UNIVERSITY OF THE WESTERN CAPE

The Efficacy of the Eigenvector Approach to South African Sign Language Identification

Vaughn Mackman Segers

A thesis submitted in fulfilment of the
degree of Master of Science

in the
Faculty of Science
Department of Computer Science

February 2010

Declaration of Authorship

I, Vaughn Mackman Segers, declare that this thesis titled, 'The Efficacy of the Eigenvector Approach to South African Sign Language Identification' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.

- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.

- Where I have consulted the published work of others, this is always clearly attributed.

- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.

- I have acknowledged all main sources of help.

- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:
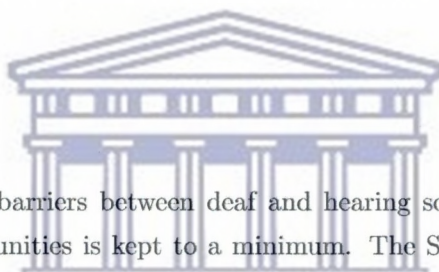
_____

Date:

_____

i

"If I am still standing at the end of the race, hit me with a board and knock me down, because that means I didn't run hard enough."

-Steve Jones, Welsh athlete and former world marathon record holder.

UNIVERSITY *of the*
WESTERN CAPE

Abstract

The communication barriers between deaf and hearing society mean that interaction between these communities is kept to a minimum. The South African Sign Language research group, *Integration of Signed and Verbal Communication: South African Sign Language Recognition and Animation* (SASL), at the University of the Western Cape aims to create technologies to bridge the communication gap.

In this thesis we address the subject of whole hand gesture recognition. We demonstrate a method to identify South African Sign Language classifiers using an eigenvector approach. The classifiers researched within this thesis are based on those outlined by the Thibologa Sign Language Institute for SASL. Gesture recognition is achieved in real-time. Utilising a pre-processing method for image registration we are able to increase the recognition rates for the eigenvector approach.

Acknowledgements

Thanks to God.

Many thanks to all who have helped me while doing my M.Sc. Many thanks to my family and friends, Mackman, Kay, Maxine and Chantal Segers and Ryan Johnson, whose patience and support have been greatly appreciated.

I never expected to study towards a Masters degree when I began my university career. Many thanks to my supervisor, James Connan, for encouraging me to study further and patience through this thesis. Thanks to my friend and fellow Master's student, Nathan Naidoo, without whom I would not have thought to study this long.

A word of thanks to my bursars. I was incredibly fortunate to have my university fees sponsored for the entire duration of my studies. My undergraduate studies were funded by Dell South Africa. My honours degree was funded through sponsorships provided by the Centre of Excellence at the Department of Computer Science. My masters degree was funded by Telkom South Africa. To all these sponsors and those who made the funding possible, many thanks for the opportunity you have given me.

Contents

UNIVERSITY *of the*

WESTERN CAPE

List of Figures

List of Tables

Abbreviations

AI	Artificial Intelligence
ASL	American Sign Language
BSL	British Sign Language
CCTV	Closed Circuit Television
EGM	Elastic Graph Matching
FERET	Facial REcognition Technology
FPS	Frames Per Second
GR	Gesture Recognition
HCI	Human Computer Interaction
HMM	Hidden Markov Models
HSV	Hue Saturation Value
IOHMM	Input Output Hidden Markov Models
JSL	Japanese Sign Language
MHM	Movement Hold Model
NGT	Nederlandse Gebarentaal
NN	Neural Networks
PCA	Principal Component Analysis
RGB	Red Green Blue
SASL	South African Sign Language
TSLI	Thibologa Sign Language Institute

Chapter 1

Introduction

1.1 Background

Gesture Recognition

There has been a recent surge in research into gesture controlled interfaces. A catalyst for this increase in interest is the application to gaming. The Nintendo Wii gaming device allows for intuitive game-play by mimicking user movements within a game. This application has prompted companies to start developing controller and camera gesture-based games and applications. Gesture controlled games generally involve players interacting with virtual environments using the movement of their bodies.

Simple gesture recognition, such as that required for gaming, has revitalised this field of research. The challenge remains, however, to use inexpensive means to achieve accurate results on an established gesture set such as sign language.

Challenges for creating a gesture recognition system include:

- Accurate recognition while still having the potential to cover the entire signed language.

- The division of signs into units for recognition, such as applying phonetic logic present in verbal language to sign language.

- Making sign language translation systems mobile, where data transfer and compression complications exist.

1

Interactive gaming offers insight into potential solutions to the question of gesture recognition but does little to contribute to the mobility of recognition systems. Sign language users require a translation solution that can be used anywhere. Mobile gesture recognition is considered pivotal to future gesture recognition applications.

Sign Language

Sign languages develop independently of spoken languages, from the need for the deaf to communicate among themselves. There is little correlation between signed and spoken languages. Holt [1] writes that the deaf are largely illiterate. Strong [2] states that the reading and writing competencies of deaf students were well below the accepted level of hearing learners in the same grade. It was also shown that speaking a language builds on the writing ability of the user [3], an avenue of learning wholly unavailable to the deaf. Therefore, the assumption that the deaf can read and write is largely false, and is another misconception that further widens the divide between the deaf and hearing community. This can make even the accepted means of computer interaction, a mouse and keyboard, a daunting prospect for the deaf user.

Sign languages are as rich and complex as spoken languages, and are built to fully express the wide range of actions, objects and emotions encountered on a daily basis. South African Sign Language is also a protected language under the South African constitution [4], and the language is used by over 600 000 South Africans [4]. South African Sign language consists of body movements, known as gestures. These gestures can be either manual or non-manual. Manual gestures are performed by moving the hands, arms, fingers and head in relation to the signer's body. Non-manual gestures are the more subtle movements of the face. Non-manual gestures are performed by the raising of the eyebrows, frowning, smiling and similar actions. Both manual and non-manual gestures are required to fully understand the emotion and intentions of the signer. There are two groups of deaf individuals, the hard of hearing with minimal hearing ability, and the completely deaf who are usually born deaf. Therefore members of the deaf community can have different levels of exposure to spoken language.

Integration of Signed and Verbal Communication: South African Sign Language Recognition and Animation(SASL) is a research group at the University of the Western Cape formed to use technology to integrate signed and verbal communication. The group is

concerned with SASL recognition and animation. The group aims to create a system that:

- Translates SASL to English, and

- Translates English to SASL

Eigenvectors

This thesis forms part of the work done by the SASL group. The SASL group aims to create a mobile system for the recognition of sign language. SASL has created a digital phrasebook known as *iSign* which:

- Translates whole-body SASL phrases into English.

- Translates English to SASL.

- Provides a phrase search feature, and

- Operates on a mobile phone with processing done on a server.

iSign does not recognise the hand shapes of the user. A fast method for hand shape recognition is needed. This thesis investigates the eigenvector approach applied to fast hand shape recognition.

Eigenvectors are used to reduce the dimensionality of a set of values. Therefore, large images can be reduced in size but still retain necessary variation. Lowered dimensionality makes comparisons of different hand shapes less computationally expensive.

1.2 Motivation

Imagine a simple trip to the doctor's office. Consider the lines of communication between the deaf patient and the English speaking doctor. The doctor would require some sort of English to sign language interpretation, as a deaf individual would likely not know how to write in English [2] to communicate with the doctor. The deaf individual would also require interpretation, to translate from sign language to English.

In these situations the services of a sign language interpreter are required. In the South African context, these services can be difficult to access. Not only are these services costly, but there are also few properly trained interpreters available [4]. In instances where a deaf parent is accompanied by a family member such as a hearing child, this offers a solution to the problem of interpretation [4]. In some cases, hearing teachers of sign language act as interpreters. These informal interpreters are not fluent in all aspects of sign language as they largely interact with a hearing world, i.e. where children of deaf parents would likely be reared by their hearing family [4]. Consider also the implications of privacy on a situation such as this. These interpreters are not versed in the obvious ethical issues involved in interpersonal communication. Privacy is also an obvious problem if professional interpreters are enlisted.

Professional interpreters also have to be scheduled in advance, making spur of the moment interpretation near impossible. Lotriet [4] highlights the problems at South African police stations and courts where "gross injustices" [4] occur as deaf members of society have little to no access to professional interpretation. It is in these situations that an automated, real-time, non human-reliant system would be of great assistance.

Beyond the aforementioned there are other areas of application of real-time gesture recognition. These areas include:

- Computer game control
- Human-robot interaction
- Human Computer interaction (HCI) and
- Automated homes.

1.3 Problem Statement

Gesture recognition has been extensively studied within the SASL project. Whole body gesture recognition has been researched by Naidoo and Connan [5]. Non-manual gestures of the face have been studied by Whitehill [6].

Hand gestures have not been researched within the SASL project. Hand gestures are difficult to recognise due to changes in shape between hand signs and individuals. The

challenge for this thesis is to accurately recognise hand gestures between different individuals.

1.4 Research Question

Can eigenvectors be applied to hand shape recognition?

Can eigenvectors be used in a sign language recognition system such as the SASL project?

1.5 Research Hypothesis

- SASL hand shapes can be separated into distinct classes.

- Eigenvectors can be applied to hand shape recognition.

- Eigenvectors can provide real-time recognition speeds.

- This system can be incorporated into the larger SASL project.

- Low resources are needed for recognition.

1.6 Technical Objectives

To address the questions raised above we must perform the following tasks:

- **Gather a data set**: Test subjects must be gathered to obtain test and training data. The data set must contain all hand shapes in our chosen classifier system. Video recordings from different individuals are needed for the training and recognition process.

- **Pre-processing**: As gesture recognition is known to decrease in accuracy with large amounts of image noise, we must determine proper image registration techniques. Image registration is needed to focus recognition on *the hand only* in the captured image. As we train the system we will consider which method provides the greatest accuracy with the least impact on real-time operation.

- **Recognition**: Accuracy of classifier recognition is the central system goal. The system must identify a particular *hand shape* from different unseen users.

- **Time Complexity**: To be usable in real-world situations we need to keep computational complexity to a minimum. The objective is for the system to perform in real-time.

1.7 Research Methodology

The methods we use to achieve our technical objectives are the following:

- Random non-natural SASL users are asked to perform the signs. The signs are described by the Thibologa Sign Language Institute and bridge discrepancies in signs from various signing groups.

- Our image registration methods find the regions of interest within the images captured. We only train our system on images containing the hands, but test images have greater freedom of movement. We use the contours of the hand and the known parameters of the image to assist the registration techniques. We re-size the images to known bounds to ensure correlation between all images.

- Our recognition follows known eigenvector techniques successfully used on facial recognition. We apply these techniques to our set of hands to gauge the usefulness of this technique to the task of hand gesture recognition.

- We implement code in an efficient manner to ensure the system remains real-time. Eigenvectors are chosen as they are suited to real-time applications. Registration methods that assist the time complexity are given preference.

1.8 Research Contributions

As stated previously, there has been great interest in the area of gesture recognition. Systems have been built around invasive hardware that is largely not intended for mobile use. Systems have been built incorporating complex methods that cannot perform in real-time. A system is required to recognise sign language hand gestures in the South African context, in real time.

1.9   Outline of Thesis

The remainder of this thesis is organised as follows:

- **Chapter 2** *Sign Language Classifier Predicates* This chapter outlines the use of classifiers within verbal and sign language systems around the world. We describe how oral languages identify classifiers. We give particular consideration to the classifier methods used in SASL. We consider the similarities and differences for various sign languages and what each defines as a classifier. We describe the chosen classifier method and outline advantages, disadvantages, and the use thereof in a SASL recognition system.

- **Chapter 3** *Communicating through Gestures* In this chapter we consider the research already done in the field of gesture recognition. We consider all forms of gesture recognition, from whole-body to hand-only gesture recognition. Popular, novel, as well as recent methods in this field are analysed. We place particular emphasis on the application of eigenvector systems to gesture recognition. To complete this argument we also review the application of eigenvectors to other areas of research, such as faces. From this review we draw conclusions on the feasibility of an eigenvector-based system.

- **Chapter 4** *Eigenvector Theory and Implementation* This chapter gives an overview of the theory behind eigenvectors. A mathematical approach is taken to give an understanding of the statistical justification of the method employed. Formulas as well as derivations are given, along with a brief introduction to the prerequisite knowledge to better understand the subject matter. A graphical illustration of the application of the theory is presented, and the ability of eigenvectors to compress matrix data efficiently is shown.

- **Chapter 5** *Image Registration* This chapter explains the important step of image registration. The pre-processing methods used for cropping and resizing the image to remove only the hand are explained. We outline the importance of such pre-processing to this particular form of recognition and the reasoning behind the use of each step. Where necessary, graphical demonstrations of the operation of the pre-processing step are provided.

- **Chapter 6** *Experimental Setup and Testing* The testing chapter outlines three areas of the system. Firstly, the implementation and experimental setup is described. This explains the creation of the system. Thereafter we describe the data acquisition methods for testing and training of the system. Finally, we present the test results of the system in terms of accuracy and time complexity on the test and training images.

- **Chapter 7** *Conclusions and Directions for Future Work* This chapter presents the conclusions and potential areas for future work for this research. We conclude on the reliability and accuracy of an eigenvector-based system and the real-time application thereof. For future work we consider what is required to extend this work to a full SASL recognition system and other areas of interest.

1.10 Summary

In this chapter we have introduced the work done within this thesis. A background to gesture recognition, SASL and eigenvectors has been provided. We have introduced our problem statement, research hypotheses and methodology. The research goals are outlined as well as a summary of the chapters within this thesis.
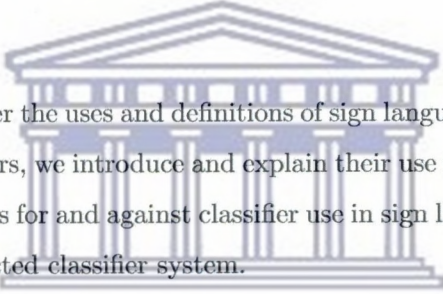
Chapter 2

Sign Language Classifier Predicates

In this chapter we cover the uses and definitions of sign language classifiers. To introduce the concept of classifiers, we introduce and explain their use in verbal language. We then consider the arguments for and against classifier use in sign language. Finally, we outline the merits of our selected classifier system.

2.1 Verbal Language Classifiers

The term classifier is largely accepted and understood when used in reference to spoken language, where parts of speech are divided into classes such as nouns and verbs. The term *classifier* is derived from the word *classify*, which means to "arrange in classes or categories" [7]. Therefore we provide an introduction to classifiers in verbal languages.

Classification systems in verbal languages are researched in terms of morphemes. A morpheme is considered "a meaningful morphological unit of a language that cannot be further divided (e.g. *in, come, -ing* forming *incoming*)" [8]. An extensive study by Aikhenvald [9] presents the nature of classification in over 500 natural languages from around the world. The languages studied are from areas such as East Asia, South America, the South Pacific and Australia. Studying the morphological patterns classifying nouns, comparisons are made between these greatly differing languages. Classifiers are found to perform similar tasks across the languages studied, such as numerical

9

Classifier Type	Semantic and Pragmatic Function
Numerical Classifier	Quantification, Enumeration
Noun Classifier	Determination
Verbal Classifier	Object/Subject Agreement
Relational Classifier	Possession
Possessed Classifier	Possession
Locative Classifier	Spacial Location
Deictic Classifier	Spatial Location, determination

TABLE 2.1: Functions of classifiers in verbal language as identified by Zwitserlood in [10].

Class	Prefix	+	Stem	Quantitative Noun	English
Class 1	**um-**	+	-fundi	**um**fundi	student
Class 2	**aba-**	+	-fundi	**aba**fundi	students

TABLE 2.2: A Table representing the numerical classification of nouns in Xhosa [11].

classifiers, which enumerate nouns. Other cross-linguistic classifiers identified include possessive classifiers, relational classifiers, verbal classifiers, locative classifiers and deictic classifiers. Though Aikenvald states that no language contains all classifiers noted previously.

In Table 2.1, we see classifiers as defined by Zwitserlood [10], similar to those described by Aikhenvald [9].

The work of Aikhenvald [9] provides a comprehensive insight into classifier use in various verbal languages. We now provide an example of noun classification from the South African languages of Xhosa and Afrikaans.

Xhosa

The Xhosa language divides nouns into classes by prefix. This example considers the Xhosa Numerical Classifier. Within the Xhosa language nouns are always divided into the parts: *prefix + stem* [11]. An example of this can be found in Table 2.2.

The example in Table 2.2 demonstrates the numerical classifier defined in Table 2.1. Class 1 defines the singular object, while Class 2 defines the plural. In Xhosa the prefix, in this example *um-* and *aba-*, indicates the class of the noun. In Class 1 and Class 2, these prefixes also quantify the noun. The stem, in this example *-fundi*, is the meaning

Class 1 (singular)	Class 2 (plural)	English (singular)
umfundi	abafundi	student
umntu	abantu	person
umntwana	abantwana	child
umfazi	abafazi	woman
umfana	abafana	young man

TABLE 2.3: A Table demonstrating the creation of noun plurals by altering the prefix.

Class 1 (singular)	Class 2 (plural)	English (singular)
tand	tande	tooth
wiel	wiele	wheel
stoel	stoele	chair
tou	toue	rope
tong	tonge	tongue

TABLE 2.4: A table demonstrating the use of the *-e* suffix in creating plural nouns in Afrikaans [12].

of the noun. Classes 1 and 2 apply to people only, with further examples shown in Table 2.3.

Afrikaans

Another example of classifiers used in verbal language can be seen in the Afrikaans language demonstrated in Table 2.4.

Just as in many other languages, Afrikaans contains word classes such as nouns, verbs and pronouns. In this example, we demonstrate the numerical classification of nouns in Afrikaans. The suffix *-e* is used to denote the plurals of many nouns in the Afrikaans language [12]. Basic use of the plural noun can be seen in Table 2.4.

The provided examples from both Xhosa and Afrikaans show how numerical classifiers are used in verbal languages. Changes in morphemes such as prefixes and suffixes are shown to alter the meaning of a word. We now consider the extension of this classifier system to sign languages.

2.2 Signed Language Classifiers

Some linguists believe that these same linguistic traits can be applied to sign languages. This idea of classifiers, or "iconicity", was first introduced by Frishberg [13] in 1975,

putting forward the notion that a pre-defined hand-shape and location are representative of an entity. There are, however, some researchers who disagree with this supposition. Not all linguists agree that classifiers in signed languages can be identified in the same way as verbal language classifiers [14] [15] [16].

Verbal Language Views of Sign Language Classifiers

We first consider the definition from researchers who agree that some commonality exists between sign and verbal language classifiers. William Stokoe was one of the first researchers in this field. Stokoe [17], the inventor of the Stokoe notation[1], says this of classifiers in American Sign Language (ASL):

> *Possibly to be counted as a kind of pronoun, [CLASSIFIERS] is a special class of signs that share some of the functions of collective nouns and indefinite in pronouns English. These signs were called "classifiers" by Kegl & Wilbur [18]. ASL like many languages requires special forms to go with antecedents semantically classified; e.g. "something long and thin", "something hollow", "something self-propelled".*

Stokoe designates the *dez, sig* and *tab* aspects of a sign language gesture. They represent the designator, action and place elements respectively within the gesture. Classifiers are considered as a *dez* aspect of a sign gesture.

Ted Supalla [19] was the first to look closely at these classifiers, with a view to classify the classifiers. He designated signs into two types: "Size and Shape Specifiers"(SASS), which, as the name implies, denotes the size and shape of a referred entity by hand shape. The position and bending of all fingers contribute to the meaning of the classifier. The second type is known as a "semantic classifier", where a particular hand shape can represent a general class, such as a vehicle. Examples of these two classifiers are shown in Figure 2.1. A third classifier, known as the "handling classifier", was introduced by Schick in 1990 [20]. This classifier is used when an entity is referred to that is handled or gripped. An example of a handling classifier can be seen in Figure 2.2.

---

[1]Stokoe notation is a scripting method for sign languages.
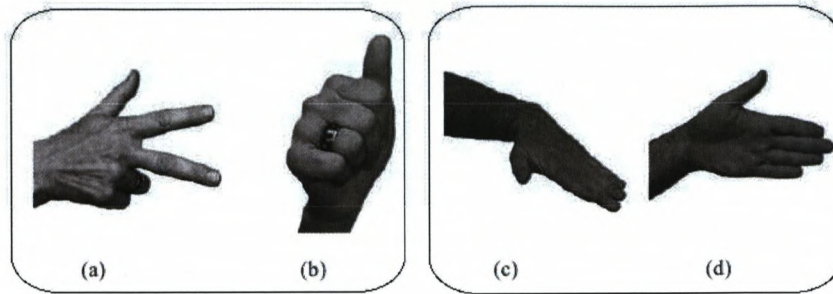
FIGURE 2.1: ASL Entity Classifiers (a) & (b): Vehicle, Object.
ISL SASS classifiers (c) & (d): Flat Object-Car, Flat Object-Bike [21].

FIGURE 2.2: Two Examples of handling classifiers in ASL
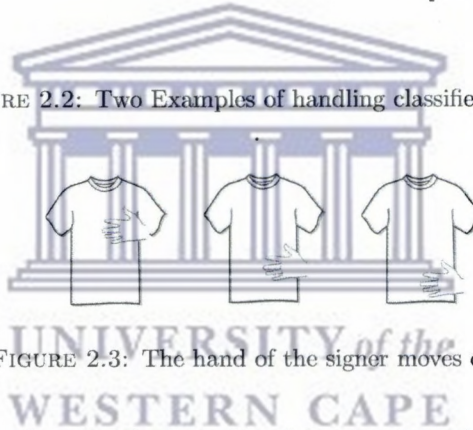
FIGURE 2.3: The hand of the signer moves down.

Inge Zwitserlood also offers insight into classifying hand configurations in sign language
in [10]. According to Zwitserlood, sign language classifiers are usually considered "mean-
ingful hand configurations". Her work shares many ideas with that of Stokoe [17] and
Supalla [19]. In her study of the sign language system of the Netherlands[2](NGT), she
notes that these "meaningful hand configurations" are accompanied by movement and
motion within a sign when gesturing. Movement refers to the articulation of the hand
within the sign, whereas motion applies to the motion of the hand referred to. The
differences between movement and motion are illustrated in Figures 2.3 and 2.4.

Zwitserlood also distinguishes between manual and body gestures. Manual gestures
do not include the use of the body of the signer, and it is these manual gestures that
are covered in her study. The classifiers for motion and movement can be seen in the
examples from NGT in Figures 2.5 and 2.6.

---

[2]The full name for Netherlands sign language is *Nederlandse Gebarentaal*.

FIGURE 2.4: The hand of the signer changes shape.
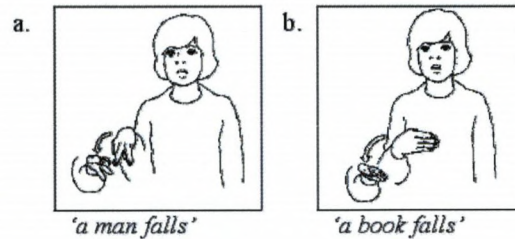
FIGURE 2.5: The NGT signage for 'a man falls' and 'a book falls' respectively [10].
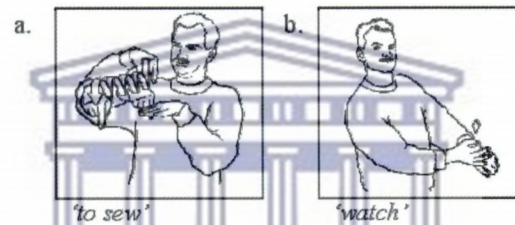
FIGURE 2.6: The NGT signage for 'to sew' and 'watch' respectively [10].

In Figure 2.5 we see the same action *falls* accompanied by different classifiers, *man* and *book* respectively. The same motion is used to represent the action of falling. The classifiers used determine the object referred to. The "meaningful hand configuration" containing the extended index and middle fingers in Figure 2.5a refers to a legged entity, in this case, a human being. In Figure 2.5b the flat open hand configuration refers to a flat entity, a book, in this instance. From here we can see how classifiers, otherwise known as the configuration of the hand, change the meaning in a signed phrase in NGT. This notion of hand shapes representing objects is not only seen in NGT. For example, the sign for 'small animals' in ASL is shown by the hand configuration in Figure 2.7.

FIGURE 2.7: The ASL signage for 'small animal' [10].

FIGURE 2.8: Clockwise from top-left, these images represent the South African Sign Language (SASL) sign for 'mouse' in computing [23].

Schematic Views on Sign Language

Alternatively, the view expressed by Cogill-Koez [22], Schembri [15] and Liddell [16] is that the division by morphemes of signed gestures, location or movement, is generally non-linguistic. Their argument views sign language gestures as a simplified gestural representation of the event or action. In other words, a specialised and sophisticated form of pantomime or *playing charades*. This particularly applies to what is classed as verbs in oral languages. The argument of Cogill-Koez [22] proposes that signs are inherently schematic in their strategy.

The SASL sign for mouse, in terms of a computer, is an example of this. The sign for computer can be seen in Figure 2.8, occurring in the first 3 frames, as a subgesture of the full sign. Movements shown across all frames are needed to perform the sign. The final two frames are seen to be an imitation of the use of a mouse. The imitation of the use of the mouse is a schematic representation of it's use.

Cogill-Koez further emphasises the separation between spoken languages and signed languages. It is concluded that not all gestures are created from a sum of basic classifiers, instead drawing from the convenience of performing the gesture. It is suggested that with this realization classifier predicates are unnecessary in the description of signed languages. It is shown that, rather than doing away with classifier predicates, classifier predicates can and must be studied alongside these schematic representations of signs [22]. The linguistic basis of signs is therefore thought to be modified to include these schematic representations into the area of sign language research.

2.3 Classifiers for SASL

As shown above, there exist many definitions for classifier predicates in sign languages. A definition was sought for the specific SASL context. The Thibologa Sign Language Institute(TSLI), established in 2005, defines a set of SASL classifiers. The purpose of the TSLI is to assist the South African deaf by easing communication between themselves and the hearing by using SASL. As such, the Thibologa Sign Language Institute has defined classifiers as shown in Figure 2.9. The Thibologa Institute has made these classifiers available to the public through their SASL instruction booklet [23].
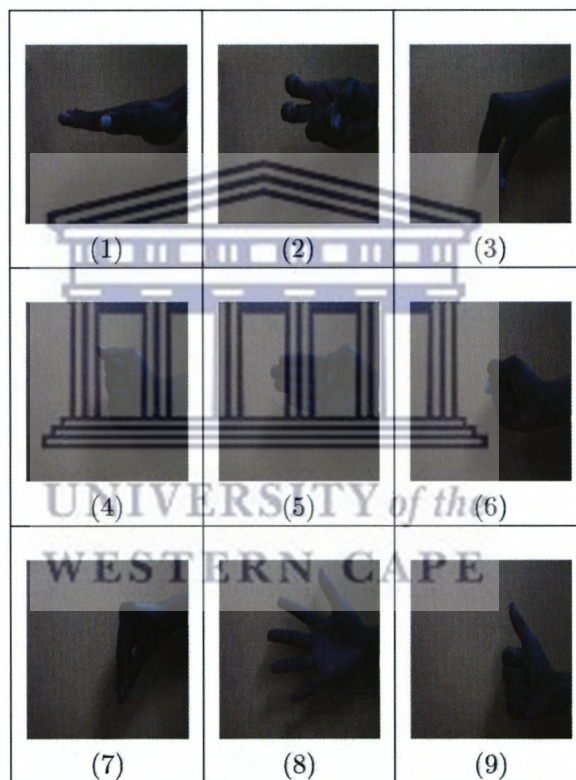
FIGURE 2.9: Classifiers defined by the TSLI described in Table 2.5

These images demonstrate that the TSLI SASL classifiers are visibly similar to those found in the work of Supalla [19] and Supalla and Liddell [24]. The meaning of the signs in Figure 2.9 are found in Table 2.5.

Index	Hand Description	Possible Sign
1.	Palm/Flat	Child, House, Care, Happy
2.	Two long thin bent extensions	Issue, Research, Book, Rat, Rules
3.	Narrow/Shallow Flat Object	Money, Video, Sandwich, Document
4.	Round/Spherical Object	Duster, Audiology, Act, Exercise, Drive
5.	Flat/Long Smooth Surface	Make-up, Butter, Brown, Baking Tray, Paper
6.	Fist	Cough, Run, Cook, Hold tight, Bath towel
7.	Flat/Triangular Object	There, Opinion, Snake, Bring, Decorate
8.	Index Five	Colour, Ball, Light, Encourage, Integrity
9.	Compact Mass with Salient Extension	Good, Bad, Lift, Neighbour

TABLE 2.5: The nine SASL classifiers and the various possible signs they represent.

2.4 Conclusion

We conclude that whether a hand shape is classified in relation to verbal language or schematic methods, the specific "meaningful hand configuration" always portrays a class of meaning to sign language, and specifically to SASL.

2.5 Summary

In this chapter we introduced and described the use of classifiers in a broad spectrum of languages studied by Aikhenvald [9], and specifically in the verbal languages of Xhosa and Afrikaans. We have extended this idea to explore classifiers in sign language. We have discussed the arguments for and against the two methods proposed for classifier interpretation in sign languages. We have also introduced the chosen classifier set used in this thesis to be that defined by the TSLI.

Chapter 3

Communicating through Gestures

In this chapter we review different recognition methods. We consider the use of gestures in human communication, from simple gesticulation to complex sign languages. We describe gesture recognition methods relating to the recognition of whole-body gestures and hand-shape specific gestures. We also describe the implementation, application and results for these various techniques.

3.1 Gestures

Gestures form a vital part of everyday human interaction. Gestures comprise movements of the hands, arms, head, face and even the body to convey messages or emotions. Gestures are used to interact with the environment, and also occur in differing social situations.

FIGURE 3.1: An example of a simple gesture [25].

18

Gestures offer a complete communication tool for sign language users. All sign language communication consists of manual and non-manual gestures. Manual gestures are movements of either the hand or arm, or a combination of the two. Non-manual gestures are movements of the face in the form of facial expressions. It is a combination of these manual and non-manual gestures that are used to communicate in sign language. It is a common misconception that sign language is merely another representation of a spoken language, as writing English is to speaking English. As diverse as spoken languages are from one another, so too does sign languages vary from region to region among different cultures. For example, British Sign Language (BSL) is used in Britain, whereas South African Sign Language is used in South Africa. Neither of these sign language systems are related to a spoken language.

General gestures, not attributed to specific sign languages, are "ambiguous" and "incompletely specified", as detailed by Mitra and Acharya in [26]. An example is the gesture to *stop talking*, generally attributed to an index finger pointed upwards, placed over the mouth. This same expression can also be gestured by a *pinching* action over the lips. There can be differences in a particular sign when considered by different cultures, people, and even the same person at differing instances. It is common for people to use gestures, i.e. *gesticulate*, when talking on the phone. Even blind people are known to gesticulate when speaking.

Gestures have also been used for Human-Computer Interaction (HCI) to make interfaces simpler and more intuitive. Typically, these have been used to improve the interactions with virtual environments, such as the cellular phone or the computer desktop. Examples of this include using a wave of the hand to navigate menus or performing a specific hand-shape to start and stop music playback.

These examples show how gestures are important to human communication, providing the basis for gesture recognition.

3.2   Gesture Recognition

With the knowledge of gestures as discussed in Section 3.1, we now look at the various methods used for gesture recognition. These methods cover areas such as statistics, artificial intelligence, and others, in their approach to gesture recognition. These approaches

include Hidden Markov Models (HMM), Neural Networks (NN), Support Vector Machines (SVM) and others. This section concentrates on whole body recognition while hand shape recognition is considered separately in Section 3.3. Systems comprising both static and dynamic recognition techniques are covered. Static gestures, where a specific shape is made and then held, are also known as postures. Dynamic gestures are gestures made over time, such as waving a hand. Figure 3.2 is an example of a static gesture.
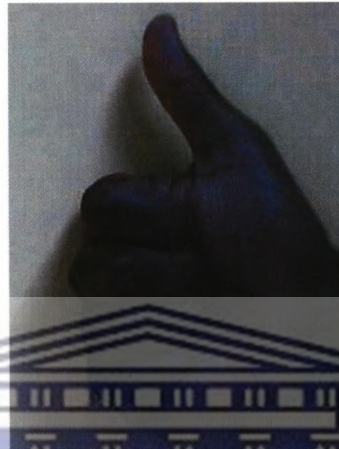
FIGURE 3.2: An example of the *thumbs up* gesture.

3.2.1 Hidden Markov Models

HMM's have been successfully used in handwriting and speech recognition [27]. The use of HMM's is seen as favourable to apply to the problem of sign language recognition when the Movement-Hold (MH) model is used. The MH model states that signs are built of a sequential series of movement and hold positions [16].

Starner, Weaver and Pentland present a HMM system for the recognition of American Sign Language on a 40 word lexicon [28]. Their system works in two modes, with the user in front of the camera and with the camera placed on the head of the user. Both methods place the user in a seated position and capture whole-body signs. The system captures the hand by skin detection and tracks the skin as blobs. These blobs are differentiated as left and right hands depending on which is leftmost and rightmost in the image. They note that their skin detection model is not reliable under differing lighting conditions, which can lead to unreliable tracking of the hands. It is useful to note that this work is an extension of earlier work done by T. Starner and A. Pentland [29] which relied

on coloured gloves to identify the hands. Gloves however are not practical to everyday situations and thus an *a priori* skin map is the next step to demonstrate effectiveness without coloured gloves. The recognition accuracy of the system is 92% when the camera is in front of the user and increases to 98% when the cap-worn system is used. Their work shows a high accuracy and gives credibility to the use of HMM's.

One significant restricting factor to the use of HMM's and the MH model is the large number of training images that need to be used. To achieve the level of accuracy obtained, 384 sentences were required for training on their 40 word lexicon. This amounts to almost ten training sentences per word, with 5 words per sentence. To put this into perspective, there are 6000 ASL signs, each requiring a minimum of 10×5 training examples for effective recognition. Thus 300000 training images are needed to cover the entire language.

3.2.2 Neural Networks

The work done by Marcel et al. [30] combines the advantages of neural networks(NN) and input-output Hidden Markov Models (IOHMMs) for a full-body gesture recognition system. The first step in recognition for this system is to detect the face, as detection is calculated on a face-centered basis. Hands are then found corresponding to skin colour blobs, after which a neural network is used to detect the hand posture on the face-centered space of the user. IOHMMs have been used in this method over the typical HMM approach as it allows for mapping of the input sequences onto the output sequences. Gesture paths are extracted in a 2D space by detecting the skin-blob corresponding to the hand and tracking the motion over time. The goal of this work is to differentiate between two classes of gestures, deictic and symbolic. Deictic gestures occur when a pointing movement is made to the left or right of the body-face space. Symbolic movements also occur on the left and right of the body-face space but denote gestures intended for commands such as grasping or rotating. Their method achieves a recognition rate of between 90% and 100%.

3.2.3 Miscellaneous Recognition Methods

This section looks at other methods that do not fall into a specific category but use differing and unique methods to achieve gesture recognition.

A novel method is put forward by Alon et al. in their work on gesture recognition and spatiotemporal gesture segmentation [31]. Their system does not require any segmentation of the hand, as bad segmentation can adversely affect the outcome of recognition. A colour-based method is used to detect the skin, but they take an interesting approach to choosing the skin region. Rather than finding the first or largest connected skin region, their system finds every skin region. This gives multiple candidate regions for the position of the hand. Unlike other methods which segment and then recognise, this method performs segmentation *with* recognition. Recognition is performed on each of these candidate locations and as such processing time increases with each possible hand region detected. Training and test data for the system is comprised of continuous numerical digits gestured by the signer. An example of this can be seen in Figures 3.3 and 3.4. A problem for this type of matching is that the gesture for the number 5 is a subgesture of the number 8. Using this method it is seen that if one gesture is fully within another, all but the lowest matching model will be pruned away. Therefore, the first gesture found will not be flagged as the gesture until it is conclusively found not to be a subgesture.

Experimentation involved users performing the numbered gestures in sequence. Tests were done to compare the effect of the subgesture reasoning on the test outcomes. Hard and easy data sets were used. The easy data set contained only the user in front of the camera, while the hard data set intentionally contains background movement including non-gesture skin regions. The accuracy was improved from 79% to 94.6% when subgesture reasoning was introduced to the easy data set. On the hard data set, an accuracy of 69.2% was achieved before subgesture reasoning, and 85% thereafter.

This method is computationally complex, and made more so when more potential hands are introduced. This is because all skin coloured objects are tracked as hands until shown to be otherwise. A pruning method [31] was later introduced so that a tracked hand, found not to be within the training boundaries, was removed immediately to speed up processing. This method offers acceptable accuracy along with novel ideas for continuous gesture recognition.

FIGURE 3.3: The gestures performed for the numbers four and five respectively [31].

FIGURE 3.4: The gestures performed for the numbers eight and nine respectively [31].

3.3 Hand shape Recognition

We now consider the specific field of hand shape recognition as applied to gesture and sign language recognition. Factors which increase the difficulty of hand shape recognition include:

1. Segmentation of the hands between signs [32],

2. Hand segmentation from the image background [32],

3. Gesture differences between different signers [32], and

4. Gesture differences between the same signer at different instances [32].

As such segmentation from cluttered backgrounds presents a challenge for this field of study.

3.3.1 Elastic Graph Matching

We consider the work by Jochen Triesch and Christoph von der Malsburg [33]. Their work called "A System for Person-Independent Hand Posture Recognition against Complex Backgrounds" proposes a method for hand shape recognition using Elastic Graph Matching (EGM). This view-based method was used for the inherent ability of EGM to

handle the variations in hand shape and as well as not requiring a perfectly segmented hand. EGM has been successfully used for face detection and recognition as well as the recognition of objects.

In this method, the images are represented by a series of graphs. This series of nodes and edges placed over an image denotes the shape of the hand. Using this manually created graph of the hand, a series of feature types are extracted at the nodes of each graph. An example of a manually created elastic graph can be seen in Figure 3.5. Gabor Jets, Bunch Graphs, Compound Jets, Compound Bunch Graphs, Color Average and the Color Gabor Jet are the features extracted from every node. These nodes contain color, 2D position, whole image and average image information of the nodes in each graph. Recognition is achieved by shifting the trained graph over the test graph until a match is found. The graph, along with the edges and nodes, moves within certain allowable parameters while searching for a match over the test image. In this way minor deformations in the hand shape, size and orientation are accounted for. The system makes use of a set of more than 1000 color images, 72 of which are used for training. Under this EGM system, combined with one or more extracted features, recognition rates of up to 92% were found on simple backgrounds and 85.8% on complex backgrounds. A large amount of manual work is done to label the images with the nodes and edges of the elastic graph. Triesh and von der Malsburg note this as a limiting factor to the amount of images placed in their training set.
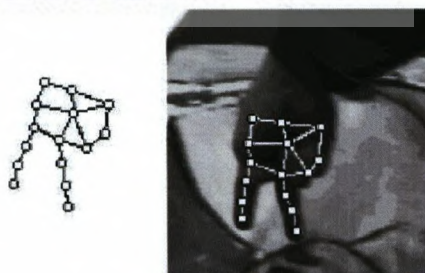

FIGURE 3.5: A visual representation of Elastic Graph Matching [33].

3.3.2 Chamfer Distance

Potamias and Athitsos [34] perform hand shape recognition using a Chamfer distance matching approach. The intention of their work is to detect 20 different hand signs using a large data set of signs. The challenges of this task are to effectively label and

index the images for searching. Embedded and hash-based methods are used to search the data set. The system then needs to find the closest match between the data set of images and the input image. To do this, edge images are created that will be compared using the Chamfer distance [35], a method known to find the distance between two edge images. To find these edge images they use the Canny edge detector [36]. They find that using the Chamfer distance on their data set takes an unacceptably long time of two minutes, thus they propose an embedding system to speed up the process.

Boostmap and Lipschitz embedding is used for embedding the edge images into a vector space. Embedding is effective in speeding-up the process significantly without a great impact on the recognition rate of the system. Test results reveal a 90% to 99% retrieval accuracy depending on the embedding method used. These results are offset against the comparative time taken for the brute force method to complete the same matching. When comparing the embedding to brute-force, a speedup factor of 300 is obtained.

3.3.3   Depth Mapping

R. Feris et al. present a method for hand-shape recognition using image-depth capturing hardware [37]. The system consists of a camera with four flashbulbs placed on either side, as well as above and below the camera.

The flashbulbs flash sequentially, highlighting depth discontinuities from the four opposing perspectives. A combination of these four images produce a depth map found to show more relevant edging than that of Canny edge detection, as seen in Figure 3.6. A shape descriptor [38] is applied to the set of training images to classify them, invariant to image translation and scale. A nearest neighbour classifier is used for recognition. When a test image is encountered and the edges found, the shape descriptor is applied. The image is identified by a nearest neighbour comparison to the training images. This method achieved a recognition rate of 96%, compared to the Canny images, which achieved a recognition rate of 88%.

Another technique for reliable hand-shape recognition is presented by Fujimura and Liu . This method has been applied to the recognition of Japanese Sign Language (JSL). A unique depth-camera is used to determine the depth of images within a scene. The
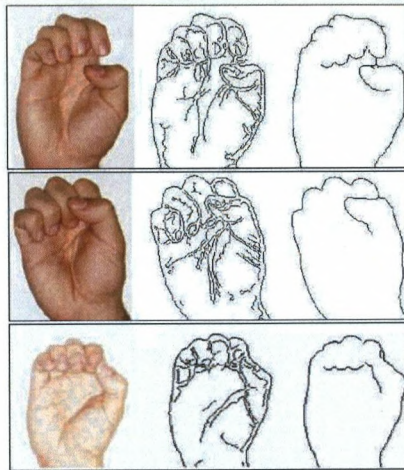
FIGURE 3.6: The fist column represents a hand image, the second column shows Canny edge detection. The third column demonstrates the depth-based edging. Each row shows a different hand shape and ambient lighting condition [37].

depth camera was developed by 3DV Systems [39] and calculates scene-depth in real-time[1]. Using a large set of training data, the system achieves an "error rate [OF] 5% or less" [40].

Depth information from the 3DV camera is lighting invariant and reduces the problems encountered due to occlusions. It is clear that depth information provides a useful aid to reliable gesture recognition.

FIGURE 3.7: An example of the depth map produced by the 3DV camera. Lighter areas are closer to the camera [40], darker areas are further away.

---

[1]3DV Systems has been acquired by Microsoft Inc. and this technology is being used with their Xbox360 gaming system.

3.4 Eigenvector Vision Systems

The eigenvector approach[2] has been used extensively in the area of facial recognition [41] [42] [43] [44]. The term *eigenfaces*[3] has since been coined to describe eigenvectors when used in a facial recognition system.

An eigenface approach is essentially a 2D matching technique, using face-forward images of the face in controlled lighting conditions.

3.4.1 Eigenfaces

The work of Sirovich and Kirby [51] [52] contributed greatly to other systems under review within this section. Their 1986 work [51] continues the use of eigenvectors in facial recognition. This initial study was done on 115 different face images. The images were all taken under the same conditions with respect to lighting and head orientation. Their later work [52] builds on this by making use of the same image set, but setting about the recognition in a different manner.

In [51] the images subjected to eigenvector matching are cropped over the eyes and nose. These fitted and cropped faces are used for training the eigenvector system. The test subjects were also carefully chosen to maintain a relatively homogeneous population. Test subjects were randomly chosen smooth-skinned Caucasian males. From these experiments it was concluded that there exists enough variance in parts of the human face (the eyes and nose) for accurate recognition. It was also found that to use the full face would increase the number of features and the number of errors in recognition.

In [52], the authors take a different approach to modifying the images before training and recognition. This approach used vertically mirrored images in the training set, thereby extending the amount of images in training. This manipulation is done to reduce the approximation error for images not included in the original ensemble of images. Final conclusions determine that these modifications are beneficial to recognition. The eigenvector approach also successfully reduces the amount of data required to recognise a human face.

---

[2]The eigenvector approach is also known as Principal Component Analysis or PCA.

[3]Those who recognise faces term them eigenfaces [45] [46] [47], while those who recognise images term them eigenimages [48] [49], *ad infinitum*. This thesis will use the term eigenvector as in previous work, [50].

In the 1991 work of Matthew Turk and Alex Pentland [47], they describe a near real-time system for facial recognition using eigenfaces. Their work, influenced by Sirovich and Kirby [51] [52], further explores the conditions around creating an effective eigenface system for facial recognition. A set of 2500 facial images are used in their study, consisting of faces in various lighting conditions, orientations and sizes. The effectiveness of eigenfaces under these varying conditions is given considerable attention, though primarily they seek to create an effective real-time facial recognition system. From this work we see how different lighting conditions, sizes and orientations have an effect on recognition rates. Specifically, according to their results, when a particular face is to be recognised, lighting would reduce recognition by 19%, orientation would reduce recognition by 39% and size would reduce recognition by 60%.

From these results it is clear that registration is a challenge facing any eigenvector-based recognition system. Registration is the ability to crop and scale the images to match the training set. Their suggested solutions are to either crop and scale the images, or to ensure that the training set contains versions of the image at various sizes. The near real-time implementation and apparent robust performance to changes in lighting conditions make the overall results a success.

This is continued in the work of Pentland et al. in [46]. They delve extensively into the use of eigenvectors for the matching and characterisation of human faces on a large set of images. Their 1994 paper on "View-Based Modular Eigenspaces for face recognition" tests the eigenface concept on the largest face set at the time.

Their research used a set of 7562 facial images of approximately 3000 different people. This was significantly greater than comparative research which only used a few hundred face images. Images were full-face, with images of the same person occurring with different facial expressions, hair styles, glasses, etc. Each of these faces were manually annotated before training. Their resultant system gave rise to a novel application called *Photobook*, which could sort a set of images on their similarity to a user selected image. Pentland, Moghaddam and Starner examine two methods of approaching the eigenspace problem, from a view-based and from a parametric view. The parametric view is simplistic in that eigenvectors are calculated over the entire face set for recognition. The view based approach first classifies the different faces by size and orientation, i.e. different views, before recognition.

In their different testing scenarios the view-based method only slightly out-performed the parametric based recognition. The test data included 189 images of 21 people in 9 different views. View-based recognition achieved 90% accuracy while parametric based recognition achieved 88% accuracy. From this, they again identify the importance of registration in eigenspace methods. A feature detection system, to find the eyes, nose and mouth, was also created. Testing on these eigenfeatures, versus eigenfaces, revealed little difference in recognition rates, as they both achieved recognition of 95%. A combination of the two systems saw recognition increase to 98%.

From this work we see the success of the eigenvector approach on a large image data set. Furthermore we can see that sufficient variation exists in faces to recognise them using an eigenface approach.

FIGURE 3.8: A visual representation of different facial characteristics emphasised by the eigenvectors [46].

3.4.2 Eigenobjects

A. Leonardis and H. Bischof present an eigenimage approach to recognition that achieves a high recognition rate [53]. Using eigenvector recognition they emphasise the negative effect that outliers can have on recognition. A novel approach to the selection of the coefficients of the eigenimages is suggested to improve the recognition rates. On the set of 15 test images, no-false positives are reported during recognition. The real-time application of this system is however not addressed.

A method for matching objects using eigenvector techniques is presented by M.J. Black and A.D. Jepson [54]. This method emphasises making the matching technique more robust by applying a least-squares reconstruction to the eigenspace images. The work is tested on objects, in this case a soda can, for which they achieve a 96% recognition rate.
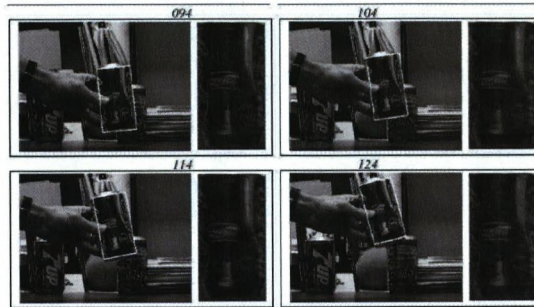
FIGURE 3.9: PCA object recognition by Black and Jepson [54].

3.4.3 Conclusions from Eigenvector Matching

We have analysed various methods of eigenface and eigenobject matching in Sections 3.4.1 and 3.4.2 above. Though differences exist between hand and face recognition, we can draw the following conclusions from these methods:

- An eigenspace approach to recognition, once trained, is a fast method of recognition.

- This fast recognition lends towards the creation of real-time systems.

- This method can be scaled to large image data sets.

- Lighting conditions have little effect on the accuracy of an eigenspace system.

- Registration in terms of image segmentation is necessary for consistent accurate recognition.

- Features within an object, such as the eyes and nose within a face, do little to improve the accuracy of the system.

It can be reasonably concluded that the eigenspace approach can be applied to objects other than faces, and forms the basis for this thesis.

Chapter 4

Eigenvector Theory and Implementation

4.1 Introduction

In this chapter we describe the theory and implementation of eigenvectors and eigenvalues. We give a mathematical representation of eigenvectors and eigenvalues, followed by an example to illustrate the derivation thereof. The technical derivation of eigenvectors is followed by a graphical representation of the algorithmic procedure for eigenvector recognition.

4.2 Mathematical Representation

Given a covariance matrix C, we define a non-zero vector u as an eigenvector[1], if it satisfies the equation, Equation 4.1.

$$Cu = \lambda u \tag{4.1}$$

Where, for some scalar value λ, λ is known as the eigenvalue corresponding to the eigenvector u.

---

[1] *Eigen* is the German word meaning *characteristic of* or *peculiar to*. Hilbert is believed to have first used the term in relation to eigenvectors and eigenvalues in 1904. Hence some authors refer to eigenvalues as characteristic values.

4.3 Example

Consider the example equations 4.2 and 4.3:

$$\begin{pmatrix} 2 & 7 \\ 0 & 3 \end{pmatrix} \times \begin{pmatrix} 2 \\ 3 \end{pmatrix} = \begin{pmatrix} 25 \\ 9 \end{pmatrix} \tag{4.2}$$

$$\begin{pmatrix} 2 & 7 \\ 0 & 3 \end{pmatrix} \times \begin{pmatrix} 7 \\ 2 \end{pmatrix} = \begin{pmatrix} 21 \\ 6 \end{pmatrix} = 3 \times \begin{pmatrix} 7 \\ 2 \end{pmatrix} \tag{4.3}$$

In Equation 4.3 we say that $\begin{pmatrix} 7 \\ 2 \end{pmatrix}$ is an eigenvector of the matrix $\begin{pmatrix} 2 & 7 \\ 0 & 3 \end{pmatrix}$ since the resulting vector is an integer multiple of the original vector. The resulting vector is 3 times the original vector. Equation 4.2 is an example of a non-eigenvector as the resulting vector $\begin{pmatrix} 25 \\ 9 \end{pmatrix}$ is not an integer multiple of the original vector, $\begin{pmatrix} 2 \\ 3 \end{pmatrix}$.

This example also introduces the eigenvalue. There is no eigenvalue in Equation 4.2 as there are no eigenvectors either. In Equation 4.3 the integer 3 is the eigenvalue corresponding to the eigenvector $\begin{pmatrix} 7 \\ 2 \end{pmatrix}$. Therefore demonstrating that eigenvectors and eigenvalues always occur in pairs [55].

As we now know what eigenvectors and eigenvalues are, we move on to more complex methods of determining them. It is only relatively simple to calculate the eigenvectors and eigenvalues on a matrix smaller than 3×3 [55], after which, programming methods are needed.

4.4 Background

To explain the concept of eigenvector and eigenvalue derivation we must first cover the prerequisite knowledge. We present a brief explanation of statistical variance and covariance.

Variance

Variance measures how data is spread within a one dimensional set. The formula for variance is shown in Equation 4.4

$$var(X) = \frac{\sum_{i=1}^{n}(X_i - \overline{X})^2}{(n-1)} \tag{4.4}$$

Where n is the number of elements within the set, and X is an element within the set. Explicitly, for each element X, subtract the mean $\overline{X}$ from X and square the result. Sum these values and divide by $(n-1)$ to determine the variance.

Covariance

Covariance extends the use of variance to two dimensions. The formula for covariance can be seen in Equation 4.5

$$cov(X, Y) = \frac{1}{(n-1)} \sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y}) \tag{4.5}$$

By expanding the squared term in Equation 4.4, we note that the variance appears similar to the covariance:

$$cov(X, X) = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(X_i - \overline{X})}{(n-1)}$$

In utilising the covariance values, we can see how much data differs from the mean in relation to each other.

4.4.1 Covariance Matrix

Situations exist where a data set has more than 2 dimensions. These situations require the use of a matrix to calculate covariance. A covariance matrix for a data set of n dimensions is defined as follows:

$$C^{n \times n} = (c_{i,j}, c_{i,j} = cov(Dim_i, Dim_j))$$

As such, a 3×3 covariance matrix with dimensions x, y and z is represented as follows:

$$C = \begin{pmatrix} cov(x,x) & cov(x,y) & cov(x,z) \\ cov(y,x) & cov(y,y) & cov(y,z) \\ cov(z,x) & cov(z,y) & cov(z,z) \end{pmatrix}$$

4.5 Computing Eigenvectors and Eigenvalues

The aim of eigen decomposition is to project a data set onto a lower dimensional space with minimal data loss. This is achieved by choosing the best eigenvectors to describe the data set. The best eigenvector corresponds to the largest eigenvalue. We demonstrate the calculation of eigenvectors in training and recognition in the sections to follow.

4.5.1 Training

Training Theory

First, let the hand image x_i be an array of size $m \times n$. Let the set $\Gamma = \{x_1, x_2, ..., x_k\}$, where k is the number of images in Γ. Re-order the elements of the set Γ to be the matrix Γ, such that the images are row elements of length mn and the matrix contains k number of rows. Therefore each row in Γ is an image.

$$\Gamma = \begin{pmatrix} x_{1_1} & x_{1_2} & \cdots & x_{1_{mn}} \\ x_{2_1} & x_{2_2} & \cdots & x_{2_{mn}} \\ \vdots & \vdots & \ddots & \vdots \\ x_{k_1} & x_{k_2} & \cdots & x_{k_{mn}} \end{pmatrix}$$

Define the average image $\bar{x}$ over all the images in the data set. The average image is computed by the sum of all images in the set Γ, divided by the number of images in Γ, k:

$$\overline{x} = \frac{1}{k} \sum_{i=1}^{k} x_i \tag{4.6}$$

Thus, each pixel in the average image contains a value representative of an average over all images in Γ. *Center* Γ by subtracting the calculated mean of the image set, $\overline{x}$, from every element in the set:

$$\Phi_i = x_i - \overline{x} \tag{4.7}$$

Kirby and Sirovich term the subtracted image Φ as the "caricature" [52]. Every caricature Φ_i is combined to form the matrix A such that $A = [\Phi_1, \Phi_2, ..., \Phi_K]$. A matrix of size $mn \times k$. The covariance matrix C is given by:

$$C = \frac{1}{k-1} \sum_{i=1}^{k} \Phi_i \Phi_i^T = AA^T \tag{4.8}$$

Finally, utilising Equation 4.9, calculate the eigenvectors and eigenvalues[2] of C:

$$Cu = \lambda u \tag{4.9}$$

The eigenvalues are ordered such that $C : \lambda_1 > \lambda_2 > ... > \lambda_N$. The eigenvectors also ordered, $C : u_1, u_2, ..., u_K$, in relation to their corresponding eigenvalues. To reduce the dimensionality of the data set the eigenvectors corresponding to the largest eigenvalues are selected.

For each training image x the projection onto the eigenspace is given by Equation 4.10.

$$x - \overline{x} = \sum_{i=1}^{q} \alpha_i u_i \tag{4.10}$$

with α_i as the principal component. Therefore, if $U = [u_1, ..., u_q]$ and $\alpha = [\alpha_1, ..., \alpha_q]^T$ then $\alpha = U^T(x - \overline{x})$. Each principle component will define an image class.

---

[2]An alternative method of calculating the eigenvectors and eigenvalues is to calculate the Singular Value Decomposition (SVD) of C.

Training Methodology

To train an eigenvector system the following is needed:

- A training set of sample hand images.

- From this set of hand shapes, calculate the eigenvectors according to the theoretical derivation described above.

- From the calculated eigenvectors, store those corresponding to the largest eigenvalues. These stored eigenvectors are known as the eigenspace.

- By taking a projection of each hand onto the eigenspace, determine the principal components which classify each hand.

Training Implementation

We have implemented our system based on functions from the OpenCV imaging library [56]. OpenCV provides useful tools for fast image processing. The OpenCV framework was developed by IBM and contains functions for many image processing techniques. Among these are the image processing functions for various methods of image manipulation such as edge detection, colour modification and resizing. Importantly, image data processing methods are also included such as those for matrix conversion, singular value decomposition and the calculation of a dot product. Our implementation makes use of the OpenCV eigenvalue constructs to perform eigen decomposition on our data set.

During this phase the system processes images to form the basis for testing new images. All images are ordered sequentially to create a complete matrix of images. We now seek to lower the dimensionality of this large set of data. The OpenCV function *cvCalcEigenObjects* reduces this image matrix to a smaller one containing the eigenvectors of the data set. These eigenvectors contain all the variance in the set and will later be used to classify new images.

Utilising our calculated eigenvectors and eigenvalues, we use the *cvEigenDecomposite* function to project each training image onto the eigenspace. This process evaluates a principal component by which we class our training images.

An illustration of the training process can be found in Figure 4.1. The OpenCV function definition for *cvCalcEigenObjects* can be found in Appendix A.

4.5.2 Recognition

Recognition Theory

When the system encounters a new image y, this image must be projected onto the eigenspace for classification. Classification is performed by first creating a caricature of y. This is done in the same way as training the system, by subtracting the average image from the input image. The resultant image is then manipulated in combination with the saved eigenvectors, as in Equation 4.11.

$$\alpha_y = \sum_{i=1}^{q} (y - \overline{x}) u_i \tag{4.11}$$

Where α_y is the principal component attributed to the image y. These values are the differentiating factors in classifying each new image into the separate hand classes. To determine the class to which a particular hand shape belongs, we look to minimise the Euclidean distance between test and training principal components. The Euclidean distance is determined as follows:

$$\epsilon = || (u_x - u_y) ||^2 \tag{4.12}$$

In this manner a hand class is identified during the recognition process. Therefore, to see if a hand to be tested belongs to a particular class, perform the following steps:

- Find the caricature of the new hand, as before, by Equation 4.7.

- Find the principal components of the new hand by Equation 4.11.

- Find the shortest Euclidean distance between the test and training images by Equation 4.12.
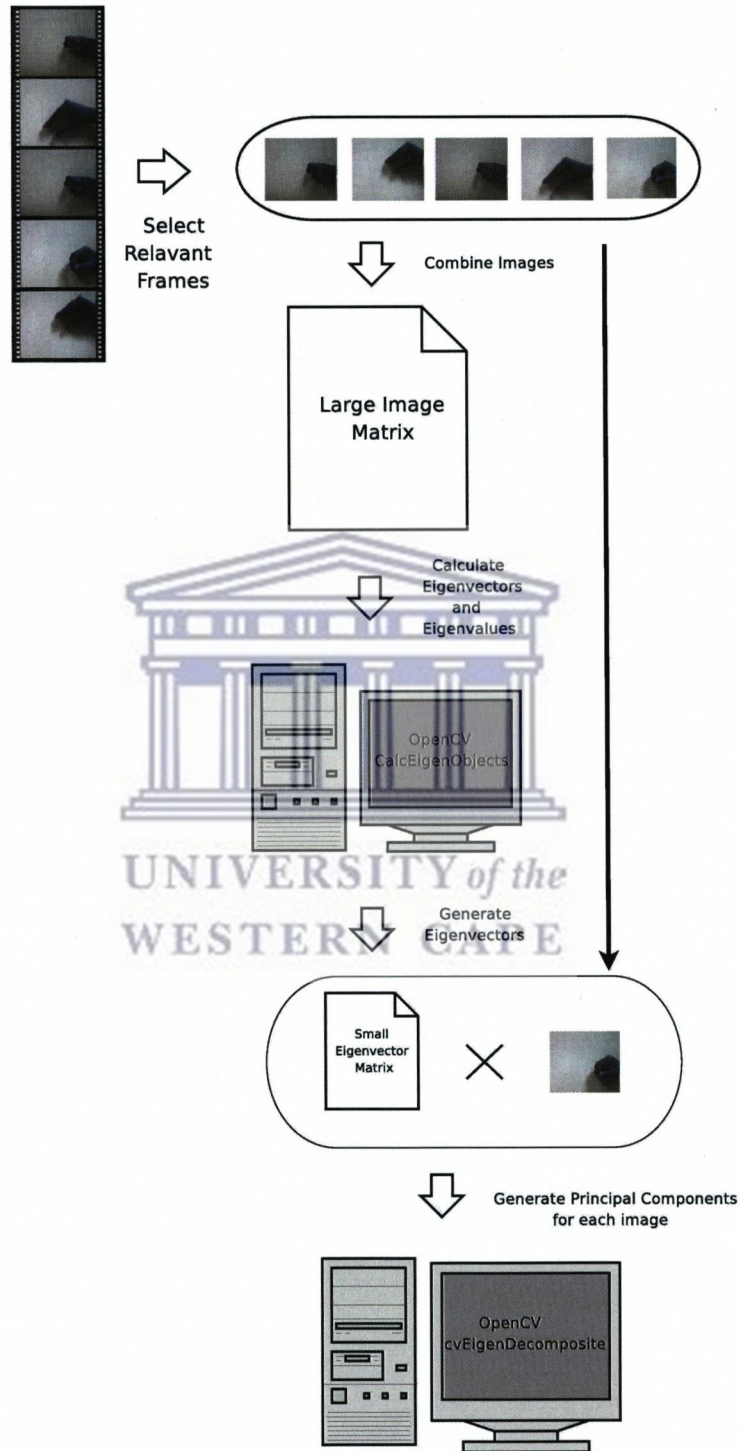
FIGURE 4.1: A diagram representing the flow of data when training the system.

Recognition Methodology

To recognise hand classes in an eigenvector system, proceed as follows:

- Create the set of eigenvectors from the original set of hand images.

- Choose eigenvectors corresponding to the largest eigenvalues.

- Determine the principal components using these eigenvectors to characterise the hand class.

- Determine the threshold which assigns a given image into the relevant hand category.

- Determine Φ for each new image to be recognised.

- Project Φ onto the eigenspace and find the corresponding principal components.

- Determine the Euclidean distance from the training to the test principal components to determine the hand class.

Recognition Implementation

Recognition was also implemented using the eigenvector functionality of OpenCV. During this phase the principal components of a test image is needed. The *cvEigenDecomposite* function is used to calculate these components from the test image.

In the work of Sirovich and Kirby, they find the use of the first 40 eigenvectors to be sufficient to approximate a face [51]. We have determined that the first 45 eigenvectors are needed to classify a hand gesture.

It is by the calculation of the Euclidean distance between this value and the saved principal components of the training images that the image class is determined. The image is classified to a certain class if the shortest Euclidean distance is found within the classification threshold.

An illustration of the recognition process can be found in Figure 4.2. The OpenCV function definition for *cvEigenDecomposite* can be found in Appendix A.
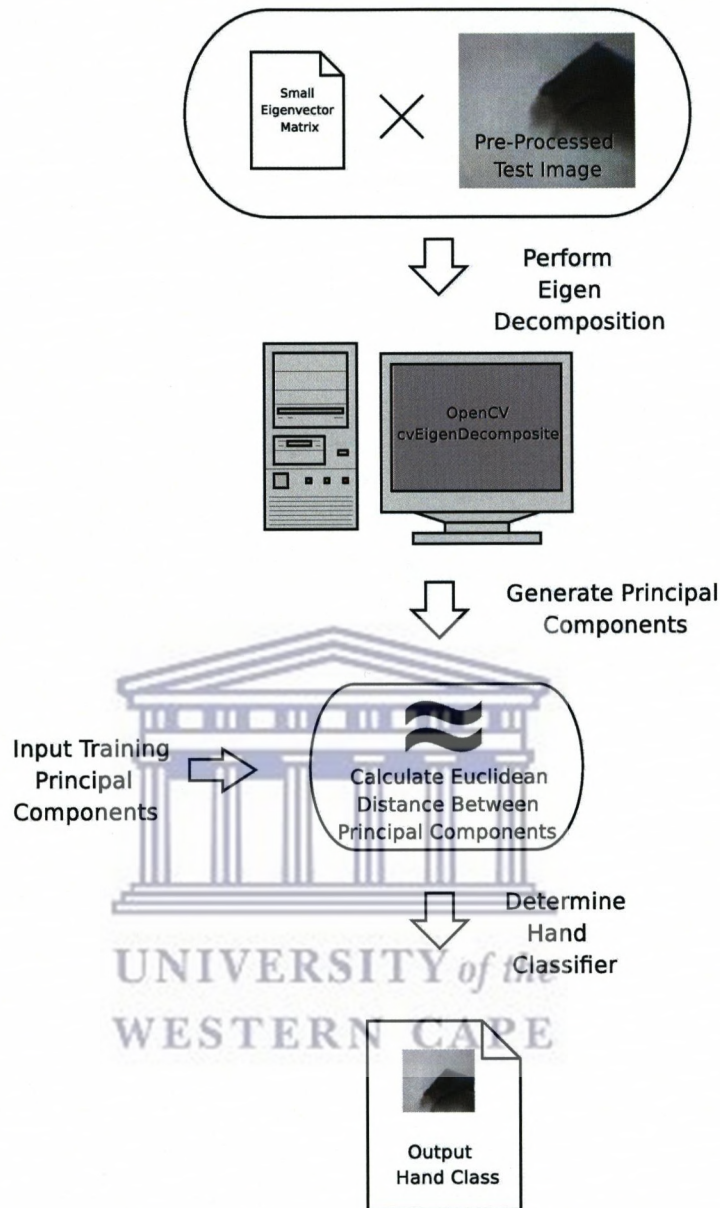
FIGURE 4.2: A diagram representing the flow of data when recognising a hand.

4.5.3 Summary

In this chapter we have given an outline of the theory of eigenvectors. We have also demonstrated the use thereof in this SASL classifier system.

Chapter 5

Image Registration

5.1 Introduction

Image registration is the acquisition of the area of interest for recognition [57], in this case, the hand. More explicitly, registration is the process of aligning two or more images by some translation of the image. We align the hands during training and testing to allow for accurate recognition. In this chapter we address the challenges faced in preprocessing an image before image recognition.

As videos are no more than a series of frames, we discuss image registration on a representative image, which is a frame from the video.

5.2 Removal of Outliers

A large problem facing the accuracy of eigenvector recognition systems is encountered in registration [57]. Outliers can have drastic negative effects on the recognition of an eigenvector system [53] [55]. Outliers are also known as noise. This negative effect occurs as unimportant noise data is recognised as genuine image variation. For this reason all areas around, and not relating to the hand, must be eliminated. Using prior knowledge of the images captured, we propose the following methods for the removal of these outliers. Three transformations are performed on the image to reduce outlier interference. These transformations are grayscale conversion, contour detection and hand extraction. This chapter outlines these transformations.
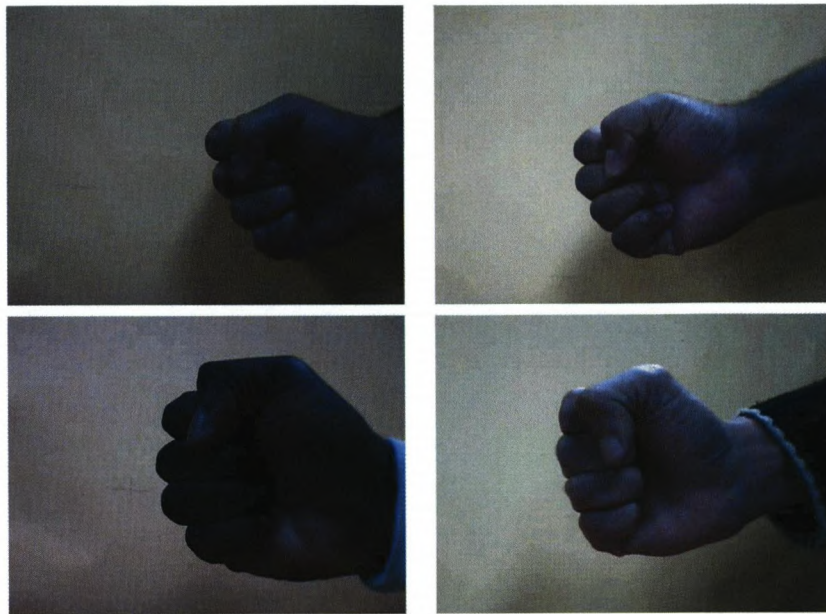
41

FIGURE 5.1: A figure displaying different skin tones from the set of hand images. Also note the differences in hand size in the images.

5.2.1 Grayscale Conversion

Skin tone varies greatly in the images collected in our test data. These differences can be seen in the sample images in Figure 5.1. An overview of the test set, training set and data acquisition methods can be found in Chapter 6.

To compensate for this visible variation in skin colour, a grayscale image is used instead of the original colour image. This provides less variation within the set of images. Eigenvector matching encodes any differences in the set of images into a matching component. Reducing the skin variation in the set allows for skin tone to be more easily discarded as a discriminating factor between hands.

OpenCV was used to convert the image to grayscale. The typical method to convert an image to grayscale is outlined below.

To convert an image to grayscale:

- The image is converted to red, green and blue(RGB) colour components.

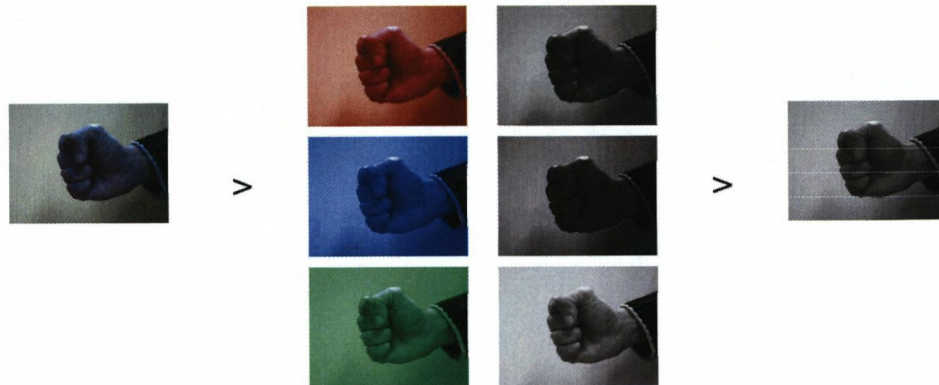- Compute 30% of the red value, 59% of the green value and 11% of the blue value.

FIGURE 5.2: A graphical representation of grayscale conversion.

- Sum these components to produce the grayscale image.

We can see the process and result of grayscale conversion on an image in Figure 5.2.

These images still contain noise, that is, sleeves, wrists, shadows and image background, which contribute negatively to recognition. These outliers are to be dealt with in the following sections.

5.2.2 Contour Detection

Our goal is to remove the hand from the image. To do this we first broadly search for a region of interest to find the hand. To accomplish the task of removing this region of interest, we extract the contour lines from the image. These contours represent the outline of the image. Using contour lines, the image can be cropped to the size of the hand while a binary image thereof is also created.

Cropping using Image Contours

Contour detection within the OpenCV environment requires either an edge or binary image. As our image contains only the hand with varying degrees of arm and wrist inclusion, we can accomplish this by converting the image to edges. Edge detection is shown in Figure 5.3.

FIGURE 5.3: Canny Edge Detection applied to an image with a threshhold of 30.

FIGURE 5.4: Clockwise from top left, edge thresholds of 10%, 100%, and the chosen threshold 30% are shown after smoothing.

Edges are computed by using the Canny Edge detection method [36]. Canny Edge Detection was chosen as it is well known to generate continuous edges. An edge threshold of 30% was used as higher thresholds lost too much information. Lower thresholds captured increasing amounts of image noise. We can see the effect different thresholds have on the edges detected in Figure 5.4.

Images are further smoothed to remove excess noise. The cvSmooth function is used to smooth the edge image. An example of this smoothing technique can be seen in Figure
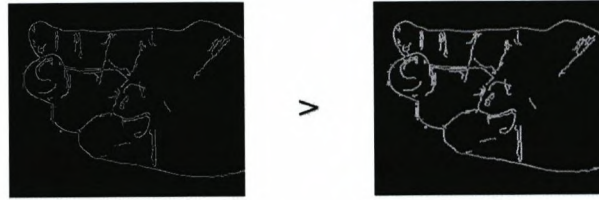
FIGURE 5.5: The left image is the original Canny Edge Detection. The image on the right is smoothed using the cvSmooth method.

5.5, where smoothing ensures continuity in edges. A large continuous edge is of greater interest than smaller, short edges.

A group of the largest of these edges is taken to be the outline of our region of interest. We know this as the *region of interest* as it requires further processing to determine if this region contains the hand.

These edges represent the outline of the hand and wrist. From these edges, selected continuous contours are extracted using OpenCV. Many erroneous edges are found and detected as contours. We determine that only the largest contours are needed to sufficiently estimate the area of interest of the hand. We have chosen to only examine the five largest contours.

As each contour is detected, a summation is made to the region of interest. We can see the detected contours and their corresponding hand data in Figures 5.6 and 5.7. We show the before and after images for the extraction of the *two long thin bent extensions* gesture in Figure 5.8.

As each contour is detected, so the region of interest grows. Within this region we identify the hand. This process removes most background elements from the image, effectively cropping the image around the gesturing region.

Binary Image from Contours

Hands vary in size and shape between people. The hand shape needs to be generalized to recognise many different people. We do this by creating a binary image of the hand.

The same contours detected above are *filled in*, as shown in Figure 5.9. We thus reduce the variation between people performing the same gesture. It is this binary image on which eigen training and recognition will be performed.
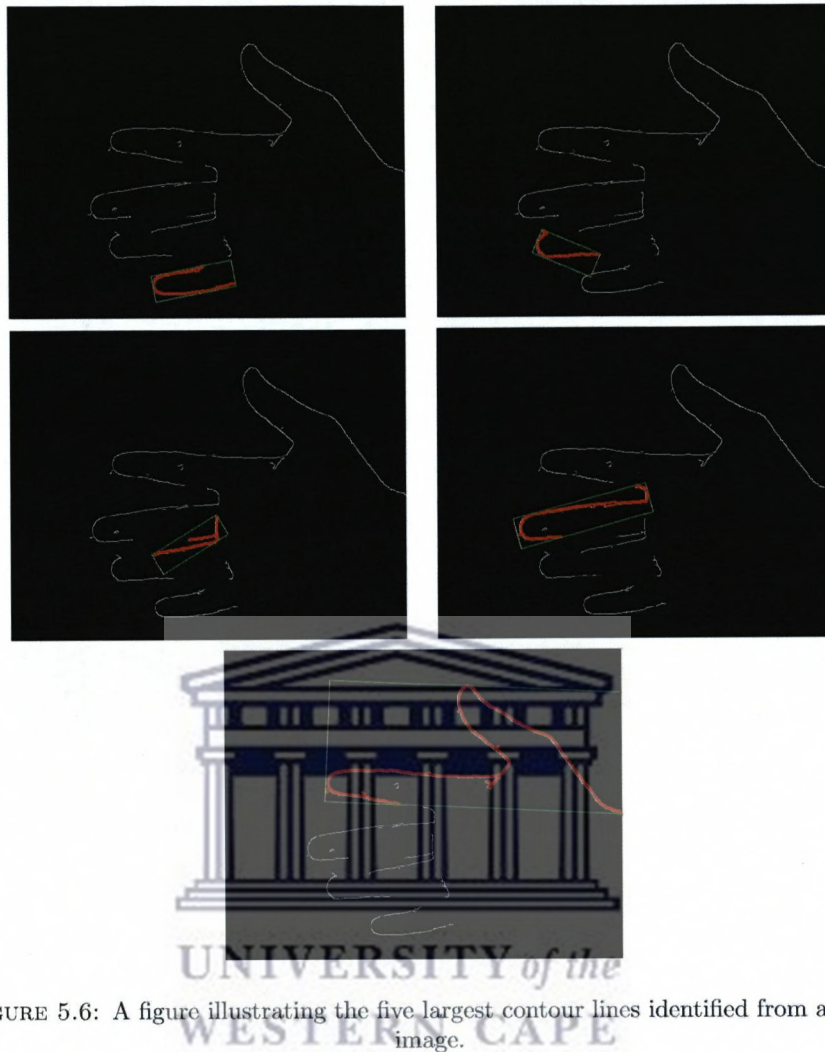
FIGURE 5.6: A figure illustrating the five largest contour lines identified from an edge image.

5.2.3 Hand Extraction

For the purpose of this thesis we have only considered images of the gesturing hand, ignoring the rest of the body. These images, however, are still not perfectly segmented to only display the hand. Image input to the system would be too restrictive to a potential user if only the hand was permitted to appear within the image. As such we see areas where the wrist and sleeves are included in the images.

The presence of the wrist, sleeve or arm presents a problem to recognition, and can reduce the accuracy of the system [53] [55]. A method is therefore needed to remove these problem areas from the images.
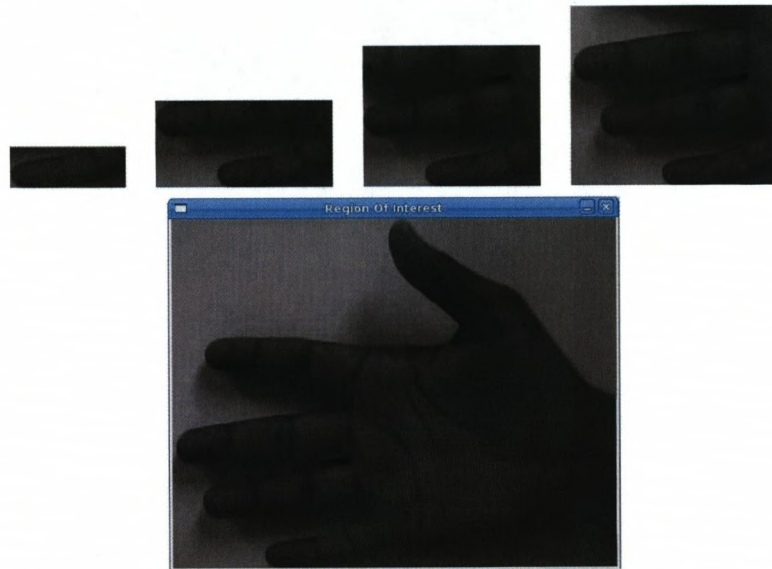
FIGURE 5.7: Images progressively extracted utilising contour information. The final image is used in hand extraction.

FIGURE 5.8: A figure illustrating the before and after images for the extraction of the *two long thin bent extensions* gesture.

We assumed that the image obtained from contour extraction contains the hand. By narrowing and resizing the image we can eliminate areas that are not part of the hand. We narrow the image from the side by five pixels recursively to create a candidate image. Each of these candidate images are compared against the image data set. Eigen recognition is performed on the re-sized region. By finding the Euclidean distance at this point, the hand is then classified.

This wrist removal process is applied to both the training and test sets. During this hand extraction phase images are also corrected for size by resizing to a pre-defined image size, 200x150 pixels. This process results in recognition of only the image region containing the hand, a process essential to eigenvector recognition. Figure 5.10 demonstrates the
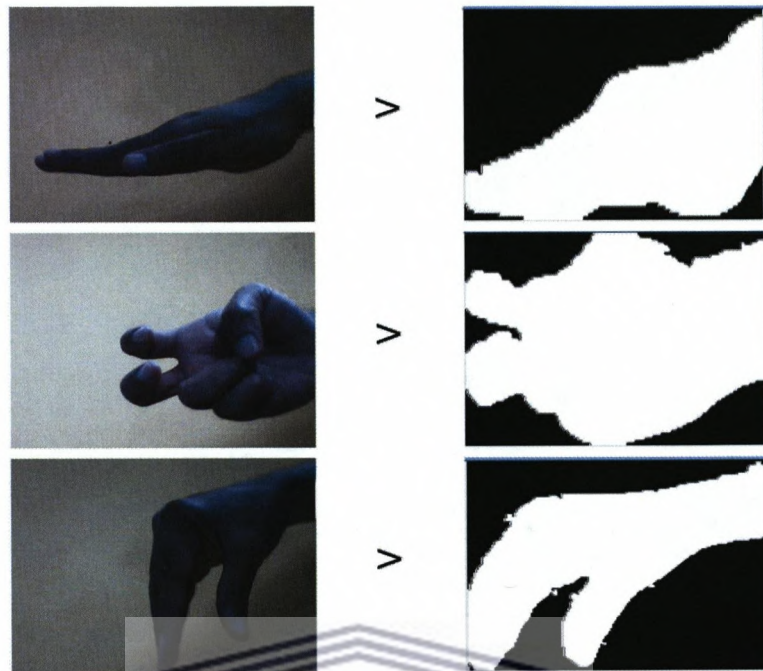
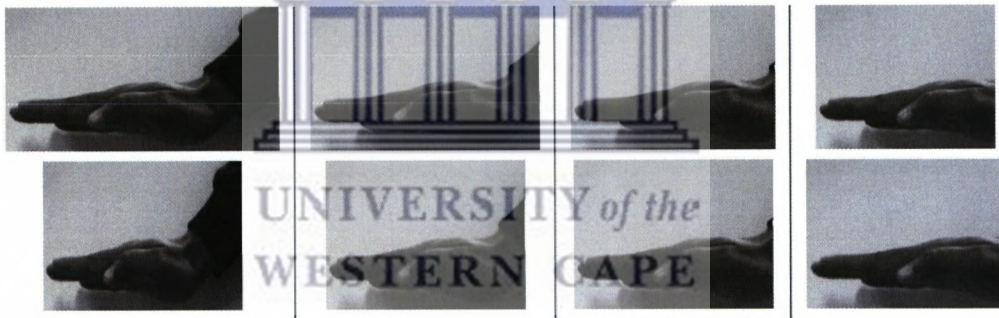FIGURE 5.9: Various hand shapes and their corresponding binary images.

FIGURE 5.10: Starting with the original contour-cropped image, we progressively remove pixels from the side of the image in the top tow. In the bottom row the image is then re-sized to the standard 200x150 pixels from which recognition is performed.

hand extraction method.

5.3   Conclusion

In this chapter we have addressed the problem of image registration for this eigenvector system.

We have addressed the problems of:

- Large skin-tone differences by grayscale conversion.

- Region of interest identification by contour cropping.

- Hand extraction by broad image resizing.

- Erroneous outlier removal by image smoothing.

All these pre-processing methods are necessary to ensure accurate identification of variance in the image set. These steps ensure that only the hand is used in recognition. This produces acceptable hand extraction to use in the recognition phase. It must also be noted that such a real-time system is allowed to operate due the computational simplicity of the eigenvector system. Thus we have reduced the registration errors usually encountered by an eigenvector system.

Chapter 6

Experimental Setup and Testing

6.1 Introduction

In this chapter we describe the setup used in the creation of this system and outline the hardware and software components used. Data acquisition techniques for obtaining test and training data is described. The test results of performing image recognition on the data gathered is then presented in tabular form.

6.2 Experimental Setup

This section describes the hardware and software components used for the creation of the SASL classifier recognition system.

6.2.1 Hardware Components

This work forms part of a larger full-gesture SASL recognition system. The full gesture recognition system is described by Naidoo and Connan in [5]. The eigenvector system is required to perform in real-time, with minimal computational cost to the larger system. The following is used during the experimental testing stage:

1. **CPU**: 1.2Ghz Intel Centrino Duo, a low power CPU used in many laptop computers.

50

2. **RAM**: 2Gb, a more than sufficient amount for our purpose.

3. **Webcam**: Logitech Quickcam Chat, a relatively inexpensive camera with acceptable image quality.

6.2.2 Software Components

The following software is used:

1. **Operating System**: Ubuntu Linux 8.04 is used. Ubuntu is a Linux distribution with regular system updates and readily available support.

2. **Image Capture**: The Gnome camera application Cheese is used for the acquisition of training and test data. Cheese is a free and open-source camera application.

3. **Computer Vision Library**: The OpenCV computer vision library is used.

4. **Programming Language**: The C++ programming language is used. C++ is commonly used for real-time image processing applications.

6.2.3 High-Level Design

The system uses either live or pre-recorded video as input. Dring live usage, the user is required to place his or her hand between the webcam and a plain background. The user is then able to see their hand on the computer monitor in real-time. A diagram of this setup can be seen in Figure 6.1.

Above the video of the user's hand is the name of the classified sign as determined by the system. A demonstration of this can be seen in Figure 6.2. When using pre-recorded video the system does not present the user with a graphical user interface (GUI).

6.2.4 Data Acquisition

Unfortunately no standard hand gesture data set exists for the recognition of hand shapes, as exists for faces with the use of the FERET face data set [58]. Our test and training data is based on the classifiers outlined by the TSLI, described in *Chapter 2*.
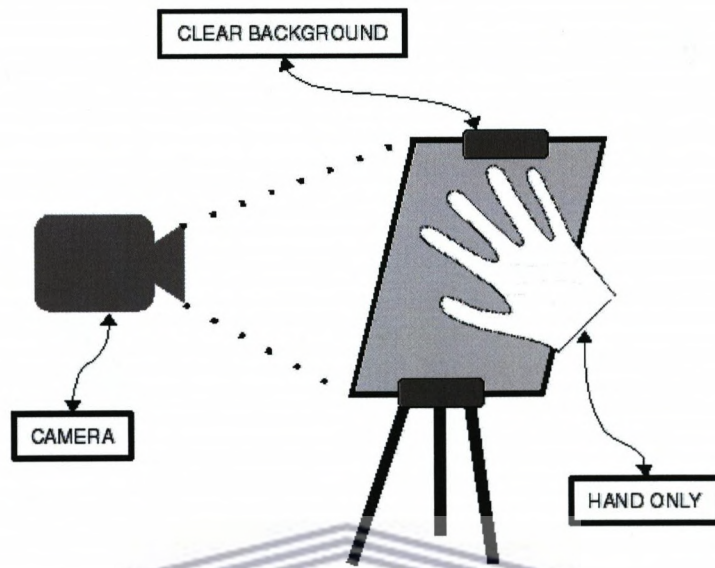
FIGURE 6.1: The user is required to use the system as demonstrated in this diagram.
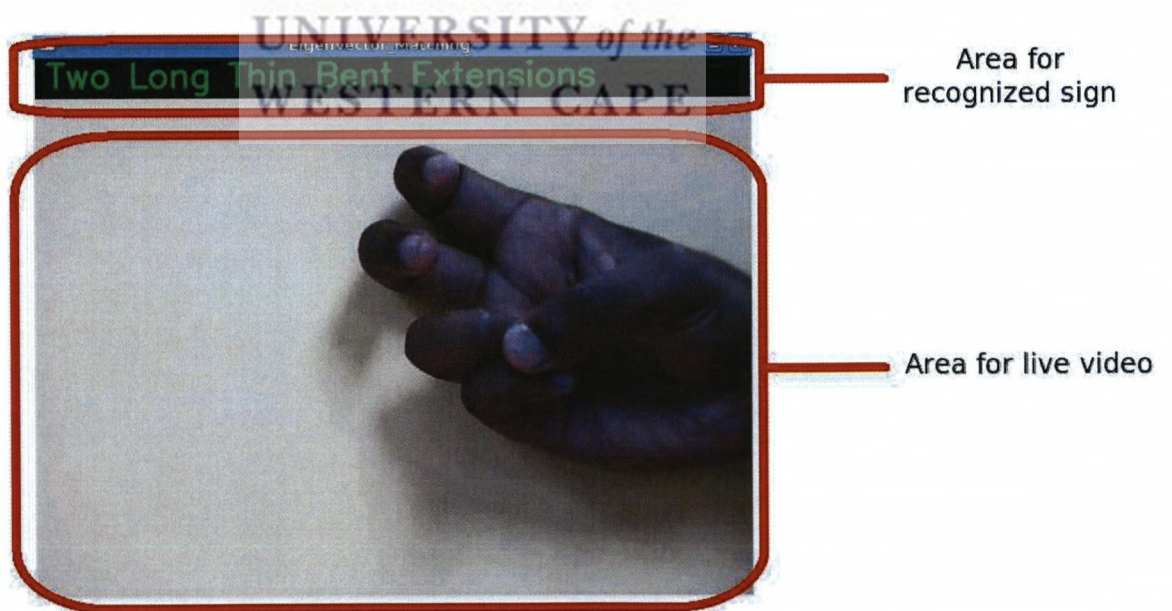

FIGURE 6.2: The live video area is placed below the recognition area in the final system.

Using this classification system does not require the use of natural SASL speakers as the hand shapes are generic in nature. It is still necessary for the set to be diverse in terms of those performing the hand shapes for training and testing. We acquired our data set by using students at the University of the Western Cape. Students were asked to give a moment of their time to perform the nine hand shapes in front of a camera.

Each student was required to observe a set of previously signed examples of the classifiers. Each participant was asked to perform the hand shapes sequentially, and a video was captured thereof. A diverse group of skin hues were sought for the test and training data sets, as this would be more reflective of the South African context.

These hand gestures were captured in a continuous video. Thus, not only were the individual hand gestures captured, but also the intermediate hand orientation between gestures. The videos were then split into individual frames for training and testing. Videos were captured over an arbitrary length of time, that is, however long the students required to perform the classifier satisfactorily. The shortest video is 18 seconds long. The longest video is 49 seconds in length.

Individual frames are used for training and testing the system. The videos were captured at a resolution of 320x240 pixels, at 25 frames per second (fps), in the .ogg video format. Each frame was manually annotated for inclusion, or exclusion, in each of the nine gesture classes. Intermediate gestures were not included, that is, those hand shapes which occurred between the gesturing of classifiers. The number of frames captured range from 183 to 1225 frames. The videos were captured in a typical computer laboratory environment with no control enforced on the surrounding lighting conditions. Confirmation that the hand gesture performed by the participant sufficiently resembled the original gesture was the task of the researcher.

We abbreviate the names of the TSLI SASL gestures as described in Table 6.1. The data set can be seen in Table 6.2.

SASL TSLI Gesture	Abbreviation
Palm/Flat	G1
Two long thin bent extensions	G2
Narrow/Shallow Flat Object	G3
Round/Spherical Object	G4
Flat/Long Smooth Surface	G5
Fist	G6
Flat/Triangular Object	G7
Index Five	G8
Compact Mass with Salient Extension	G9

TABLE 6.1: Abbreviations for the TSLI SASL gestures used within this chapter.

6.2.5 Training and Testing Phases

Training

For training the system, videos of each participant are used. Individual frames are taken from these videos. Training is outlined as follows:

1. The training images are loaded by the system.

2. OpenCV is used to compute the Eigenvectors and Eigenvalues of these images.

3. This eigenvalue and eigenvector data is saved to the hard disk for later recognition use by the system.

Testing

The system can operate on either live or pre-recorded video. *Pre-recorded video testing* is used to determine the overall accuracy of the system.

The software performs the following operations:

1. Eigenvector data generated by the training phase is loaded from the hard disk.

2. Each frame is subjected to image registration, cropping the hand from the image.

3. Eigen decomposition is performed on the cropped hand using the OpenCV eigenvector functionality [56].

4. The matching classifier is determined by a nearest neighbour calculation.

	Video1	Video2	Video3	Video4	Video5	Video6	Video7	Video8	Video9	Video10	Video11	Video12	Total
G1	14	3	11	17	30	25	18	17	8	8	10	14	**175**
G2	6	4	20	15	17	20	17	10	38	11	37	12	**207**
G3	19	5	22	17	27	18	9	17	26	14	17	9	**200**
G4	41	12	41	16	31	16	14	14	22	14	14	9	**244**
G5	45	12	27	15	36	13	14	20	20	16	15	13	**246**
G6	23	13	33	14	27	13	11	18	17	16	20	13	**218**
G7	11	6	32	23	32	9	13	19	17	10	13	13	**198**
G8	12	6	31	11	33	10	12	35	11	8	25	12	**206**
G9	8	13	18	18	24	12	14	17	26	16	32	15	**213**
Total	**179**	**74**	**235**	**146**	**257**	**136**	**122**	**167**	**185**	**113**	**183**	**110**	**1907**

TABLE 6.2: A table representing the data gathered from videos of 12 users. Values indicate the number of frames manually identified containing the relevant TSLI SASL classifier performed by a user in a particular *Video*.

	Excluding Image Registration	Including Image Registration
Mean	49.43ms	109.96ms
Standard Deviation	2.47ms	2.62ms

TABLE 6.3: A table demonstrating the increase in recognition times when image registration is used. The mean and standard deviation are given.

6.3 System Performance

The following properties of the system are investigated:

- Does the system perform in real-time?

- How accurately are hands classified?

- Which gestures are incorrectly classified?

Test cases are presented to determine the performance of the system in relation to each of these questions.

6.3.1 Real-time operation

Different systems use different time-scales to define real-time operations. For example, virtual-human animation requires the movements of an avatar to appear realistic to a human viewer. In this instance, Kennaway defines real-time to be 15 frames per second (fps) [59]. For CCTV (Closed Circuit Television) systems, however, Keval et al determined that operators require at least 8fps to correctly identify persons from video [60].

Our system shows live video to the user. If 8fps is sufficient for security purposes in a CCTV system, then this is an adequate lower-bound for viewing a hand-shape. The upper-bound is the speed at which we record video, which is 25fps.

- Therefore our system must perform between:

    - The **Upper Bound** of 40ms

    - The **Lower Bound** of 125ms

Note: All timing-based tests have been computed multiple times (100 times) and the average of these tests taken, in order to account for CPU task scheduling which may influence timing results.

The mean time to recognise an image is more than doubled when using our image registration technique. Our system is able to operate at approximately 9 fps. This is slower than the upper bound, but faster than the lower bound for real-time recognition.

Training the system would have no effect on real-time performance. This is due to training occurring before the system is tested, i.e. *offline*, with training results stored to the hard disk. The time taken for training, however, does change with the inclusion of more training data, due to images being loaded by the system and the increased number of computations necessitated by the larger image matrix. In Figure 6.3 we present results for the training times on various image-set sizes with comparison to the inclusion or exclusion of image registration. Note: All images used are frames taken from training videos.
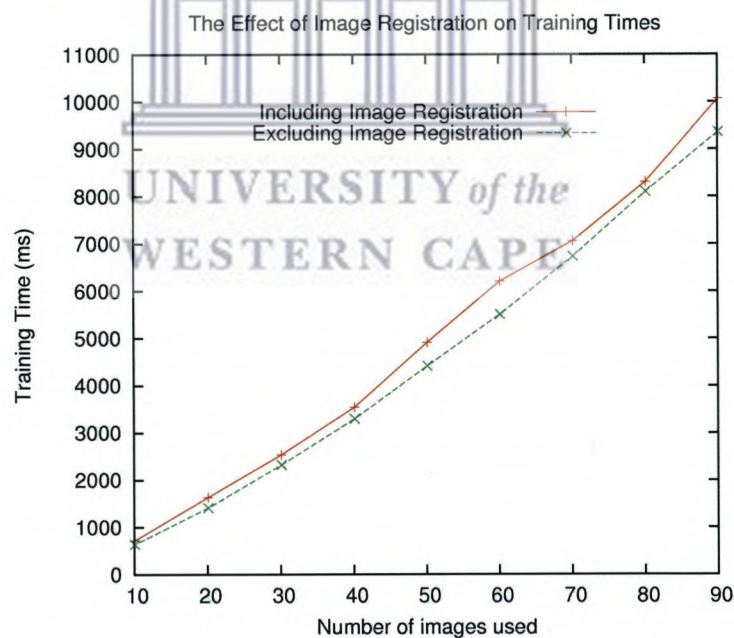

FIGURE 6.3: A graph demonstrating the changes in training time compared to the number of images used. The results with and without image registration are shown.

Figure 6.3 shows a linear increase in training time as the number of training images increases.

| Number of users	12
Number of Signs per User	9
Total Number of Images	1907
Recognition Accuracy	100%

TABLE 6.4: The system was tested on seen data and performed as expected.

6.3.2 Seen User Recognition

Our first evaluation of the system was to test the accuracy when using seen user data. This test is performed to check that our system works as expected. For this test we used videos from seen users split into frames. These frames were manually grouped into the 9 different SASL hand classifiers. Every frame was used for training as well as testing.

Under these circumstances the system performed correctly, predicting all hands accurately, as in Table 6.4.

This was the expected result of an eigenvector approach to the recognition of seen data. In cases where the input image is the same as the training image, the new image perfectly maps onto the eigenspace. When eigen decomposition is performed on the same training and test images the exact same principal component value is returned. In these instances the Euclidean distance between the test image and the training images is zero, a perfect match.
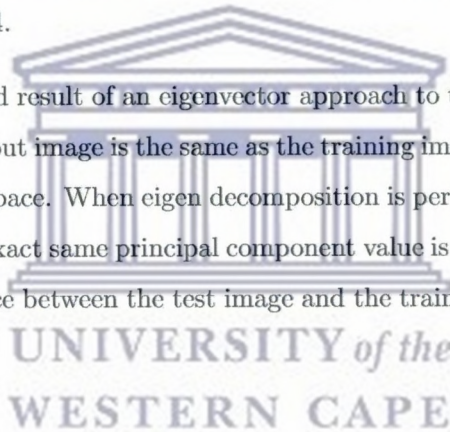
6.3.3 Recognition on Multiple Users

In this experiment we test the recognition of multiple hand shapes from multiple users. For this test, we train our system on half of our users while testing is performed on the remaining users.

6.3.3.1 Recognition of a Single Frame

In this test we separate our data into two randomly disjoint sets for training and testing. We are testing on unseen data and incorporating all 12 users in our training and testing.

Training Data:

- 54 frames

- 6 users.

- 9 signs.

- 1 frame per sign, per user.

Test Data

- 432 frames

  - 6 remaining users not used for training.

  - 9 signs.

  - 8 frames per sign, per user.

We chose users that have a minimum of 8 frames for each sign for testing the system. Users with less than 8 frames per sign is used for training. The results of our system testing on a single frame can be seen in Table 6.5.

In Table 6.6 we present the accuracy of the system when used with and without the use of our image registration step. The number of misclassifications by the system is approximately halved when using our image registration step. Therefore we see the importance of image registration to an eigenvector recognition system.

Table 6.7 represents the average recognition rates when observed by sign, as well as the mean and standard deviation. Table 6.6 shows great variation in the classification accuracy of system, dependent on the gesture performed.

Some SASL classifiers are similar in shape and more likely to be misclassified, such as those in Figure 6.4. Figure 6.5 shows the *palm/flat* classifier, considered easy to perform and demonstrates the consistencies in execution by different participants.

The *two long-bent extensions* shape was considered difficult to perform by participants. This is expected to reflect negatively on classifier recognition. In Figure 6.6 we see the inconsistent interpretation of the *two long-bent extensions* classifier by different participants. Some signers performed the *two long-bent extensions* classifier with the two extended fingers close together, while others spread these same fingers. Signers are also inconsistent in terms of the extent to which fingers were bent. This was not due to a lapse on the part of the signers, but rather an inability to perform the specific hand shape due to the restrictions on the articulation of their hands and fingers.

	G1	G2	G3	G4	G5	G6	G7	G8	G9	Total Accuracy with Registration	% Accuracy with Registration
User1	8/8 (100%)	0/8 (0%)	8/8 (100%)	8/8 (100%)	8/8 (100%)	8/8 (100%)	1/8 (12.5%)	3/8 (37.5%)	7/8 (87.5%)	51/72	70.83
User2	8/8 (100%)	0/8 (0%)	8/8 (100%)	8/8 (100%)	2/8 (25%)	0/8 (0%)	8/8 (100%)	8/8 (100%)	7/8 (87.5%)	49/72	68.06
User3	8/8 (100%)	0/8 (0%)	8/8 (100%)	0/8 (0%)	0/8 (0%)	4/8 (50%)	8/8 (100%)	1/8 (12.5%)	8/8 (100%)	37/72	51.39
User4	8/8 (100%)	0/8 (0%)	8/8 (100%)	8/8 (100%)	2/8 (25%)	1/8 (12.5%)	0/8 (0%)	0/8 (0%)	1/8 (12.5%)	28/72	38.89
User5	8/8 (100%)	8/8 (100%)	1/8 (12.5%)	0/8 (0%)	8/8 (100%)	0/8 (0%)	0/8 (0%)	8/8 (100%)	2/8 (25.5%)	35/72	48.61
User6	8/8 (100%)	0/8 (0%)	7/8 (87.5%)	0/8 (0%)	8/8 (100%)	1/8 (12.5%)	4/8 (50%)	1/8 (12.5%)	8/8 (100%)	45/72	62.5
Total Accuracy with Registration	48/48	8/48	40/48	24/48	28/48	17/48	21/48	21/48	33/48	245/432	
% Accuracy with Registration	100	16.67	83.33	66.67	58.33	29.17	43.75	43.75	68.75		56.71

TABLE 6.5: Table representing the results of the system tested on each frame from multiple unseen signers.

Test Subject	% Accuracy with Registration	% Accuracy without Registration
User1	70.83	16.7
User2	68.06	41.7
User3	51.39	26.4
User4	38.89	0
User5	48.61	48.6
User6	62.50	37.5
Mean	56.71	28.47
Standard Deviation	12.68	17.98

TABLE 6.6: In this table we show test results as recognition is performed on a single frame.

Gesture	% Accuracy with Registration
Palm/Flat	100
Two long thin bent extensions	16.67
Narrow/Shallow Flat Object	83.33
Round/Spherical Object	66.67
Flat/Long Smooth Surface	58.33
Fist	29.17
Flat/Triangular Object	43.75
Index Five	43.75
Compact Mass with Salient Extension	68.75
Mean	56.71
Standard Deviation	26.29

TABLE 6.7: We demonstrate the recognition rates of our system analysed by sign.

66.67% Accuracy   29.17% Accuracy

FIGURE 6.4: The hand gesture representing the *round/spherical object* and *fist* classifiers respectively. Similarities in these gestures were expected to weaken recognition performance. Recognition accuracy is given below the respective gesture.
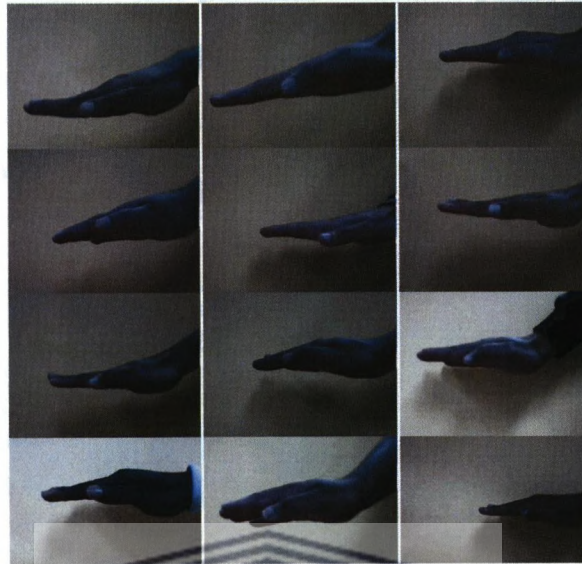
FIGURE 6.5: The *Palm/Flat* classifier as performed by test participants.
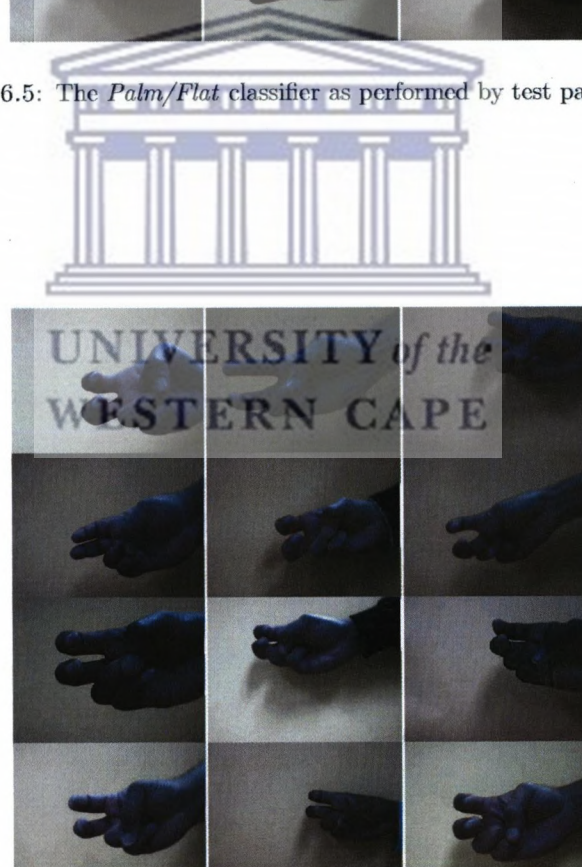
FIGURE 6.6: The classifier representing *Two Long Thin Bent Extensions* was described as difficult to perform by some users and thus showed inconsistency in execution between users. This was expected to impact negatively on recognition performance.

These results (57.71% Accuracy) do not compare favourably to the performance of eigenvectors in facial recognition systems (88 -98% Accuracy).

The aim of this system is to form part of the larger SASL system. In order to achieve this, the hand shape does not need to be classified within each frame, only each time it changes. Therefore we can submit a series of frames to the system, and when a gesture is held over a number of frames, it needs to be recognised only in a single frame in order to be considered successful. We can safely assume that the video input to the system will contain multiple frames of a hand in a given hand shape. Only one of these frames needs to be recognised.

We therefore re-look at the results from this experiment, taking this construct into consideration.

6.3.3.2   Recognition on Multiple Frames

The same test results obtained in Section 6.3.3.1 are used to demonstrate the effectiveness of the system when multiple frames are used. Under the test conditions shown in Table 6.5, the system is required to recognise at least one image in 8 frames to be considered a success. In Table 6.8 we demonstrate this for the 9 signs for each user. This result (74.07) compares much more favourably to to face recognition systems using eigenvectors.

The basic characteristics of a sign language gesture consist of movement and hold positions of the hand [16]. The hold positions are required when dividing the testing video into frames. The hand class is identified from these hold positions, where the hand remains stationary or moves slowly.

Table 6.8 also shows the accuracy of the system for different users when presented with 8 frames for each user. From Table 6.8 we can see that the system performs well when recognition is undertaken on multiple frames. Table 6.9 shows each classifier and the recognition accuracy associated with .

	G1	G2	G3	G4	G5	G6	G7	G8	G9	Total	% Accuracy
User1	✓	✗	✓	✓	✓	✓	✓	✓	✓	8/9	88.89
User2	✓	✗	✓	✓	✓	✗	✓	✓	✓	7/9	77.78
User3	✓	✗	✓	✗	✗	✓	✓	✓	✓	6/9	66.67
User4	✓	✗	✓	✓	✓	✓	✗	✗	✓	6/9	66.67
User5	✓	✓	✓	✗	✓	✗	✗	✓	✓	6/9	66.67
User6	✓	✗	✓	✗	✓	✓	✓	✓	✓	7/9	77.78
Total	6/6	1/6	6/6	3/6	5/6	4/6	4/6	5/6	6/6	40/54	
% Accuracy	100	16.66	100	50	83.33	66.67	66.67	83.33	100		74.07

TABLE 6.8: Successfully recognised hand gestures within 8 frames.
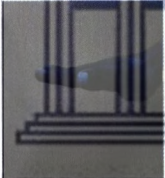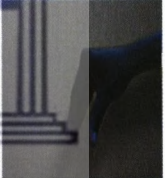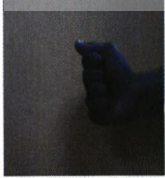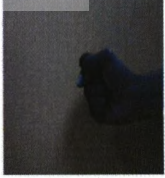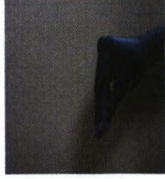
TABLE 6.9: TSLI SASL Classifiers and their respective recognition accuracies

Classification Rate on Unseen Data									
		Projected Gesture								
		G1	G2	G3	G4	G5	G6	G7	G8	G9
Actual Gesture	G1	**48**	0	0	0	0	0	0	0	0
	G2	11	**8**	8	8	5	8	0	0	0
	G3	0	0	**40**	0	0	0	8	0	0
	G4	0	0	0	**32**	0	16	0	0	0
	G5	0	14	0	0	**28**	0	0	6	0
	G6	0	8	0	17	0	**14**	9	0	0
	G7	0	4	0	9	0	7	**28**	0	0
	G8	0	0	9	0	4	0	10	**21**	4
	G9	0	0	0	6	0	0	8	0	**34**

TABLE 6.10: The number of correctly classified and incorrectly classified TSLI SASL gestures are shown in the table above

6.3.4 Cross-Testing

Identified in Table 6.10 are the recognised gestures as well as their incorrect classifications.

Table 6.10 shows that the only gesture on which 100% recognition rate is achieved is G1, the *palm/flat* gesture. The large misclassification rate of gesture G2 is also noted, as expected from the different interpretations of the same sign by different signers.

Gesture G3, the *narrow/shallow flat object* gesture is misclassified to the *flat/triangular object*, G7. These gestures appear similar and this result is understandable. The same can be said for gesture G4, the *round/spherical object*, which is mostly misclassified as G6, the *fist* gesture.

We observe that other incorrect classifications are more random.

6.4 Summary

In this chapter we have discussed the experimental setup of the system, including hardware, software, high-level design and data acquisition.

In testing the system we have studied the real-time performance, seen and unseen user accuracy and the cross testing results on multiple users. The system is found to perform at an adequate real-time frame-rate. Due to the fast recognition time, the system is found to be appropriate for integration into the larger SASL project.

Accuracy on seen data was found to be 100%, as expected of this system. The recognition of the system on unseen data consisting of a single frame from multiple users was found to be 56.71% when using the image registration technique, halving the errors encountered without image registration.

When applying the system to to multiple frames from multiple users, the accuracy of the system improves to 74.07%. The *two long thin bent extensions* gesture is frequently incorrectly classified and has the lowest recognition rate. Therefore recognition can be significantly improved with the removal of this inconsistently recognised gesture.

Chapter 7

Conclusions and Directions for Future Work

7.1 Introduction

Human computer interaction (HCI) and computer vision are advancing rapidly as a consequence of ubiquitous hardware improving. The use of cameras, accelerometers, and gyroscopes have spurred research into more intuitive HCI systems. Every PC user can now have access to interactive technologies other than the ordinary keyboard and mouse. PC's and mobile phones are commonly equipped with cameras. The average technology user now expects intuitive interaction with their computing devices. Therefore exciting research is happening in the field of computer vision. The average person is now *expecting* their PC to conform to their HCI needs and moving away from the traditional methods of computer interaction.

The expectations of people are the same for linguistic translation. It does not feel natural to use a digital dictionary where words need to be typed-in and then read. People expect to speak to their PC or mobile phone and have their voice automatically translated. Frequent international travel leads to greater numbers of people constantly needing simpler methods of interaction through the linguistic divide.

67

7.2   Conclusions

The aim of this work is to apply the eigenvector technique to hand detection, thereby making gesture recognition better for SASL translation. To do this we needed to perform image registration as well as apply eigenvector recognition to the hands.

Image registration was used to remove the hand from a uniform background. We perform image registration to generalize the hand shape across different individuals. A uniform background was used, as ultimately the hand will be extracted from a whole-body video by the larger SASL system. Background noise is to be dealt with at the time of integration. Image registration removes the hand from the image by first creating a grayscale and edge-detected image. Contour detection determines the outline of the hand in the image and this information is used to crop the hand from the larger image. Once the hand is cropped this resultant binary image is filled and ready for the recognition phase.

To recognise the hands we need to apply eigenvector recognition to the binary images. On single frames our performance was 57%. This does not compare favourably to the recognition rates found with eigenvector-based face recognition systems. Testing on multiple frames improves the system performance (74%) and is closer to the recognition rates achieved by face recognition systems. Though the TSLI SASL classifier set is small, the classifiers represent real-world gestures. The system requires low computational resources and is fast when recognizing gestures. Therefore the larger SASL system, which has greater processing needs, will not be affected when hand recognition is included.

Referring to the research hypothesis, we have shown that SASL hand shapes can be divided into specific classes, namely those defined by the TSLI. These hand shapes can be recognised by using eigenvectors in real-time. The fast processing time demonstrates that the system can also be incorporated into the larger SASL project because of the low computational resources needed for recognition.

Referring to our research question, we have found that eigenvectors can be applied to hand shape recognition. Recognition rates of 57% were found when applied to single frames and 74% when applied to multiple frames. This includes bad training classifiers, such as the *two long thin bent extensions*, without which the system accuracy would increase. Therefore eigenvectors can be used in a sign language system such as the SASL sign language recognition system.

7.3 Directions for Future Work

- **Registration:**

  - Image registration is a difficult task and can be seen as a separate area of research, especially in the area of hand registration. Background noise needs to be addressed in image registration. Certain outliers can weaken image registration and therefore more can be done to build on the present image registration technique.

- **Testing Set Size:**

  - Future work can be extended by increasing the size of the test set. More users can be obtained to perform the gestures and to create a large-scale implementation of the system.

- **Real World Video:**

  - The system uses only video captured within laboratory conditions. Extending to real-world video is also an area for continued research.

- **Native SASL Signers:**

  - The test and training data was compiled on non-native SASL signers. The performance of the classifiers by native signers can add to future research, highlighting the similarities or differences between the different groups.

- **Other Applications:**

  - This thesis has shown the application of eigenvectors to SASL recognition. The application of the same technique to other recognition tasks is an interesting direction for future research.

7.4 Final Comments

This thesis has provided the author with insight into the world of the deaf that was hereunto unknown. I have also gained knowledge of the challenges facing the computer vision community and the potential solutions proposed. It is hoped that this active area of research will continue to grow and be of benefit to more marginalised communities

around the world, as well as furthering the aims of the SASL group at the University of the Western Cape.

Bibliography

[1] J. Holt. Stanford Achievement Test-8th edition: Reading Comprehension Subgroup Results. *American Annals of the Deaf*, 2(138):172–175, 1993.

[2] M. Strong and P.M. Prinz. A study of the relationship between american sign language and english literacy. *Journal of Deaf Studies and Deaf Education*, 2(1): 37–46, 1997.

[3] C.A. Perfetti and R. Sandak. Reading optimally builds on spoken language: Implications for deaf readers. *Journal of Deaf Studies and Deaf Education*, 5(1):32–50, 2000.

[4] A. Lotriet. *Sign Language Interpreting in South Africa: Meeting the Challenges.* John Benjamins Publishing, Vancouver, 1998.

[5] N. Naidoo and J. Connan. Gesture recognition using feature vectors. *Southern Africa Telecommunication Networks and Applications Conference 2009*, pages 333–337, September 2009.

[6] J. R. Whitehill. *Automatic Real-Time Facial Expression Recognition For Signed Language Translation.* Master's thesis, University of the Western Cape, 2006.

[7] W. Branford, editor. *The South African Pocket Oxford Dictionary of Current English.* Oxford University Press, 1994. ISBN 0195707605.

[8] S. Tulloch, editor. *Reader's Digest Oxford Complete Wordfinder.* Claredon Press, 1996. ISBN 0276421019.

[9] A.Y. Aikhenvald. Classifiers: A typology of noun categorization devices. *Journal of Linguistics*, 38(01):137–172, 2002. doi: 10.1017/S0022226702211378.

[10] I. Zwitserlood. *Classifying Hand Configurations in Nederlandse Gebarentaal.* PhD thesis, Utrecht, The Netherlands, 2003. Promotor : Prof. Dr. W. Zonneveld Co-promotor: Prof.Dr. A.E. Baker.

[11] A. Munnik. *Learn Xhosa with Anne Munnik.* Shutter and Shooter, 1996. ISBN 0796009813.

[12] M.P.O. Burgers. *Teach Yourself Books Afrikaans.* David McKay Company, 1971. ISBN 034005770.

[13] N. Frishberg. Arbitrariness and iconicity: Historical change in american sign language. *Language*, 51(3):696–719, 1975.

[14] K. Emmorey, editor. Psychology Press, USA. ISBN 9780805842692.

[15] A. Schembri and T. Johnston. Variable 'subject presence in australian sign language narratives. In Timothy Jowan Curnow, editor, *Selected Papers from the 2007 Conference of the Australian Linguistic Society*, 2007. ISBN 9780980281521.

[16] S.K. Liddell and R.E. Johnson. American sign language: The phonological base. *Sign Language Studies*, 64:195–277, 1989.

[17] W.C. Stokoe Jr. Sign language structure: An outline of the visual communication systems of the american deaf. *Journal of Deaf Studies and Deaf Education*, 10(1): 3–37, 2005. doi: 10.1093/deafed/eni001.

[18] J. Kegl and R. Wilbur. When does structure stop and style begin? Syntax, morphology and phonology vs. stylistic variation in American Sign Language. In *Papers from CLS 12*, pages 376–396. Chicago Linguistic Society, Chicago, 1976.

[19] T. Supalla. The classifier system in American Sign Language. *Noun Classification and Categorization*, pages 181–213, 1986.

[20] B. Schick. Classifier predicates in american sign language. *International Journal of Sign Linguistica*, 1:15–40, 1990.

[21] M. Aronoff, I. Meir, C. Padden, and W. Sandler. *Classifier Constructions and Morphology in Two Sign Languages.* Psychology Press, USA, 2003. ISBN 9780805842692.

[22] D. Cogill-Koez. A model of signed language classifier predicates as templated visual representation. *Sign Language and Linguistics*, 3:209 236(28), 1 August 2000.

[23] Thibologa Sign Language Institution. *South African Deaf People and their language.* Thibologa Sign Language Institution, 2007. ISBN 978798855709.

[24] S.K. Liddell and M. Metzger. Gesture in sign language discourse. *Journal of Pragmatics*, 30:657 697, 1998.

[25] A.D. Wilson and A.F. Bobick. Parametric Hidden Markov Models for gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21 (9):884 900, 1999. ISSN 0162-8828. doi: http://doi.ieeecomputersociety.org/10. 1109/34.790429.

[26] S. Mitra and T. Acharya. Gesture recognition: A survey. *SMC-C*, 37(3):311 324, May 2007.

[27] X.D. Huang, Y. Ariki, and M.A. Jack, editors. *Hidden Markov Models for Speech Recognition.* Edinburgh University Press, Edinburgh, 1990.

[28] A.P. Pentland, J. Weaver, and T.E. Starner. Real-time american sign language recognition using desk and wearable computer based video. In *Vismod*, pages 1371 1375, 1998.

[29] T. Starner and A. Pentland. Visual recognition of american sign language using hidden markov models. In *In International Workshop on Automatic Face and Gesture Recognition*, pages 189 194, 1995.

[30] S. Marcel, O.J. Bernier, J.E. Viallet, and D. Collobert. Hand gesture recognition using input-output hidden markov models. In *AFGR00*, pages 456 461, 2000.

[31] J. Alon, V. Athitsos, Q. Yuan, and S. Sclaroff. A unified framework for gesture recognition and spatiotemporal gesture segmentation. *PAMI*, 31(9):1685 1699, September 2009.

[32] Y. Wu and T.S. Huang. Vision-based gesture recognition: A review. In *GW99*, pages 103 115, 1999.

[33] J. Triesch and C. von der Malsburg. A system for person-independent hand posture recognition against complex backgrounds. *PAMI*, 23(12):1449 1453, December 2001.

[34] M. Potamias and V. Athitsos. Nearest neighbor search methods for handshape recognition. In *PETRA '08: Proceedings of the 1st International Conference on Pervasive Technologies Related to Assistive Environments*, pages 1–8, New York, NY, USA, 2008. ACM. ISBN 9781605580678.

[35] H.G. Barrow, J.M. Tenenbaum, R.C. Bolles, and H.C. Wolf. Parametric correspondence and chamfer matching: Two new techniques for image matching. In *IJCAI77*, pages 659–663, 1977.

[36] J. Canny. A computational approach to edge detection. *PAMI*, 8(6):679–698, November 1986.

[37] K. Tan, R. Feris, M. Turk, R. Raskar, and G. Ohashi. Exploiting depth discontinuities for vision-based fingerspelling recognition. In *In IEEE Workshop on Real-time Vision for Human-Computer Interaction (in conjunction with CVPR04*, pages 155–155, 2004.

[38] K.H. Tan and R. Raskar. Non-photorealistic camera: Depth edge detection and stylized rendering using multi-flash imaging. In *ACM transactions on graphics*, pages 676–685, 2004.

[39] G. Yahav, G.J. Iddan, and D. Mandelboum. 3d imaging camera for gaming applications. In *International Conference on Consumer Electronics, 2007. ICCE 2007. Digest of Technical Papers.*, pages 1–2, 2007.

[40] K. Fujimura and X. Liu. Sign recognition using depth image streams. In *FGR06*, pages 381–386, 2006.

[41] T. Mandal and Q.M. Jonathan Wu. Face recognition using curvelet based pca. In *ICPR*, pages 1–4. IEEE, 2008. ISBN 9781424421756.

[42] I. Atalay. *Face Recognition Using Eigenfaces*. Master's thesis, Istanbul Technical University Institute of Science and Technology, 1996.

[43] N. Muller. *Image Recognition Using the Eigenpicture Technique (with Specific Applications in Face Recognition and Optical Character Recognition)*. Master's thesis, University of Cape Town, 1996.

[44] N. Muller. *Facial Recognition, Eigenfaces and Synthetic Discriminant Functions.* PhD thesis, University of Stellenbosch, Stellenbosch, South Africa, November 2000. Promotor: Prof. B. Herbst.

[45] R. Cendrillon and B. Lovell. Real-time face recognition using eigenfaces. In *VCIP*, pages 269 276, 2000.

[46] A. Pentland, B. Moghaddam, and T. Starner. View-based and modular eigenspaces for face recognition. pages 84 91, 2009.

[47] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71 86, 1991.

[48] H. Bischof and A. Leonardis. Robust recognition of scaled eigenimages through a hierarchical approach. In *CVPR98*, pages 664 670, 1998.

[49] B.M. Herbst and N. Muller. Building a representative training set based on eigenimages. In *ICPR98*, pages Vol II: 1846 1848, 1998.

[50] V. Segers. Real-time gesture recognition using eigenvectors. *Southern Africa Telecommunication Networks and Applications Conference 2009*, pages 363 366, September 2009.

[51] L. Sirovich and M. Kirby. Low dimensional procedure for the characterization of human faces. *Journal of the Optical Society of America*, 4(3):519 524, 1987.

[52] M. Kirby and L. Sirovich. Application of the Karhunen-Loeve procedure for the characterization of human faces. *PAMI*, 12(1):103 108, January 1990.

[53] A. Leonardis and H. Bischof. Dealing with occlusions in the eigenspace approach. In *CVPR 96*, pages 453 458, 1996.

[54] M.J. Black and A.D. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. In *International Journal of Computer Vision*, pages 329 342, 1996.

[55] L.I. Smith. A tutorial on principal component analysis. 2002. URL http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf. (accessed December 2009).

[56] R. Hewitt. Seeing with OpenCV, part 5: Implementing eigenface. *SERVO Magazine*, May 2007.

[57] S. Periaswamy and H. Farid. Elastic registration in the presence of intensity variations. *MedImg*, 22(7):865–874, July 2003.

[58] P. Jonathon Phillips, Hyeonjoon Moon, Syed A. Rizvi, and Patrick J. Rauss. The feret evaluation methodology for face-recognition algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:1090–1104, 2000.

[59] R. Kennaway. Synthetic animation of deaf signing gestures. In *GW '01: Revised Papers from the International Gesture Workshop on Gesture and Sign Languages in Human-Computer Interaction*, pages 146–157, London, UK, 2002. Springer-Verlag. ISBN 3540436782.

[60] H. Keval and A. M. Sasse. To catch a thief — you need at least 8 frames per second: The impact of frame rates on user performance in a CCTV detection task. In *MM '08: Proceeding of the 16th ACM International Conference on Multimedia*, pages 941–944, New York, NY, USA, 2008. ACM. ISBN 9781605583037. doi: http://doi.acm.org/10.1145/1459359.1459527.

[61] OpenCV. OpenCV open source computer vision library. December 2009. URL http://sourceforge.net/projects/opencvlibrary/.

Appendix A

OpenCV Eigen Functions

Below are the definitions for the two OpenCV eigen funtions from the OpenCV documentation [61]:

A.1 CalcEigenObjects

Calculates orthonormal eigen basis and averaged object for a group of input objects

```
void cvCalcEigenObjects( int nObjects, void* input, void* output, int ioFlags,
                int ioBufSize, void* userData, CvTermCriteria* calcLimit,
                       IplImage* avg, float* eigVals );
```

nObjects Number of source objects.

input Pointer either to the array of *IplImage* input objects or to the read callback function according to the value of the parameter *ioFlags*.

output Pointer either to the array of eigen objects or to the write callback function according to the value of the parameter ioFlags .

ioFlags Input/output flags.

ioBufSize Input/output buffer size in bytes. The size is zero, if unknown.

userData Pointer to the structure that contains all necessary data for the callback functions.

77

calcLimit Criteria that determine when to stop calculation of eigen objects.

avg Averaged object.

eigVals Pointer to the eigenvalues array in the descending order; may be *NULL* .

The function cvCalcEigenObjects calculates orthonormal eigen basis and the averaged object for a group of the input objects. Depending on *ioFlags* parameter it may be used either in direct access or callback mode. Depending on the parameter *calcLimit*, calculations are finished either after first *calcLimit.maxIters* dominating eigen objects are retrieved or if the ratio of the current eigenvalue to the largest eigenvalue comes down to *calcLimit.epsilon* threshold. The value *calcLimit -> type* must be *CV_TERMCRIT_NUMB, CV_TERMCRIT_EPS*, or

CV_TERMCRIT_NUMB | CV_TERMCRIT_EPS . The function returns the real values *calcLimit -> maxIter* and *calcLimit -> epsilon* .

The function also calculates the averaged object, which must be created previously. Calculated eigen objects are arranged according to the corresponding eigenvalues in the descending order. The parameter *eigVals* may be equal to *NULL*, if eigenvalues are not needed.

The function cvCalcEigenObjects uses the function cvCalcCovarMatrixEx.

A.2   EigenDecomposite

Calculates all decomposition coefficients for an input object

```
void cvEigenDecomposite( IplImage* obj, int nEigObjs, void* eigInput,
                         int ioFlags, void* userData, IplImage* avg, float* coeffs );
```

obj Input object.

nEigObjs Number of eigen objects.

eigInput Pointer either to the array of *IplImage* input objects or to the read callback function according to the value of the parameter *ioFlags*.

ioFlags Input/output flags.

userData Pointer to the structure that contains all necessary data for the callback functions.

avg Averaged object.

coeffs Calculated coefficients; an output parameter.

The function cvEigenDecomposite calculates all decomposition coefficients for the input object using the previously calculated eigen objects basis and the averaged object. Depending on *ioFlags* parameter it may be used either in direct access or callback mode.

Appendix B

Image Registration Code

Below is the code for the image registration technique:

B.1 Image Registration

```
IplImage* ImageReg(IplImage* input_image, int resize_me)
{
    CvBox2D box2d; //for drawing the rectangle
    CvPoint2D32f box_vtx[4]; //for drawing the rectangle
    int i; //for drawing rectangle
    CvPoint pt0, pt; //for drawing rectangle
    CvPoint Copy_of_pt0; //saving a copy for getting the ROI
    IplImage* roi = NULL; //for showing the ROI
    int lo_x;
    int lo_y;
    int hi_x;
    int hi_y;
    int x_diff;
    int y_diff;
    int edge_thresh = 44;
    int count_contours; //count the number of contours in a point set
    int count_points; //number of points in a contour
```

80

```
CvSeq* temp; //store re-orederd sequences
CvSeq* Top5; //accumulate all 5 point sets
int size_arr_totals = 500;
int arr_totals[size_arr_totals]; //array to store sequence totals
int count_totals; //variable to keep track of totals



IplImage* my_original_image  = input_image;
IplImage* my_black_image  = NULL;


my_black_image  = cvLoadImage( placeholder_image,
                                    CV_LOAD_IMAGE_GRAYSCALE );
IplImage* resize_temp  = cvCreateImage( cvGetSize(my_black_image), 8, 1 );
IplImage* resize_temp2  = cvCreateImage( cvGetSize(my_black_image), 8, 1 );
cvResize(my_original_image , resize_temp);
resize_temp2  = cvCloneImage(resize_temp);



IplImage* img_mine  = cvCreateImage( cvGetSize(my_black_image), 8, 1 );
IplImage* img_edge  = cvCreateImage( cvGetSize(my_black_image), 8, 1 );
IplImage* subtacted_img  = cvCreateImage(cvGetSize(my_black_image), 8, 1);
CvMemStorage* storage  = cvCreateMemStorage();
CvSeq* first_contour  = NULL;
cvSub(resize_temp, my_black_image, subtacted_img);
cvCanny(subtacted_img, subtacted_img, (float)edge_thresh,
(float)edge_thresh*3, 3);
cvSmooth(subtacted_img, img_mine, CV_GAUSSIAN, 3, 3,1);
cvThreshold( img_mine, img_edge, 40, 200, CV_THRESH_BINARY );



int Nc = cvFindContours(
    img_edge,
    storage,
    &first_contour,
```

```
        sizeof(CvContour),
        CV_RETR_LIST
    );
    int n=0,k;


lo_x  = INT_MAX;
lo_y  = INT_MAX;
hi_x  = 0;
hi_y  = 0;


    for(  count_totals=0; count_totals < size_arr_totals; count_totals++ )
    {
    arr_totals[count_totals] = 0;
    }
count_contours = 0;
CvSeq* c = first_contour;
count_totals = 0;


    for(  c=first_contour; c!=NULL; c=c->h_next )
    {
    if(c->total > 15)
    {
    arr_totals[count_totals] = c->total;
    count_totals++;
    }
    }
    qsort(arr_totals, size_arr_totals, sizeof(int), compare_int);


    for( c=first_contour; c!=NULL; c=c->h_next )
    {
    if( c->total == arr_totals[0] || c->total == arr_totals[1] ||
        c->total == arr_totals[2] || c->total == arr_totals[3] ||
        c->total == arr_totals[4])
    {
```

```
        for( int i=0; i<c->total; ++i )
        {
CvPoint* p = CV_GET_SEQ_ELEM( CvPoint, c, i );
//BEGIN: save the highest and lowest x and y values
if(hi_x < (p->x)) hi_x = p->x;
if(hi_y < (p->y)) hi_y = p->y;
if(lo_x > (p->x)) lo_x = p->x;
if(lo_y > (p->y)) lo_y = p->y;
//END
}


//BEGIN: display the found image region of interest
      roi = cvCloneImage(resize_temp2);
      cvResetImageROI( roi );
x_diff  = hi_x-lo_x;
y_diff  = hi_y-lo_y;
if(x_diff < 1) x_diff = 1;
if(y_diff < 1) y_diff = 1;
cvSetImageROI(roi,cvRect(lo_x, lo_y, x_diff - resize_me, y_diff));
//END
    n++;
}
}
   cvReleaseImage( &my_original_image );
   cvReleaseImage( &resize_temp );
   cvReleaseImage( &img_edge );
   cvReleaseImage( &img_mine );
   cvReleaseImage( &subtacted_img );
   cvReleaseImage( &my_black_image );


   return roi;
}
```

Appendix C

Notes on Image Registration

C.1   Image Registration Output

This evaluation shows the output of the image registration step and how this normalizes the image of the hand. We show the differences in output on different hands and demonstrate how these images are changed by pre-processing.

Pre-processing in this way removes outliers which can reduce recognition efficacy.

Table C.1 demonstrates that the output of our image registration remains consistent among different signers. Discrepancies in the output occur when:

- There are clearly visible shadows

  - The Canny edge detector will sometimes see these shadows as edges.
  - We did not eliminate shadows as system is intended to be used on live video. This tested the system for robustness.

- Different users perform a particular sign differently

  - Sign G2 has different outputs for all signers. We can therefore see the differences in each users interpretation of the sign.
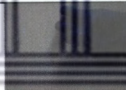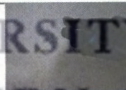
TABLE C.1: Image registration results shown on 2 signs from 4 different signers. The original image is on the left with the corresponding image registration output image visible on the right.