2k^2\pi^2} \sum_{i=1}^{n} \frac{\Delta x_i}{\Delta t_i} \left[ \cos(\frac{2k\pi t_i}{T}) - \cos(\frac{2k\pi t_{i-1}}{T}) \right] \tag{3.13}$$

$$b_k = \frac{1}{2k^2\pi^2} \sum_{i=1}^{n} \frac{\Delta x_i}{\Delta t_i} \left[ \sin(\frac{2k\pi t_i}{T}) - \sin(\frac{2k\pi t_{i-1}}{T}) \right] \tag{3.14}$$

$$c_k = \frac{1}{2k^2\pi^2} \sum_{i=1}^{n} \frac{\Delta y_i}{\Delta t_i} \left[ \cos(\frac{2k\pi t_i}{T}) - \cos(\frac{2k\pi t_{i-1}}{T}) \right] \tag{3.15}$$

$$d_k = \frac{1}{2k^2\pi^2} \sum_{i=1}^{n} \frac{\Delta y_i}{\Delta t_i} \left[ \sin(\frac{2k\pi t_i}{T}) - \sin(\frac{2k\pi t_{i-1}}{T}) \right] \tag{3.16}$$

For each Fourier harmonic calculated, we obtain an ellipse that yields four invariant features [62]. The first two features are the major and minor axis lengths of the

calculated ellipses. The latter two features can be derived from the first two features and vice versa. In our experiments with hand contours, we found ten harmonics sufficient to represent hand gestures, yielding 40 sized feature vectors. The formulas for the features are given in Equations 3.17- 3.20

$$\mathbf{A}_k^2 = \frac{I_k + \sqrt{I_k^2 - 4J_k^2}}{2} \tag{3.17}$$

$$\mathbf{B}_k^2 = \frac{J_k^2}{A_k^2} \tag{3.18}$$

$$\mathbf{I}_k = a_k^2 + b_k^2 + c_k^2 + d_k^2 \tag{3.19}$$

$$\mathbf{J}_k = (a_k d_k) - (b_k c_k) \tag{3.20}$$

## 3.3. Local Binary Patterns

Local Binary Patterns (LBP) were introduced by Ojala [49] for texture representation. The LBP is used across various computer vision fields (e.g. image synthesis, light normalization, face detection, face/expression recognition). It has been successfully used for hand detection in cluttered images [2]. We use LBP for hand shape description.

First a LBP image is computed. The algorithm moves a defined patch along all the pixels in an image. For each pixel, a unique value depending on its neighbours illumination is calculated. The LBP image can be calculated for different patch size and distributions. We use a circular 8-neighbourhood patch with radius one and two

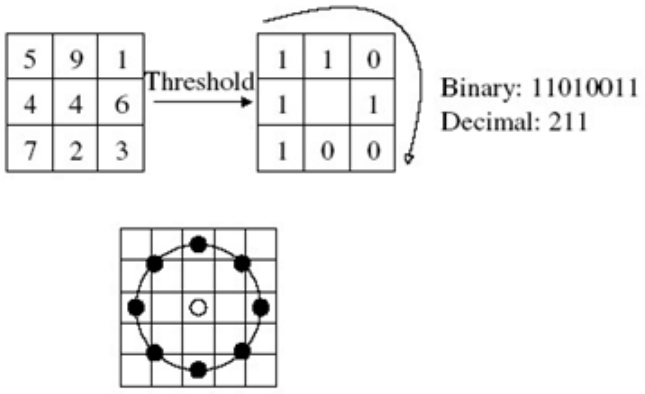



Figure 3.1. Local Binary Pattern Calculation

pixels. An overview of LBP calculation and the choice of neighboring pixels for a LBP radius of 2 can be seen in 3.1. If the brightness of a pixel in the patch is greater than or equal to the evaluated pixel's brightness we assign a binary label 1 to the corresponding location in the patch. If the patch brightness is lower than the evaluated brightness, we assign label 0 to it.

In each patch, we evaluate eight locations which yields us eight binary numbers. Writing these numbers side by side in order, we obtain an 8-bit number. This number converted to octal mod and is assigned to the location of the evaluated pixel. Then the patch moves to the next pixel and calculates a value for the entire image. Next, we compute a histogram of the LBP image, which we use as a descriptor vector. In regular LBPs there are 256 histogram bins, each bin representing one pattern. In practice it has been shown that all the patterns are not important for recognition. By removing these low frequency patterns, we can reduce the dimensionality of the feature vector. Most of the information is clustered in patterns that have two or fewer changes between 0 and 1s in binary form. Such LBPs are called uniform LBP's [49] and they can be seen in figure 3.2. There exist 58 such patterns and all the other patterns are moved to the 59th bin of the histogram. This means that for a uniform LBP, the feature vector is of size 59 .We implemented both uniform and non-uniform LBPs with patch radii of one and two.

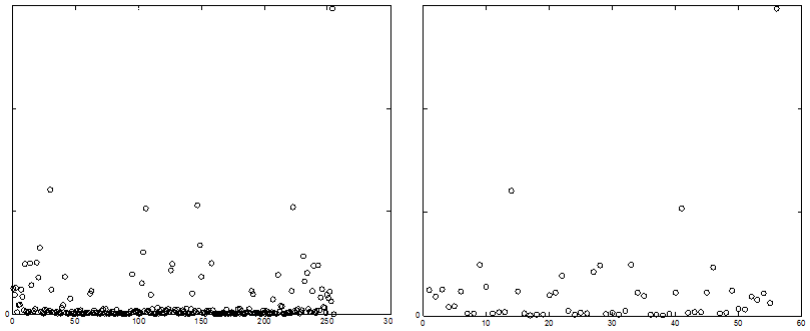


Figure 3.2. Uniform and non-uniform LBP descriptors

Being a local shape descriptor, LBP descriptors are often calculated in one of two ways. In the first method, LBP features are calculated for an equally spaced grid. As this method will generate multiple descriptors for a single object, a single feature can be obtained to represent the image using bag of visual words histograms. In the second method, whole hand histograms can be calculated from scale normalized hand images. Since the former method requires much more space to cluster each grid point to a visual word, we have opted to use the latter method in our studies. In our representation of hand shapes, we chose to resample our hand images to a fixed width, thus yielding us 58 sized uniform LBP features to represent each normalized hand image.

## 3.4. Radial Distance Function

The radial distance function method, presented in [50] is another external contour based method. Using the distance of a seed-point to the closest background pixel in all directions, a description of the image is given. The ideal choice of seed-points can vary depending on the context of the shape. For hand recognition, centers of mass or elbow hand intersections are commonly used seed-points. The calculation of feature vectors for a hand shape can be seen in Figure 3.3

The calculated image descriptors are invariant to translation and they can be made invariant to size and rotation. Scale invariance is achieved by dividing each

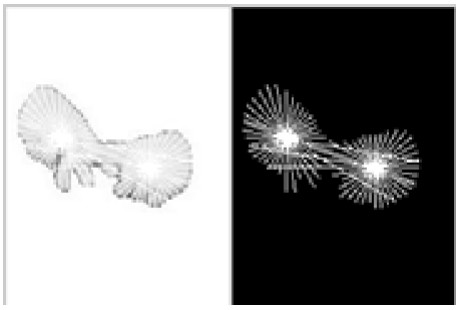


Figure 3.3. Radial Distance Function Calculation

distance feature to the sum of all distance features in the vector. Rotation invariance can be achieved by choosing the angle with the smallest radial distance as the origin point for each seed point. While describing an isolated hand, the radial distance function is an efficient measure as it is possible to represent finger locations and notable extensions of the hand. However, when describing blobs consisting of not completely overlapping, but touching hands, obtaining a point which has a straight line distance to both hands may not be possible. For this reason, we attempt to find the centers of gravity belonging to both hands. Using image moments, we calculate the parameters of the smallest ellipse that covers both hands using the formulas 3.21-3.23.

$$\mathbf{a} = \frac{M_{20}}{M_{00}} - x^2; \mathbf{b} = 2(\frac{M_{11}}{M_{00}} - x_c y_c); \mathbf{c} = \frac{M_{02}}{M_{00}} - y^2 \tag{3.21}$$

$$l_1 = \frac{(a+c) + \sqrt{b^2 + (a-c)}}{2}; l_2 = \frac{(a+c) - \sqrt{b^2 + (a-c)}}{2} \tag{3.22}$$

$$\phi = \frac{1}{2} * \tan_{-1}(\frac{b^2}{a-c}) \tag{3.23}$$

In the calculations, $l_1$ and $l_2$ are the principal axes of the bounding ellipse centered on the centroid of the image and $\phi$ is their rotation angle. By dividing the image using the minor axis $l_2$ as a separator, we effectively divide the blob into two approximately

equal parts belonging to different hands. After calculating the centroid of each part using image moments, we obtain seed locations for two radial distance functions that are sufficient to describe a hand blob consisting of two hands. The calculation of radial distance function for 2 hands can be seen in Figure 3.4.

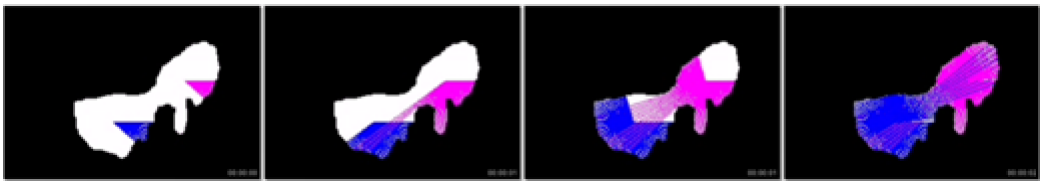


Figure 3.4. Radial Distance Function Calculation For Two Hands

In our experiments, we have modeled our hand shapes using 72(360/5) distance calculations from two seed points. Thus, for each image, we obtained descriptors of size 144. The extracted feature sets are used in the subsequent classification stage.

# 4. TEMPORAL SEGMENTATION AND CLASSIFICATION

In this chapter, we discuss our fingerspelling recognition methodologies. In chapter 3, we elaborated on the spatial modeling of hand gestures to represent sign language gestures by obtaining individual descriptors for each frame containing a gesture. However, for the task of online sign language recognition, the time component of gestures also needs to be taken into account. As the overall success of the fingerspelling recognition task depens on the systems ability to distinguish gesture sequences correctly, classification models that take both shape based and temporal sign language features are necessary.

In our continuous fingerspelling recognition scenario, we attempt to resolve two separate tasks using temporal segmentation. First, given a continuous sequence of images containing hand gestures, we try to designate when a sign starts and finishes. Secondly, having split sequences into chunks of images containing a single gesture, we try to decide on the letter, which is best suited to represent the entire sequence. To solve these issues, we have experimented with two tiered classification approaches to handle continuous sign language recognition.

## 4.1. Keyframe Detection

An important challenge in continuous fingerspelling recognition is the problem of designating which frames the signer actually performs a gesture and which frames the signer does not. We use a keyframe based system, where hand motion and motion blur are utilized to automatically separate keyframes from transition frames. In a given sequence of images, we classify the images into two categories depending on its semantic content. If the frame contains a snapshot of a hand gesture which can be used to represent a concept, we call the frame a keyframe. However, if the frame does not contain any hand gestures or is a transition between two gestures, we call the frame

a transition frame. This information is crucial to temporally segmenting the frames containing semantic meaning.

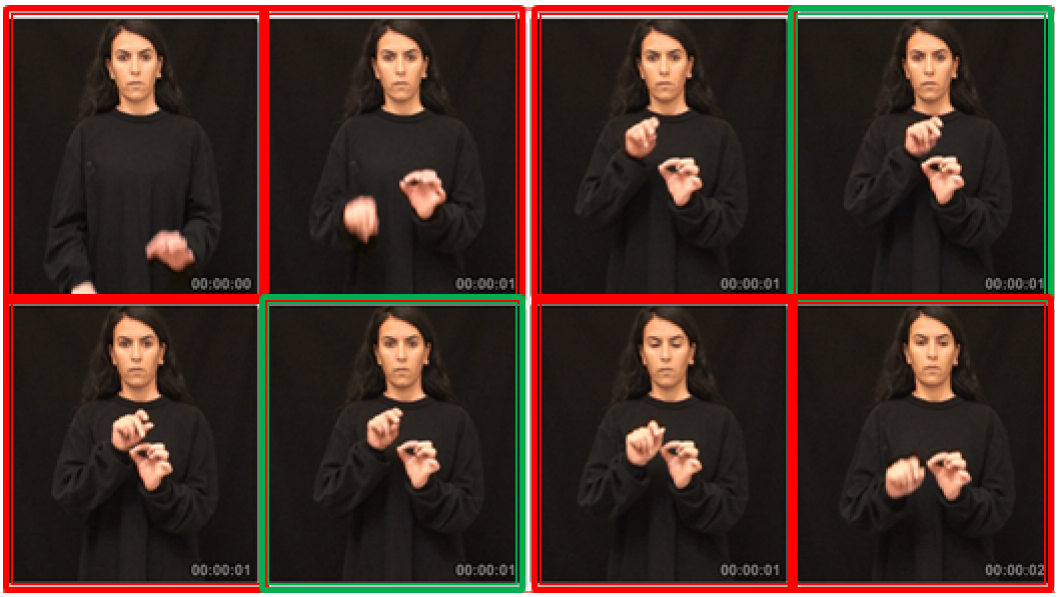



Figure 4.1. Hand gesture performance split into keyframes (green) and transition frames (red)

4.1.0.3. Hand Movement. An effective measure for keyframe detection is the motion of signer hands. In static gestures where the signer displays image snapshots, begining and end of gesture performances can easily be inferred from the motion of signer hands. In dynamic gestures, although differentiating between transition frames and movement between the phases of the same gestures can be difficult, the presence of momentary stops like changing hand movement detection still makes it possible.

While signing, the overall motion of the human hands can be classified into two categories. The first is the global motion of the hands which is its relative position with respect to the camera. The global motion is obtained through keeping track of the tracking boxes in the camshift algorithm. The second kind of motion is the local hand motion, which can be summarized as the change in the shape of the hand. We keep track of the change in the hands external contour.

For the classification of movement features, we use two methods, one gaussian and one heuristic based. In the gaussian method, we train the system with positive and negative keyframe examples. However, in the online system, the heurisitic method where the user can manipulate the thresholds to adjust sytem performance to his speed was deemed more usable, as it was easier to calibrate for different users.

4.1.0.4. Motion Blur. In a video or image capture device, the captured frame does not always represent a single instant. To capture a single frame, the camera captures the scene over a certain period of time designated by its shutter speed. Thus, any object that is displaced relative to the camera during the frame capture interval appears to look blurred along its direction of relative motion. In our sign language recognition setup, the user constantly moves his/her hands to perform certain hand gestures. Therefore, depending on the capture hardware there usually is some motion blur present due to the mobility of hands.

In sign language recognition systems, a major problem that hampers the systems accuracy is motion blur, which is caused by the rapid movement of hands while moving from one gesture to another. Since the presence of motion blur prevents accurate segmentation in a sequence of gestures, it is desirable to choose the frame with the minimal blur to maximize segmentation and therefore recognition accuracy.

In addition, in a hand gesture recognition setup where the signer performs continuous gestures, motion blur becomes a possible feature in keyframe selection. In keyframe selection, the goal is to distinguish probable sign language hand gestures from transition gestures that occur while moving hands from one gesture to another. Although the motion of hands seems to be the key distinguishing feature for such a case, individual usage of such a feature does not produce extremely accurate results as the motion of hands varies with the signer, with the accuracy of hand tracking, with the current sign and with the co articulation of previous and following signs. Therefore, supplementing hand motion with a feature that is less dependent on the above

conditions and more dependent on the quality of captured images becomes useful in the selection of meaningful and accurate key frames.

In this study, exploring the features presented in [55], we attempted to solve a more constrained and focused problem. Although partial blur detection is a complicated problem that needs to take into account global changes in saturation and illumination in an image, our task in hand gesture keyframe selection is relatively more constrained. As our objective is determining the presence of blur in the signers hands, we are only interested in detecting blur in a constrained area. In addition, since we are given the possible contours of the hands from the previous hand tracking stage, the amount of useless background information can be severely reduced.

For detecting motion blur, we employ a method that makes use of the gradient strengths of hand edges. Compared to unblurred images, a major characteristic of blurred images is that edges tend to be smoother and contain smaller gradient values. Therefore, focusing on the distribution of gradient values in a certain image patch can give us an idea about the presence of partial motion blur.

Figure 4.2. Motion blur detection in hand images

Firstly, we obtain hand images as seen in Figure 4.2 and use the image derivative masks in Equation 4.1 to obtain the derivative images $I_x$ and $I_y$.

$$Dx = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}; Dy = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \tag{4.1}$$

Then we convolve the images to obtain the derivative images $I_x$ and $I_y$. From those images, using a Gaussian window, we obtain the smoothed squared image derivatives $I_x^2$ and $I_y^2$. The corresponding images can be seen in Figure 4.3.

Then we obtain the trace of the image(4.4) with the Equation 4.2.

$$A = k * (I_x^2 + I_y^2)^2 \tag{4.2}$$

Figure 4.3. From left to right : I, $I_x$, $I_y$, $I_x^2$, $I_y^2$

Figure 4.4. Trace Image

As the trace image yields the most significant results in the edge areas that separate the hands from the background, we compute the trace image for small windows around the hand contour. We capture nxn sized small images around the hand contours

that are at least $\sqrt{(n^2)}/4$ apart from each other. After obtaining the gradient trace values from these boxes, we extract two different features for blur detection. The first feature is the span of the gradient distribution histogram. Since unblurred edges tend to be sharper and contain relatively small gradients, they tend to have a distribution with a smaller variance. On the other hand, blurred edges tend to have a larger variance as they consist of more zero and large valued gradients. As suggested by Liu, we use Gaussian mixture models to fit a two component gaussian model to the gradient distribution. In Figure 4.5, the distribution with the smaller variance represents the background pixels while the edges (both blurred and unblurred) tend to have a distribution with smaller peak and a larger variance. Thus, we make use of the variance of the smaller peak as a feature to classify image patches as blurred and unblurred.

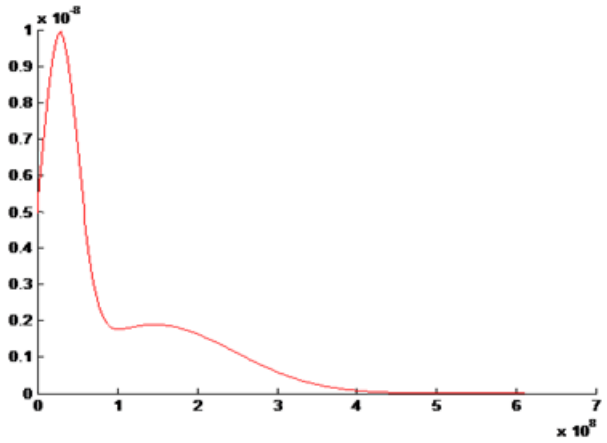



Figure 4.5. The distributions of Gradient Magnitudes

In addition, we also make use of the local difference in gradient vectors to classify blurred and unblurred images. In our sign language recognition scenario, hands are often partially blurred instead of being totally distorted. Therefore, although the difference between the maximum and minimum gradient values seem to be meaningless when the entire patch is blurred and unblurred, it becomes a distinguishing element when half of the image is blurry and the other half is unblurred.

## 4.2. Hand Gesture Classifiers

In this section, we describe the base classifiers which we have chosen to use with our hand gesture features. The hand descriptors which we use to model 88 classes consist of 14 to 144 length vectors.

In our dataset, for some classes, the ratio of (number of samples) / (feature vector size) is smaller than 1. As we do not have enough data to generate additional validation sets, we perform validation on our training sets using k-fold cross validation(k=10). In 10-fold cross validation, the training set is divided into k random subsets. For each subset, we construct a classification model using the data belonging to the remaining k-1 subsets. We report the average of these k-folds as our training accuracy. The cross validation method only inputs the training set. The test images are only used with the model whose parameters are optimized by cross validation.

Using cross validation benefits the process of model construction in two different ways. Using cross validation, adaptive comparison of model parameters improves the robustness of the model towards new data. In addition, through cross validation, it becomes possible to see if training a model with some of the sub-training sets actually performs better than using the entire training set.

### 4.2.1. Gaussian Classifier

The first classifier we have employed is the linear classifier known as the Gaussian (minimum-error) classifier. For this classifier, we have assumed that the data is normally distributed and used equal covariance matrices. The reason for our covariance matrix choice is the fact that for some classes and features, our feature vector sizes greatly outnumbers our sample size. The equation for calculating the log likelihood $g_i(x)$ is given in Equation 4.3. While calculating likelihoods for each class we assume all covariane matrices are the same ($\Sigma_i = \Sigma$)

$$g_i(x) = -\frac{1}{2}(x - \mu_i)^T \Sigma_i^{-1}(x - \mu_i) \tag{4.3}$$

While training the classifier, we perform a 10-fold cross-validation. Since we want to estimate and minimize test error without using the test set in system training, we split the training set into 10 folds and take the one with the best performance. If the classifier with the best performance performs much better than the average result of cross validation, that subset of the training set is preferred over using the entire training set.

### 4.2.2. K-Nearest Neighbour Classifier

The k-Nearest Neighbour method is a non-parametric robust classification algorithm. This method assigns the input data x to the class which has the most number of elements among the k-closest neighbors of $x$. In kNN classification, a distance metric needs to be chosen to represent the distance between two samples. In this study, we have opted to use Euclidean Distance as it did not provide a significant performance decrease over Chi-square distance. Using this metric, an input test samples distance to all training samples are calculated. Then, this input sample is assigned to the class having the most examples among the k closest neighbors. In kNN classification, all neighbors are given equal priorities in voting. Thus the number k needs to be an odd number in order to avoid ties while voting.

### 4.2.3. Support Vector Machines

Support Vector Machines(SVM) are robust, non-probabilistic classifiers. Given a data that belongs to multiple classes, the SVM classifier tries to find optimal separating hyperplanes between class pairs one by one. SVM tries to find the optimal hyperplane that separates the data while trying to maximize its distance to the closest data point on either class.

While training the SVM's we aim to calculate a discriminant for a set of points represented as $x$ belonging to the class $k_i \in \{+1, -1\}$. Using kernel functions, the SVM maps $x$ to a higher dimensional feature space. The decision function for the kernel can be written as seen in equation 4.4 as described in[63].

$$g_i(x) = (\sum_i \alpha_i k_i K(x_i.x) + b) \tag{4.4}$$

Support vector machines are two class classifiers, which are not directly capable of handling multi-class classification problems. To use this method with multi class datasets, two approaches known as one-vs-rest and one-vs-one classifiers with voting are widely used. The implementation of LibSVM [64], which we use in this study makes use of the one-vs-one approach, where SVM's between each class are calculated for a given training sample. Then by using voting among all the results, the system decides on the class that includes the given sample in the largest number of iterations.

## 4.3. Temporal Segmentation

In temporal segmentation of hand gestures, the system integrates two different types of information. Spatial and appearance information contains descriptive information on the shape and possible meaning of individual frames. Temporal information, on the other hand, can be used to ascertain which frame follows which in a sequence and which of these are actually meaningful keyframes. By incorporating these two types of information with different inputs, the system performs several different tasks.

In isolated fingerspelling gesture videos, the system starts performing a single gesture from a start pose and ends the gesture by returning to the start pose. The beginning and end of hand gestures are manually annotated to indicate when a gesture starts and ends. On the other hand, continuous videos do not contain manual annotation. In these videos, a signer begins signing by performing letter after letter

and only returns to the start pose when he finishes signing a word. Thus, to perform fingerspelling recognition with continuous videos, identifying the start and end of each gesture is necessary.

### 4.3.1. Keyframe Based Gesture Detection

In order to segment gesture sequences from a continuous sequence of frames, we try to exploit the uniform performance characteristics of fingerspelling signs. From a temporal point of view, gestures can be classified into two different categories as static and dynamic. In static gestures, the user only moves his/her hands when starting and finishing the gesture. As can be seen in figure 4.6, the hands are static only during keyframes.

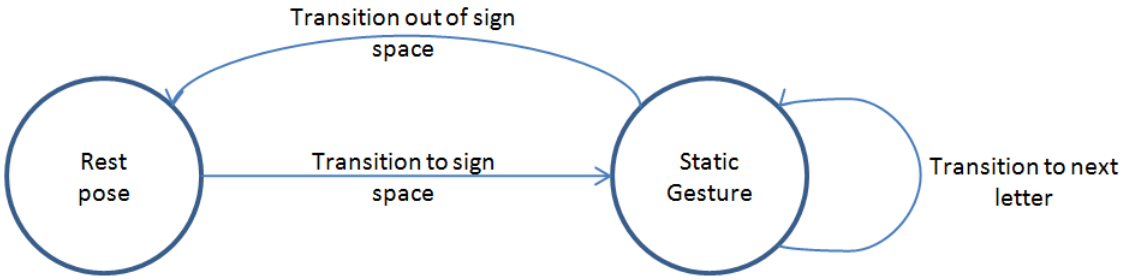



Figure 4.6. Continious Performance of Static Gestures

Unfortunate from a recognition point of view,some fingerspelling gestures are dynamic. Detecting the start and end positions of dynamic gestures is a more challenging task then annotating static gestures. As can be seen from figure 4.7, dynamic gestures include movement both between and during gestures. Therefore, a necessity of distinguishing gesture movements from transition movements during temporal segmentation arises.

The dynamic gestures, present in our multilingual dataset, contain movements in one or more segments. Thus, motion based keyframe detection yields positive results either when the hands make a complete stop at a static pose or when hands change direction between two dynamic poses. As a result of coarticulation caused from moving hands to the pose start and end positions, hands change direction at least two times while performing a gesture. Those hesitance points of time are actually candidates to serve as keyframes, since the hands move relatively slower and much less numbers of pixels are smeared into background.

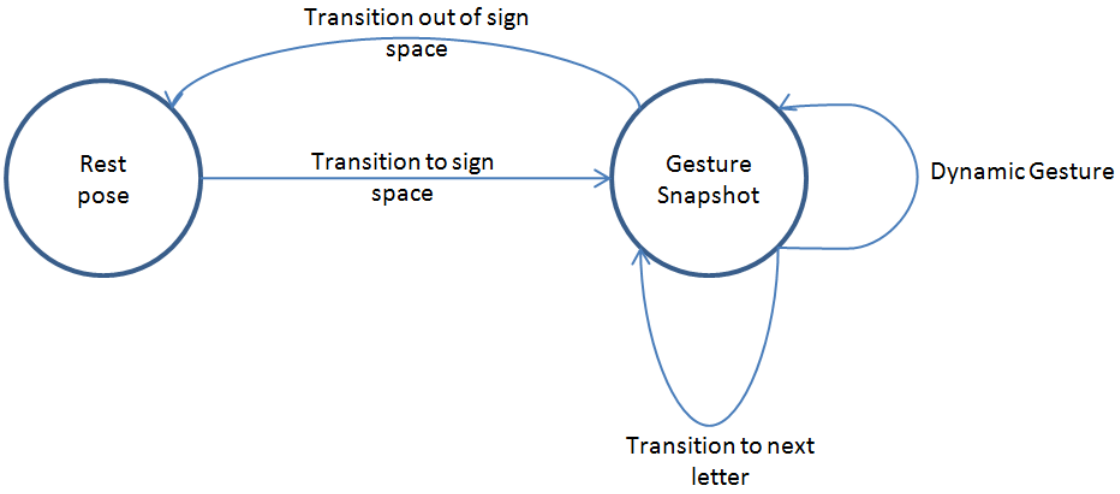



Figure 4.7. Continious Performance of Dynamic Gestures

The model displayed in figure 4.8 attempts to segment gesture start and end positions in a continuous image sequence. In a continuous sequence of images marked as keyframes and transition frames, the first encountered keyframe is designated as the sequence start frame. Then, each new frame is added to the sequence until a continuous transition frame of certain length arrives. In this method, we make use of two thresholds to optimize segmentation. First, by making sure that the recognized sequence is at least of length $(0.30 seconds/system fps)$ frames, we attempt to prevent short noise from being recognized as a sequence. Secondly, we set the minimum length of continious transition frames used to determine the end of a sequence to a fixes threshold of $(0.20 seconds/system fps)$. This way, movements of dynamic gestures are segmented from transitions between gestures.

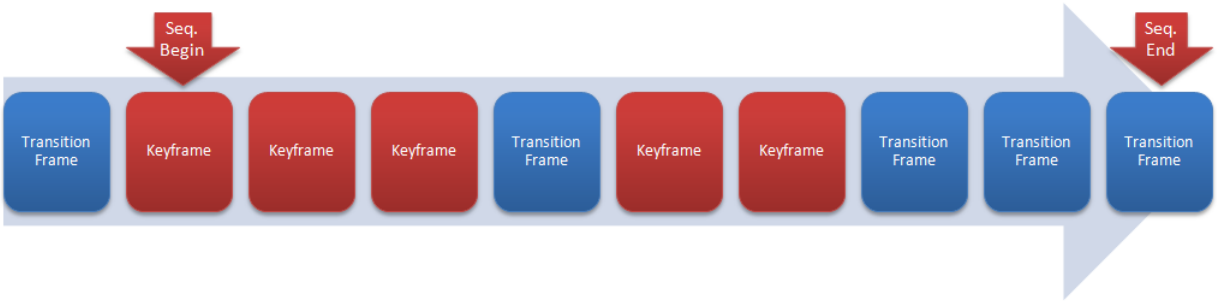



Figure 4.8. Keyframe based gesture start and end point detection

The thresholds setting minimum transition and minimum sequence length were chosen based on the fingerspelling performance of our native signer subjects performing approximately 3 signs per second. Choosing higher thresholds for slower, non-native signers prevents signer movement noise from being recognized as a gesture or a single gesture from being interpreted as two or three gestures.

### 4.3.2. Fusion Techniques

In our study, we describe fingerspelling hand gestures using different appearance based models over time. Both the appearance features and the time dimension effect the meaning conveyed by the gesture. So the combination and synchronization method of these information among and with each other affects overall system performance.

In our system we make use of four different appearance based classifiers with different strengths. Therefore in our search to obtain the highest appearance based fingerspelling recognition accuracy, we have attempted to compare the fusion of these features on isolated hand gesture keyframe classification.

As a first attempt, we applied feature level fusion by normalizing feature vectors, concatenating them and performing PCA on the results. There were certain challenges at this stage mostly due to feature incomparability. We had four features of lengths 14,80,96 and 144 of different scales. While some of these features benefited from tech-

niques like z-normalization, others such as EFD whose internal feature weights needed preservation performed worse in a combined setting.

In another method, we applied score level fusion on our appearance based classifiers by handling each feature separately. As discussed in section 4.2 we have used kNN and linear discriminant classifiers to classify each feature separately. Then, according to the training/cross-validation accuracy of each set, we performed score level weighted majority voting to reach a final decision.

To fuse appearance based results with the time knowledge, which consists of sequence and keyframe information, we have made use of three methods. The first two methods involve classifying each frame separately and reaching a decision by modeling the frame-wise results of a sequence.

In the first method, we make a sequence-wise decision using a majority voting approach. In this method we only take into account the frames that are designated as keyframes. By looking at a sequence of classified pixel-wise class labels, the decision on the sequence is made using the most numerous class its frames belong to.

A second approach we have used is the modeling of pixel-wise classification results using discrete HMM's. With this approach, we classify the frames belonging to each gesture using a pixel-wise classifier and obtain output label sequences. Then, inputting these discrete sequences into HMM's we decide on the class which the sequence is more likely to belong to.

Lastly, we make use of continuous HMM's to perform sequence level classification directly. Using Gaussian Mixture Models, we model each feature of appearance based feature vectors and construct hidden Markov models from these features. Then, a decision on a given input sequence is made by finding which class the best fitting model belongs to.

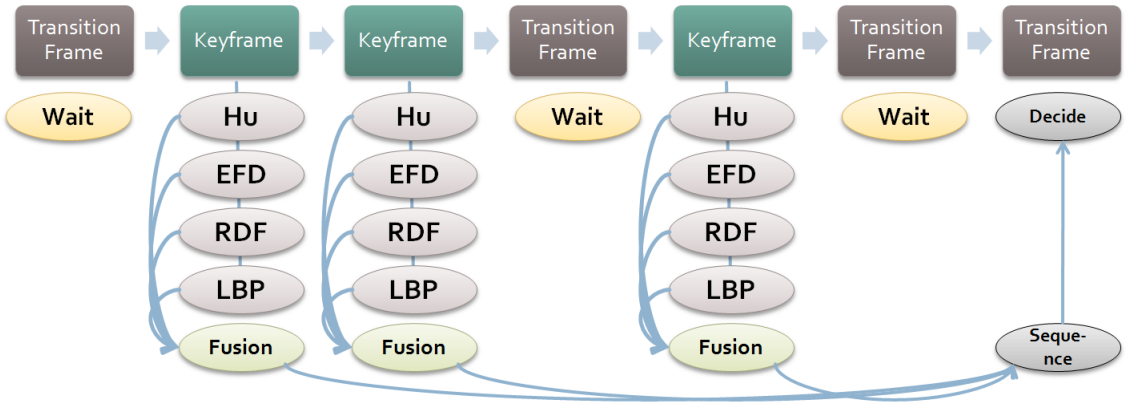



Figure 4.9. Keyframe based gesture start and end point detection

### 4.3.3. Hidden Markov Models

Hidden Markov Models(HMM)'s are probabilistic models that are frequently used to model time series. They are widely used in many speech and gesture recognition applications. Likewise, we make use of HMM's in modeling sequences of fingerspelling gestures. HMM's can have two types of input as discrete and continious inputs. In the task of fingerspelling recognition, we have used HMM's with both discrete and continious observations. In the former case, for each sequence of continuous features describing hand gesture images frame by frame, we perform frame-wise classification to obtain a discrete class information. Each of these labels are considered as the observations O = $\{O_1, O_2, .....O_t\}$ of our HMM. In the model, each hidden state of an output produces a likelihood value of an output belonging to the observation, O. The discrete hidden markov model can be parameterized as seen in Equation 4.5.

$$\lambda = (A, B, \pi) \tag{4.5}$$

In our system each image sequence containing a single hand gesture is represented by an HMM (Figure 4.10). We have used a seven state model to represent our hand gestures. The usage of HMM's can be broken down into three operations[65].

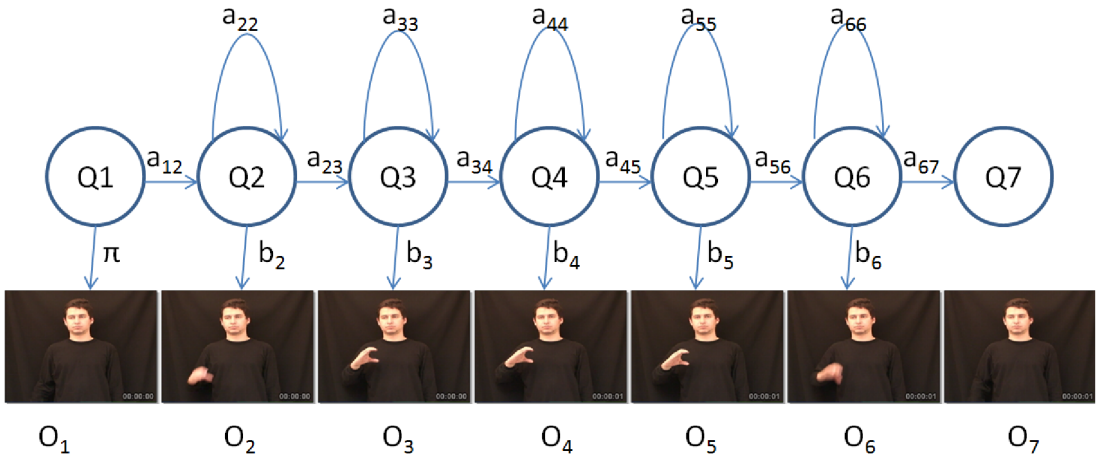



Figure 4.10. The seven state Hidden Markov Model

(i) Given an observation sequence $O(O_1, O_2, ....O_t)$ for testing and a model $\lambda(A, B, \pi)$, find the likelihood $P(O \mid \lambda)$ of the observation sequence.

(ii) Given an observation sequence $O(O_1, O_2, ....O_t)$ and a model $\lambda(A, B, \pi)$; find the state sequence $Q(Q_1, Q_2, ....Q_t)$ that has the highest probablity of generating O.

(iii) Given training observation sequences $X = \{O^k\}_k$ and a model $\lambda(A, B, \pi)$, find the parameters of a model $\lambda$ that maximizes $P(X \mid \lambda)$.

In order to train a hidden Markov model, given training observation sequences $X = \{O^k\}_k$, we use a maximum likelihood approach. Our goal in this method is to find the model $\lambda$ that maximizes the likelihood of our training observations. The maximum likelihood algorithm we use is called the Baum-Welch algorithm.

We begin iterating the Baum-Welch algorithm by first calculating alpha and beta using the forward variable $\alpha$ in equation 4.6 and the backward variable $\beta$ in equation 4.7.

$$\begin{aligned} Initialization : \alpha_1(i) &= \pi_i b_i(O_1) \\ Recursion : \alpha_{t+1}(j) &= [\sum_{i=1}^{N} \alpha_t(i) a_{ij}] b_j(O_{t+1}) \end{aligned} \tag{4.6}$$

$$\begin{aligned} Initialization : \beta_T(i) &= 1 \\ Recursion : \beta_t(i) &= \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j) \end{aligned} \tag{4.7}$$

After calculating alpha and beta, these are used in the calculation of $\gamma$ and $\xi$. $\gamma$ (equation 4.8) is the probability of being in a state at a given time given the observation and model parameters. $\xi(i, j)$ (equation 4.9) is the probability of being in state i at time $t$ and in state j at time $t + 1$, given O and $\lambda$.

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)} \tag{4.8}$$

$$\xi(i, j) = \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_k \sum_l \alpha_t(k) a_{kl} b_l(O_{t+1}) \beta_{t+1}(l)} \tag{4.9}$$

Finally, using $\gamma$ and $\xi$, we estimate the updated transition, observation and prior probabilities, that form our Hidden Markov model.

$$\pi_i = \frac{\sum_{k=1}^{K} \gamma_1^k(i)}{K} \tag{4.10}$$

$$\hat{a}(i,j) = \frac{\sum_{k=1}^{K}\sum_{t=1}^{T_k-1}\xi_t^k(i,j)}{\sum_{k=1}^{K}\sum_{t=1}^{T_k-1}\gamma_t^k(i)} \tag{4.11}$$

$$\hat{b}_j(m) = \frac{\sum_{k=1}^{K}\sum_{t=1}^{T_k-1}\gamma_t^k(j)l(O_t^k = v_m)}{\sum_{k=1}^{K}\sum_{t=1}^{T_k-1}\gamma_t^k(i)} \tag{4.12}$$

The update of the parameters are executed until the difference in A, B and $\pi$ becomes negligible. At that point the model is considered to be converged, thus yielding the optimal model.

Next, to achieve prediction with the optimized models, we solve the evaluation problem. Given an observation and k models belonging to k different classes, we find the probablity that the observed sequence can be generated by each model P(O| $\lambda$). By evaluating the observation sequence against each model, we designate the model with the highest probability as the recognized sign.

To calculate the probability of an observation sequence being generated by a model $P(O\ |,\lambda)$, we need to calculate likelihoods for all possible $Q = (Q_1, Q_2.....Q_t)$, which is impractical[65]. For that reason we use a procedure called the forward-backward procedure. It effectively divides a sequence into two parts one from time 1 to t and the other from t+1 to T. The forward variable $\alpha_i$ was descirbed in equation 4.6. and the backward variable $\beta$ was described in equation 4.7. Then, to find the state sequence $Q = (Q_1, Q_2.....Q_t)$, we can calculate $\gamma_t(i)$ (equation 4.8), which gives us the probability of being in state i at time t.

For each iteration, using the $\gamma_t(i)$ variable with the Viterbi algorithm, the likelihood of the model for a given observation can be calculated. The Viterbi algorithm produces an efficient means of evaluating a set of HMMs by taking only the maximum-likelihood path at each time step instead of all paths. It is a form of dynamic programming which finds the HMM states that maximizes the probability of the observations given the model.

Another method of Hidden Markov Models that does not use discretized feature vectors is continious HMM's. If the inputs of HMM's are continious, we make use of gaussian mixtures to model each feature. In a case, where we use $J$ gaussians to model our each feature, our observations in state $S_j$ are drawn from normal distributions with mean $\mu_j$ and variance $\sigma_j$. In the maximization step of the baum welch algorithm the new mean and variance values are calculated as in Equation 4.13 and 4.14.

$$\hat{Mu}_j = \frac{\sum_t \lambda_t(j) O_t}{\sum_t \lambda_t(j)} \tag{4.13}$$

$$\hat{\sigma}_j^2 = \frac{\sum_t \lambda_t(j)(O_t - \hat{\mu}_j)^2}{\sum_t \lambda_t(j)} \tag{4.14}$$

# 5. EXPERIMENTAL RESULTS

In this chapter, we present recognition results on our self collected Multi-lingual Fingerspelling datasets. In section 5.1, we present applications, which use the methods presented. In section 5.2, we describe the collection, contents and challenges of the dataset. In section 5.3, we present and discuss our recognition results on fingerspelling keyframes and sequences.

## 5.1. Multi-Lingual Fingerspelling to Speech Translator

Being the first prototype application stemming from the work in this thesis, this application helps the communication of two people, one hearing impaired and one visually impaired by converting speech to fingerspelling and fingerspelling to speech. In the system, different spoken languages and sign languages such as English, Russian, Turkish and Czech are considered. By using recognition and synthesis techniques for speech and fingerspelling modalities the system enables the communication of the deaf and the blind, who have no other natural shared medium to communicate with each other. A screen shot from the system demo presentation can be seen in Figure 5.1.

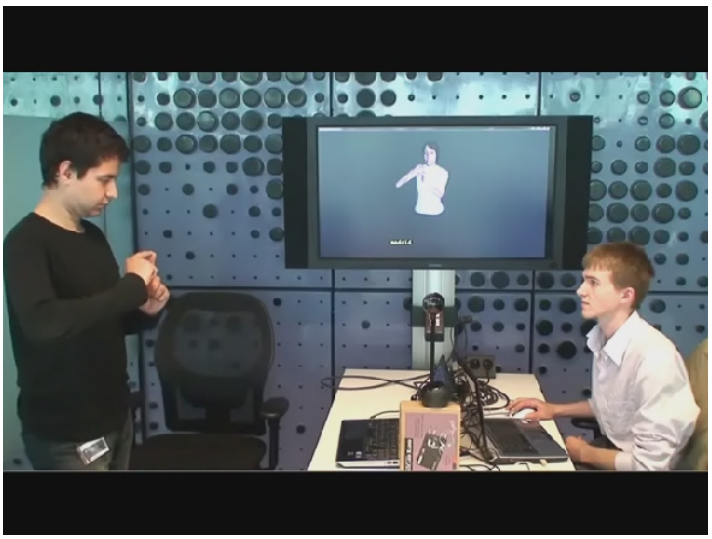


Figure 5.1. FingerSign to Speech translator system flowchart

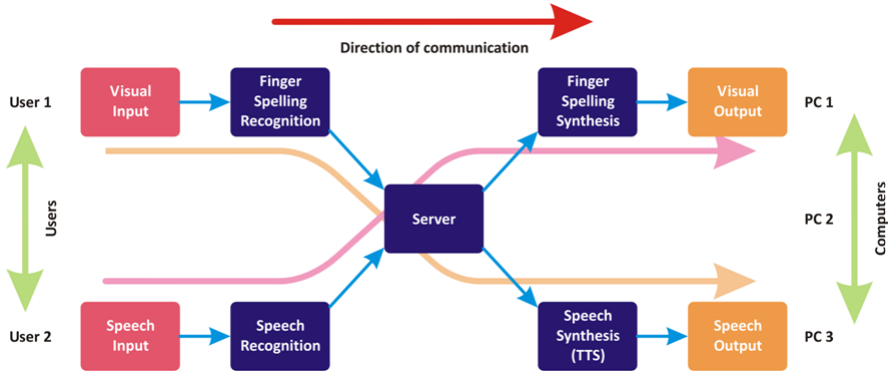



Figure 5.2. FingerSign to Speech translator system flowchart

The overall system is implemented in client-server architecture. The system contains four standalone modules as fingerspelling synthesis, fingerspelling recognition, speech synthesis and speech recognition. The modules are the client applications and they communicate through the server. The system is operating in close to real time. It takes the fingerspelling input from the camera, or the speech input from the microphone and converts it to synthesized speech or fingerspelling. The input and output can be selected among the supported languages for each module. The translation between different languages is handled via Google translate APIs. The system flowchart can be seen in Figure 5.2.

In this system, the fingerspelling recognition module is presented a continuous gesture input and the system provides word output. For word level recognition, the motion modeling algorithm with weighted voting of hand gesture sequences described in Section 4.3.1 are used. An additional "End of Word" sign is performed after each word to separate different words. This approach yields a sequence of letters that are assumed to belong to the same word. However, due to possible errors in the recognition of individual letters, the letter sequences that form up words tend to contain one or more errors. To reduce the word error, we make use of a language model, which contains 2000 words per different language. Using Levenshtein distance [66] we compare the recognized letter sequence with the entire language model to find the best matching word. The word with the closest score is returned as the final result of the fingerspelling recognition module.

## 5.2. Database

The aim of this thesis is to realize and evaluate a fingerspelling recognition system in a multi-lingual setting. For this purpose, video inputs consisting of all the signs to be recognized were deemed necessary. Although various datasets belonging to each of these gesture sets exist in different sources, they all seem to vary in different critical aspects such as lighting, background constrictions or body pose. Therefore, in order to train a compatible multilingual system, we have collected a video database of fingerspelling sings. The dataset contains the fingerspelling alphabets of the Czech, Russian and Turkish sign languages. Gestures of the Turkish fingerspelling alphabet can be seen in Figure 5.3, the Czech fingerspelling alphabet in Figure 5.4 and the Russian fingerspelling alphabet in Figure 5.5

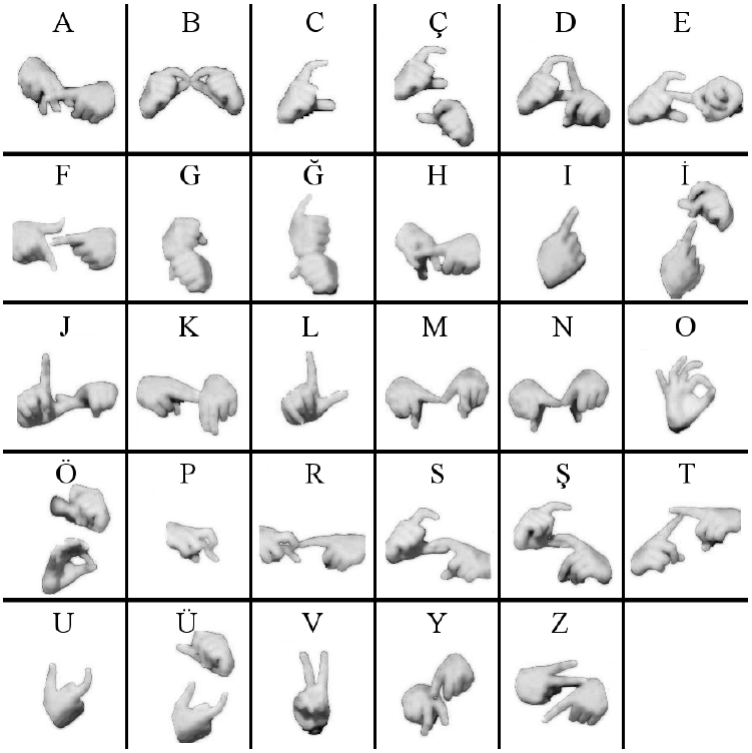



Figure 5.3. The Fingerspelling Turkish Alphabet

Currently, the database contains videos of five subjects performing the Turkish sign alphabet, three subjects performing the Czech sign alphabet and three subjects performing the Russian sign alphabet. Of these subjects, one is a natural signer and a

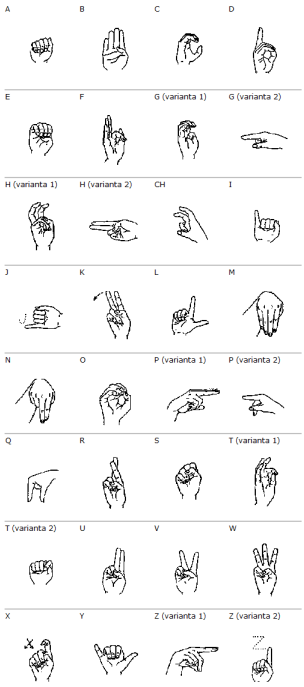



Figure 5.4. The Czech Fingerspelling Alphabet

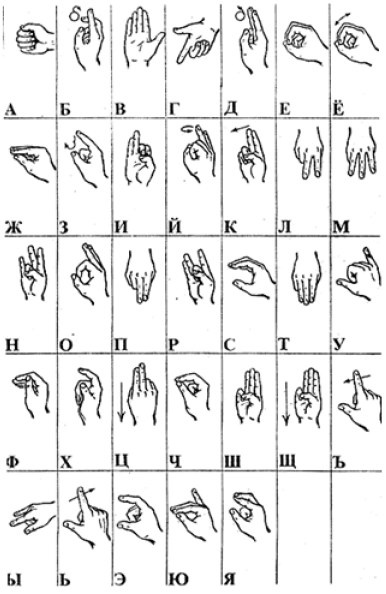



Figure 5.5. The Russian Fingerspelling Alphabetl

signing instructor. The rest of the subjects are university students who have performed fingerspelling gestures through visual guidance. The gestures of the Czech and Russian alphabets were taught to the performers by our study partners from those respective countries during the eNTERFACE'10 workshop[67]. Sample Images of the dataset can be seen in Figure 5.6.

Figure 5.6. Samples images from the multi-lingual fingerspelling dataset

In total, there are 88 letters performed in the database, covering 29 Turkish, 26 Czech and 33 Russian sign alphabet letters. Some letters can be represented using only one hand, whereas some require the use of both hands. Moreover, one keyframe is enough to represent some letters whereas representation of some letters have a more dynamic nature since they can be performed with a movement rather than a static

frame, for instance X. Each subject repeats each letter in the database five to nine times.

In the videos, certain restrictions are imposed on the signers to make skin color based segmentation easy. The signers wear dark colored clothes with long sleeves that leaves only hands and faces bare. The signs are performed in front of a black background using a constant camera distance and angle. Before and after each performance, the signer brings his hands down to his sides, which is designated as the rest pose. The videos were recorded from a frontal pose approximately one and a half meters from the signers that captured the signers face and hands down to his/her waist. While the usage of certain lighting conditions allowed for easier segmentation, we have used normal room conditions in the recording of some videos to make the training set more robust.

The signs are recorded by a mini-dv camera at 25fps on resolutions varying from 1072x768 to 640x480. The total length of the training and test videos for each subject is between 25 and 30 minutes. Thus the length of the entire dataset borders four hours. In the videos, the interlacing effect caused by the camera interlacing effect is removed by applying a deinterlacing algorithm.

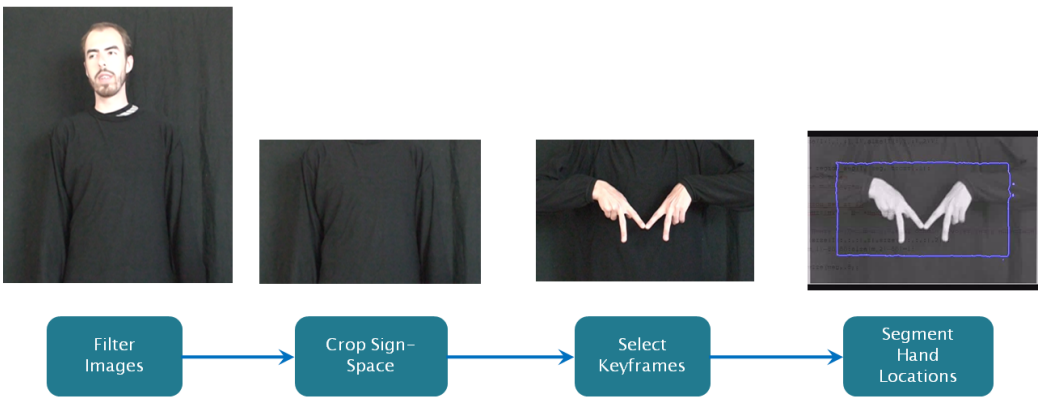



Figure 5.7. The Russian Fingerspelling Alphabet

In the dataset, gestures with many differing properties are present. All of the gestures in the database use hands as the only modality. However, the composition of different gestures shows a lot of variance. While some gestures are represented by static shapes others are represented by dynamic movements of hands. Likewise, while all gestures of the Russian language make use of one handed gestures, the Czech and Turkish alphabets contain both one and two handed gestures.

Each recorded video in the database has been preprocessed and annotated. As the shape of the human hand is highly deformable with hard to draw features, manual annotation of a hand is very difficult and time consuming. For that reason we developed a semi-automatic annotation pipeline to detect and segment hand images as shown in 5.7. In the pipeline, the signspace is cropped for each image. Then the presence of a hand in the signspace is checked using a basic global skin color model. If a hand is present, a large approximate bounding box is drawn and the hands are segmented using Active Contour Models[68].

## 5.3. Fingerspelling Recognition

In this section, we evaluate the performance of our fingerspelling representation and recognition methodologies. For each gesture in our dataset we use appearance based descriptors explained in Section 3 to represent hands and classifiers described in section 4.2. Then moving one step further, sequence information is incorporated into this classification to make a keyframe sequence-wise decision.

In our experiments we have tested several different modules of the proposed fingerspelling system. For this purpose we have performed different tests aimed at illustrating and analyzing recognition accuracies under differing conditions. The recognition performance of isolated fingerspelling gestures can be compared using recognition accuracies. Recognition accuracy is calculated by dividing the number of correctly classified samples (true positive and true negative) over all (true positive,true negative, false positive, false negative) samples in the dataset as seen in Equation 5.1.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{5.1}$$

### 5.3.1. Evaluation of Automatic Keyframe Segmentation

In both the isolated and continuous fingerspelling videos, the videos contain frames where no signs are performed. Therefore we attempt to recognize these frames through our automatic keyframe recognition methodology described in Section 4. To measure the effectiveness of our keyframe segmentation methodology we performed two separate tests.

In the first test, in a dataset of nearly 6000 frames belonging to annotated continuous hand gesture videos, we attempted to classify each frame as a keyframe or transition frame. The videos belonging to performances from the Turkish manual alphabet were manually annotated as keyframe or transition frames. The hand contours in the videos were segmented using active contour models and separated into 4 groups as blurred, unblurred, indistinguishable and hands not available. Then, using a fusion hand displacement and partial motion blur, each frame was classified as a keyframe or a transition frame. In these tests performed using K-nearest neighbor classifiers, the use of partial blur detection through the variance of the gradient disribution yields an accuracy of 0.7456 per cent. On the other hand, classification with partial blur detection using maximum gradient difference of contours yielded 0.8314 per cent accuracy.

To clearly analyze the effect of automatic keyframe recognition on fingerspelling recognition, we performed isolated fingerspelling recognition experiments on manual and automatically selected hand gesture keyframes. In the first group of images, snapshots of each sign language video were extracted from fingerspelling videos. As these snapshots are manually extracted from image sequences to represent gestures, they are not affected by possible propagating image segmentation or temporal segmentation errors. Therefore, the tests with annotated snapshot images provide us information on

feature descriptor and classifier performances free of segmentation and keyframing errors. In Table 5.1 we compare the recognition accuracies for the Turkish fingerspelling alphabet using both manually selected and automatically extracted keyframes.

Table 5.1. Turkish fingerspelling recognition accuracy with different keyframe selection using kNN

| | **HU** | **EFD** | **RDF** | **LBP** | **Fusion** |
|---|---|---|---|---|---|
| **Manual** | 0.5416 | 0.7041 | 0.7283 | 0.8662 | **0.8814** |
| **Automatic** | 0.4924 | 0.6465 | 0.7606 | 0.7639 | **0.7793** |

While comparing the results in Table 5.1 one must take into account that isolated snapshots contain no temporal ordering information. Therefore, transition frames that are selected and reduce recognition accuracy in automatic keyframe selection should not be considered useless, as they are simply left out in this kind of modeling.

### 5.3.2. Effects of Feature Selection and Dimensionality Reduction

In Table 5.2, we display results on the effects of dimensionality reduction on keyframe based recognition accuracy. For dimensionality reduction, we make use of the principal component analysis algorithm. In these tests, the kNN algorithm is used to classify the features belonging to the keyframes of the Turkish Fingerspelling alphabet. In the experiment, we have chosen to preserve energy at different levels by choosing different number of coefficients whose eigenvalues sum up to a certain percentage of the total energy. Figure 5.8 shows that except for Elliptic Fourier Descriptors, no feature shows performance improvement when projected onto a lower dimensional space.

We explicitly feel the need to note here that each EFD coefficient is of a different magnitiude giving higher weight to higher harmonic features. Performing z-normalization on the vector actually reduces overall descriptor accuracy by negating high level shape characteristics. Thus, the success of PCA algorithm to converge on features with higher harmonics increases the desciptors overall efficiency.

Table 5.2. Effects of dimensionality reduction on Turkish fingerspelling recognition using kNN

| | **HU** | **EFD** | **RDF** | **LBP** |
|---|---|---|---|---|
| **NO PCA** | 0.4924 | 0.6465 | 0.7606 | **0.7639** |
| **PCA %95** | 0.4633 | **0.6964** | 0.3896 | 0.2343 |
| **PCA %99** | 0.4924 | **0.7704** | 0.6643 | 0.5649 |
| **PCA %100** | 0.4924 | 0.7629 | 0.7609 | **0.7636** |

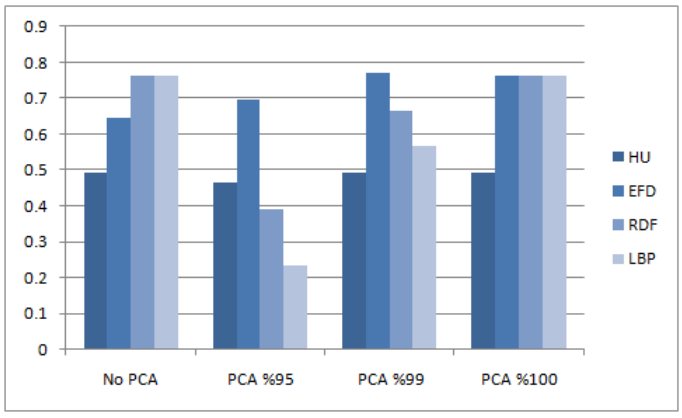



Figure 5.8. Effects of PCA on fingerspelling recognition accuracy

### 5.3.3. Performance Evaluation of Hand Gesture Features

In Tables 5.3 and 5.4, we display recognition accuracy results for Turkish, Czech and Russian fingerspelling alphabets using different features and different classifiers. The results presented in this section are calculated using automatically selected keyframes on isolated fingersign videos. Best results are obtained using local binary patterns with a recognition accuracy of 0.8013 per cent in kNN classifcation. Radial distance function also shows good performance with especially two handed Turkish gestures, but fails to perform well in Russian and Czech Languages. Hu Moments have the lowest classification accuracy. Using naive bayes classification, elliptic fourier descriptors outperform other descriptors.

Table 5.3. Fingerspelling recognition accuracy with with automatically selected keyframes using kNN classifier

| | **HU** | **EFD** | **RDF** | **LBP** | **Fused** |
|---|---|---|---|---|---|
| **Turkish** | 0.4924 | 0.6465 | 0.7606 | 0.7639 | **0.7793** |
| **Russian** | 0.3298 | 0.5616 | 0.5501 | **0.7764** | 0.7752 |
| **Czech** | 0.5409 | 0.5766 | 0.6773 | 0.7847 | **0.8013** |

Table 5.4. Fingerspelling recognition accuracy with automatically selected keyframes using naive bayes classifier

| | **HU** | **EFD** | **RDF** | **LBP** | **Fused** |
|---|---|---|---|---|---|
| **Turkish** | 0.3276 | **0.5373** | 0.4906 | 0.2936 | 0.5653 |
| **Russian** | 0.3457 | **0.6089** | 0.4586 | 0.3166 | 0.5642 |
| **Czech** | 0.3328 | **0.6434** | 0.5683 | 0.4934 | 0.6167 |

### 5.3.4. Comparison of Signer Independent and Dependent FRR

In our experiments we report two sets of results as signer independent and signer dependent recognition accuracies. In the signer dependent case, training and test examples are chosen from the same individuals. For each subject, half of the repetitions of each gesture are placed into the training set and the remaining half are placed into the test set. In the signer independent recognition, a subject is chosen as test subject and the system is trained with sample gestures belonging to the rest of the users.

Using the signer dependent and independent test sets, recognition accuracy results (fingersign recognition rate - FRR) belonging to different subjects are obtained and shown in Tables 5.5 and 5.6. The goal of these tests is to estimate the recognition accuracy of an online system whose test videos for every possible user can not be present in the database.The recognition accuracy difference of these two recognition sets show us how much the systems recognition is reliable, if a signer's articulation deviates from the training data.

Table 5.5. Multi-lingual user dependent fingerspelling recognition accuracy with manually selected keyframe sequences

| | **Language** | **HU** | **EFD** | **RDF** | **LBP** | **Fused** |
|---|---|---|---|---|---|---|
| Subject | **Turkish** | 0.53 | 0.63 | 0.67 | 0.85 | **0.88** |
| Subject | **Turkish** | 0.56 | 0.70 | 0.78 | 0.91 | **0.93** |
| Subject | **Turkish** | 0.54 | 0.75 | 0.77 | **0.87** | **0.87** |
| Subject | **Turkish** | 0.48 | 0.61 | 0.61 | 0.74 | **0.78** |
| Subject | **Turkish** | 0.59 | 0.79 | 0.76 | 0.94 | **0.96** |
| Average | **Turkish** | 0.54 | 0.70 | 0.72 | 0.86 | **0.88** |
| Subject | **Russian** | 0.64 | 0.61 | 0.72 | 0.69 | **0.75** |
| Subject | **Russian** | 0.38 | 0.54 | 0.75 | **0.8** | **0.8** |
| Subject | **Russian** | 0.63 | 0.61 | 0.64 | 0.69 | **0.73** |
| Average | **Russian** | 0.55 | 0.59 | 0.7 | 0.73 | **0.76** |
| Subject | **Czech** | 0.6 | 0.52 | 0.62 | **0.75** | 0.72 |
| Subject | **Czech** | 0.48 | 0.61 | 0.78 | 0.76 | **0.84** |
| Subject | **Czech** | 0.47 | 0.65 | 0.67 | 0.73 | **0.78** |
| Average | **Czech** | 0.52 | 0.59 | 0.69 | 0.75 | **0.78** |

Table 5.6. Multi-lingual user independent fingerspelling recognition accuracy with manually selected keyframe sequences

| | **Language** | **HU** | **EFD** | **RDF** | **LBP** | **Fused** |
|---|---|---|---|---|---|---|
| Subject | **Turkish** | 0.31 | 0.15 | 0.39 | 0.28 | **0.43** |
| Subject | **Turkish** | 0.23 | 0.06 | 0.24 | 0.21 | **0.28** |
| Subject | **Turkish** | 0.36 | 0.30 | **0.48** | 0.29 | 0.45 |
| Subject | **Turkish** | 0.07 | 0.28 | 0.61 | 0.48 | **0.54** |
| Subject | **Turkish** | 0.32 | 0.20 | **0.45** | 0.23 | 0.40 |
| Average | **Turkish** | 0.26 | 0.20 | **0.43** | 0.30 | 0.42 |
| Subject | **Russian** | 0.36 | 0.16 | 0.44 | 0.44 | **0.47** |
| Subject | **Russian** | 0.36 | 0.25 | 0.39 | **0.42** | 0.4 |
| Subject | **Russian** | 0.27 | 0.28 | **0.43** | 0.31 | 0.33 |
| Average | **Russian** | 0.33 | 0.23 | **0.42** | 0.39 | 0.4 |
| Subject | **Czech** | 0.39 | 0.34 | 0.46 | 0.41 | **0.48** |
| Subject | **Czech** | 0.27 | 0.18 | **0.49** | 0.44 | 0.46 |
| Subject | **Czech** | 0.22 | 0.18 | 0.37 | 0.37 | **0.38** |
| Average | **Czech** | 0.29 | 0.23 | **0.44** | 0.4 | **0.44** |

### 5.3.5. Comparison of Temporal Segmentation Methodologies:

As described in Section 4, we have used three different methodologies to temporally model our image sequences. In the first two methods, we attempt to quantify features belonging to each frame to class labels and model the class labels. In the last method, we use features of image sequences as continuous HMM observations and model them with GMM's.

In Tables 5.7 and 5.8, we classify features belonging to each sequence using pixelwise kNN or Naive Bayes classifiers. Then using majority voting, we find the class that contains the most number of keyframes in a sequence to make a sequence-wise decision. As this model incorporates the transition frames of dynamic gestures, it produces much better results than pixelwise classification.

Table 5.7. Fingerspelling recognition accuracy on keyframe sequences using majority voting with kNN classifier

| | **HU** | **EFD** | **RDF** | **LBP** | **Fused** |
|---|---|---|---|---|---|
| **Turkish** | 0.6919 | 0.7677 | 0.8729 | 0.8753 | **0.9218** |
| **Russian** | 0.5684 | 0.7222 | 0.7863 | 0.8889 | **0.9103** |
| **Czech** | 0.6629 | 0.7079 | 0.8652 | 0.9045 | **0.9213** |

Table 5.8. Fingerspelling recognition accuracy on keyframe sequences using majority voting with naive bayes classifiers

| | **HU** | **EFD** | **RDF** | **LBP** | **Fused** |
|---|---|---|---|---|---|
| **Turkish** | 0.3472 | 0.6235 | 0.6015 | 0.3276 | **0.6333** |
| **Russian** | 0.2735 | 0.4744 | 0.3205 | 0.3462 | **0.4573** |
| **Czech** | 0.3427 | 0.7584 | 0.7416 | 0.5657 | **0.7886** |

Using the majority voting method with kNN classifiers, we achieve combined feature recognition accuracies of approximately 92 per cent. Trained with signer dependent data, this classifier yields the best overall fingerspelling gesture recognition accuracies. Confusion matrices for the Turkish, Russian and Czech manual alphabets are shown in Figures 5.9, 5.10 and 5.11, respectively.

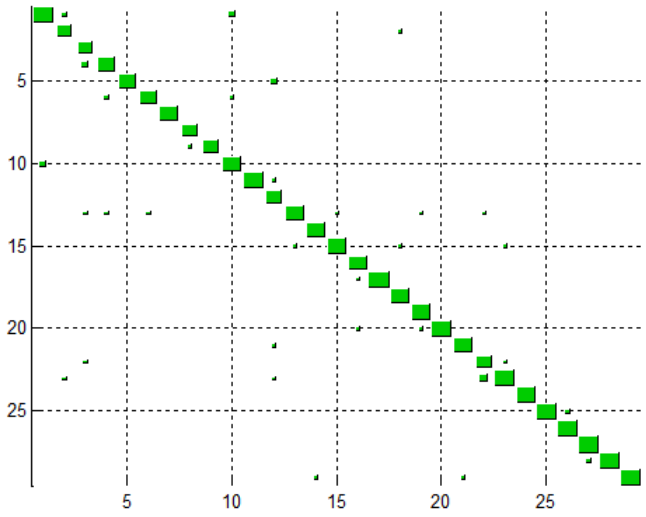



Figure 5.9. Turkish fingerspelling recognition confusion matrix

Table 5.9 displays the recognition results of the combined 88 class fingerspelling recognition rate. Pixel-wise classification results are combined with majority voting to reach sequence level decisions. The multilingual confusion matrix is given on Figure 5.12.

Table 5.9. 88 class multilingual fingerspelling recognition on keyframe sequences using majority voting

| | **HU** | **EFD** | **RDF** | **LBP** | **Fused** |
|---|---|---|---|---|---|
| **kNN** | 0.5513 | 0.6618 | 0.8276 | 0.8316 | **0.8445** |
| **Gaussian** | 0.2263 | 0.5039 | 0.5395 | **0.6627** | 0.6427 |

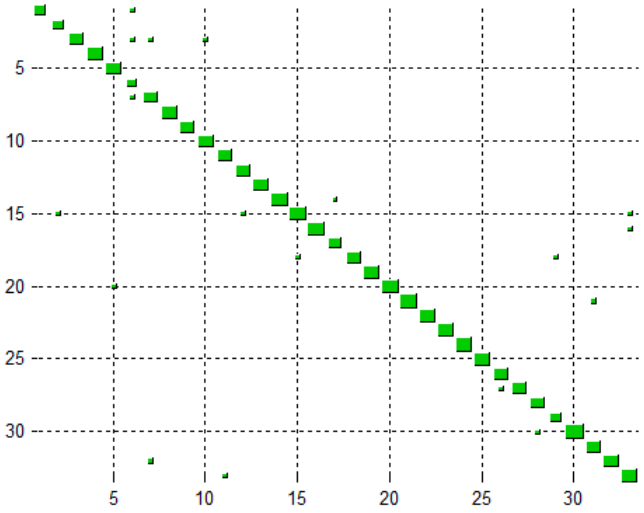



Figure 5.10. Russian fingerspelling recognition confusion matrix

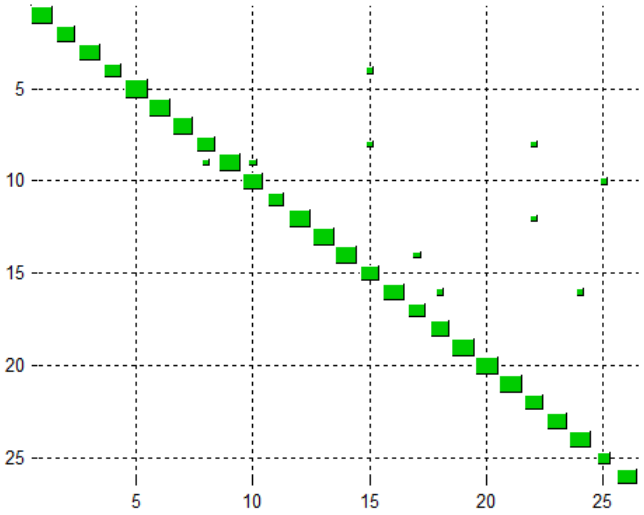



Figure 5.11. Czech fingerspelling recognition confusion matrix

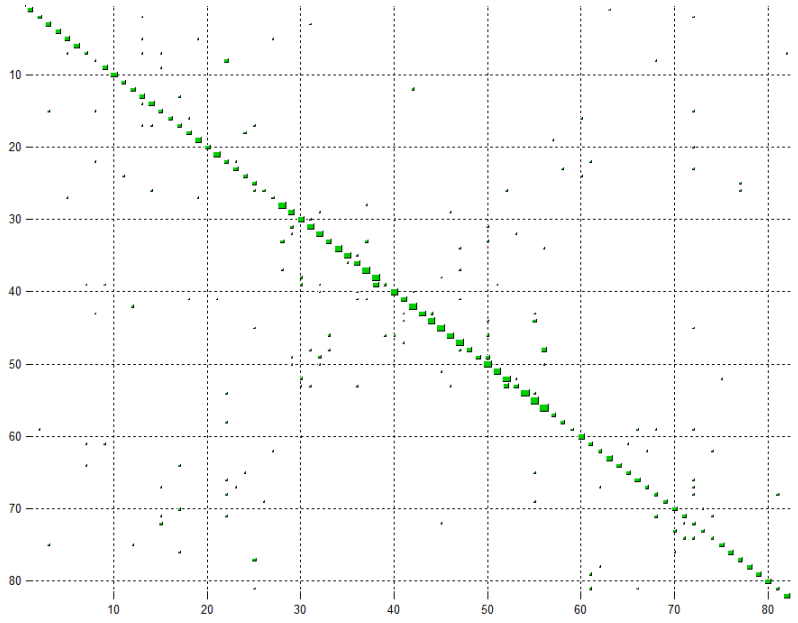



Figure 5.12. Combined fingerspelling recognition confusion matrix

In Table 5.10, we present the results of isolated gesture recognition using discrete Hidden Markov Models. Like the majority voting method, we classify features belonging to each sequence using pixel-wise kNN classifiers. After obtaining a sequence of class label outputs, we use them to build discrete HMM's for each class. We decide on the class label of a sequence depending on how well it fits models belonging to each class.

Table 5.10. Fingerspelling recognition accuracy on keyframe sequences using discrete HMM's

| | **HU** | **EFD** | **RDF** | **LBP** | **Fused** |
|---|---|---|---|---|---|
| **Turkish** | 0.4695 | 0.4299 | 0.4977 | **0.6051** | 0.5316 |
| **Russian** | 0.2965 | 0.4267 | 0.3785 | **0.5034** | 0.4684 |
| **Czech** | 0.2870 | 0.4122 | **0.5422** | 0.4717 | 0.5159 |

Lastly, we present the results of our continuous HMM based classification in the results shown in Tables 5.11 and 5.12. In this method, feature vectors belonging to each descriptor are modeled with Gaussian Mixture models and combined in Hidden Markov Models. Each feature vector of an individual gesture is modeled using a single HMM with five Gaussian mixture components, which was determined through cross-validation. While modeling gestures using HMMs, an effective parameter of the model is the number of states used to generate a gesture. We have used two different methods to estimate the optimal state count of our models. In the first approach we have trained several HMMs with different number of states. Using five-fold cross validation, we have compared system recognition accuracies of these HMMs to determine the state number of the best fitting model for each feature (Figure 5.13). In Table 5.11, we present the recognition of modeling with a fixed HMM with five states.

Table 5.11. Fingerspelling recognition accuracy on keyframe sequences using continuous HMM's

| | **HU** | **EFD** | **RDF** | **LBP** | **Fused** |
|---|---|---|---|---|---|
| **Turkish** | 0.2714 | 0.6156 | 0.3291 | 0.6457 | **0.6407** |
| **Russian** | 0.2371 | 0.6983 | 0.1905 | 0.5086 | **0.6147** |
| **Czech** | 0.2768 | 0.7288 | 0.2373 | 0.6271 | **0.7054** |

Since our dataset contains hand gestures of differing lengths and shape combinations, we have hypothesized that the optimal number of states necessary to represent different gestures may be different from one gesture to another. For this reason, we attempted to find the state count of the best model for each gesture. As seen in Figure 5.13, we obtained log likelihood values determining the fitness of each fold to HMM's generated with different number of states. We compared the maximum log-likelihood values for each fold and obtained the state count of the model that the validation samples fit best as seen in Table 5.13. Using the HMMs with unique state numbers for each different gesture, we calculated the fingerspelling recognition accuracies for the Turkish fingerspelling alphabet seen in Table 5.12.

Table 5.12. Turkish FRR with dynamic and static numbers of state continious HMM's

| | **HU** | **EFD** | **RDF** | **LBP** | **Fused** |
|---|---|---|---|---|---|
| **Static Number of States** | 0.2714 | 0.6156 | 0.3291 | 0.6457 | **0.6407** |
| **Dynamic Number of States** | 0.2764 | 0.6256 | 0.3510 | 0.6608 | **0.6884** |

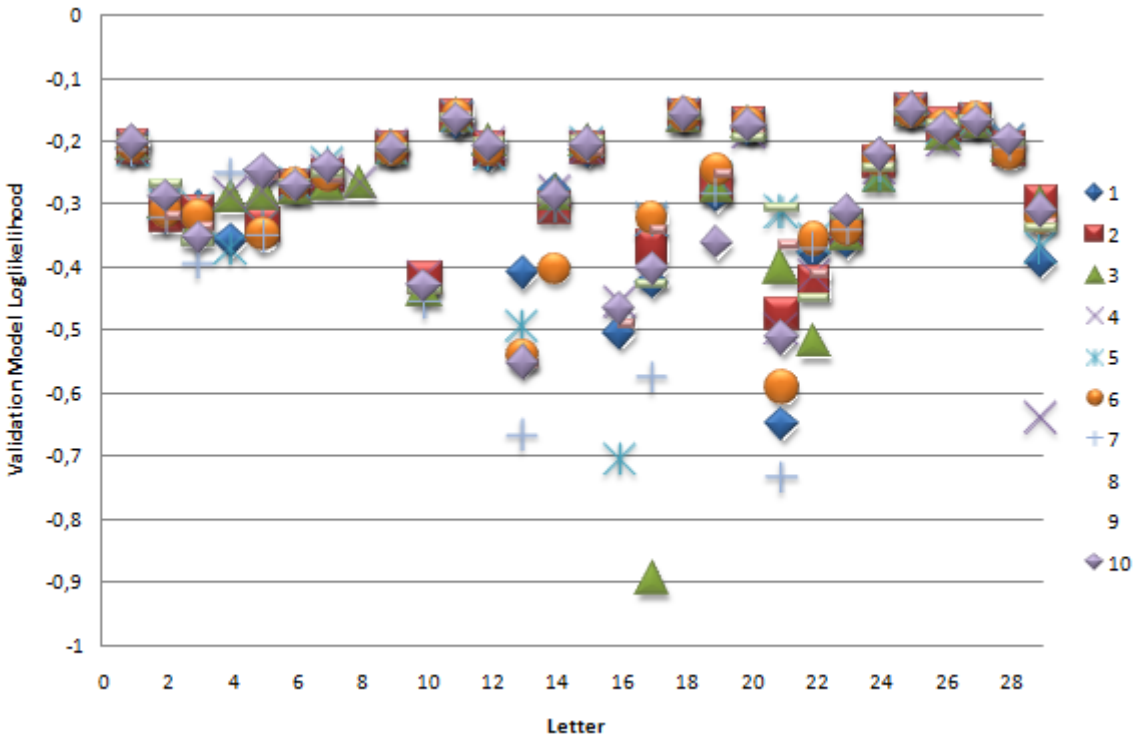



Figure 5.13. Cross-validation results of continious HMM log-likelihood values with different number of states for Turkish fingerspelling gestures

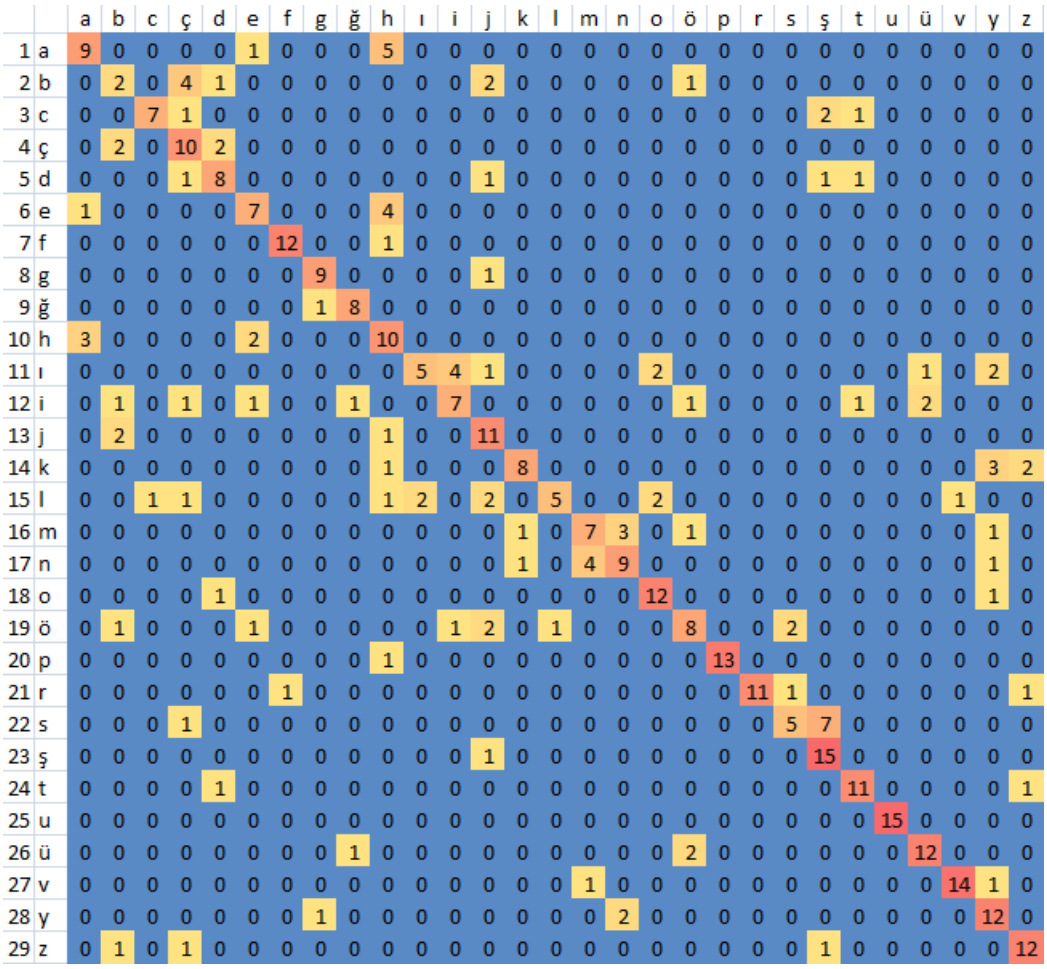
Table 5.13. Gesture properties of Turkish Fingerspelling Alphabet and optimal number of HMM states

| | Hand Count | Dynamic | Optimal State Number |
|---|---|---|---|
| A | 2 | No | 9 |
| B | 2 | No | 9 |
| C | 1 | No | 4 |
| Ç | 2 | Yes | 7 |
| D | 2 | No | 1 |
| E | 2 | No | 5 |
| F | 2 | No | 5 |
| G | 2 | No | 3 |
| Ğ | 2 | Yes | 9 |
| H | 2 | No | 4 |
| I | 1 | No | 4 |
| İ | 2 | Yes | 8 |
| J | 2 | Yes | 2 |
| K | 2 | No | 2 |
| L | 1 | No | 3 |
| M | 2 | No | 4 |
| N | 2 | No | 6 |
| O | 1 | No | 5 |
| Ö | 2 | Yes | 6 |
| P | 1 | No | 2 |
| R | 2 | No | 9 |
| S | 2 | No | 5 |
| Ş | 2 | Yes | 10 |
| T | 2 | No | 10 |
| U | 1 | No | 3 |
| Ü | 2 | Yes | 2 |
| V | 1 | No | 6 |
| Y | 2 | Yes | 10 |
| Z | 2 | No | 2 |

Table 5.12 shows a four percent increase in overall system FRR from 64 to 68 per cent. Further improvement of recognition results requires a breakdown of recognition accuracy for different gestures. Figure 5.14 shows the confusion matrix for the fused Turkish FRR from the experiment in Table 5.12. By closely examining the confusion matrix, we can detect letter pairs that are most likely to be confused with one another.

It is observed that the letter pairs "A-H", "E-H" and "M-N" are often confused with each other. The most likely reason of this confusion is the similarity in the posture of hands which only slightly differs by the position of middle and ring fingers of the right hand. Likewise, the letter pairs "S-Ş" and "K-Y" also often appear to be confused. The interesting fact to note about these letter pairs is that, while they have similar hand postures, "S" and "K" are static gestures and "Ş" and "Y" are dynamic. Therefore, while recognizing these gestures, the system fails to distinguish between letters that may contain similar gestures but contain different movements over time. Another highly confused pair of letters is "I-İ" which is highly unlikely and unexpected. While "I" is performed with one hand, the letter "İ" is performed using two hands. Through experimentation, this confusion was found to be a failure in tracking. During tracking, a redetection failure of the right hand while performing the letter "İ" caused the system to create both one handed and two handed models for that letter. Likewise, due to failures in the tracking of the loosely merged two handed letter "B", the letter is often confused to separated two handed letters such as "Ç" and "J".

Table 5.14 displays two class classification accuracy results for the letter pairs that are most likely to be confused with each other. In these tests, we used cross validation to train the models with the ideal state numbers to distinguish the classes from each other. In these tests we saw that, while it was possible to distinguish all test instances of "A-H", "K-Y" and most instances of "E-H", it was not possible to distinguish multiple samples of he pairs "S-Ş", "I-İ" and "M-N".



| | a | b | c | ç | d | e | f | g | ğ | h | ı | i | j | k | l | m | n | o | ö | p | r | s | ş | t | u | ü | v | y | z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | a | 9 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | b | 0 | 2 | 0 | 4 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | c | 0 | 0 | 7 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 1 | 0 | 0 | 0 | 0 | 0 |
| 4 | ç | 0 | 2 | 0 | 10 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | d | 0 | 0 | 0 | 1 | 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 6 | e | 1 | 0 | 0 | 0 | 0 | 7 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | f | 0 | 0 | 0 | 0 | 0 | 0 | 12 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | g | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 9 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | ğ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 10 | h | 3 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 11 | ı | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 4 | 1 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 2 | 0 |
| 12 | i | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 7 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 2 | 0 | 0 | 0 |
| 13 | j | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 11 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 14 | k | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 2 |
| 15 | l | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 0 | 2 | 0 | 5 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 16 | m | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 7 | 3 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 17 | n | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 4 | 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 18 | o | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 19 | ö | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 0 | 1 | 0 | 0 | 0 | 8 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 20 | p | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 13 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 21 | r | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 11 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 22 | s | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 7 | 0 | 0 | 0 | 0 | 0 | 0 |
| 23 | ş | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 15 | 0 | 0 | 0 | 0 | 0 | 0 |
| 24 | t | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 11 | 0 | 0 | 0 | 0 | 1 |
| 25 | u | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 15 | 0 | 0 | 0 | 0 |
| 26 | ü | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 12 | 0 | 0 | 0 |
| 27 | v | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 14 | 1 | 0 |
| 28 | y | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 12 | 0 |
| 29 | z | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 12 |

Figure 5.14. Confusion matrix for Turkish FRR using continuous HMM's

Table 5.14. Two class recognition accuracy of commonly confusing letter pairs

| | HU | EFD | RDF | LBP | FUSED |
|---|---|---|---|---|---|
| A-H | 0,8 | 0,7667 | 0,6333 | **1** | 0,9667 |
| S-Ş | 0,7241 | 0,7241 | 0,6207 | **0,7586** | 0,7241 |
| I-İ | 0,5667 | 0,7667 | 0,7667 | 0,8 | **0,8333** |
| K-Y | 0,8276 | 0,8276 | 0,8966 | 0,9655 | **1** |
| E-H | 0,8148 | **0,963** | 0,7407 | 0,9259 | 0,9259 |
| M-N | 0,5357 | **0,8929** | 0,75 | 0,6071 | 0,7857 |

# 6. CONCLUSIONS

In this study, we investigate the extraction and recognition of fingerspelling gestures from sign language videos. We work with Czech, Russian and Turkish sign alphabets and deal with the segmentation,tracking, representation, temporal modeling and classification of hand gestures.

We have analyzed and used several methods for each of these challenging tasks. In hand segmentation, we used color histogram backpropagation and region growing methods in conjunction with a camshift based multi object tracking algorithm. We used invariant Hu moments, Elliptic Fourier Descriptors, Local Binary Patterns and Radial Distance function to represent hand shapes. We classified these features using kNN and Naive Bayes approaches. We extracted temporal keyframe information using hand motion and motion blur. Finally we used these two modalities with heuristic voting and HMM's to perform recognition of hand gesture videos.

The contributions of this thesis include the collection of a multi-lingual fingerspelling recognition dataset, the realization of a fingerspelling recognition system, the analysis of different hand gesture representation and classification methodologies and the realization of a fingersign to speech module.

The system was tested on a self collected dataset composed of 11 different signers from three different languages. A total of 88 signs are used in the fingerspelling recognition experiments. The videos of the dataset contain hand annotations for usage in frame based recognition tests. In addition, the start and end locations of isolated gestures are manually annotated by signers.

We have presented a method for fast and effective skin color based tracking of hands and the face on natural fingersign signing videos. The method is based on a camshift approach and takes into account the expected positions of the hands and the face. This joint modeling enables us to accurately track occluding objects and to

maintain the correct labeling after reappearance of tracked body parts. The algorithm is designed to be usable in applications where tracking is needed to perform for long durations, thus minimizing tracking errors and recovering quickly from them.

The experiments performed with isolated fingerspelling gestures show that the system is capable of recognizing isolated gestures. The KNN classification with majority voting of sequences achieved approximately 92 percent recognition accuracy with all three languages. While the results with other approaches seem comparatively lower, they can be improved through the availability of more training data and examples for each class. By modeling gestures with Hidden Markov Models using Gaussian Mixture Models, we obtain a fingerspelling recognition accuracy of approximately 64 per cent. By optimizing the number of HMM states to model each gesture using cross validation, we obtain an accuracy increase of approximately four per cent. However, a disadvantage of using cross validation to determine the number of states for an HMM is that training multiple HMMs from scratch is computationally expensive. In addition, since the success of training a model depends on initial parameters, it is possible to compare ideal models with models that have been poorly initialized. To overcome this, a multiple number of models with the same state number may be trained to choose the best among them. However, this also causes an increase in computational complexity. Therefore in order to train the system for new gestures, the usage of methods such as State Splitting HMM's may be preferred over cross validation for determining the number of states[69].

Our tests with signer independent gestures shadow the acquired signer dependent accuracy ratings. We observe that the recognition rate of fingersign keyframes drops from 82 per cent for the signer dependent test set to 42 per cent for the signer independent test set. This result clearly shows that as the signers articulation derivate from the training data, the recognition accuracy decreases. We attribute this performance decrease in signer independent recognition to several factors such as variances in signer performances, differences in interpretation of signs, signing speed and hand shape or sizes. Better results are obtained through the usage of language models at word level recognition. Further improvement of these results is possible through the application

of dedicated adaptation methods successfully applied in automatic speech and sign language recognition.

Of our different feature representation methods, we have found Local Binary Patterns and Radial Distance Function to provide the highest recognition accuracies. Elliptic Fourier Descriptors also show a good performance with GMM's modeled using HMMs. A problem in recognizing dynamic letters becomes apparent from comparing the recognition results of all three languages. The experiments showed that the applied features are effective in distinguishing hand gestures, but improvements in accuracy for dynamic hand gestures are possible. Our experiments with different temporal modeling techniques yielded different performances. The method of sequence decision through weighted voting outperformed the use of discrete and continuous HMM's with an accuracy of 92 per cent.

A possible weakness of our fingerspelling recognition method is that we do not make use of spatial position of hands while describing hand gestures. Although we make use of the location of hands and face while initializing tracking and keeping track of keyframes, the spatial coordinates of hand gestures are not represented in any of our features. Although this provides us with an advantage in dealing with performance variations of different signers, it also causes the system to neglect valuable information while dealing with two handed gestures. Therefore, a possible feature level improvement of the system can be achieved by incorporating normalized spatial information to our feature vectors.

Among the three fingerspelling alphabets, the Turkish fingerspelling alphabet yields the highest accuracy rates. We can attribute this to several factors, the most likely being the presence of a higher percentage of two handed gestures and static gestures. In addition, the tracking, feature extraction and recognition modules were all implemented and calibrated using the videos from Turkish fingerspelling dataset. The Russian and Czech languages were incorporated into the system at a later point in the development schedule, which may be a reason of their lower performance.

# REFERENCES


1. Aran, O., *Vision based sign language recognition: modeling and recognizing isolated signs with manual and non-manual components*, Ph.D. thesis, Bogazici University, 2008.

2. Ong, S. C. W. and S. Ranganath, "Automatic sign language analysis: a survey and the future beyond lexical meaning.", *IEEE transactions on pattern analysis and machine intelligence*, Vol. 27, No. 6, pp. 873–91, June 2005.

3. Wu, Y. and T. Huang, "Vision-based gesture recognition: A review", *Gesture-Based Communication in Human-Computer Interaction*, pp. 103–115, 1999.

4. Sturman, D. J., *Whole-Hand Input*, Ph.D. thesis, Massachusetts Institute of Technology, 1992.

5. Starner, T. and A. Pentland, "Real-Time American Sign Language Recognition from Video Using Hidden Markov Models", *Proceedings., International Symposium on Computer Vision 1995*, pp. 265–270, 2002.

6. Birk, H., T. Moeslund, and C. Madsen, "Real-time recognition of hand alphabet gestures using principal component analysis", *In 10th Scandinavian Conference on Image Analysis*, 1997.

7. Ahmad, T., "Tracking and recognising hand gestures, using statistical shape models", *Image and Vision Computing*, Vol. 15, No. 5, pp. 345–352, May 1997.

8. Lien, C., "Model-based articulated hand motion tracking for gesture recognition", *Image and Vision Computing*, Vol. 16, No. 2, pp. 121–134, February 1998.

9. Handouyahia, M., D. Ziou, and S. Wang, "Sign language recognition using moment-based size functions", *Proc. of the Int'l Conf. on vision interface. Kerkyra: CRC*

*Press*, pp. 210–216, 1999.

10. Eikvil, L., "Video Analysis of Sign Language", 2000.

11. Lockton, R. and A. Fitzgibbon, "Real-time gesture recognition using deterministic boosting", *Proceedings of the British Machine Vision Conference*, Vol. 2, pp. 817–826, Citeseer, 2002.

12. Tanibata, N., N. Shimada, and Y. Shirai, "Extraction of hand features for recognition of sign language words", *The 15th International Conference on Vision Interface*, pp. 391–398, Citeseer, 2002.

13. Utsumi, A., N. Tetsutani, and S. Igi, "Hand detection and tracking using pixel value distribution model for multiple-camera-based gesture interactions", *Knowledge Media Networking, 2002. Proceedings. IEEE Workshop on*, pp. 31–36, IEEE, 2002.

14. Yang, M. and N. Ahuja, "Extraction and classification of visual motion patterns for hand gesture recognition", *Computer Vision and Pattern Recognition, 1998. Proceedings. 1998 IEEE Computer Society Conference on*, pp. 892–897, IEEE, 2002.

15. Feris, R., M. Turk, R. Raskar, and G. Ohashi, "Exploiting Depth Discontinuities for Vision-Based Fingerspelling Recognition", *2004 Conference on Computer Vision and Pattern Recognition Workshop*, pp. 155–155, 2004.

16. Goh, P., "Automatic recognition of Auslan finger-spelling using hidden Markov models", *Undergraduate thesis, University of Western Australia*, 2005.

17. Altun, O., S. Albayrak, A. Ekinci, and B. Bükün, "Turkish Fingerspelling Recognition System Using Axis of Least Inertia Based Fast Alignment", *AI 2006: Advances in Artificial Intelligence*, pp. 473–481, 2006.

18. Dreuw, P., D. Keysers, T. Deselaers, and H. Ney, "Gesture Recognition Using

Image Comparison Methods", *Gesture in Human-Computer Interaction and Simulation*, pp. 124–128, 2006.

19. Marnik, J., "The Polish Finger Alphabet Hand Postures Recognition Using Elastic Graph Matching", *Computer Recognition Systems 2*, pp. 454–461, 2007.

20. Assaleh, K., T. Shanableh, M. Fanaswala, H. Bajaj, and F. Amin, "Vision-based system for continuous Arabic Sign Language recognition in user dependent mode", *2008 5th International Symposium on Mechatronics and Its Applications*, pp. 1–5, May 2008.

21. Liwicki, S. and M. Everingham, "Automatic recognition of fingerspelled words in British Sign Language", *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, , No. iv, pp. 50–57, June 2009.

22. Santemiz, P., *Alignment and Multimodal Analysis In Signed Speech*, Ph.D. thesis, Bogazici University, 2009.

23. Erenshteyn, R., P. Laskov, R. Foulds, L. Messing, and G. Stern, "Recognition approach to gesture language understanding", *Proceedings of 13th International Conference on Pattern Recognition*, pp. 431–435, 1996.

24. Erenshteyn, R. and P. Laskov, "A multi-stage approach to fingerspelling and gesture recognition", *Proceedings of the Workshop on the Integration of Gesture in Language and Speech*, 1996.

25. Hernandez-Rebollar, J., R. Lindeman, and N. Kyriakopoulos, "A multi-class pattern recognition system for practical finger spelling translation", *Proceedings. Fourth IEEE International Conference on Multimodal Interfaces*, pp. 185–190, 2002.

26. Jiangqin, W. and G. Wen, "The recognition of finger-spelling for Chinese sign language", *Gesture and Sign Language in Human-Computer Interaction*, pp. 96–

100, 2002.

27. Shahabi, C. and F. Parvini, "UTGeR: A User-Independent Technique for Gesture Recognition", 2005.

28. Lew, Y., A. Ramli, S. Koay, R. Ali, and V. Prakash, "A hand segmentation scheme using clustering technique in homogeneous background", *Student Conference on Research and Development*, pp. 305–308, 2002.

29. Askar, S., Y. Kondratyuk, K. Elazouzi, P. Kauff, and O, "Vision-based skin-colour segmentation of moving hands for real-time applications", *Proc. of 1st European*, 2004.

30. Birdal, A. and R. Hassanpour, "Region Based Hand Gesture Recognition", *ia-son.zcu.cz*, 2008.

31. Black, M. and A. Jepson, "Eigentracking: Robust matching and tracking of articulated objects using a view-based representation", *International Journal of Computer Vision*, Vol. 26, No. 1, pp. 63–84, 1998.

32. Malima, A., E. Ozgur, and M. Çetin, "A fast algorithm for vision-based hand gesture recognition for robot control", *Signal Processing and Communications Applications, 2006 IEEE 14th*, pp. 1–4, IEEE, 2006.

33. Holden, E.-J., G. Lee, and R. Owens, "Automatic Recognition of Colloquial Australian Sign Language", *IEEE Workshop on Motion and Video Computing (WACV/MOTION'05) - Volume 2*, pp. 183–188, January 2005.

34. Hasanuzzaman, M., T. Zhang, V. Ampornaramveth, and M. A. Bhuiyan, "Gesture Recognition for Human-Robot Interaction", *Robotics and Autonomous Systems*, Vol. 55, No. 8, pp. 643–657, 2007.

35. Gui, L., J.-p. Thiran, and N. Paragios, "Finger-spelling recognition within a collaborative segmentation/behavior inference framework", *Proceedings of European*

*Signal Processing Conference, Lausanne, Switzerland*, pp. 2741–2744, 2007.

36. Kakumanu, P., S. Makrogiannis, and N. Bourbakis, "A survey of skin-color modeling and detection methods", *Pattern Recognition*, Vol. 40, No. 3, pp. 1106–1122, March 2007.

37. Keskin, C., O. Aran, and L. Akarun, "Real time gestural interface for generic applications", *European Signal Processing Conf., EUSIPCO Demonstration Session, Antalya*, Citeseer, 2005.

38. YANG, M.-H. and N. AHUJA, "Gaussian mixture model for human skin color and its applications in image and video databases", *SPIE proceedings series*, pp. 458–466.

39. Allen, J., R. Xu, and J. Jin, "Object tracking using camshift algorithm and multiple quantized feature spaces", *Proceedings of the Pan-Sydney area workshop on Visual information processing*, Vol. 36, pp. 3–7, Australian Computer Society, Inc., 2004.

40. Bradski, G., "Real time face and object tracking as a component of a perceptual user interface", *Proceedings Fourth IEEE Workshop on Applications of Computer Vision. WACV'98 (Cat. No.98EX201)*, pp. 214–219, 1998.

41. Kindiroglu, A. A., H. Yalcin, O. Aran, M. Hruz, P. Campr, L. Akarun, and A. Karpov, "Multi-Lingual Fingerspelling Recognition for Handicapped Kiosk", *Pattern Recognition and Image Analysis*, pp. 33–37, St. Petersburg, 2010.

42. Hernandez, A. and M. Reyes, "Spatio-Temporal GrabCut Human Segmentation for Face and Pose Recovery", *CVPR10*, 2010.

43. Goh, P. and E. Holden, "Dynamic fingerspelling recognition using geometric and motion features", *Image Processing, 2006 IEEE International Conference on*, pp. 2741–2744, The University of Western Australia, IEEE, 2007.

44. Fujimura, K., "Sign Recognition using Depth Image Streams", *7th International*

*Conference on Automatic Face and Gesture Recognition (FGR06)*, pp. 381–386, 2006.

45. Chetty, V., "Microsoft's Project Natal for Xbox 360", July 2009.

46. Chen, F., "Hand gesture recognition using a real-time tracking method and hidden Markov models", *Image and Vision Computing*, Vol. 21, No. 8, pp. 745–758, August 2003.

47. Yang, M., K. Kpalma, and J. Ronsin, "A survey of shape feature extraction techniques", *Pattern Recognition*, , No. November, 2008.

48. Suraj, M., "Appearance based recognition methodology for recognising fingerspelling alphabets", *Proceedings of 20th International Joint Conference*, pp. 605–610, 2007.

49. Ojala, T., M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 24, No. 7, pp. 971–987, July 2002.

50. Akakin, H. and B. Sankur, "DCT Based Facial Feature Extraction", *Signal Processing and Communications Applications, 2006 IEEE 14th*, pp. 1–4, IEEE, 2006.

51. Lin, C. and C. Hwang, "New forms of shape invariants from elliptic Fourier descriptors", *Pattern Recognition*, Vol. 20, No. 0031-3203, pp. 535–545, 1987.

52. Fang, Y., J. Cheng, K. Wang, and H. Lu, "Hand Gesture Recognition Using Fast Multi-scale Analysis", *Fourth International Conference on Image and Graphics (ICIG 2007)*, pp. 694–698, August 2007.

53. Ranjan, A., "Using A KNN and MOG Based Algorithm for Static Hand Posture Recognition", *Relation*, Vol. 10, No. 1.92, p. 2328, 2008.

54. Eisenstein, J., S. Ghandeharizadeh, L. Huang, C. Shahabi, G. Shanbhag, and R. Zimmermann, "Analysis of clustering techniques to detect hand signs", *Proceedings of 2001 International Symposium on Intelligent Multimedia, Video and Speech Processing. ISIMP 2001 (IEEE Cat. No.01EX489)*, pp. 259–262, 2001.

55. Liu, R., Z. Li, and J. Jia, "Image partial blur detection and classification", *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, June 2008.

56. Wozniak, M., "Classifier Fusion Based on Weighted Voting - Analytical and Experimental Results", *2008 Eighth International Conference on Intelligent Systems Design and Applications*, pp. 687–692, November 2008.

57. Cheng, Y., "Mean shift, mode seeking, and clustering", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 17, No. 8, pp. 790–799, 1995.

58. Viola, P. and M. Jones, "Robust real-time object detection", *International Journal of Computer Vision*, Vol. 57, No. 2, pp. 137–154, 2002.

59. Xia, J., J. Wu, H. Zhai, and Z. Cui, "Moving vehicle tracking based on double difference and camshift", *Proceedings of the International Symposium on Information Processing*, Vol. 1, pp. 3–6, 2009.

60. Hu, M., "Visual pattern recognition by moment invariants", *Information Theory, IRE Transactions on*, Vol. 8, No. 2, pp. 179–187, 1962.

61. Kuhl, F. and C. Giardina, "Elliptic Fourier Features of a Closed Contour", *Computer graphics and image processing*, Vol. 18, No. 0146-664x, pp. 236–258, 1982.

62. Tort, A., "Elliptical Fourier Functions as a Morphological Descriptor of the Genus Stenosarina (Brachiopoda, Terebratulida, New Caledonia)", *Mathematical Geology*, Vol. 35, No. 7, pp. 873–885, October 2003.

63. Ari, I., *Facial Feature Tracking and Expression Recognition For Sign Language*,

Ph.D. thesis, Bogazici University, 2008.

64. Chang, C.-C. and C.-J. Lin, "{LIBSVM}: a library for support vector machines", 2001.

65. Alpaydin, E., *Introduction to Machine Learning*, MIT Press, London, UK, second edition, 2004.

66. Levenshtein, V., "Binary codes capable of correcting deletions, insertions and reversals", *Soviet Physics Doklady*, Vol. 10, No. 707-10, 1966.

67. Campr, P., E. Dikici, A. Kindiroglu, M. Hruz, Z. Krnoul, A. Ronzhin, H. Sak, D. Schorno, L. Akarun, O. Aran, A. Karpov, and M. Saraclar, "Automatic Fingersign to Speech Translator", 2010.

68. Kass, M., A. Witkin, and D. Terzopoulos, "Snakes: Active Contour Models", *International Journal of Computer Vision*, Vol. 1, No. 4, pp. 321–331, 1988.

69. Siddiqi, S., G. Gordon, and A. Moore, "Fast state discovery for HMM model selection and learning", *Proc. AISTATS*, Citeseer, 2007.