UNIVERSITY OF THE WESTERN CAPE

South African Sign Language Recognition Using Feature Vectors and Hidden Markov Models

Nathan Lyle Naidoo

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, Nathan Lyle Naidoo, declare that this thesis titled, 'South African Sign Language Recognition Using Feature Vectors and Hidden Markov Models' and the work presented in it is 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: ______________________________

Abstract

This thesis presents a system for performing whole gesture recognition for South African Sign Language. The system uses feature vectors combined with Hidden Markov models. In order to constuct a feature vector, dynamic segmentation must occur to extract the signer's hand movements. Techniques and methods for normalising variations that occur when recording a signer performing a gesture, are investigated. The system has a classification rate of 69%.

Acknowledgements

First and foremost I wish to thank the Lord Jesus Christ, the Son of God, who gave me the use of all my faculties and a sound mind, apart from which I would not be able to complete this thesis. I wish to thank my father, Prof. Anthony Naidoo and my late mother, Charmaine Naidoo for instilling in me the need for education. To my twin brother, Marc Naidoo who has stood by me through everything, your support and love will always be appreciated. To my loving wife Adrienne Naidoo, you have been my pillar of strength and your support has carried me through. To my colleagues, especially Mehrdad Ghaziasgar and Vaughn Segers, our hard work has finally paid off.

Finally, I wish to thank my supervisor Mr. James Connan for the wisdom and knowledge that he has demonstrated. The support, encouragement and drive you have shown, has been all inspiring and has pushed me beyond what I could of ever imagined. For this I am immensely thankful.

Contents

Declaration of Authorship i

Abstract ii

Acknowledgements iii

List of Figures vii

List of Tables ix

Abbreviations x

1 Introduction 1
- 1.1 Thesis Objectives . . . 2
- 1.2 Problem Statement . . . 3
- 1.3 Research Questions . . . 3
- 1.4 Technical Objectives . . . 4
- 1.5 Research Methodology . . . 5
- 1.6 Thesis Outline . . . 5

2 Gesture Recognition 7
- 2.1 Gestures . . . 7
- 2.2 Stokoe's Model . . . 8
- 2.3 Segmental Model . . . 9
  - 2.3.1 Movement-Hold Model . . . 9
- 2.4 Comparing the Two Models . . . 10
- 2.5 3D and 2D Modelling . . . 10
  - 2.5.1 3D Modelling . . . 10
  - 2.5.2 2D Modelling . . . 11
  - 2.5.3 Comparing 3D and 2D Modelling . . . 11
- 2.6 Gesture Recognition . . . 13
  - 2.6.1 Gesture Recognition using Computer Vision . . . 14
    - 2.6.1.1 Static Gesture Recognition . . . 14
    - 2.6.1.2 Summary On Static Gesture Recognition . . . 15
    - 2.6.1.3 Dynamic Gesture Recognition . . . 16

2.6.2 Summary on Dynamic Gesture Recognition 19
2.6.3 Conclusion 20
2.6.4 Summary 20

3 Preprocessing Techniques 21
3.1 Skin Segmentation 21
3.1.1 Colour Space 22
3.1.2 The RGB Colour Model 22
3.1.3 The CMYK Colour Model 23
3.1.4 The HSI Colour Model 23
3.1.5 Conversion from RGB to HSI 24
3.1.6 Comparing the RGB and HSI Colour Space 25
3.1.6.1 Skin Segmentation With RGB 26
3.1.6.2 Skin Segmentation With HSI 26
3.2 Background Subtraction 27
3.2.1 Adaptive Modelling 28
3.2.2 Non-Adaptive Modelling 28
3.2.3 A Comparison of Adaptive and Non-Adaptive Modelling 31
3.3 Normalising Method 31
3.3.1 Find Head Method 32
3.3.2 Centring Method 34
3.3.3 Construct Grid Method 35
3.3.4 Feature Extraction 37
3.4 Summary and Conclusions 39

4 Hidden Markov Models 40
4.1 Markov Process 40
4.2 Hidden Markov Model 42
4.2.1 The Three Basic Problems for HMMs 43
4.2.2 A Solution to the Evaluation Problem 43
4.2.3 A Solution to the Decoding Problem 44
4.2.4 A Solution to the Learning Problem 46
4.3 Summary 47

5 Experimental Setup 48
5.1 iSign Desktop 48
5.1.1 System Description 49
5.2 iSign Mobile 50
5.2.1 System Description 50
5.3 Data Collection and Training 50
5.4 Feature Vectors and Training 52
5.5 Summary and Conclusions 54

6 Results and Discussion 55
6.1 Preliminary Parameters and Techniques 55
6.1.1 Cropping a Frame Sequence of a SASL Gesture 55
6.1.2 Training 57
6.2 Results 58

6.2.1 Testing on Seen Data . . . 59
6.2.2 Testing on Unseen Data . . . 67
6.2.3 Testing the Normalising Method . . . 69
6.2.3.1 Test A . . . 70
6.2.3.2 Test B . . . 71
6.2.3.3 Test C . . . 73
6.2.3.4 Further Testing . . . 77
6.3 Summary . . . 78

7 Conclusion and Direction for Further Research **80**
7.1 Skin Segmentation . . . 80
7.2 Background Modelling . . . 81
7.3 Normalising method . . . 81
7.4 Feature Vector . . . 81
7.5 Future Work . . . 82

Bibliography **83**

List of Figures

1.1 The Waterfall Model . . . 5

2.1 A Taxonomy of Gestures . . . 8
2.2 The Movement-Hold Model . . . 9
2.3 Depth Mapping . . . 11
2.4 The 3D hand . . . 13
2.5 3D to 2D . . . 13
2.6 Frame Difference . . . 14
2.7 Vertical Histograms . . . 15
2.8 Hand Shape Classifiers . . . 15
2.9 Motion Sequence . . . 16
2.10 Motion Intensity . . . 17
2.11 Generalising Vectors . . . 18
2.12 Decision Tree . . . 18

3.1 The Skinmap . . . 22
3.2 RGB Primaries . . . 23
3.3 RGB Schematic . . . 24
3.4 HSI Schematic . . . 24
3.5 Different Lighting Conditions . . . 25
3.6 RGB Skinmap . . . 26
3.7 HSI Skinmap . . . 27
3.8 Background Subtraction . . . 29
3.9 Frame Differenced Images . . . 29
3.10 Background Images . . . 30
3.11 Adaptive Background Subtraction . . . 30
3.12 Haar Features . . . 32
3.13 The Classifier Cascade . . . 33
3.14 Classified Face . . . 34
3.15 Marr/Hildreth and Canny . . . 34
3.16 Centring the Signer . . . 35
3.17 The Vitruvian Man . . . 35
3.18 Overlay of the Grid . . . 37
3.19 Skin Detection Using the Grid . . . 38
3.20 The Feature Vector . . . 38

5.1 Setup A . . . 49
5.2 Setup B . . . 50

5.3 Experimental Results 52
5.4 Feature Vector for the SASL Word *Hello* 53

6.1 Motion Sequence for the SASL Gesture *Hello* 56
6.2 Motion Intensity for the SASL Gesture *Hello* 56
6.3 Ergodic HMM 58
6.4 Three State HMM 58
6.5 Movement and Hold Elements for the SASL Gesture *Help you* 65
6.6 Movement and Hold Elements for the SASL Gesture *Help me* 65
6.7 Feature Vectors for *Help you* and *Help me* 66
6.8 Side View of the SASL Gesture *Help me* 66
6.9 Side View of the SASL Gesture *Help you* 67
6.10 A Graph on the Testing Performance 69
6.11 Various Signers 70
6.12 Various Grid Sizes 70
6.13 Distance Manipulation 72
6.14 Cellphone Video 73
6.15 Movement Simulation 74
6.16 Feature Vectors for the SASL Gesture *Goodbye* 76
6.17 Manipulated Frames 78
6.18 Manipulated Data Results 78

List of Tables

2.1 ASL and the Movement-Hold Model 10
2.2 High Level Features 17
2.3 Comparison of GR Systems 19

5.1 Hardware Specifications 48
5.2 SASL and the Movement-Hold Model 51
5.3 Binary Representation 53

6.1 Sample Set for Seen Data 59
6.2 Testing Performance on Seen Data 59
6.3 Cross-Testing 61
6.4 Correct and Incorrectly Classified 62
6.5 SASL Gestures and their Hand Movements 63
6.6 Misclassified Gestures 64
6.7 Misclassification Rate 65
6.8 Training Sample Set on Unseen Data 67
6.9 Testing Sample Set on Unseen Data 67
6.10 Testing Performance on Unseen Data 68
6.11 Training and Testing Sample Set for Test A 71
6.12 Testing Results for Test A 71
6.13 Training Sample Set for Test B 72
6.14 Testing Sample Set for Test B 72
6.15 Results of Test B 73
6.16 Results of Test C1 75
6.17 Results of Test C2 75
6.18 Results of Test C3 76
6.19 Results of Test C4 76

Abbreviations

2D	**T**wo **D**imension
3D	**T**hree **D**imension
3G	**T**hird **G**eneration
ASL	**A**merican **S**ign **L**anguage
CMYK	**C**yan **M**agenta **Y**ellow **K**ey
CPU	**C**entral **P**rocessing **U**nit
EDGE	**E**nhanced **D**ata **G**lobal System For Mobile Communication **E**nvironment
GB	**G**igabyte
GHz	**G**igahertz
GPRS	**G**eneral **P**acket **R**adio **S**ervice
GR	**G**esture **R**ecognition
HMM	**H**idden **M**arkov **M**odel
HSI	**H**ue **S**aturation **I**ntensity
JSL	**J**apanese **S**ign **L**anguage
NN	**N**eural **N**etwork
PC	**P**ersonal **C**omputer
RAM	**R**andom **A**ccess **M**emory
RGB	**R**ed **G**reen **B**lue
RGBD	**R**ed **G**reen **B**lue **D**epth
RPM	**R**evolutions **P**er **M**inute
SASL	**S**outh **A**frican **S**ign **L**anguage

Chapter 1

Introduction

Verbal communication forms an integral part of our daily lives. It is used to convey, share or exchange information and ideas, and it helps human beings to connect with one another as individuals and as independent groups, such as families or work colleagues. Verbal communication channels are ubiquitous and are evident everywhere in society—telephones or cellphones, radio, television and voice over internet protocol. Communication is the very fabric, that binds societies together and allows them to function not only interdependently and independently, but also cohesively.

Not all members of society, however, are able to participate fully in the exchange of verbal communication. Deaf people, for example, are unable to hear verbal communication and for this reason have largely been marginalised in society. Throughout the world and especially in developing countries such as South Africa, deaf communities are faced with severely limited access to information, education and work opportunities in comparison to hearing communities. This has severely marginalised and impeded development in deaf society. An automated translation system that converts spoken language into sign language and sign language back into spoken language, would greatly benefit both the hearing and deaf communities and allow facilitated verbal communication between individuals from these communities. In the past decade there have been great strides in the field of human-computer interaction [1]. The reliance on the keyboard and mouse is seen as an inefficient and obtrusive way of interacting with a personal computer (PC). This has resulted in the development of alternative ways of interacting with the PC, such as speech and gesture recognition [2]. Advances in these technologies in recent years may offer a solution that can allow the deaf and the hearing to communicate with each other.

The Integration of Signed and Verbal Communication: South African Sign Language Recognition and Animation (SASL) project at the Computer Science Department at the University of the Western Cape (UWC), of which this research is part, aims to develop a full, unobtrusive translation system which will not only aid in the communication between the deaf and hearing, but also aid in deaf to deaf communication. This computerised system would need to identify three components of a sign simultaneously, namely, hand gestures, hand articulation and facial expressions. These three components would then need to be analysed linguistically to determine the meaning, before converting it into a spoken language [3, 4].

This research focuses solely on dynamic hand gesture recognition, and adds to the body of knowledge already acquired in the SASL group at UWC. Some of the research completed at UWC, pertaining to SASL recognition include: a real-time facial recognition system which was developed by J.R. Whitehill[4] and a hand shape recognition system developed by V.M. Segers [3].

1.1 Thesis Objectives

The research goals of this thesis are three-fold:

1. We wish to develop an automatic gesture recognition system that recognises dynamic hand gestures, and is able to convert gestures signed in SASL into English, without the use of additional, obtrusive equipment. Such obtrusive equipment includes: datagloves, coloured gloves and coloured markers on the hand. To this end we employ computer vision techniques that use a camera and image manipulation algorithms to interpret gestures.

2. We wish to maximise gesture recognition (GR) rates across different signers by developing an algorithm that normalises each video, before attempting to classify the gesture. The objective of this method is to minimise variations that can occur in multiple videos of signs, using multiple signers. The method must cater for the following variations:

   - Variations in body dimensions,
   - Variations in the distance of the signer from the camera, and
   - Variations in the position of the signer within the video frame.

3. Using the GR system and the normalising method that we aim to develop, we aim to construct and design a software prototype and framework for the GR system to work on. The prototype system must allow us to record SASL using a standard webcam attached to a personal computer or cellphone camera. The prototype

system must interpret the gestures and output English text or audio using text-to-speech technology. Ultimately this framework can be used in future work to allow the constitutive contributions such as that of Whitehill and Segers [3, 4] to be merged into the system.

Our GR system consists of three processes:

1. Image preprocessing,
2. Feature extraction, and
3. Feature classification.

In this thesis we investigate several methods and techniques used to implement and develop each of the above mentioned processes.

1.2 Problem Statement

Our goal, as mentioned in Section 1.1, is to create an unobtrusive GR system, using hand gestures, which is be able to convert SASL words or phrases into English text. In order to achieve this goal, we need to address the following problems:

- Describe the hand movements within a sign for a computer to interpret them, and
- To maximise GR rates across multiple signers, we need to minimise variations that occur in videos of signed gestures using different signers.

1.3 Research Questions

In order to solve the above problem statements, we need to analyse and answer the following research questions:

- GR systems use frames from a video as input. The system relies on the extraction of appropriate information from a video. What are the appropriate features that will allow us to classify a signed gesture accurately and how are these features extracted?
- Since we are using only hand gestures to attempt to classify a sign gesture, it is imperative to question whether it is possible to interpret a SASL gesture by only using hand gestures?

- SASL gestures are dynamic. We therefore need to ask: "What is an appropriate manner for describing hand movement in a video that can be used to train and test the system?"
- Given that no two signers have the same physical body structure, how do we dynamically normalise the features that would need to be extracted?

1.4 Technical Objectives

In order to address the above mentioned research questions, we need to achieve the following technical objectives:

- Gesture definition: Breaking down hand gestures in SASL into smaller components can aid the classification process and allow for easy identification of the beginning and end points of a sign.
- Preprocessing: Image preprocessing and feature extraction are paramount to the success and accuracy of a GR system. The image preprocessing consists of the following attributes:
  - Normalisation that caters for variances in different videos taken of different signers,
  - Background modelling in order to differentiate the background from the foreground, and
  - Feature extraction that extracts motion information of the hands, from each frame in a video.

  For each of these attributes we identify and implement a method that is appropriate for our purpose.
- Feature vector development: What distinguishes gestures from one another is their dynamic behaviour. We therefore need to identify an appropriate way to store extracted features and temporal information from a video of a SASL gesture.
- Performance analysis: We need to verify that our system can indeed correctly identify different signed gestures from several different signers.

1.5 Research Methodology

In order to build our system and achieve the above mentioned technical objectives, we use an adapted version of the waterfall model used to design and implement software. The waterfall model is shown in Figure 1.1.

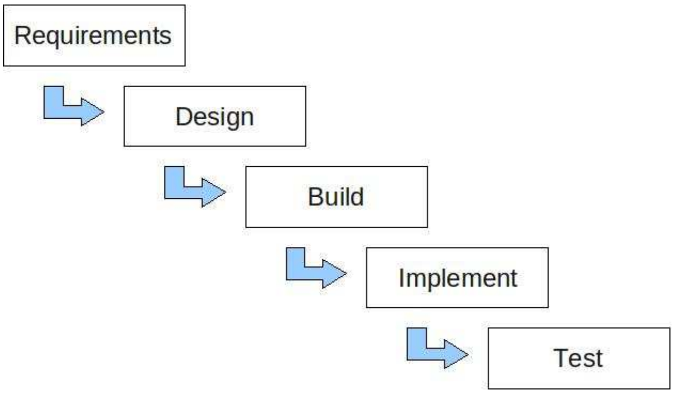


Figure 1.1: The waterfall model is used to design our system.

We use the waterfall model must be used to design, implement and test three core components of our system.

- Implementation of an image preprossessing algorithm which first, normalises variance that may occur in the video. Second, uses background modelling to distinguish the foreground from the background. Third, extracts hand gesture features that are appropriate for our objectives,
- Construction of a feature vector, using the features extracted in the preprocessing stage, and
- Classification of different gestures must be achieved by using the discriminating nature of Hidden Markov Models (HMMs).

1.6 Thesis Outline

The chapters of this thesis are outlined as follows:

In Chapter 2 we look at the taxonomy of gestures to find the context of gestures in South African Sign Language (SASL). We evaluate the structure of signed words or phrases by looking at two models proposed by Stokoe, and Liddell and Johnson [5, 6], which breaks up signs into sub-components (called cheremes) but differ substantially. We then describe gesture recognition and evaluate systems that have been implemented.

In Chapter 3 we discuss our preprocessing techniques. We begin by defining skin segmentation. An evaluation of different colour spaces is provided. We then assess both adaptive and non-adaptive modelling in background subtraction techniques. Finally, we look at the normalising of a signer in a video frame and the feature extraction of skin regions.

HMMs are discussed in chapter 4. The theory and application of HMMs in real-world solutions is discussed. The evaluation, decoding and recognition problems associated with HMMs are reviewed, followed by their solutions.

In Chapter 5 and 6 we describe our experimental setup and present our results. Concluding remarks and recommendations for future work are presented in Chapter 7.

Chapter 2

Gesture Recognition

In this chapter we review the literature of gesture recognition systems. We discuss the taxonomy of gestures to set the context of gestures in South African Sign Language. A comparison between three-dimensional (3D) and two-dimensional (2D) modelling determines an appropriate approach for our study. We examine several static and dynamic gesture solutions that have been developed. We compare these solutions, followed with a conclusion and summary.

2.1 Gestures

Gestures encapsulate a large family of body movements that express ideas or meaning [7, 8]. Gestures can be categorised as follows:

- Manipulative gestures (e.g. picking up a ball) are used to move objects and convey no direct meaning in conversation [9], and
- Communicative gestures (e.g. waving goodbye) are used to express meaning and intent in conversation [10–12].

Figure 2.1 illustrates the gesture hierarchy. Gestures can be broken down into two main classes: manipulative and communicative. We refer to the different sub-classes of communicative gestures to identify an appropriate context for South African Sign Language recognition. For our research we only consider the manual movement class. The manual class is sub-divided into two categories, static and dynamic. Static gestures denote gestures when no perceived change takes place in the orientation, shape or position of the hands for a period of time. Dynamic gestures are gestures where the hands are moving. There are three distinguishable types of dynamic manual movements in SASL:

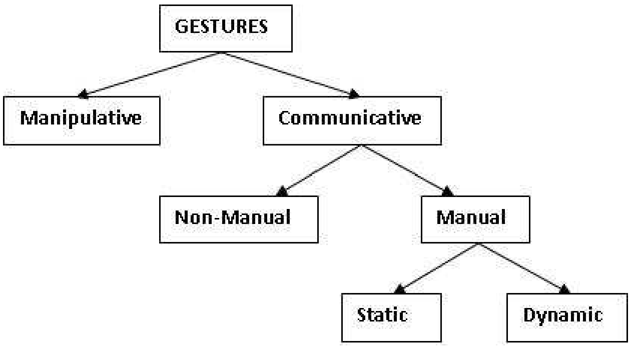


Figure 2.1: A taxonomy of the gesture classes used to find the context of sign gestures[13].

- Global movement is where the start and end location of the signer's hands are in a different place in front of the signer (sign space). These include movements from side to side, up and down, toward and away from the signer, and circular movement along these three axes,
- Secondary or local movement usually accompanies the global movement and can be characterised by localised movements of the fingers, and
- Epenthesis movement. This is not part of any particular sign. It is usually found at the beginning of the sign or between signs [14].

2.2 Stokoe's Model

Stokoe realised that signs can be broken down into smaller units, cheremes or phonemes[1] [5]. A phoneme is defined as the smallest unit in a language. Phonemes allow humans to distinguish one word from another. He demonstrated the applicability of phonemes in sign language. For GR purposes phonemes are suitable, as there is a limited number of these units in any language, as opposed to an unlimited number of words that can be built from the phonemes. Stokoe used three parameters or descriptors to describe a phoneme in sign language. These parameters are the location of the hands (tabula or *tab*), the hand shape (designator or *dez*) and the movement of the hand (signation or *sig*). He emphasised that phonemes occur simultaneously in sign language and assumed that variations in the sequence of these descriptors within a sign are not considered to be significant [15].

[1] The term chereme was coined in 1960 by William Stokoe as part of an attempt to demonstrate that signed languages are full languages. It is a technical term for the smallest meaningful units of a sign language. Phoneme is the technical term for the smallest unit in the sound system of a spoken language. The position is now universally accepted that phonemes and cheremes are organisationally and functionally equivalent at every level of linguistic structure.

2.3 Segmental Model

Liddell and Johnson argued against Stokoe's assumption [6]. They outlined the phonological structure and processes of sign language and emphasised, in contrast to Stokoe, that phonemes occur sequentially in sign language. Such models are called segmental models [15]. Liddell and Johnson called their model the Movement-Hold model.

2.3.1 Movement-Hold Model

Signs are made up of sequences of movements and holds [6, 15]. A complex sign is divided into two elements, that is, movement and hold elements as indicated in Figure 2.2. This model can also be classified as segmental, where each sign is broken down into a series of phonemes. Movement elements are characterised by segments where some aspect of the sign's configuration has changed, such as a change in hand shape, orientation or movement. Hold elements are defined as those segments where no change has taken place (e.g. a stationary hand).

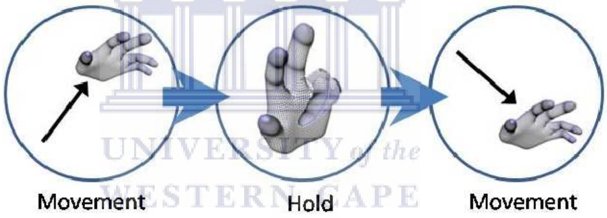


Figure 2.2: The structure of the Movement-Hold model.

A hold is a static sub-unit of a gesture, which is composed of three simultaneous and inseparable components as shown in the Equation 2.1. Let P represent the pose, h, o and l denote hand shape, palm orientation and hand location, respectively.

$$P = (h, o, l) \tag{2.1}$$

A movement, M, is a dynamic sub-unit of a gesture composed of the velocity, v, and the direction, d, of the hands as they travel between successive poses in sign space. This is shown in Equation 2.2.

$$M = (d, v) \tag{2.2}$$

A manual gesture is therefore made up of a sequence of holds (H) and movements (M) elements. Table 2.1 shows an example of words in American Sign Language (ASL) with the sequence of hold and movement elements.

Word	**Sequence**
Good	H-M-H
Sit	M-H
Chair	M-M-M-H

TABLE 2.1: American Sign Language words and their sequence of holds and movements derived from the Movement-Hold model [15].

2.4 Comparing the Two Models

Liang and Ouhyoung adopted Stokoe's model by using modified Cybergloves to extract the *tab*, *dez* and *sig* features of a sign [16]. They use Hidden Markov Models (HMMs) to recognise 250 words in Taiwanese Sign Language. Their system is continuously able to recognise gestures in real-time and achieve a classification rate of 80.4%. Volger and Metaxas use the Movement Hold model when designing their system [15]. They experimented on a 22 word ASL vocabulary. Even though their vocabulary is considerably smaller compared to that of Liang and Ouhyoung's vocabulary, they achieve a higher recognition rate of 91.82%. Volger and Metaxas, and Liddell and Johnson argued that the fundamental weakness in Stokoe's model [6, 15], is that his model cannot distinguish between gesture components with a single movement and repeated movements, with all other features being equal [17]. Even though the Movement-Hold model does not include non-manual features and inherits some redundancy, its sequential nature is ideally suited for HMMs and has widely been used to classify gestures [15, 18].

2.5 3D and 2D Modelling

2.5.1 3D Modelling

3D modelling entails capturing a gesture or sign in a three-dimensional space [19]. Datagloves have been widely used to track the 3D movement of hands. It uses an array of sensors fitted to gloves that the user wears. The sensors are able to record information such as hand shape, hand orientation, hand global position and hand velocity. A pinch glove was used by Kim and Waldron to obtain a sequence of 3D positions of a hand's trajectory [20]. The sequential data obtained from the datagloves is used to classify the gesture. Kim and Waldron were able to achieve a GR accuracy of 86% using the pinch glove. 3DV systems have developed an image sensor which is capable of producing RGBD signals, where R stands for red, G for green, B for blue and D stands for the distance of each pixel relative to the camera's position [21]. This makes it possible to track hand movements in 3D without the signer having to wear datagloves. While still under development 3DV systems let Fujimura and Liu develop a GR system for Japanese Sign Language (JSL) [22]. Fujimura and Liu use the information captured by 3DV system's image sensor [2] to classify JSL. Depth information is displayed in an image produced by the camera. Dark regions denote objects that are far away from the camera with lighter regions describing objects that are closer. An example of an image produced by the 3DV system's camera is shown in Figure 2.3.

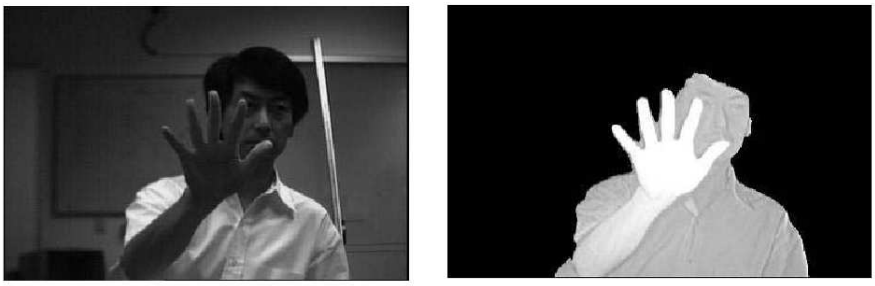


FIGURE 2.3: An example of an image containing depth information produced by the 3DV system's camera. Lighter areas are closer to the camera [22].

Using this technology Fujimura and Liu were able to achieve an error rate of less than 5% with a vocabulary size of 15 JSL gestures.

2.5.2 2D Modelling

2D modelling is perspective based, that is, where 2D image data is captured from a single camera's point of view. Image segmentation and manipulation algorithms are used to extract information from the image. This information is used to classify the gesture. Grobel and Assam achieve a classification rate of 91.3% by extracting features from a video of signers wearing coloured gloves [23]. The coloured gloves made segmenting and extracting the hands' position and shape more robust. 2D modelling techniques rely on computer vision algorithms to extract information of a gesture, rather than using specialised equipment.

2.5.3 Comparing 3D and 2D Modelling

Grobel and Assam use HMMs to classify 262 isolated signs with a 91.3% accuracy. Starner and Pentland use a similar technique to obtain two-dimensional features [24].

[2] 3DV Systems has been acquired by Microsoft Inc.. Attempts to acquire this technology has been unsuccessful.

They treated signs as whole units and made no attempts to break them down into phonemes. They trained HMMs on a 40 word vocabulary and obtained a 91.9% accuracy. Both teams' results indicate that comparable recognition rates can be achieved without the cost of three-dimensional techniques. Kim and Waldron trained neural networks (NN) with data obtained from a modified pinch glove, to recognise phonemes within sign language [20]. The phonology included 36 hand shapes, 10 locations, 11 orientations, and 11 hand motions. Kim and Waldron were able to achieve an overall GR accuracy of 86%.

There are a number of complexities with 3D modelling as argued by Nam, Korea and Wohn [19]:

- *Temporal Variance* : Variations exist in speed and location of movement between different people performing the same sign,
- *Spatial complexity* : The human body possesses a high degree of freedom in 3D space. This is due to the following aspects:
  - Large variations of shape,
  - Rotation variance, and
  - Translation variance.
- *No start and end point* : There is no explicit start and end points in a gesture,
- *Repeatability and connectivity* : The way gestures are connected to each other makes it difficult to classify them. Segmentation is therefore crucial, and
- *Multiple attributes* : Features such as the hand shape, hand orientation and hand position needs to be simultaneously processed.

In addition to these challenges, hardware for retrieving 3D information is extremely specialised and expensive. Nam, Korea and Wohn argue that 3D modelling is an extremely complex task, when classifying gestures [19]. They describe how 3D data of hand movements can be reduced to 2D, thus reducing the complexity. Figure 2.4 shows some of the attributes tracked in three-dimensions. The process of fitting 3D to 2D is shown in Figure 2.4(c).

The hand is shown moving in three-dimensional space, illustrating a high degree of freedom in the rotational and transformation aspects. They used a chain encoding

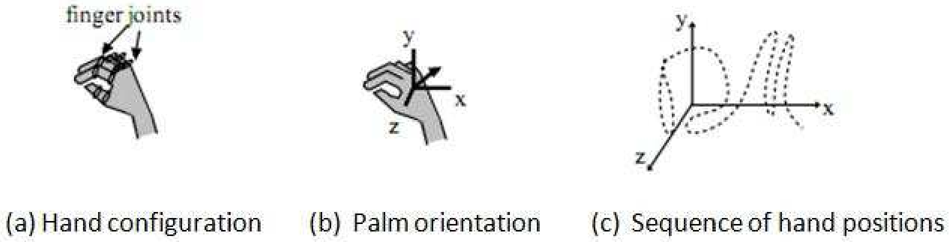


Figure 2.4: 3D hand gesture attributes.

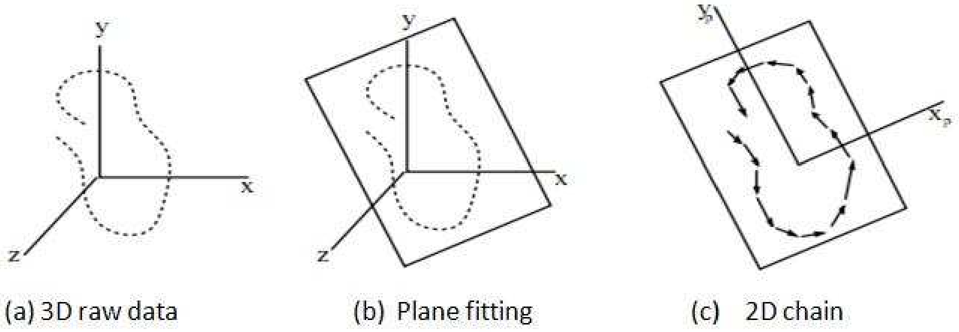


Figure 2.5: 3D to 2D reduction by plane fitting.

scheme for describing the hand movement path in 2D, shown in Figure 2.5. They were able to achieve a GR accuracy of 80%, which compares well to the pinch glove systems accuracy of 86%, developed by Kim and Waldron.

2.6 Gesture Recognition

Communication is a complex platform that is made up of several components. It is often carried out by means of gestures, facial expressions and vocal sounds. In addition to these components, body language plays a key role when interpreting communication. In many cases, such as a person waving, most of the communicative information is contained in the gesture. [25–27]. Gestures can either be used to assist voice communication, or independently in sign language. Virtual reality applications have previously used this technology for virtual manipulation and communication. Gestures have proven effective as a suitable means of communication in such applications [28, 29].

Datagloves and computer vision-based techniques are the two well-known means of recognising hand gestures [17]. To date, most research on gesture and sign language recognition have made use of wired datagloves. Datagloves are input devices that the user wears. They use an array of motion and pressure sensors. These sensors are used to capture the global position, rotation and deformation of the hand. These systems have predominantly focused on finger spelling, where the user spells each word with the appropriate hand gestures [30]. However, this is counter intuitive and time consuming as sign language consists of gestures that represent whole words. Glove based gestural interfaces require users to wear cumbersome hardware, which inhibits the natural movement of the signer.

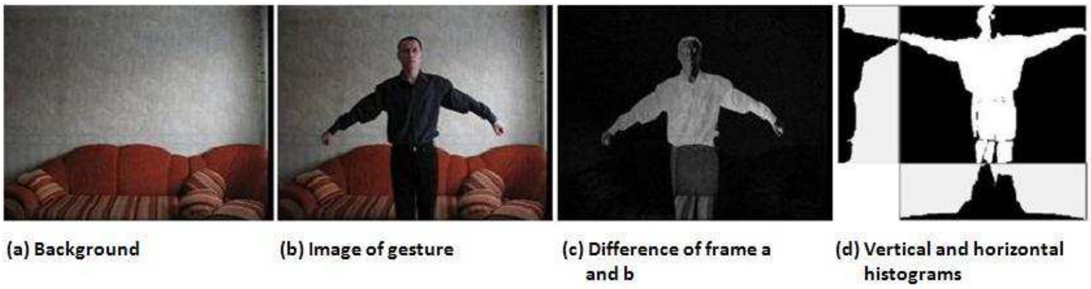
Computer vision based techniques use a camera and image manipulation algorithms to interpret gestures. This provides a more natural way of interactions between the system and user. The process of finding and analysing hand postures in cluttered images is extremely complex and troublesome. This has led to the development of methods involving wearing of coloured markers or coloured gloves on the hands and restricting the background of the video. The wearing of extra equipment and restricting the background of the video are widely acknowledged limitations of computer vision based techniques.

2.6.1 Gesture Recognition using Computer Vision

In this section we will examine two types of manual gestures namely, static and dynamic gestures. First, an examination on two solutions for static GR using histograms and eigenvectors is provided. Second, a discussion on several dynamic GR solutions is then provided.

2.6.1.1 Static Gesture Recognition

Kirillov uses a simple yet efficient gesture recognition method based on analysing horizontal and vertical histograms [31]. Background modelling is used to extract the signer from the image. This is done by subtracting the n^{th} frame from a reference frame, where the reference frame is the background. Figure 2.6(c) shows the absolute difference between image a and b. The difference produces a grey scaled image with only the signer present, without the background. Darker areas indicate areas of less difference, whereas lighter areas show areas of higher difference. Using the frame differenced image,



(a) Background (b) Image of gesture (c) Difference of frame a and b (d) Vertical and horizontal histograms

Figure 2.6: Framed differenced image, used to construct a vertical and horizontal histogram signature for classification by Kirillov' system 2.6.

the system generates a vertical and horizontal histograms of key regions in the video frame. The vertical histograms of the arm regions are shown in Figure 2.6(d). By using statistical methods Kirillov compares peaks in both histograms of an image to that of a predefined histogram data set. Figure 2.7 shows the vertical histograms and associated arm gesture used by the system. In a controlled environment and lexicon size of 15 static gestures the system is able to achieve an accuracy of 97%. Segers implemented a static GR system for the classification of hand shapes found in SASL [3]. He used eigenvectors to classify 9 different hand shapes in real-time as shown Figure 2.8. Using 10 images to train an eigenvector on each hand shape, Segers is able to achieve a real-time recognition rate of 88.9%.

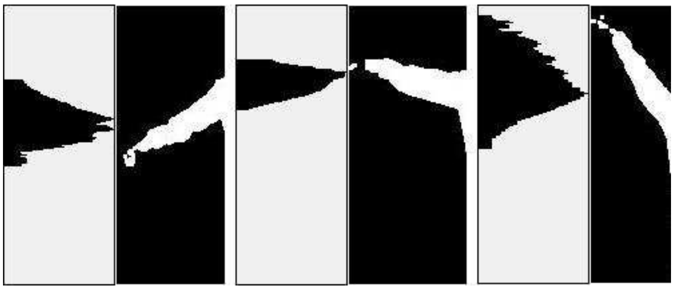


Figure 2.7: Vertical histograms of the arm regions.

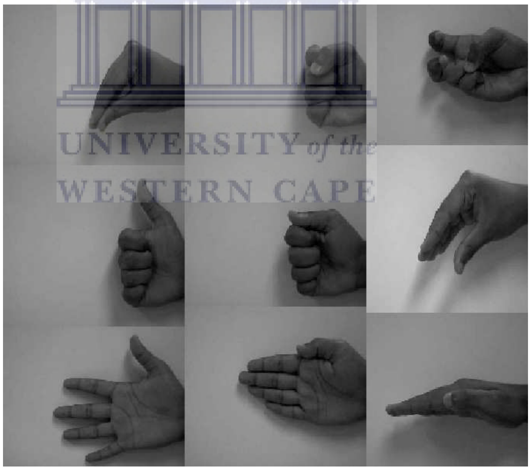


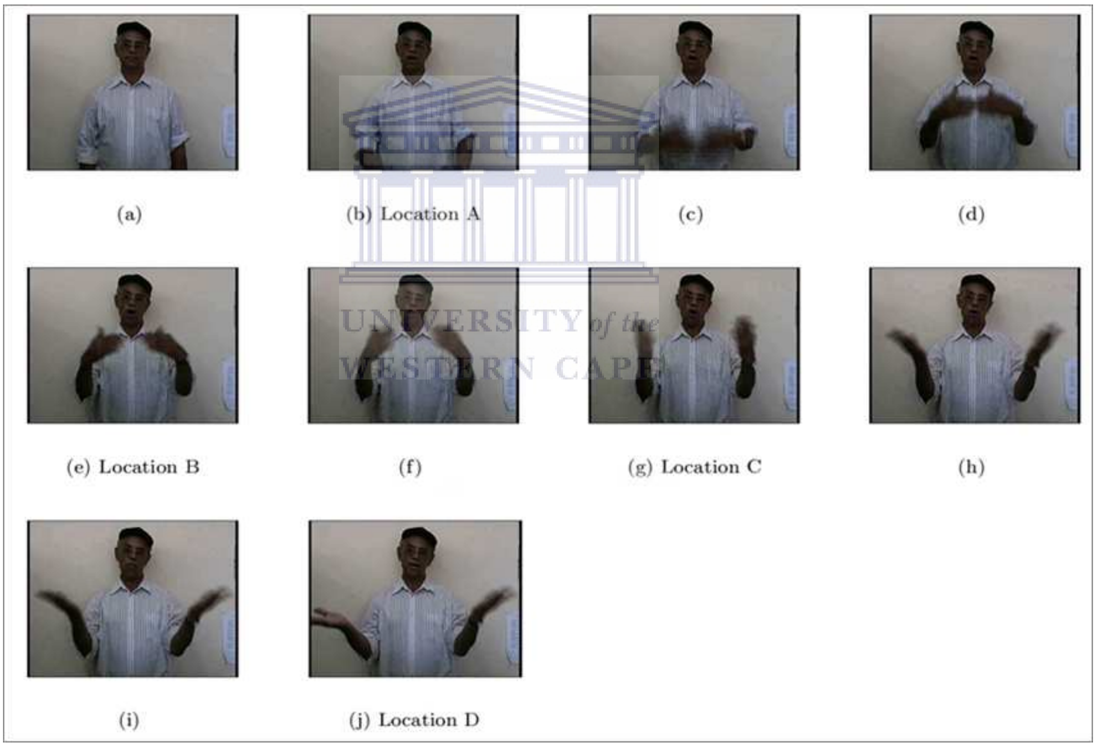
Figure 2.8: Hand shape classifiers used in Segers' system [3].

2.6.1.2 Summary On Static Gesture Recognition

Most GR systems have focused on dynamic GR solutions. As a result, there are fewer systems that focus solely on static gestures. Though Kirillov's and Segers' systems have a high recognition rate, both lexicon sizes are small. In addition to this, their system fails when there is background noise. In Kirillov's case, his system fails when shadows fall on the wall behind the signer, creating two peaks in the vertical histogram.

2.6.1.3 Dynamic Gesture Recognition

Rajah describes a gesture recognition system based on recognising sub-components of a gesture [17] (i.e. phonemes). He argues that it is sensible to break down signs into their smallest component in order to handle variations in two signs performed by different people. He employs the Movement-Hold model and defines phonemes by extracting the intensity flows of the signer in a video. His segmentation involves searching for natural abrupt variations in movement. This involves identifying changes in the signer's arm and hand movements. He thus ignores the hand shapes and orientations and only focuses on the velocity and direction of skin regions. Rajah calculates the local minima and maxima of velocity for skin regions. He uses this to identify the movements and holds within a sign. Figure 2.9 shows sample frames of a signer signing the SASL word *Open* and Figure 2.10 plots the intensity flows for each frame. He trains HMMs to recognise these intensity flows and identified 23 phonemes based on 102 training signs. He achieves an individual phoneme recognition rate of 71%.


(a) (b) Location A (c) (d)

(e) Location B (f) (g) Location C (h)

(i) (j) Location D

FIGURE 2.9: Motion sequence for "open" sign [17].

Bowden et al. describe a HMM based recognition system for sign language recognition with a lexicon size of 49 signs [32]. They use high level descriptors to classify signs.

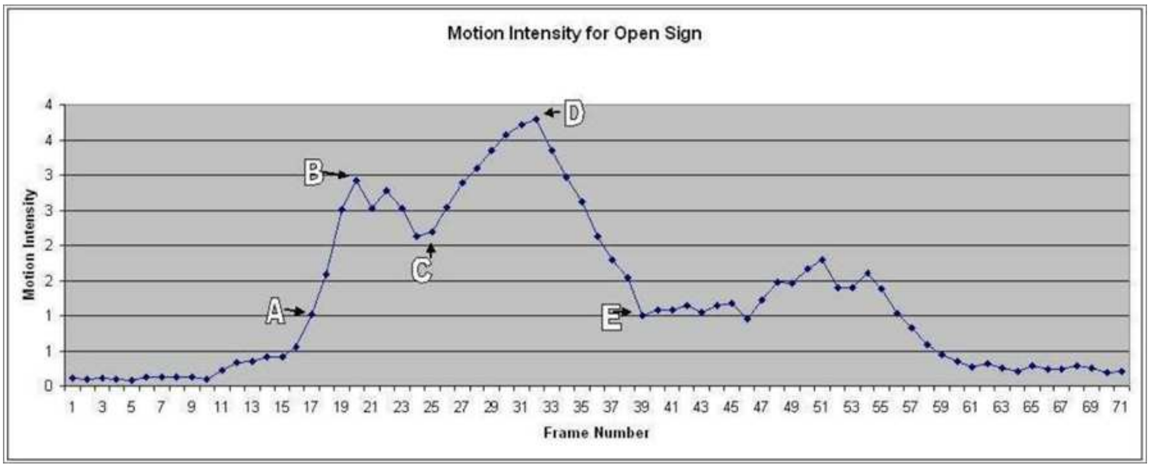


FIGURE 2.10: Intensity of Motion [17].

They argue that features such as hand shape, hand orientation and hand position are important and can be used not only to shorten training a HMM, but also to improve GR rates [32]. They employ the high level descriptors that were developed by Stokoe, shown in Table 2.2, to describe a series of actions in a sign. HMMs are trained on each of the 49 signs using the descriptors. A high classification rate of 97.67% is achieved. This result compares well to Starner and Pentland's system which uses coloured gloves and has a vocabulary of 40 signs. Starner and Pentland's system achieves an accuracies of 91.3% [33]. Bowden et al. show that by extracting *HA*, *TAB*, *SIG* and *DEZ* from a video sequence they are able to eliminate a significant amount of noise and thus reduce the amount of transition states required in the Markov chains [32]. Their system is able to generalise these features across different individuals making it possible for the system to classify signs from different people to that of the training set. Figure 2.11(a) shows the ASL word *Different* being performed by two people with the extracted features represented by the binary vectors. The two binary vectors are very similar and are classified correctly by their system.

Name	**Description**
HA	Position of the hands relative to each other
TAB	Position of the hands relative to key body locations
SIG	Relative movements of the hands
DEZ	The Shape of the hands(s)

TABLE 2.2: The high level features used to describe a gesture [32].

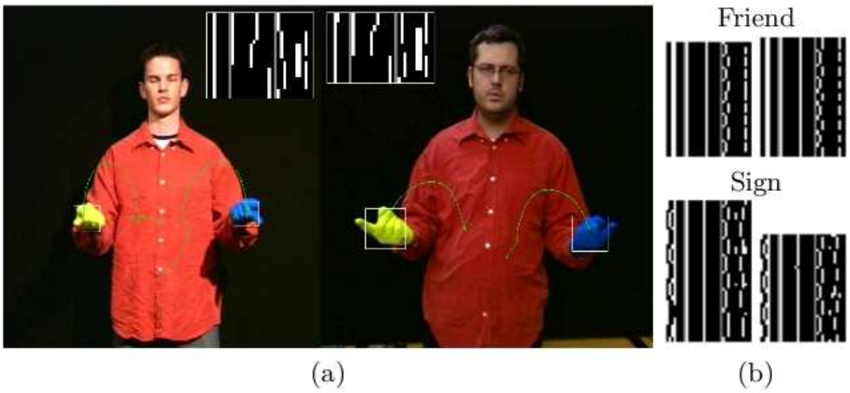


Figure 2.11: Generalisation of feature vectors across different individuals. [32].

Figure 2.11(b) shows two more sign feature vectors, indicating how similar binary feature vectors are produced for the same sign.

Kadous developed a gesture recognition system using *PowerGloves* [34]. The gloves are worn on both the left and right hands and use an array of sensors to provide real time information about their shape orientation and movement. Instrumented gloves have in the past been extensively used in direct manipulation of 3D virtual environments [35]. They have more recently been applied to gesture recognition [34]. Kudous uses a PowerGlove originally designed for gaming on the Nintendo 8 bit console and applies it in gesture recognition. Using the data captured by the PowerGloves, he constructed a decision tree using C4.5 [36] for a particular sign. An example of a decision tree for deciding whether or not to play golf is shown in Figure 2.12. He achieves a classification rate of 80%.

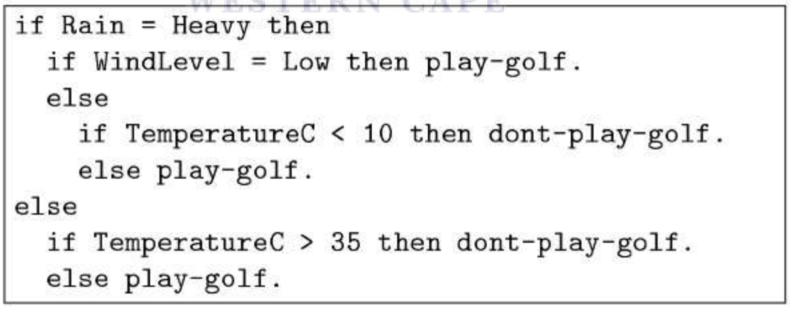

```
if Rain = Heavy then
  if WindLevel = Low then play-golf.
  else
    if TemperatureC < 10 then dont-play-golf.
    else play-golf.
else
  if TemperatureC > 35 then dont-play-golf.
  else play-golf.
```

Figure 2.12: A simple decision tree for deciding to pay golf. [34].

Wang et al. describe a Chinese sign language system that uses instrumented gloves [37, 38]. However, they use HMM isolated recognition for their classification. They have an extensive lexicon of 5100 signs. Due to the size of their vocabulary, a fast matching algorithm was implemented to run in real-time [37]. Each of the 5100 signs was performed five times, four were used for training and one was used for testing. They achieved real-time recognition rate of 95%.

2.6.2 Summary on Dynamic Gesture Recognition

Table 2.3 presents the above discussed systems, highlighting techniques and methodologies used. Rajah uses the Movement-Hold model and omits the use of hand shape and orientation but rather focuses on their movements. It should be noted that he does not achieve a high accuracy and he does not use any specialised equipment. Bowden et al., Starner and Pentland use coloured gloves, making extracting hand features easier. Bowden et al. use red long sleeved shirts, shown in Figure 2.11, worn by different signers making segmentation easier. The black background also ensures no background noise. A common hurdle with gesture classification systems is signer-independent recognition.

System Created By:	**Model:**	**3D/2D Modelling:**	**Specialised Equip.:**	**Classification Technique**	**Lexicon Size:**	**Accuracy**
Kirillov	n/a	2D	none	Correlation	15	97%
Segers	n/a	2D	none	Eigenvectors	9	80%
Rajah	Movement -Hold	2D	none	HMMs	23	71%
Bowden et al.	Stokoe	2D	Coloured gloves	HMMs	49	97.67%
Starner and Pentland	Movement -Hold	2D	Coloured gloves	HMMs	40	91.3%
Kadous	Stokoe	3D	Datagloves	Decision tree	95	80%
Wang et al.	Stokoe	3D	Datagloves	HMMs	5100	95%

Table 2.3: A comparison of gesture recognition systems.

Kadous cited a major caveat of his system: that people who were not used to train the system achieve accuracies of about 12% to 15% [34]. This weak classification can be partially attributed to the strict deterministic nature of decision trees. The inability of the system to generalise input across interpersonal variations severely impedes the scalability and feasibility of such a solution. Vamplew and Adams describe a gesture recognition system for 52 Australian signs [39]. It uses neural networks to classify gestures and is able to achieve classification rate of 94% on seen data and 85% on unseen data [39]. Akyol and Canzler describes an information terminal that has a vocabulary size of 16 German signs [40]. Ten people were recorded performing these signs, seven were used for training the HMM and three were used for testing. They achieved a classification accuracy of 94% [40].

2.6.3 Conclusion

In the preceding sections we have described several implementations of gesture recognition systems, for both static and dynamic gestures. For the purpose of this study we focused only on dynamic gestures. Decision trees are not feasible for interpersonal classification systems, however their rigidity can be exploited to determine a subset of the vocabulary for the system to classify, rather than searching through the whole vocabulary. HMMs have been used extensively to implement gesture recognition solutions. Most of these systems use cumbersome equipment i.e. datagloves and coloured gloves. Our research focuses on developing an interpersonal HMM classification system without the use of restrictive equipment. Our study focuses on how to normalise input across different signers. This is an important aspect as this greatly affects the performance of the system.

2.6.4 Summary

We have discussed the taxonomy of gestures and provided a context of gestures in SASL. Our research will focus on dynamic GR. A comparison between 3D and 2D modelling was discussed, showing that 2D modelling can be as effective as 3D modelling. We have compared a range of gesture recognition systems and highlighted their strengths and weaknesses.

Chapter 3

Preprocessing Techniques

In this section, we describe preprocessing techniques and feature extraction methods used by our GR system. The human form consists of several complex appendages. Tracking the face, hand and torso's simultaneous movements is very complicated in comparison to tracking artificial objects [41]. There are a number of cues or features which can be considered when trying to narrow the search space in tracking or identification systems. The most important of these features are spatial, temporal and textural [41]. We begin by discussing skin colour segmentation as it has proved to be a useful and robust feature for localisation and tracking. Many researchers have built up a wide range of knowledge on this tracking technique [8, 42–44]. This is followed by a discussion on background modelling. Background noise is one of the main factors that influence a computer vision system's accuracy. Many systems have used constrained backgrounds to control this factor [32]. A discussion on methods used to normalise features across different people is provided. Finally we describe how the system extracts features from the preprocessed image for classification.

3.1 Skin Segmentation

The objective of skin colour segmentation is to build a decision rule that determines if a pixel is skin or not. Pixel-based skin detection methods functions by sequentially and independently analysing each pixel's colour in an image and determining whether it is skin or non-skin [45]. Methods that incorporate pixel spatial relationships, take into account what the current pixel's neighbour's values are in order to classify the current pixel. However, spatial relationship algorithms depend on certain aspects of pixel-based methods and can be partially dependent on the performance of the pixel-based methods

[45–48]. For this reason, we focus on pixel-based approaches to classify skin regions rather than spatial relationship approaches.

Skin colour pixels vary under different lighting conditions, making pixel-based classification a non-trivial task. Identifying what is considered to be skin colour, involves finding the range of values, attributed to skin, in a given colour space. An example of skin segmentation is shown in Figure 3.1, with 3.1(a) being the source image and 3.1(b) being the output binary image, which is also known as the skinmap.

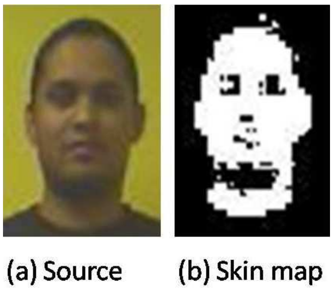

(a) Source (b) Skin map

FIGURE 3.1: Skin segmentation with skin pixels marked with white.

3.1.1 Colour Space

In order to optimise skin recognition, an appropriate colour space is needed. The objective of using a colour space or a colour model is to standardise the specification of colours in some standard format. A colour space is a specification of a coordinate system and a subspace within that system where each colour is represented by a single point [49]. When designing a skin based segmentation system an important factor to consider is choosing the correct colour space. Most colour models that are in use today focus either towards hardware interpretation or towards applications where colour manipulation is key. Colour spaces that are generally used include the Red, Green, Blue or RGB model, the Cyan, Magenta, Yellow, Key or CMYK model and finally the Hue, Saturation, Intensity or HSI model. Pixel-based skin colour localisation is extremely sensitive to environmental influences, such as noise and illumination changes. Choosing the correct colour space minimises false positive skin detections.

3.1.2 The RGB Colour Model

RGB is probably the most commonly hardware-oriented model used today. It is used in colour monitors and a wide range of colour video cameras [49]. Each colour in the RGB model appears in its primary spectral components of red, green and blue. Using 24 bit colour, each RGB component value ranges from 0, being the lowest intensity that can be displayed on a monitor, to 255 being the highest. All three components are used to produce a single colour. Figure 3.2 represents an example of the RGB colour space. When all three components are combined at their highest intensity, white is produced.

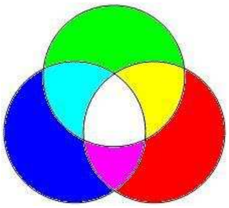


FIGURE 3.2: RGB primaries.

3.1.3 The CMYK Colour Model

Cyan, Magenta, and Yellow are the secondary colours of light and are the primary colours of pigments [49]. Figure 3.2 shows how CMY component colours are created by combining half of any two primary colours e.g. when red and blue are combined in equal amounts, Magenta is formed. The K, also known as Key, refers to the black colour component, it is formed when CMY colours are combined. The CMYK colour space is typically used in colour printers and copiers.

3.1.4 The HSI Colour Model

The HSI model describes colour in terms of hue, saturation and brightness. The hue attribute describes a pure colour. Hue is associated with the prevailing wavelength in a mixture of light waves which represent the dominant colour as viewed by an observer. Saturation measures to what degree pure colour is diluted with white light. Collectively hue and saturation is called chromaticity. Colour can be characterised by its colour sensation or brightness and chromaticity. *Tristimulus* is the quantity of red green and blue that is required to form a particular colour and is denoted, X, Y, and Z,respectively.

$$x = \frac{X}{X+Y+Z} \tag{3.1}$$

$$y = \frac{Y}{X+Y+Z} \tag{3.2}$$

and

$$z = \frac{X}{X+Y+Z} \tag{3.3}$$

From these equations it is noted that

$$x + y + z = 1 \tag{3.4}$$

The intensity element expresses colour sensation by using brightness as a subjective descriptor. The HSI model owes its usefulness to two important facts. First, by separating the intensity component from colour carrying information (hue and saturation), the HSI colour model is ideally suited for algorithmic image manipulation that is based on a colour description [49]. Second, the chromaticity component is related to the way in which human beings perceive colour [49]. The HSI model is ideally suited for skin segmentation, however hardware uses the RGB colour space to communicate with monitors. Since the HSI colour space is not natively used by hardware, the RGB colour space is converted to the HSI colour space.

3.1.5 Conversion from RGB to HSI

The RGB model can be defined with respect to a cube as shown in Figure 3.3. The attributes of the HSI model are defined with respect to a diamond shown in Figure 3.4. To ascertain how hue can be determined from a given set of RGB values, consider Figure

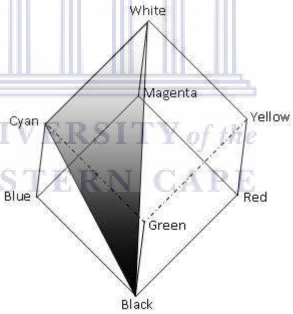


FIGURE 3.3: Schematic of the RGB colour cube.

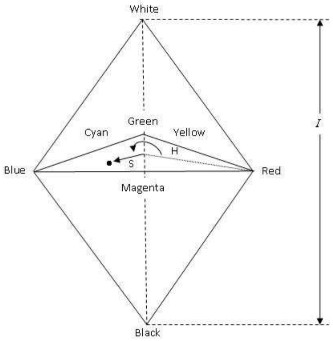


FIGURE 3.4: Schematic of the HSI colour triangle.

3.3, which shows a plane defined by three points(black, white, and cyan). The black and white points on the plane map onto the HSI intensity attribute on Figure 3.4. Furthermore, we note that all points contained in the plane segment, defined by the intensity axis and the boundaries of the cube, have the same hue (cyan). It can be concluded that all colours generated by RGB are contained in the HSI colour triangle which is defined by those colours. Given an image in RGB colour space, the HSI equivalent is obtained using the following formulas. Hue is calculated by: with

$$h = \begin{cases} \theta & \text{if} B \leq G \\ 360 - \theta & \text{if} B > G \end{cases} \tag{3.5}$$

$$\theta = \cos^{-1}\left\{ \frac{\frac{1}{2}[(R-G)+(R-B)]}{[(R-G)^2+(R-B)(G-B)]^{\frac{1}{2}}} \right\} \tag{3.6}$$

The saturation component is given by:

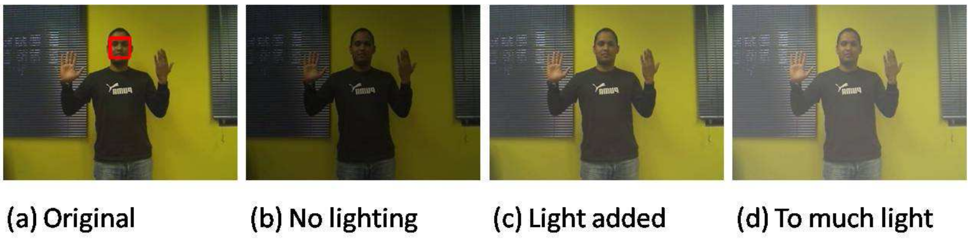
$$s = 1 - \frac{3}{(R+G+B)}[\min(R,G,B)] \tag{3.7}$$

The intensity component is given by:

$$i = \frac{1}{3}(R+G+B) \tag{3.8}$$

3.1.6 Comparing the RGB and HSI Colour Space

The RGB colour space on its own is not sufficient for the classification of skin regions as it not only represents colour but luminance as well. Various skin tones reflect light differently due to varying ambient lighting conditions. These variations make it difficult to classify, let alone different skin types accurately, using a pixel-based approach in the RGB colour space.



(a) Original (b) No lighting (c) Light added (d) To much light

Figure 3.5: Images captured under different lighting conditions.

3.1.6.1 Skin Segmentation With RGB

Figure 3.5(a) was captured in adequate lighting conditions, 3.5(b,c,d) were captured under varying light conditions. Skin values for the signer are calculated by averaging the RGB values of the face region in image 3.5(a). The region used for extracting the skin values is indicated with the red square. The RGB values for the skin of this signer is calculated to be in the range: $R : 65 - 135, G : 30 - 100, B : 0 - 60$. Using this information an algorithm is applied to 3.5(a,b,c,d) to obtain the skinmap. The red, green and blue values for the skinmap are computed as:

$$\text{isSkin} = \text{red} \geq 65 \text{ and red} \leq 135 \text{ and green} \geq 30 \text{ and green} \leq 100 \text{ and blue} \leq 60$$

The resultant skin maps using the RGB colour space are shown in Figure 3.6. When the algorithm is applied to image 3.5(a), from where the skin values was extracted, it yields the best results, show in Figure 3.6(a). However the lighter region of the palm is not correctly classified. In the darker image 3.5(b) the intensity value of the RGB colour space decreases, diminishing the overall accuracy, shown in Figure 3.6(b). When lighting is increased slightly, shown in Figure 3.5(c), more background noise is introduced. This skinmap produced a marginally higher false positive detection rate, but is comparable to 3.6(a). However when lighting is drastically increased in Figure 3.5(d), no skin regions are detected, as no pixel values fall within the predetermined ranges. This experiment confirms that the RGB colour space and pixel based classification methods are not reliable for identifying skin coloured pixels in varying lighting conditions.

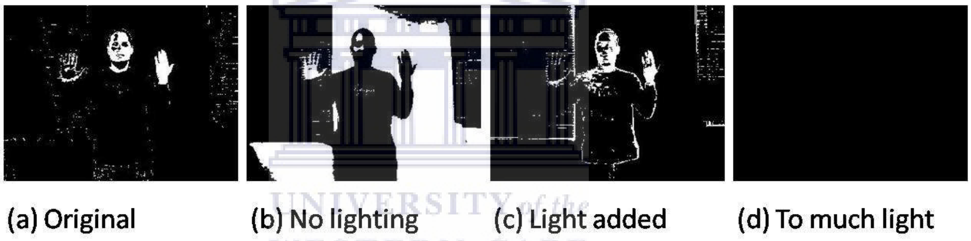


FIGURE 3.6: Skin maps produced in the RGB colour space.

3.1.6.2 Skin Segmentation With HSI

Chromatic colours are more reliable and are derived by separating luminance from the colour information. The HSI colour space owes its usefulness to this very fact. By separating the luminance, the HSI colour space is reliable when classifying the same skin coloured pixels across varying lighting conditions.

The images in Figure 3.5 are converted into the HSI colour space by using the formulas 3.5,3.6,3.7 and 3.8 respectively. The skin (h, s, i) values are extracted from inside the red square in Figure 3.5(a). The HS values for the skin of this signer is calculated to be in the range: $H : 0 - 0.8, S : 0.2 - 0.8$. Using this information an algorithm is then applied to Figures 3.5(a,b,c,d). The algorithm is computed as:

$$\text{h} \geq 0 \text{ and h} \leq 0.8 \text{ and s} \geq 0.2 \text{ and s} \leq 0.8$$

Lighting variation needs to be ignored as we want to classify skin under different light conditions. We therefore exclude the i value, as it contains the intensity value of each pixel. The skinmap results that are computed in the HSI colour space, are shown in Figure 3.7. The results show that the algorithm classifies skin colour accurately across all images with varying light intensities. All skin maps in Figure 3.7 indicate that very little or no loss of accuracy was produced by the algorithm in the HSI colour space. We choose to work in the HSI colour space as this produces better results than the RGB colour space. We therefore covert all images to the HSI colour space before attempting to segment them.

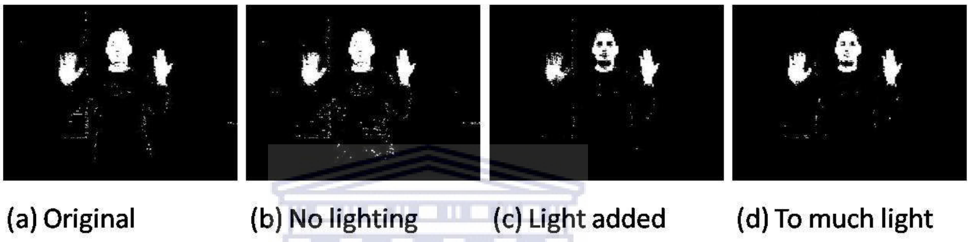


FIGURE 3.7: Skin maps produced in the HSI colour space.

3.2 Background Subtraction

Background subtraction involves extracting information of the background in order to distinguish between the background and foreground. This eliminates the noise produced by background objects. There are two well known approaches: adaptive and non-adaptive background modelling [50]. We look at these two approaches to determine which one is optimally suited for our implementation of a GR system. Adaptive modelling involves some sort of learning process in which the algorithm tries to learn what the background is over a series of frames, so that the background can be eliminated. Non-adaptive approaches subtract the current frame from a reference frame, or a previous frame. The resultant image contains only the pixels that have changed between the two subtracted frames, showing where motion has taken place or where pixel colour has changed.

3.2.1 Adaptive Modelling

A real-time adaptive background mixture model was proposed by Stauffer and Grimson [51]. Each pixel is modelled as a mixture of Gaussian distributions. Every pixel's value in a series of images is tracked over time in what is know as the *pixel process.* The probability of observing the current pixel's value t is given by:

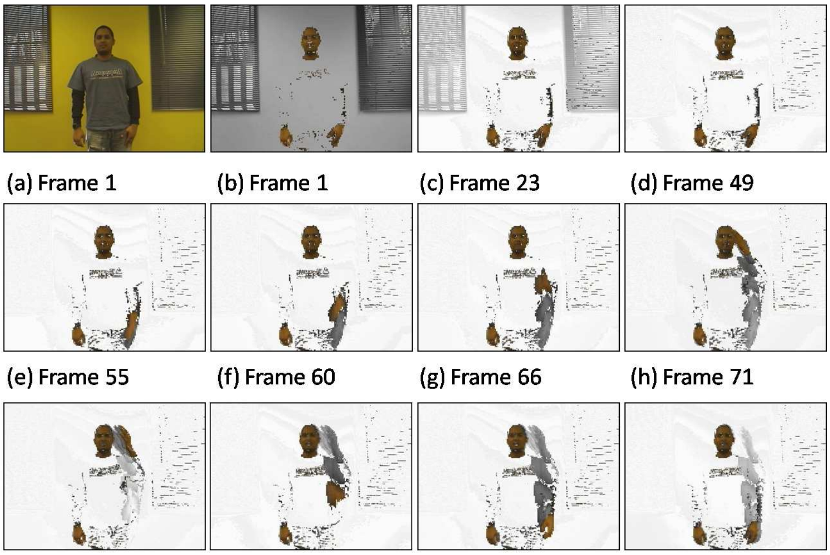
$$P(X_t) = \sum_{i=1}^{K} \omega_{i,t} \eta(X_t, \mu_{i,t}, \Sigma_{i,t}) \tag{3.9}$$

The recent history of each pixel is modelled by a mixture of K Gaussian distributions. Where K is the number of distributions, $\omega_{i,t}$ represents the estimate of the weight of the i^th Gaussian in the mixture and $\mu_{i,t}$ is the average Gaussian at time t. Every new pixel obtained in the next sequential image of the video is validated against the existing K Gaussian distributions. When a match is found, using a threshold, it is considered to be part of the background.

This adaptive modelling method was modified to work with our GR system. When background modelling is applied to an incoming image, all skin regions are ignored. This is to ensure that important static regions, such as the face, are not computed in the Gaussian distributions. This guarantees that all skin regions stay in the foreground. An example of this is shown in Figure 3.8(b). Figure 3.8 exhibits a series of frames depicting a signer, signing the SASL word *Hello.* The video consists of a total of 129 frames. Figure 3.8(a) shows the signer with the background included. When the adaptive model classifies a background pixel, the system grey-scales it according to how recently it has been recognised as part of the background. Dark regions indicate a recent classification and lighter regions indicate older classifications. The background takes between 10 and 20 frames to stabilise, leaving the skin regions behind, shown in Figure 3.8(d). The adaptive background mixture model has been successfully applied to this sequence of frames and accurately recognises and subtracts the background, over a period of time, as shown in the series of images in Figure 3.8.

3.2.2 Non-Adaptive Modelling

For our non-adaptive modelling approach, we build on Kirillov's simple method of frame differencing [31]. In comparison to his method however, we assume that we have no prior



(a) Frame 1 (b) Frame 1 (c) Frame 23 (d) Frame 49

(e) Frame 55 (f) Frame 60 (g) Frame 66 (h) Frame 71

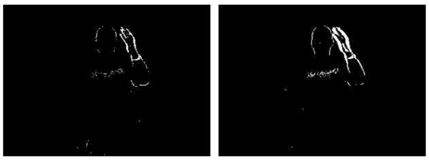
(i) Frame 83 (j) Frame 94 (k) Frame 104 (l) Frame 129

FIGURE 3.8: Background subtraction using an adaptive background mixture model.

knowledge of the background. For this reason our method needs to construct an image containing background information. This new image is used as the reference frame for the frame differencing method.

We realise that the entire background information is not needed. We limit the background only to where motion takes place. Areas that do not include motion, other than the head, carry no meaning, are redundant and can be ignored. This simple step eliminates all of the background information that is not needed and greatly reduces noise created by background objects.

To build the reference frame, the system first converts all images to their equivalent grey scale. The system then frame differences all successive frames in the video. The same series of images used in the adaptive model approach is applied here, in order to compare the methods. Example binary images produce by the differencing method are shown in Figure 3.9. The white segments in the image indicate where pixel values have changed, thus showing where movement has taken place. Black indicates no pixel value change or no movement. To build the background image, the system evaluates each binary



(a) Frame 83 (b) Frame 84

FIGURE 3.9: Images produced by frame differencing.

image and compiles a new image, of all motion in the video, shown in 3.10(a). The system subsequently extract all the white regions, where movement has taken place, from the original series of colour images. This process produces the reference image shown in Figure 3.10(b). The new background image contains skin colour information that should be in the foreground. The system therefore extracts all skin regions from the background image producing a new image, shown in Figure 3.10(c). The system now frame differences the original set of images with the reference image by taking into account only to difference the current frame's pixels that correspond to the position of the non-black pixels in the reference image. Once again this produces a set of binary images. The binary images are evaluated by analysing the white areas in the differenced image, and extracting skin colour pixels from the original set of colour images. The non-adaptive approach has been applied successfully to the same set of images used in the adaptive approach, shown in Figure 3.11. The background is limited to where only movement has taken place to ensure minimal background noise. This produces a *clean* set of skin coloured segmented images shown in Figure 3.11(b-l).

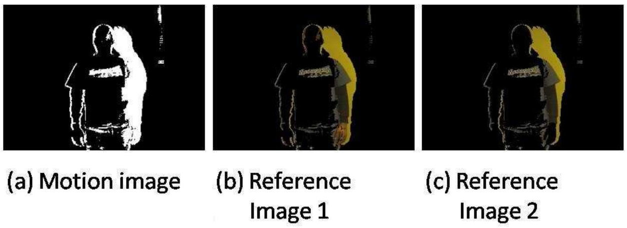

(a) Motion image (b) Reference Image 1 (c) Reference Image 2

Figure 3.10: Background images.

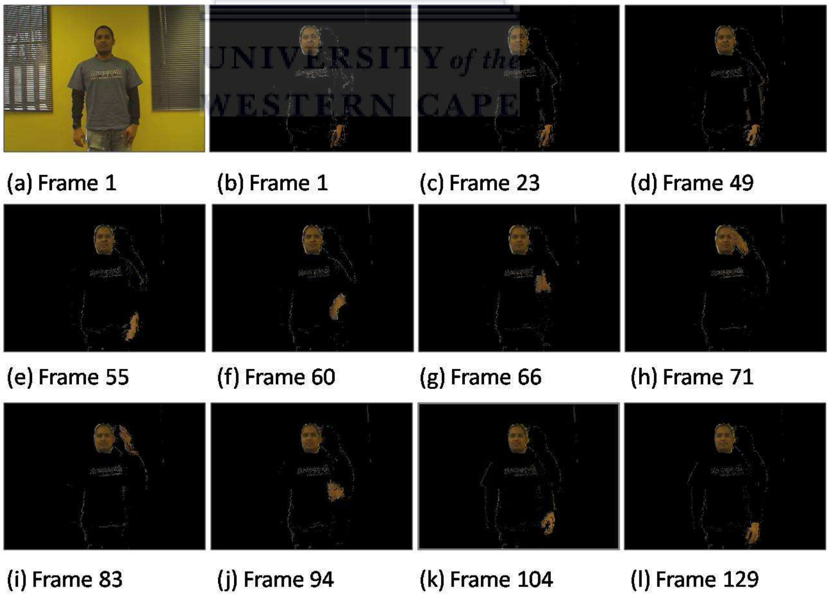

(a) Frame 1 (b) Frame 1 (c) Frame 23 (d) Frame 49
(e) Frame 55 (f) Frame 60 (g) Frame 66 (h) Frame 71
(i) Frame 83 (j) Frame 94 (k) Frame 104 (l) Frame 129

Figure 3.11: Background subtraction using an adaptive background mixture model.

3.2.3 A Comparison of Adaptive and Non-Adaptive Modelling

Both adaptive and non-adaptive methods produced promising results. Both methods generate similar results when comparing the segmented images produced. In addition both methods do not require any prior knowledge of the background. In the example above, the non-adaptive model eliminates an average of 90.97% of the background in all 129 frames, effectively eliminating most of the background noise in this video. Skin regions can be extracted from the first frame, as no time is needed for the background to stabilize. For this method to fail two criteria need to be satisfied: first, a background object needs to be the same colour as the colour of the signer's skin and second, the same object must move. The adaptive mixture model needs time for the background to be stabilised. For the adaptive model to fail, any of the following two criteria must be met: first, a background object is the same colour as the signer's skin and second, a background object must move. For the purpose of our research, a quick and reliable way of eliminating background noise is needed. Therefore the non-adaptive approach proves to be a more suitable means for our purpose.

3.3 Normalising Method

In this section we explain methods and techniques used to normalise input for classification across different signers. The objective is to minimise variations that can occur in multiple videos of signs, using multiple signers. Some variations that may adversely affect the performance of the system are:

- Variations in body dimensions,
- Variations in the distance of the signer from the camera, and
- Variations in the position of the signer within the video frame.

The normalising method is divided into several steps. We designed these steps with the above variations in mind. The normalise methods perform the following steps sequentially:

- Find Head. This method finds the location and the dimensions of the signers head,
- Centring. This method centres the signer in the video frame, using the location of the head found in the first step, and

- Construct Grid. This method constructs a grid around the signer's head. The grid size is proportioned to the signer's head as the signer's head dimensions, found in the first step, is used to construct the grid.

Each of these steps is applied to every frame extracted from the video, before attempting to classify the sign.

3.3.1 Find Head Method

The Find Head Method is important for three reasons. First, we need to determine if a signer is present in the video. Second, the coordinates of the signer's position within the video frame is needed in the Centring Method. Third, the dimensions of the signer's head are essential in the Construct Grid Method.

To find the head in a given frame, the method described by Viola and Jones (Viola-Jones method) is used [52, 53]. The OpenCV library's implementation of the Viola Jones method is used by our system [54–56]. The Viola-Jones face detector method uses a Haar Cascade classifier. Viola and Jones combined four fundamental concepts in the face detector method:

- Haar features,
- An integral image for rapid feature detection,
- The AdaBoost machine-learning method, and
- A cascade classifier to combine many features.

The method is based on Haar wavelets. Haar wavelets are single wavelength square waves, i.e one high interval and one low interval. An example of these Haar wavelets used in the original Viola-Jones cascade is shown in Figure 3.12. A square wavelet is a pair of adjacent rectangles, one light and the other dark. The Haar feature is determined by differencing the average dark region pixel value from the average light region pixel

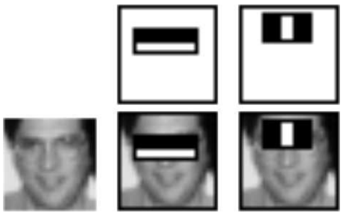


FIGURE 3.12: The first two Haar features in the original Viola-Jones cascade [53].

value. Using a threshold determined in the training phase, the resulting value determines the presence or absence of a Haar feature. This produces many Haar features at every image location over multiple scales. To find all the Haar features in an image efficiently, Viola and Jones implemented an integral image technique. The integral value for each pixel is the sum of all the pixels to its left and above it. The entire image can be integrated with a few operations per pixel.

For the training method Viola and Jones used a machine learning method called AdaBoost. The idea behind AdaBoost is to combine many *weak* classifiers to create a *strong* one. A *weak* classifier is defined as a classifier that correctly recognises a feature more often than random guessing would. By combining many weak classifiers, and each classifier ultimately *boosting* the final outcome, a strong classifier is built. AdaBoost first selects a set of weak classifiers to combine, then each classifier is weighted. The resultant strong classifier consists of a weighted combination of classifiers.

Viola and Jones used an efficient filter chain, shown in Figure 3.13, to classify image regions. Each filter consists of one Haar feature. If a region at any moment fails to pass through the cascade, it is classified as non-face, and the next region is applied to the cascade. The order of filters is determined by the AdaBoost assigned weights. The heavier weighted filters are assigned first in the cascade, in order to eliminate regions as quickly as possible. If a region passes through all of the filters without failing, it is considered to contain a face region. The OpenCV implementation of the Viola-Jones face detector method comes trained with a frontal face pose classifier. We apply the Viola-Jones method to all images in the video sequence. An example of the face detector output is shown in Figure 3.14. Any frames that do not contain a face are discarded. Once the face region is identified, the region is grey-scaled and the Marr/Hildreth and

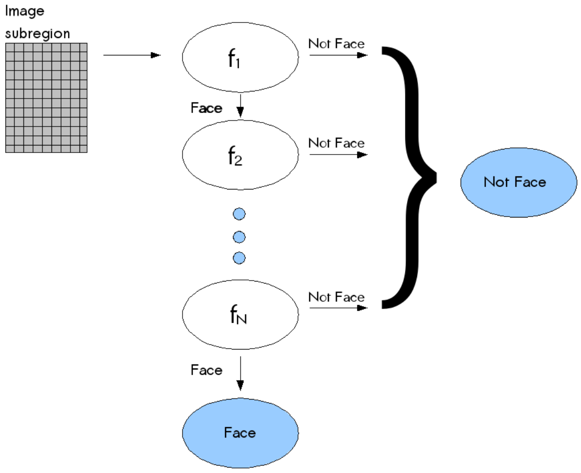


FIGURE 3.13: The classifier cascade is a chain of single feature filters that determines if a region is a face or not.

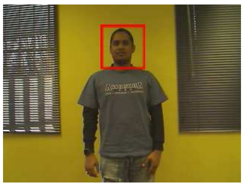


Figure 3.14: A correctly classified face using the Viola-Jones face detector method.

Canny edge detection algorithm is applied to find the outline of the face [57]. The edge detection method applies a 7 × 7 convolution mask to every pixel and analyses each surrounding pixel's intensity. A sharp change in intensity, most likely indicates an edge. The entries of the mask are calculated using a variation of the Gaussian equation show in Equation 3.10.

$$\bigtriangledown^2 g(x,y) = \frac{1}{\sigma^2}(2-(\frac{x^2+y^2}{\sigma^2})(e^{-\frac{x^2+y^2}{\sigma^2}})) \tag{3.10}$$

An example of an image after the edge detection algorithm has been applied is shown in Figure 3.15. The dimensions of the head (in pixels) are measured by our system. The height and width dimensions of the head in 3.15 are measured to be 84 and 63 pixels respectively, with the x, y coordinates of the center of the head being $x = 158, y = 75$. These dimensions are forwarded to the Construct Grid method for further use. The coordinates of the detected face are sent to the Centring method to centre the signer in the frame. In addition, the system averages the colour of the original image within the edge detected regions, producing an average skin model of the signer. This skin information is sent to the Feature Extraction method.

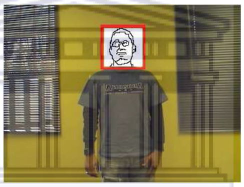


Figure 3.15: Marr/Hildreth and Canny edge detection applied to the face region classified by Viola-Jones face detector.

3.3.2 Centring Method

The Centring method uses the coordinates of the location of the head found in the Find Head method. Predefined x and y coordinates for a 320×240 image are chosen to centre signers in a video frame, with $x = 160$ and $y = 60$. Each frame in the video is analysed and adjusted, shifting the signer left/right or up/down, aligning the centre of the head on the predefined set of coordinates. With the example frame shown in Figure 3.14, the system shifts the signer right 2 pixels and up 15 pixels, centring the signer in the frame as shown in Figure 3.16. The extra pixels are filled in red in order to illustrate, in this example, the right and upward shift of the signer. The Centring method is applied to all frames, thus normalising the location of different signers in various locations within a frame. Since this method is applied to each frame, it not only normalises the position of the signer but also stabilises the vertical and horizontal movement of the video camera. This allows the video camera to move while a video of a sign is recorded. This feature is invaluable when recording a video on an unstable platform, such as using a cellphone or hand held camera. Once every frame is adjusted by this method, they are sent to the Construct Grid method.

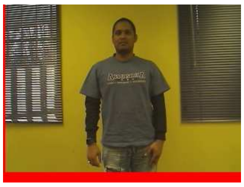


Figure 3.16: The is signer shifted up and to the right, to a predefined location on the frame.

3.3.3 Construct Grid Method

Our system needs to normalise different body sizes in order to achieve similar results when signing the same sign across signers with various body types. To this end we adapt a technique, used by artists throughout history to draw or sculpt the human body accurately, to work with our system. The artistic technique of using body proportions to draw the human form is best illustrated by Leonardo da Vinci famous drawing of the Vitruvian Man around 1487, shown Figure 3.17 [58]. The drawing depicts a man

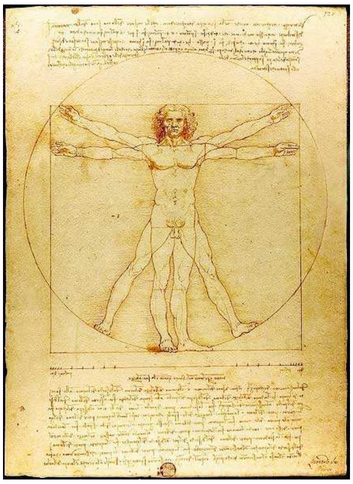


Figure 3.17: The Vitruvian Man drawn by Leonardo da Vinci to illustrate body proportions. [59]

superimposed in a circle and square and is also referred to as the *Canon of Proportions* or *Proportions of a Man.* The drawing is based on the correlations of ideal human body geometrical proportions described by an ancient Roman architect Marcus Vitruvius Pollio [59]. Vitruvius's works was based on the famous Greek philosopher Pythagoras, who concluded that the human body itself is a construct of both the square and circle [60]. Da Vinci merged science with art to prove the existence of proportions in nature. According to Da Vinci [61], the Vitruvian Man was a study on human body proportions as described by Vitruvius to be:

- a palm is the width of four fingers,
- a foot is the width of four palms,
- a cubit is the width of six palms,
- a pace is four cubits,
- a man's height is four cubits (and thus 24 palms),
- the length of a man's outspread arms (arm span) is equal to his height,
- the distance from the hairline to the bottom of the chin is one-tenth of a man's height,
- the distance from the top of the head to the bottom of the chin is one-eighth of a man's height,
- the distance from the bottom of the neck to the hairline is one-sixth of a man's height,
- the maximum width of the shoulders is a quarter of a man's height,
- the distance from the middle of the chest to the top of the head is a quarter of a man's height,
- the distance from the elbow to the tip of the hand is a quarter of a man's height,
- the distance from the elbow to the armpit is one-eighth of a man's height,
- the length of the hand is one-tenth of a man's height,
- the distance from the bottom of the chin to the nose is one-third of the length of the head,
- the distance from the hairline to the eyebrows is one-third of the length of the face,

- the length of the ear is one-third of the length of the face, and
- the length of a man's foot is one-sixth of his height.

Even though this does not prove true for everybody, it gives a good approximation of the human body. For the purpose of our study, which focuses on GR in sign language, we concentrate on the body regions above the waist. Using Da Vinci's work, we can now infer the general location of key body regions in videos of various signers.

The Construct Grid method uses the head dimensions obtained from the Find Head method. A grid is constructed around the signers head, shown in Figure 3.18 in red. The dimensions are then halved and an additional grid is added to the image, to increase the resolution, shown in Figure 3.18 in blue. Once the grid is constructed, we can infer the location of regions of interests in the image. Figure 3.18 illustrates this: Shoulder regions are indicated in green; chest regions are indicated in blue, and stomach regions are indicated in yellow. Using the adapted body proportion techniques, the system is able to infer key body regions accurately from the head dimensions. This allows the signer to stand at various distances from the camera. Due to the system using the head dimensions of the signer, the grid is in proportion to the signer's body.

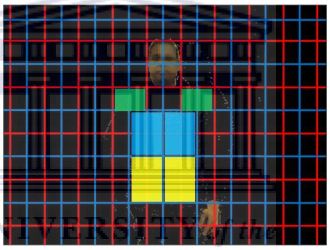
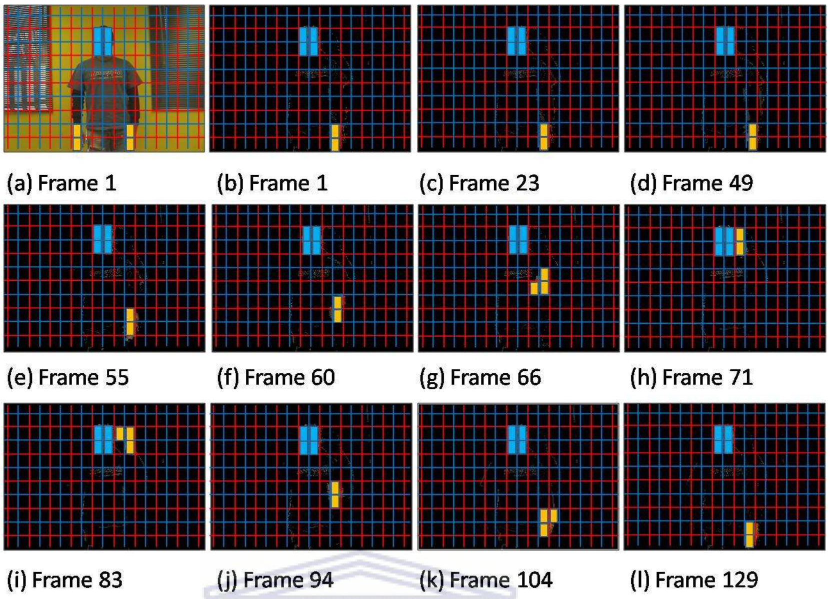


FIGURE 3.18: Image with grid superimposed on the signer.

3.3.4 Feature Extraction

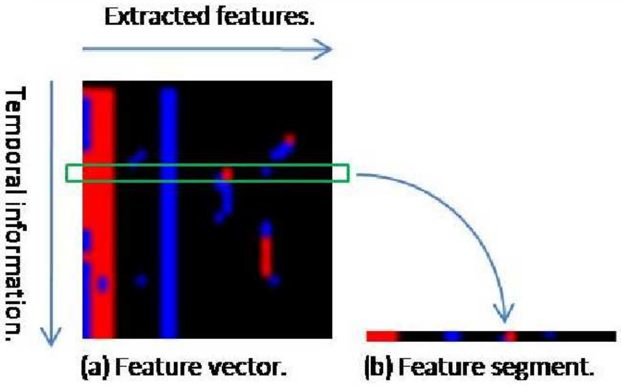
The Feature Extraction method uses the grid which was constructed in the previous method. Each frame in the video is analysed with the grid superimposed on it. For each frame, the system computes a skinmap for each block in the grid using the skin information obtained in the Find Head method. A block is said to be fired if 40% or more of its pixels are skin values. Figure 3.19 shows the grid applied to a video of a signer signing the word *Hello* in SASL. The blue blocks indicate the head regions and the orange regions indicate the hand regions. Figure 3.19(a) shows the Grid method applied to the original image without the use of background modelling. Both the left and right hand regions are identified by the system, even though the right hand is not used in the *Hello* sign. Figure 3.19(b-l) shows the Grid method applied to the non-adaptive background subtracted image. Only the left hand is identified as movement only occurs with this hand. For each frame, the fired blocks (indicating skin regions) are recorded and compiled in sequential order into a feature vector, shown in Figure 3.20(a). The y axis depicts temporal information, while the x axis depicts locations of blocks of the grid in sign space. The red colour shows skin regions identified with 40% or more skin coloured pixels within it, while blue indicates skin regions with less than 40% skin coloured pixels. Figure 3.20(b) illustrates that segments in the feature vector represent one frame in the video. It can be said that the feature vector contains movement information pertaining to the hand in frames over a period of time.


(a) Frame 1 (b) Frame 1 (c) Frame 23 (d) Frame 49

(e) Frame 55 (f) Frame 60 (g) Frame 66 (h) Frame 71

(i) Frame 83 (j) Frame 94 (k) Frame 104 (l) Frame 129

Figure 3.19: Grid superimposed on frames, showing blocks that have been triggered by skin regions.



(a) Feature vector. (b) Feature segment.

Figure 3.20: Feature vector produced by our system.

3.4 Summary and Conclusions

In the preceding sections we have first described skin segmentation. We evaluated several colour spaces to work in, and compared two well known coordinate colour systems, HSI and RGB. Due to the separation of the light information from the colour information in the HSI colour space, we chose to work in the HSI colour space.

Two methods for background subtraction were implemented and compared. Both methods assumed no prior knowledge of any background information. Adaptive background modelling uses machine learning techniques to classify the background in an image. However, several frames were required for the algorithm to classify the background. This would mean the signer would have to stand stationary for several frames before commencing with the sign. The non-adaptive modelling approach, which used several background subtraction techniques, proved to be the better option for our system. This non-adaptive method classifies the background regions where no movement takes place, effectively reducing the overall amount of background noise in a frame. In addition, this method does not require the signer to stand stationary at the beginning of the sign, increasing the effectiveness of this method.

We then developed three methods to normalise information pertaining to signs across different signers. We developed a novel approach by adapting body proportion techniques used by artists. These methods allow the signer not to be restricted by signing in a specific position and distance from the camera. In addition, the system caters for signers of various body types.

Lastly we discussed a method for extracting hand movement information from a series of frames in a video. This information is then stacked into a feature vector. The feature vector is used to classify the sign and is further discussed in Chapter 5.

Chapter 4

Hidden Markov Models

For the sake of completeness we look at a statistical model of Markov source also known as hidden Markov modelling, named after Russian mathematician Andrey Markov. It was studied and introduced in the late 1960s and has become increasingly popular. According to L. R. Rabiner, this is due to two important reasons [62]. First, the models are based on mathematical structures and therefore can form the theoretical basis for use in a wide range of applications. Second, Markov models when implemented properly, work extremely well in practice.

4.1 Markov Process

When evaluating real-world processes, we are often interested in finding patterns in the signal produced over time, by these processes. The signals can either be discrete in nature, i.e. letters in an alphabet, or continuous, i.e. speech samples or temperature readings. These patterns help us to understand what is currently happening, but can also help us understand what may happen in the future. Some examples of such patterns that occur include: fluctuation of stock prices, birth and mortality rates, interest rates, sequences of phonemes in spoken words, foreign exchange rates, etc. Such patterns may be generated deterministically or non-deterministically.

To solve a non-deterministic pattern recognition problem such as weather prediction, where there are 3 observable states:

- State 1: Rain,
- State 2: Cloudy, and
- State 3: Sunny.

It is often useful to assume that each state is independent and all states are only influenced by the values of the state that directly precedes it. This assumption is known as the Markov assumption and greatly simplifies the complexity of planning a sequence of actions. It allows for the elimination of possible states once they have left a specified time window because it is assumed that the state no longer has any effect on the current state.

In the weather example, the Markov model would look at a combination of the 3 states in a long sequence, and analyse the likelihood that one kind of weather is proceeded by another. Let's say it was found that 25% of the time, a sunny day was followed by a cloudy day, and 75% of the time a rainy day was followed by another rainy day. In addition sunny days were followed by a rainy day 50% of the time, and a rainy day was followed by a cloudy day 25% of the time by a cloudy day. Given this analysis, we can generate a new sequence of states that yields statistically similar weather events to that which we observed. To generate a new sequence, first start with today's weather. Second, given today's weather, choose a random number to pick tomorrow's weather. Third, make tomorrow's weather today's weather and repeat from the second step. This would produce sequence of states such as:
$\{Sunny, Cloudy, Rain, Rain, Sunny, Cloudy, Rain, Rain, Sunny, Sunny....\}$
It can be said that the output chain or sequence would reflect statistically the transition probabilities derived from the weather we observed.

A Markov process transitions from state to state depending solely on the information obtained from the previous n states. This process is called an order n Markov model, where n is the number of states effecting the current state's decision to move to the next state. In the non-deterministic weather example where the current state only depends on the previous state, the simple Markov process is known as a first-order process.

In contrast to the non-deterministic pattern recognition example, a deterministic system's, transitions between states are performed probabilistically. For a first order Markov process with M states, there are M^2 possible transitions between states, since all states are connected to every state. Associated with each state is a *state transition probability*, this is a the probability, given state n, that state n will move to the next state. The transition probabilities are stored in the *state transition matrix.* A first-order Markov process can be defined as:

- States: Number of observed states in the model,
- π: The probability of the system being in each of the states at the time of initialisation, and
- State transition matrix: The probability of moving from one state to the another.

Any system that can be described in this manner is a Markov process.

4.2 Hidden Markov Model

A HMM, denoted by λ, is a double stochastic process [63]. The first stochastic layer is the underlying first-order Markov process. The second stochastic layer of the HMM is the set of output probabilities for each state. Given a sequence of observations, the actual states are ambiguous, in other words it is *hidden* from the observer. For example, we apply an HMM to a non-trivial problem in speech recognition. The audible sound that we hear when someone speaks is a product of air passing through the vocal chords, shape of the mouth, position of the tongue and several other factors. We can observe some processes in this example, such as the actual utterance or the shape of the mouth. These observed output sequences can be related probabilistically to an underlying Markov process. This underlying Markov process can be non-observable and in our example will include the state of the vocal chord, position of the tongue and force of the air passing through the vocal chords. Therefore, for the recognition of any vocal utterance, a model consists of a number of observed and non-observed states.

A formal definition of HMMs consists of the following elements:

1. N is the number of states in the model,

2. M is the number of distinct observation symbols per state,

3. A is the state probability distribution. $A = \{a_{ij}\}$ where $a_{ij} = P[q_{t+1} = j|q_t = i], 1 \leq i, j \leq N$ and q_t denotes the current state.
   The transition probabilities must satisfy the normal stochastic constraints $a_{ij} \geq 0,\ 1 \leq i,\ j \leq N$ and $\sum_{j=1}^{N} a_{ij} = 1,\ 1 \leq i \leq N$,

4. B is the observation symbol probability distribution in state j, $B = \{b_j(k)\}$, where $b_j(k) = P[\mathrm{v}_k \text{ at } t|q_t = S_j]$
   The observation symbol probabilities must satisfy the following stochastic constraints $1 \leq j \leq N$ and $1 \leq k \leq M$, and

5. π is the initial state distribution. $\pi = \{\pi_i\}$ where $\pi_i = p\{q_1 = i\},\ 1 \leq i \leq N$.

We can now use the compact notation, shown in Equation 4.1 to denote HMMs with discrete probabilities.

$$\lambda = (A, B, \pi) \tag{4.1}$$

4.2.1 The Three Basic Problems for HMMs

Given the HMM in the previous section, there are three fundamental problems that must be solved for the model to be useful in real-world applications, as noted by L. R Rabiner [62]. These problems are summarised as follows:

1. Given the observation sequence $O = O_1O_2...O_T$, and a model $\lambda = (A, B, \pi)$, how do we compute $P(O|\lambda)$ the probability of the observation sequence efficiently.
2. Given the observation sequence $O = O_1O_2...O_T$, and a model λ where T it the number of observations, how do we choose a corresponding state sequence $Q = q_1q_2...q_T$ which bests explains the observations?
3. How do we adjust the model parameters $\lambda = (A, B, \pi)$ to maximize $P(O|\lambda)$?

4.2.2 A Solution to the Evaluation Problem

The evaluation problem deals with how well a model matches a given observation. This is extremely useful when trying to choose which model out performs other models [62]. Given an observation sequence $O = O_1O_2...O_T$ and model $\lambda = (A, B, \pi)$, we need to calculate $P(O|\lambda)$. Using probabilistic reasoning, it is realised that the number of operations needed to compute $(O|\lambda)$, is of the order of N^T. This proves exponentially inefficient even for moderate values of T and N.

An alternative method found in the *forward-backward* algorithm, with considerably lower time complexity, can be used. The *forward-backward* algorithm uses two auxiliary variables, namely the forward and backward variables. Only the forward section of the algorithm is used in the solution of the evaluation problem. The backward variable is used in the solution to the learning problem. The forward variable, $\alpha_t(i)$, is defined as the probability of the partial observation sequence $O = O_1O_2...O_T$ until termination at state S_i at time t. $\alpha_t(i)$ is defined in equation 4.2.

$$\alpha_t(i) = P(O_1O_2...O_t, q_t = S_i|\lambda) \tag{4.2}$$

$\alpha_t(i)$ can be solved by induction, using the following equations.

1. Initialisation:

$$\alpha_t(i) = \pi_i b_i(O_1),\ 1 \leq i \leq N. \tag{4.3}$$

2. Recursion:

$$\alpha_{t+1}(j) = \Big[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\Big] b_j(O_{t+1}),\ 1 \leq t \leq T-1,\ 1 \leq j \leq N. \tag{4.4}$$

3. Termination:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i). \tag{4.5}$$

Similar to $\alpha_t(i)$, the backward variable, defined by $\beta_t(i)$, is the probability of the partial observation sequence $O = O_1 O_2 ... O_T$ from time $t+1$ until termination at state S_i at time T. The time complexity for both the forward and backward section of this method is proportional to $N^2 T$, which is linear with respect to T. This is considerably shorter compared to the initial time complexity of N^T.

4.2.3 A Solution to the Decoding Problem

The solution to the decoding problem aims to find the optimal sequence of states for a given sequence of observations $O = O_1 O_2 ... O_T$. A problem arises in some solutions where invalid state sequences are produced. In other words, given a HMM with some states having a probability of 0 and given observations $O = O_1 O_2 ... O_T$, the solution may result in a meaningless set of state sequences. This is due to the fact that this solutions simply determines the most likely state at every instant, without regard to the probability of occurrence of sequence of states [62].

The *Viterbi* algorithm does not suffer from the above mentioned complication. The *Viterbi* algorithm was created by Andrew Viterbi in 1967 for error correction in noisy communication links [64]. The algorithm tries to find the entire state sequence which maximises the probability of observing each state. It dynamically finds the most *correct* sequence of states given an observation, this sequence is also known as the *Viterbi path*. To find the *Viterbi path* or the most *correct* sequence of states $Q = q_1, q_2 ... q_r$, given observations $O = O_1 O_2 ... O_T$, an auxiliary variable $\delta_t(i)$ is defined in Equation 4.6.

$$\delta_t(i) = \max_{q_1, q_2, \ldots, q_{t-1}} P[q_1 q_2 ... q_t = i,\ O_1 O_2 ... O_t | \lambda] \tag{4.6}$$

$\delta_t(i)$ is the *Viterbi path* with the highest probability of observing the first t observations, until termination at state S_i and time t. Given $\delta_t(i)$, the following recursive relationship holds:

$$\delta_{t+1}(j) = \big[\max_{1 \leq i \leq N} \delta_t(i) a_{ij}\big],\ 1 \leq N,\ 1 \leq t \leq T-1 \tag{4.7}$$

where

$$\delta_1(i) = \pi_i b_i(O_1),\ 1 \leq i \leq N. \tag{4.8}$$

In order to retrieve the *Viterbi path*, the algorithm needs to keep track of each state that was computed to have the highest probability in equation 4.7. This is done by using an array $\psi_t(j)$. The whole procedure can now be solved as:

1. Initialisation:

$$\delta_1(i) = \pi_i b_i(O_1),\ 1 \leq i \leq N, \tag{4.9}$$

$$\psi_1(i) = 0. \tag{4.10}$$

2. Recursion:

$$\delta_t(j) = \max_{1 \leq i \leq N} \big[\delta_{t-1}(i) a_{ij}\big] b_j(O_t), 2 \leq t \leq T, 1 \leq j \leq N, \tag{4.11}$$

$$\psi_t(j) = \text{argmax}_{1 \leq i \leq N} \big[\delta_{t-1}(i) a_{ij}\big], 2 \leq t \leq T, 1 \leq j \leq N. \tag{4.12}$$

3. Termination:

$$P^* = \max_{1 \leq i \leq N} \big[\delta_T(i)\big], \tag{4.13}$$

$$q_t^* = \text{argmax}_{1 \leq i \leq N} \big[\delta_T(i)\big]. \tag{4.14}$$

4. Path backtracking:

$$q_t^* = \psi_{t+1}(q_{t+1}^*), t = T-1, T-2, \cdot\cdot\cdot, 1. \tag{4.15}$$

For our application, this formulation of the Viterbi algorithm can lead to underflow as the probabilities can become very small. Thus, we used logarithmic scaling in order to prevent an underflow from occurring. The equations for the Viterbi algorithm thus become:

1. Initialisation:

$$\delta_1(i) = \log(\pi_i) + \log(b_i(O_1)). \tag{4.16}$$

2. Recursion:

$$\delta_t(j) = \max_{1 \leq i \leq N} \big[\delta_{t-1}(i) \log(a_{ij})\big] + \log(b_j(O_t)). \tag{4.17}$$

3. Termination:

$$\log(P^*) = \max_{1 \leq i \leq N} \big[\phi_T(i)\big]. \tag{4.18}$$

The other equation remains unchanged.

4.2.4 A Solution to the Learning Problem

Even though this solution is not used in our application, for completeness we briefly discuss it. The optimisation of the parameters in HMMs, so that it best describes a series of observations, can effect the performance of a model. The series of observations used to optimise the model, is known as the *training sequence.* The learning problem aims to find a method that can adjust the parameters of a model (A, B, π) optimally, given a set of training sequences. However, with a finite number of training sequences, there is no known way of estimating the model's parameters optimally [62].

To get around this dilemma, we can select $\lambda = (A, B, \pi)$ so that $P(O|\lambda)$ is iteratively locally maximised. This procedure is called the *Baum-Welch* method. It was created by the mathematicians, Leonard E. Baum and Lloyd R. Welch. The method is a generalised expectation maximisation method, in other words it is able to compute the transition and emission parameters of a model given a set of training sequences. In order to calculate the *Baum-Welch* algorithm we need to define $\xi_t(i,j)$ as the probability of being in state s_i at time t and state S_j at time $t+1$, given the HMM $\lambda = (A, B, \pi)$ and the training sequence:

$$\xi_t(i,j) = P(q_t = s_i, q_{t+1} = S_j|O, \lambda). \tag{4.19}$$

We define $\gamma_t(i)$ as the probability of being in state S_i at time t as:

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j) \tag{4.20}$$

with the re-estimation formula for (A, B, π) being:

$$\bar{\pi}_i = \text{ expected frequency in state } S_i \text{ at time } (t = 1) = \gamma_1(i), \tag{4.21}$$

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}, \tag{4.22}$$

and

$$\bar{b}_j(k) = \frac{\sum_{t=1, \text{ with } O_t = v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}. \tag{4.23}$$

The re-estimation formula is denoted by $\bar{\lambda} = (\bar{A}, \bar{B}, \bar{\pi})$. The $\bar{A}$ variable is the probability or expected number of times that a transition will be made from state i to state j or remain in state i . The $\bar{B}$ variable is the ratio of observing symbol o_k while being in state j. Lastly, the $\bar{\pi}$ variable is the probability of being in state i at time t

4.3 Summary

In this chapter we have discussed the HMM model. We have evaluated the 3 problems associated with HMMs, namely the evaluation problem, the decoding problem and the learning problem. We then discussed each of their solutions. In the next chapter we will discuss the implementation of HMMs in our GR system.

Chapter 5

Experimental Setup

In this chapter we discuss the physical setup for our experiments. We use two different equipment setups. Their performance is evaluated in the next chapter. The first equipment setup uses of a webcam attached to a PC to record a gesture. The second uses a camera on a cellphone to record a gesture. The system developed in conjunction with this study is called iSign. It consists of two applications: iSign desktop and iSign mobile. Although neither iSign desktop nor iSign mobile are used in their entirety in this study, for completeness, we will describe them briefly. We then look at the training procedure of the HMMs and describe how the feature vector is used, followed by a summary and conclusion.

5.1 iSign Desktop

Our first setup consists of a single webcam connected to a PC. The specifications of the system hardware is shown in Table 5.1. The system captures fifteen frames per second using the webcam, producing a series of 320 × 240 images. These images are processed

Hardware	**Description**
Operating System	Microsoft Windows XP professional Service pack 2
CPU	Intel Core 2 Extreme 2.93 GHz
RAM	2GB of Corsair 1800 mhz
Hard Drive	150GB Western Digital Raptor 10000 RPM
Webcam	Logitech notebook Quick Cam Chat

Table 5.1: Hardware Specifications.

using the techniques and methods outlined in Chapter 3. The process of capturing and interpreting SASL gestures commences by: first, a signer stands in front of the camera in the neutral pose. The neutral pose helps us to identify the start and end of a sign easily. In our experiments the neutral pose is characterised by the signer standing facing the camera with his or her arms down, located at the side of the body. Second, the system is activated and begins to record frames. Third, the signer commences with the SASL gesture and returns to the neutral pose when done and the system is stopped. The captured frames are processed, producing the feature vector for the specific gesture. The feature vector is classified using the trained HMMs, by applying it to each HMM in our lexicon. The HMM which produces the highest probability is chosen as the most likely interpretation.

5.1.1 System Description

A high level view of the iSign desktop application is shown in Figure 5.1. A brief description is provided by means of an example.

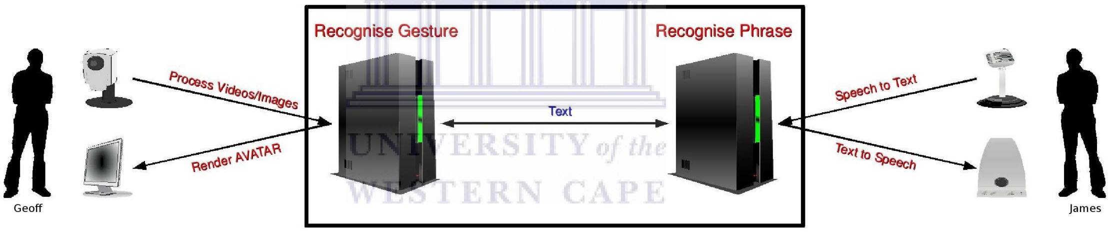


Figure 5.1: Experimental setup A.

Lets say a deaf person, Geoff, wanted to communicate with a hearing person, James. Geoff would record himself performing a SASL gesture, which would be interpreted by our system on his PC. Via a network, a text translation of the gesture would be forwarded to James' computer. Using text to speech, James will be able to listen to the message sent on by Geoff. Alternatively, James can record himself speaking and by using speech to text the message gets forwarded on to Geoff's computer as text. The system on Geoff's computer then renders the message in SASL as a 3D humanoid Avatar, as shown in 5.1.

This system is also capable of deaf to deaf and hearing to hearing communication.

5.2 iSign Mobile

The second setup, shown in Figure 5.2, uses of a cellphone's video camera to record a gesture. The video is processed using the same PC used for iSign desktop, set up as a remote server. The cellphone used in our experiments is a Sony Ericsson C905. The process of capturing and classifying a SASL gestures using the cellphone begins by: first, recording a signer performing a SASL gestures using the cellphone. The signer starts and ends in the neutral pose. Second, the video is sent to the server to be processed using an available wireless technology such as 3G, EDGE, or GPRS. Third, the system extracts the frames from the video received and processes them, producing a feature vector. The feature vector is then applied to our database of HMMs and the HMM that produces the highest probability is chosen as the interpretation.

5.2.1 System Description

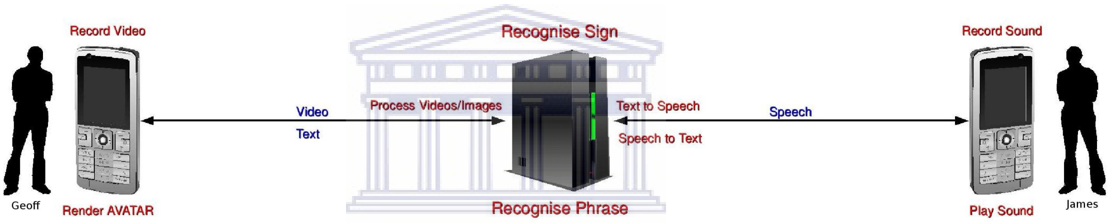


FIGURE 5.2: Experimental setup B.

Continuing with the example in Section 5.1.1: If Geoff replies using the iSign mobile setup, he records himself performing a SASL gesture using the camera on his phone. The video recording is then sent to the server and subsequently interpreted into English text. The text is converted into an audio file using text to speech and forwarded to James' phone which in turn plays the audio file.

5.3 Data Collection and Training

For our experiments a lexicon of twenty SASL gestures was randomly chosen from an English phrase book. We enlisted the help of deaf and hard of hearing students at the Dominican School for the Deaf in Wynberg Cape Town, to translate these words and phrases into SASL. A list of these words and phrases, with their sequences of holds and movements are shown in Table 5.2 The first and last hold and movement elements of each SASL gesture is attributed to the neutral state that the signer starts and ends with.

Word	**Sequence**
Hello	H-M-M-H-M-H
How are you	H-M-H-M-H-M-H
Goodbye	H-M-M-M-H
Water	H-M-H-M-H
Restaurant	H-M-M-H-M-H
Right	H-M-H-M-H
Left	H-M-H-M-H
Bus	H-M-M-H-M-H
Help me	H-M-M-M-H-M-H
Help you	H-M-M-M-H-M-H
Where is the toilet	H-M-H-M-H-M-H
Medicine	H-M-M-M-H
Doctor	H-M-M-M-M-H
Good evening	H-M-H-M-H-M-H
South Africa	H-M-M-H-M-H-M-H
Sick	H-M-H-M-H
Soccer	H-M-M-M-M-H
I have stomach pain	H-M-H-M-M-M-H
How	H-M-H-M-H
Thank you	H-M-M-H-M-H

TABLE 5.2: South African Sign Language words and phrases with their sequences of holds and movements derived from the Movement-Hold model.

To determine an appropriate sample size for our training data, we selected 3 SASL gestures at random. Using the information gained from the Dominican School for the Deaf, we recorded 30 students at UWC performing each SASL gesture twice. This is done first with a short sleeve shirt and then with a long sleeve shirt. This was to ensure the robustness of the system for varying amounts of skin that is detected. A total of 60 recordings per SASL gesture was recorded yielding total of 180 recordings. We then trained each model in increments of two gestures and tested each model with all the training data. The average accuracy of the 3 selected SASL gestures was recorded at each increment. The results are shown in a line graph in Figure 5.3.

The graph indicates that the average accuracy of the 3 chosen gestures increases as more training samples are used. However when training samples exceeded 42, the average accuracy stabilises at around 93%. We therefore chose a sample size of 42 gestures to use in training each of our 20 SASL gestures. A total of 840 sample videos was recorded for all gestures.

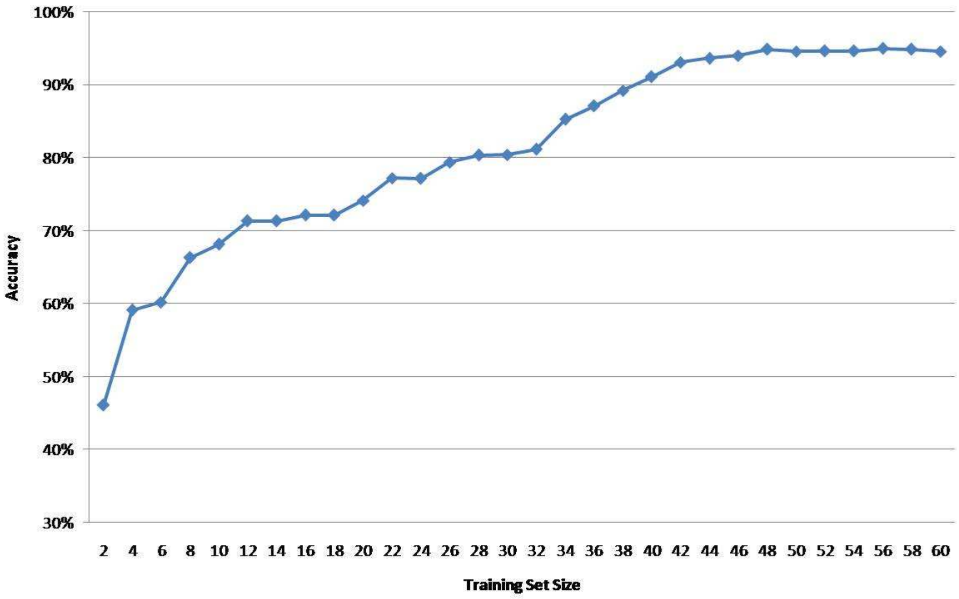


FIGURE 5.3: Graphical representation on experimental results.

In addition, 3 different students were asked to perform each gesture and were recorded using the web camera as well as the cellphone camera. They were again first recorded with a short sleeve shirt and then with a long sleeve shirt. The feature vectors of these recordings were computed. This subset of feature vectors was then used to test the HMM's performance for each of our twenty SASL gestures on unseen data.

5.4 Feature Vectors and Training

In Chapter 3 we discussed how skin regions within each frame of a video are extracted and compiled into a feature vector. Figure 5.4 shows a feature vector produced by our system for the SASL word *Hello*. Each segment in the feature vector describes the skin location of the signer within each frame of the video. The start of the video begins at the top and ends at the bottom of the feature vector. Both iSign applications use the feature vector by extracting each segment of the vector beginning from the top. Each segment is converted to a binary representation, the black colour is represent by a 0 and red or blue colour is represent as a 1. An example of this representation, for the vector shown in Figure 5.4, is shown in Table 5.3. Each unique row of the binary values is a state in our HMMs, shown in the third column in Table 5.3. For training a model $\lambda = (A, B, \pi)$, A, the state probability distribution and B, the observation symbol probability distribution are computed using the binary values of a feature vector, with each unique row of binary values being a state. The training data for a given SASL gesture consistes of 42 sets of states for each of the 42 feature vectors. These states are

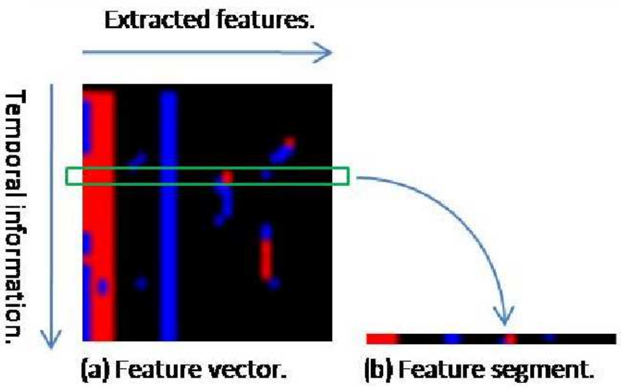


FIGURE 5.4: Feature vector produced by our system for the SASL word *Hello*.

Frame	Binary Value	State
1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	1
2	1 1 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	12
3	1 1 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	12
4	1 1 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	12
5	1 1 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	12
6	1 1 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	12
7	1 1 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0	14
8	1 1 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0	14
9	1 1 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0	15
10	1 1 1 1 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0	4
11	1 1 1 1 0 0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	12
12	1 1 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0	7
13	1 1 1 1 0 0 0 0 0 0 1 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0	8
14	1 1 1 1 0 0 0 0 0 0 1 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0	8
15	1 1 1 1 0 0 0 0 0 0 1 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0	8
16	1 1 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0	3
17	1 1 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0	3
18	1 1 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0	3
19	1 1 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0	3
20	1 1 1 1 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	2
21	1 1 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0	10
22	1 1 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0	10
23	1 1 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0	10
24	1 1 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0	10
25	1 1 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0	10
26	1 1 1 1 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0	11
27	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	1
28	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	1
29	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	1
30	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	1
31	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	1
32	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	1

TABLE 5.3: Binary representation of a feature vector for the SASL word *Hello*.

then used to train the model. For testing, the binary representation of the feature vector is applied to all the trained models of each SASL word in our lexicon. The model that produces the highest probability is chosen as the interpretation of the SASL gesture.

5.5 Summary and Conclusions

In this section, we have highlighted the hardware that we used. We have discussed two hardware setups and described both of the systems which were used to run our experiments. Finally, an overview of our data collection was provided followed by our training procedure.

Chapter 6

Results and Discussion

This chapter presents our results and contributions to GR of SASL. We first describe certain preliminary parameters and techniques that are common to all our experiments.

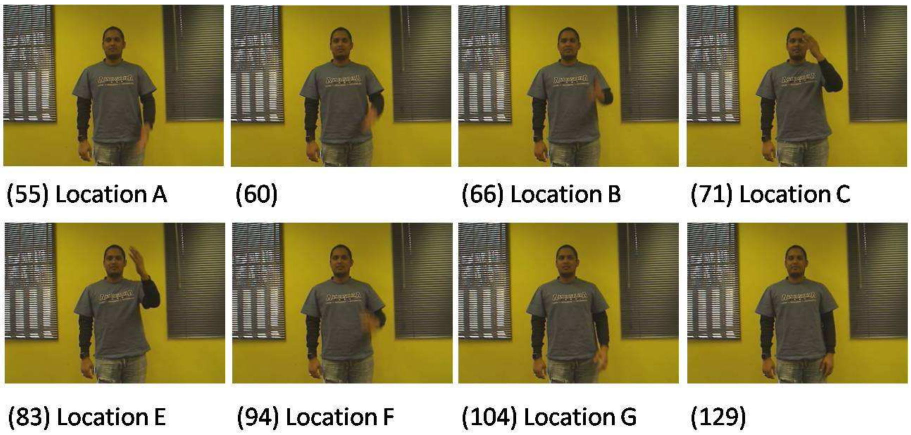
6.1 Preliminary Parameters and Techniques

For each video sample in the testing set, we perform the following processes:

1. Preprocess all videos to minimise variances pertaining to the signer,
2. Perform temporal segmentation using methods outlined in Chapter 3,
3. Build a feature vector with the information extracted in Process 2, and
4. Apply the feature vector to all HMMs and determine the most likely match.

6.1.1 Cropping a Frame Sequence of a SASL Gesture

Before we can start with Process 1, it is imperative that we find the beginning and end of a SASL gesture. We employ a similar technique to that used by Rajah [17]. It is essential to discard any unwanted frames that do not contain useful information. In addition this process will also determine the movement and hold elements of a SASL gesture. For illustrative purposes, we show how the SASL gesture for the word *Hello* is segmented using frame difference techniques outlined in Chapter 3.


(55) Location A (60) (66) Location B (71) Location C

(83) Location E (94) Location F (104) Location G (129)

FIGURE 6.1: Motion sequence for *Hello* SASL gesture at different stages of execution.

Using frame differencing techniques, we are able to calculate the intensity of motion in each frame. This allows us to detect the start and end of a gesture accurately as well as the hold and movement elements shown in Table 5.2. Figure 6.2 shows the graph of the motion intensity calculated from the frames in Figure 6.1.

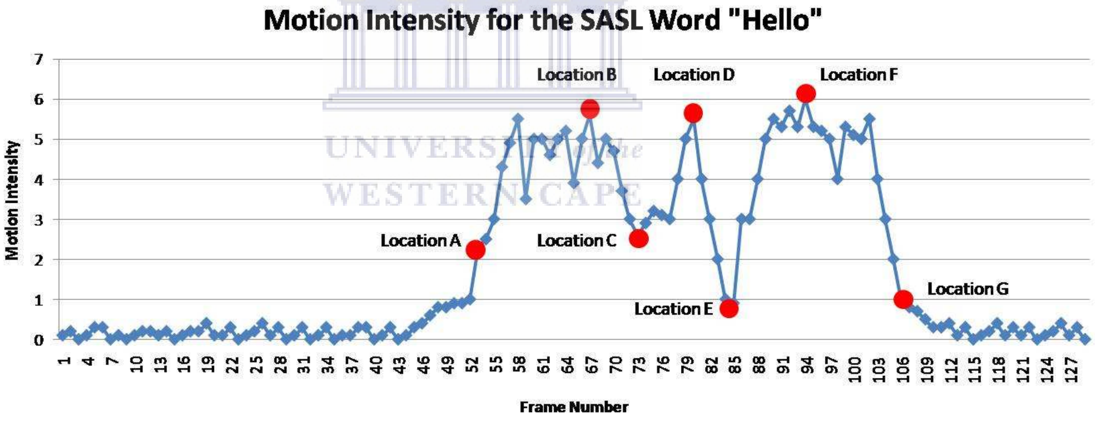


FIGURE 6.2: Intensity of motion is plotted to find the start and end of a SASL gesture.

From Figure 6.2 we can make the following observations.

1. To find the beginning of a SASL gesture we can simply detect where motion has started. This is done by frame differencing a series of frames from a video. In addition this allows us to align the gesture to the movement-hold model as outlined in Chapter 2. In Figure 6.2 we note that by using a threshold of 1, we can detect the dynamic unit of the SASL gesture. In this case, for the SASL word *Hello*, it is represented between points *A* and *G*,

2. Frames before point A and frames after point G can be ignored as they contain no movement and therefore no information pertaining to dynamic SASL gestures,

3. The frames between points A and B represent the hand moving towards the top of the head, as shown in the frames in Figure 6.1,

4. Frames between points B and C represent the hand slowing down as it reaches the head, before it changes direction away from the head, as shown in the frames in Figure 6.1,

5. The frames between points C and D represent the hand moving away from the head, as shown in the frames in Figure 6.1,

6. Frames between points D and E represent the hand slowing down and coming to a complete stop, as shown in the frames in Figure 6.1. The motion intensity drops below the threshold of 1 illustrating that in this example the hand has stopped moving,

7. The frames between points E and F represent the hand picking up speed as the signer returns to the neutral pose, as shown in the frames in Figure 6.1, and

8. Frames between points F and G represent the hand slowing down as the signer reaches the neutral pose, as shown in the frames in Figure 6.1.

6.1.2 Training

A fully connected HMM, in which every possible state can be reached from every other state, would require a large amount of memory. A three state fully connected or ergodic HMM is shown in Figure 6.3. Recall that an HMM is give as:

$$\lambda = (A, B, \pi) \tag{6.1}$$

Where:

1. A is the state probability matrix of size $N \times N$,

2. N is the number of hidden states,

3. B denotes the observation symbol probability distribution matrix of size $N \times M$, and

4. M represents the number of distinct observation symbols per state.

With thirty two features in the binary feature vector, an ergodic HMM would produce 2^{32} possible states, in which every state is connected to every other state. However, due the constraints of the human body, only a small number of these states will ever occur. For example, the left hand cannot be in two places at the same time. Thus, an ergodic HMM was not constructed. Rather we add new states to the model as they appear in training. The resulting model consists of only the required states with transitions between states determined by the state probability distribution matrix. An example of such a 3 state HMM, is shown in Figure 6.4.

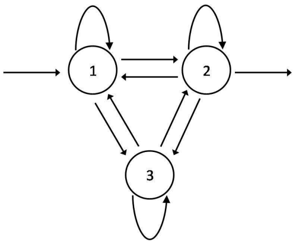


FIGURE 6.3: A three state ergodic HMM.

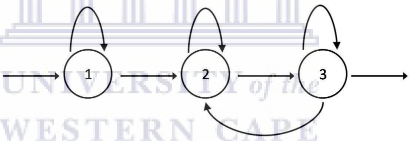


FIGURE 6.4: A three state HMM.

6.2 Results

In this section we present our results. The results of the tests listed below, are examined, followed by a summary of our tests:

1. Test on seen data,

2. Test on unseen data, and

3. Test the normalising method.

6.2.1 Testing on Seen Data

In order to validate that our trained HMMs have learned, i.e., are able to discriminate between SASL gestures, we first test each of them on seen gestures. That is, we used the same data for training and for testing. Forty two samples of each gesture were used in training, with the same 42 sample videos used for testing. A break down of the testing and training sample videos are shown in Table 6.1.

Training and Testing Sample Sets on Seen Data for each SASL Gesture		
Sample Size per SASL Gesture	**Native SASL Signers**	**UWC Students**
42	10	32

Table 6.1: Training and testing sample sets on seen data.

We expect the models to perform well, as the HMM for each of the SASL word(s) or phrases in our lexicon has been trained on each of the SASL gesture. The results of these tests are shown in Table 6.2.

Test on Seen Data				
	Word(s)	**Right**	**Wrong**	**Accuracy**
1	Bus	34	8	80.95%
2	Doctor	39	3	92.86%
3	Good evening	22	20	52.38%
4	Goodbye	40	2	95.24%
5	Hello	39	3	92.86%
6	Help me	16	26	38.1%
7	Help you	19	23	45.24%
8	How	29	13	69.05%
9	How are you	12	30	28.57%
10	Stomach pain	30	12	71.43%
11	Left	42	0	100%
12	Medicine	39	3	92.86%
13	Restaurant	31	11	73.81%
14	Right	39	3	92.86%
15	Sick	27	15	64.29%
16	Soccer	20	22	47.62%
17	South Africa	42	0	100%
18	Thank you	35	7	83.33%
19	Water	22	20	52.38%
20	Toilet	36	6	85.71%
Average:				73.38%

Table 6.2: **Testing Performance:** The table shows the test results on seen data, for each trained HMM. The column named *Right*, lists the number of correctly classified gestures and the column named *Wrong* lists the number of misclassified gestures.

Sixteen out of the twenty HMMs classify gestures correctly more than 50% of the time.

1. 10 out of 20 SASL gestures obtain an accuracy of over 80%,
2. 12 out of 20 SASL gestures obtain an accuracy of over 70%, and
3. 16 out of 20 SASL gestures obtain an accuracy of over 50%.

Even though the above test results are very promising for the majority of SASL gestures, we need to determine why the rest performed poorly. To determine why some models performed poorly, we cross-test each SASL gesture with all gestures in our lexicon. We use the same testing set as above, whereby each HMM is tested on data that was used in training. The results for this test are shown in Tables 6.3 and 6.4, with the average log probability $\log P(B|M_i)$ for each SASL gesture shown in Table 6.3 and the number of correctly classified SASL gestures shown in Table 6.4.

Probabilities Produced by the HMM Test On Seen Data																					
		HMMs																			
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
Word(s)	**1**	**-30.164**	-98.797	-46.325	-57.535	$-\infty$	-46.123	-40.016	-64.200	-51.324	-41.432	-59.083	-29.187	-43.317	-90.654	-95.238	-44.506	-32.282	-85.189	-76.222	-32.810
	2	$-\infty$	**-16.623**	$-\infty$	-26.532	-35.027	-95.268	$-\infty$	-97.547	$-\infty$	-90.628	$-\infty$	-99.061	-72.688	-26.290	-37.253	-94.306	-81.350	-35.282	-49.164	$-\infty$
	3	-75.313	-83.590	**-57.663**	$-\infty$	$-\infty$	-54.413	-37.060	-61.542	-65.718	-57.284	-53.722	-35.057	-47.329	-87.403	-78.278	-60.488	-44.667	-85.167	-82.153	-31.311
	4	$-\infty$	-22.803	-89.662	**-19.608**	-48.292	-86.641	-97.649	-96.152	-99.017	$-\infty$	-92.395	$-\infty$	-85.163	-19.513	-41.450	-94.042	-93.323	-22.728	-33.022	-93.679
	5	$-\infty$	-32.025	-80.649	-52.333	**-16.625**	-96.350	$-\infty$	$-\infty$	-92.561	-81.761	$-\infty$	-90.151	-94.280	-23.301	**-32.568**	-93.127	-81.208	-15.517	-43.636	-89.271
	6	-93.712	$-\infty$	-65.127	-95.386	-74.733	**-56.400**	**-31.388**	-55.731	-63.105	-38.233	-93.261	-28.079	-37.188	$-\infty$	-84.351	-53.691	-62.193	-97.222	-74.675	-44.623
	7	-76.712	-93.582	-37.559	-70.314	-68.704	**-20.195**	**-60.207**	-67.024	**-20.817**	-31.482	-92.530	-36.013	-43.253	-92.275	-68.016	-41.276	-42.489	$-\infty$	-84.582	-31.629
	8	-38.254	-99.647	-25.557	$-\infty$	-92.182	-58.687	-44.575	**-54.133**	-47.528	-42.247	-73.316	-33.125	-59.311	-82.057	-91.767	-58.859	-53.646	-77.733	-94.164	-29.810
	9	-49.013	-83.603	-80.024	-97.051	$-\infty$	-50.079	-36.645	-76.241	**-79.220**	-35.277	$-\infty$	-35.216	-33.903	-86.442	-86.553	-45.581	-45.573	-80.494	-97.722	-49.429
	10	$-\infty$	-90.264	$-\infty$	-84.524	-90.727	-55.706	-47.864	-68.611	-57.407	**-26.548**	-79.278	-46.116	-42.592	-92.323	-79.143	-51.770	-36.506	-89.526	-72.800	-27.937
	11	$-\infty$	$-\infty$	-91.578	$-\infty$	$-\infty$	$-\infty$	-99.679	-94.350	$-\infty$	$-\infty$	**-8.738**	-93.476	-91.661	$-\infty$	$-\infty$	$-\infty$	-99.148	-93.545	$-\infty$	$-\infty$
	12	-69.600	$-\infty$	**-19.489**	$-\infty$	$-\infty$	-38.164	-43.614	-57.260	-61.097	-31.500	-86.625	**-25.024**	-42.812	-96.146	-78.375	-58.122	-24.351	-98.618	-88.511	-39.333
	13	-76.766	-76.508	-26.424	-69.511	-90.284	-44.143	-59.242	-58.241	-63.129	-30.823	-69.151	-48.648	**-29.820**	-75.694	-94.770	-39.085	-34.118	-80.350	-98.781	-37.202
	14	$-\infty$	-28.485	-83.314	-26.263	-21.576	$-\infty$	$-\infty$	-92.038	$-\infty$	-80.067	$-\infty$	$-\infty$	-87.056	**-14.139**	-37.644	-84.091	-78.494	-30.656	-34.643	-79.690
	15	$-\infty$	-45.015	-92.583	-31.494	-30.241	-76.533	-83.179	$-\infty$	-86.671	$-\infty$	-98.041	-98.173	-90.578	-18.184	**-55.415**	-89.344	-95.443	-14.174	-58.426	-87.229
	16	-48.892	-82.790	-33.548	$-\infty$	-81.441	-42.727	-38.205	-58.481	-42.313	-40.230	$-\infty$	-30.662	-39.267	-71.442	-80.012	**-62.630**	-51.127	-71.387	-91.273	-30.091
	17	-40.048	-89.730	-34.179	-81.182	$-\infty$	-32.549	-49.047	-60.466	-35.443	-35.676	-85.456	-53.535	-32.289	-80.067	-86.310	-66.533	**-20.030**	-28.058	-83.443	-27.180
	18	$-\infty$	-18.714	-95.429	-38.011	-38.118	-96.455	-84.069	$-\infty$	-87.831	-80.439	-95.365	-91.081	$-\infty$	-40.475	-40.200	-93.290	-92.529	**-10.45**	**-29.384**	-87.676
	19	$-\infty$	-22.595	$-\infty$	-42.917	-22.039	-87.078	-98.048	-99.691	-93.515	-94.377	$-\infty$	-84.743	-93.667	-25.305	-47.732	-79.115	-84.584	-21.092	**-59.250**	-75.839
	20	-62.414	-92.365	-62.524	-92.414	-94.615	-58.684	-42.781	-67.622	-37.607	-39.617	-72.367	-36.579	-45.062	-71.236	-77.506	**-35.748**	-48.157	-99.135	-92.538	**-26.809**

TABLE 6.3: **Cross-Testing Performance:** The table shows the average logarithmic probability $\log P(B|M_i)$ produced by each HMM. The largest value for $\log P(B|M_i)$ is chosen as the most likely interpretation by our system.

Classification Rate on the Test on Seen Data																					
		HMMs																			
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
Word(s)	**1**	**34**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	2	0	**39**	0	**1**	0	0	0	0	0	0	0	0	0	0	**2**	0	0	0	0	0
	3	0	0	**22**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	4	0	**1**	0	**40**	0	0	0	0	0	0	0	0	0	**2**	0	0	0	0	**1**	0
	5	0	0	0	0	**39**	0	0	0	0	0	0	0	0	0	**13**	0	0	**6**	0	0
	6	0	0	0	0	0	**16**	**23**	**8**	0	0	0	**3**	0	0	0	0	0	0	0	0
	7	0	0	**3**	0	0	**24**	**19**	0	**23**	**6**	0	0	0	0	0	0	0	0	0	0
	8	**8**	0	**2**	0	0	0	0	**29**	0	0	0	0	0	0	0	0	0	0	0	0
	9	0	0	0	0	0	0	0	0	**12**	0	0	0	**8**	0	0	0	0	0	0	0
	10	0	0	0	0	0	0	0	0	0	**30**	0	0	0	0	0	0	0	0	0	0
	11	0	0	0	0	0	0	0	0	0	0	**42**	0	0	0	0	0	0	0	0	0
	12	0	0	**15**	0	0	0	0	**3**	0	**2**	0	**39**	0	0	0	0	0	0	0	0
	13	0	0	0	0	0	0	0	**1**	0	**4**	0	0	**31**	0	0	0	0	0	0	0
	14	0	0	0	**1**	**2**	0	0	0	0	0	0	0	0	**39**	0	0	0	0	0	0
	15	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**27**	0	0	**1**	0	0
	16	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	**20**	0	0	0	0
	17	0	0	0	0	0	**2**	0	0	**5**	0	0	0	**3**	0	0	0	**42**	0	0	**6**
	18	0	**2**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**35**	19	0
	19	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	**22**	0
	20	0	0	0	0	0	0	0	0	**2**	0	0	0	0	0	0	**22**	0	0	0	**36**

TABLE 6.4: **Cross-Testing Performance:** The table shows the number of correctly classified and incorrectly classified SASL words or phrases, for each trained HMM on seen data.

For analysis purposes, we divided the gestures in our data set into three distinct categories:

1. Single left hand movements (*left*),
2. Single right hand movements (*right*), and
3. Both left and right hand movements (*both*).

We expect that the HMMs should be able to discriminate between *left*, *right* and *both* accurately due to the distinct differences in the patterns of the motion of the hands. The twenty gestures and their corresponding hand movements are shown in Table 6.5.

	Gesture	**Hand Movements**
1	Bus	*both*
2	Doctor	*right*
3	Good evening	*both*
4	Goodbye	*right*
5	Hello	*right*
6	Help me	*both*
7	Help you	*both*
8	How	*both*
9	How are you	*both*
10	Stomach pain	*both*
11	Left	*left*
12	Medicine	*both*
13	Restaurant	*both*
14	Right	*right*
15	Sick	*right*
16	Soccer	*both*
17	South Africa	*both*
18	Thank you	*right*
19	Water	*right*
20	Toilet	*both*

Table 6.5: SASL Gestures divided up into their corresponding hand movements.

Recall Chapter 5, where an incoming feature vector produced by a gesture is applied to every trained HMM in our lexicon. The HMM which produces the highest probability is then chosen as the most likely interpretation. Analysing Table 6.3 and 6.4 shows that our system fails to accurately classify certain gestures, while others perform extremely well. As expected, the trained HMMs are able to discriminate among gestures with different patterns of hand motion, i.e. *left*, *right* and *both*. However for gestures that display similar patterns of hand motion, such as gestures within the *both* category, the HMMs fail to accurately classify them. All 20 HMMs and their misclassified interpretations are shown in Table 6.6.

	Misclassified Gestures			
	Input Gestures	**Word/Phrase**	**Number**	**Total**
1	Bus (*both*)	How (*both*)	8	8
2	Doctor (*right*)	Goodbye (*right*) Thank you (*right*)	1 2	3
3	**Good evening (*both*)**	Help you (*both*) How (*both*) Medicine (*both*)	3 2 15	20
4	Goodbye (*right*)	Doctor (*right*) Right (*right*)	1 1	2
5	Hello (*right*)	Right (*right*) Water (*right*)	2 1	3
6	**Help me (*both*)**	Help you (*both*) South Africa (*both*)	24 2	26
7	**Help you (*both*)**	Help me (*both*)	23	23
8	How (*both*)	Help me (*both*) Medicine (*both*) Restaurant (*both*) Soccer (*both*)	8 3 1 1	13
9	**How are you (*both*)**	Help you (*both*) South Africa (*both*) Toilet (*both*)	23 5 2	30
10	Stomach pain (*both*)	Help you (*both*) Medicine (*both*) Restaurant (*both*)	6 2 4	12
11	Left (*left*)	n/a	0	0
12	Medicine (*both*)	Help me (*both*)	6	6
13	Restaurant (*both*)	How are you (*both*) South Africa (*both*)	8 3	11
14	Right (*right*)	Goodbye (*right*) Sick (*right*)	2 1	3
15	Sick (*right*)	Doctor (*right*) Hello (*right*)	2 13	15
16	**Soccer (*both*)**	Toilet (*both*)	22	22
17	South Africa (*both*)	n/a	0	0
18	Thank you (*right*)	Hello (*right*) Sick (*right*)	6 1	7
19	**Water (*right*)**	Goodbye (*right*) Thank you (*right*)	1 19	20
20	Toilet (*both*)	South Africa (*both*)	6	6

TABLE 6.6: The table shows misclassified gestures and the number of misclassified gestures for each trained HMM on seen data.

The six gestures indicated in **bold** text in Table 6.6 show the SASL gestures which were incorrectly classified more 45% of the time. While the remaining 14 gestures achieve and error rate of less than 35%. Gestures which were misclassified more than 45% of the time are listed in Table 6.7.

Most Frequently Misclassified	
Trained HMM	**Accuracy**
HMM 3 (Good evening)	52.38%
HMM 6 (Help me)	38.1%
HMM 7 (Help you)	45.24%
HMM 9 (How are you)	28.57%
HMM 16 (Soccer)	47.62%
HMM 19 (Water)	52.38%

TABLE 6.7: **Misclassified:** The table shows the frequently misclassified SASL gestures for the test on seen data.

To ascertain the reason for these misclassifications, we need to decompose the gestures into their movement and hold elements. As an illustration, we will dissect the gesture *Help you*, used to train HMM 7 in Table 6.7. It obtained a misclassification for the SASL gesture *Help me*, 54% of the time. Figure 6.5 shows the movement and hold elements of the SASL gesture *Help you*. We compare the elements in Figure 6.5 of the gesture *Help you*, with that of the *Help me* gesture shown in Figure 6.6.

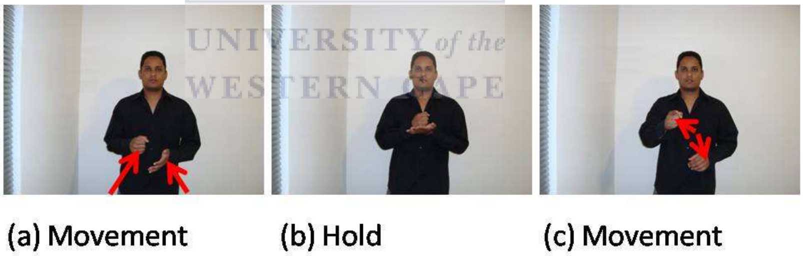

(a) Movement (b) Hold (c) Movement

FIGURE 6.5: Movement and hold elements for the SASL gesture *Help you*.

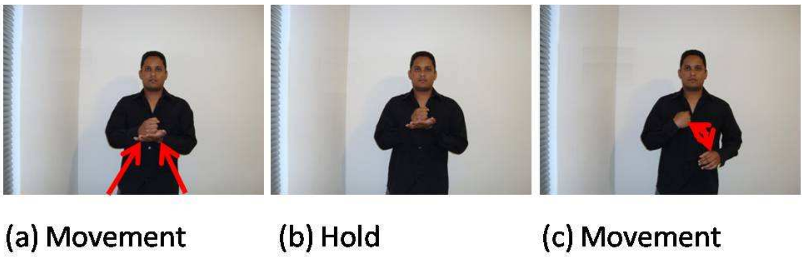
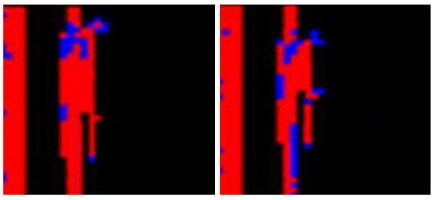

(a) Movement (b) Hold (c) Movement

FIGURE 6.6: Movement and hold elements for the SASL gesture *Help me*.

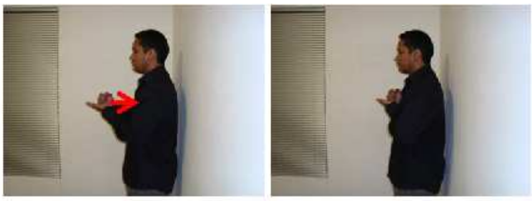
The movement and hold frames shown in Figure 6.5 and 6.6 indicate that these two SASL gestures have similar patterns of hand motions when viewed from the front. The two feature vectors generated for these two gestures are shown in Figure 6.7



(a) Feature vector 1 (b) Feature vector 2

Figure 6.7: Feature vectors produced for the gestures *Help you* and *Help me*.

The feature vector for the SASL gesture *Help you* is shown in Figure 6.7(a), with the SASL gesture *Help me* shown in Figure 6.7(b). The two similar feature vectors produced by the gestures make it virtually impossible for an HMM to discriminate between them. When viewing Figures 6.5(b) and 6.6(b) from the side, it can be seen that movement and hold information is lost due to the lack of 3D information in the frontal 2D view. Figure 6.8 and 6.9 show the lost movement and hold information. With this additional information we can now clearly see the differences between the SASL gestures *Help me* from *Help you*. It can be concluded, for this example, that the misclassified results obtained in Table 6.7 can be attributed to two factors. First, signs that have very similar patterns of hand motions can cause misclassifications. Second, due to frontal point of view of the camera, crucial movement and hold information can be lost, and therefore may hinder the classification performance of the HMMs.



(a) Movement (b) Hold

Figure 6.8: Side view of the SASL gesture *Help me*.

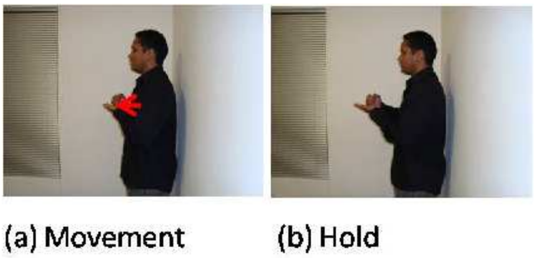


FIGURE 6.9: Side view of the SASL gesture *Help you.*

6.2.2 Testing on Unseen Data

In the next test, we assess the discrimination performance of the trained HMMs on unseen data. Unseen data refers to data that was not used during training. Five signers were required to sign all twenty SASL gestures, first wearing long sleeved shirts and then with short sleeved shirts. A total of 200 signs were recorded with 10 samples per SASL gesture used for testing. The same training sample, as used for training the HMM on seen data, was used for training the models. An overview of the training and testing sample videos for each HMM are shown in Table 6.8 and 6.9 respectively.

Training Sample Set on Unseen Data for each SASL Gesture		
Sample Size per SASL Gesture	**Native SASL Signers**	**UWC Students**
42	10	32

TABLE 6.8: Training sample set on unseen data.

Testing Sample Set on Unseen Data for each SASL Gesture		
Sample Size per SASL Gesture	**Native SASL Signers**	**UWC Students**
10	4	6

TABLE 6.9: Testing sample set on unseen data.

The test results are shown in Table 6.10. The test on unseen data produced an overall accuracy of 69%. We note that the HMMs that produced an accuracy below 50% in Table 6.10, are the same six trained models that produced misclassified results in the test on seen data shown in Table 6.7. The low overall accuracy can therefore be attributed to the HMMs inability to discriminate between SASL gestures with similar hand movements.

Even though our testing sample set size is small, by plotting the results of the seen data test in Table 6.3 against the results on the unseen data test in Table 6.10, we can see a trend appearing. Figure 6.10 shows a graph comparing the results on the seen data test with the results on the unseen data test.

Test on Unseen Data				
	Word(s)	**Right**	**Wrong**	**Accuracy**
1	Bus	9	1	90%
2	Doctor	9	1	90%
3	Good evening	5	5	50%
4	Goodbye	8	2	80%
5	Hello	9	1	90%
6	Help me	4	6	40%
7	Help you	4	6	40%
8	How	9	1	90%
9	How are you	2	8	20%
10	Stomach pain	9	1	90%
11	Left	10	0	100%
12	Medicine	8	2	80%
13	Restaurant	8	2	80%
14	Right	9	1	90%
15	Sick	4	6	40%
16	Soccer	3	7	30%
17	South Africa	9	1	90%
18	Thank you	8	2	80%
19	Water	4	6	40%
20	Toilet	9	1	90%
Average:				69%

Table 6.10: **Testing Performance:** The table shows the test results on unseen data, for each trained HMM.

By comparing the two results we notice:

1. Both achieve a similar average classification rate, with seen data achieving 73.38% and unseen data achieving 69%,

2. The same six models that performed poorly in the seen data test, performed poorly in the unseen data test, and

3. All trained models that achieved an accuracy of 80% and more in the seen data test, performed equally well in the unseen data test.

To evaluate how well these test results fit each other, we use the Pearson product moment correlation. It measures the correlation between two variables and reflects the degree of linear relationship between two variables. It ranges from -1 to $+1$, 0 reflecting no linear relationship, $+1$ reflecting a perfect positive linear relationship between variables

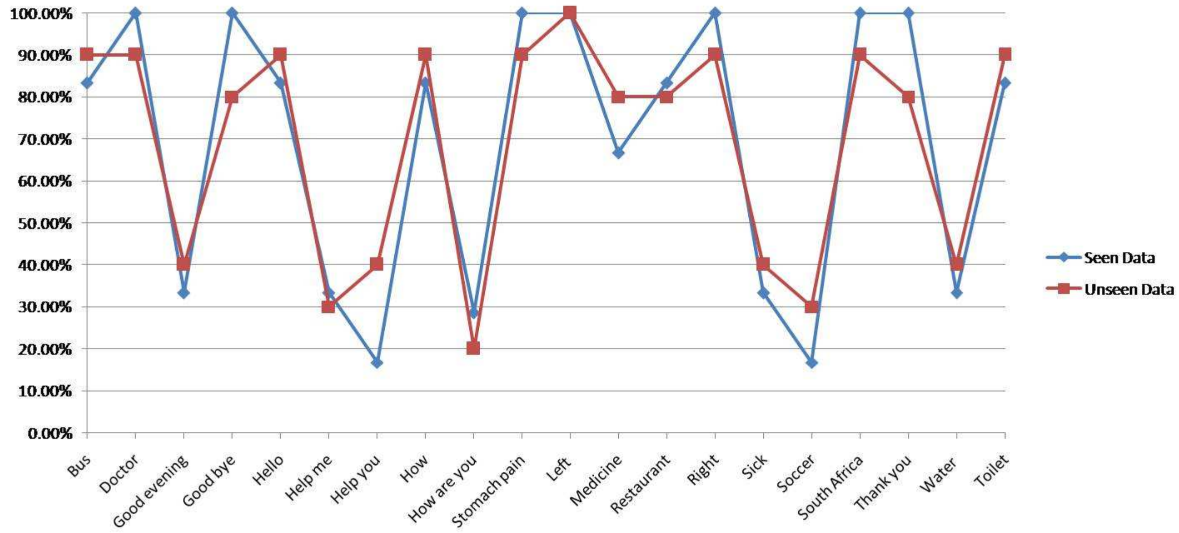


FIGURE 6.10: **Testing Performance:** The graph shows the results on the test seen data from Table 6.3 plotted with the results of the test on unseen data from Table 6.10.

and −1 reflecting a perfect negative linear relationship between variables. The Pearson product moment correlation produces a high correlation of 0.90 and a standard deviation of 0.29 for the two data sets, indicating a strong positive relationship between the two test results. We can see that HMMs are able to generalise well and classify gestures across different signers, both with seen and unseen data.

6.2.3 Testing the Normalising Method

In Chapter 3, we discussed variations that may occur when recording different signers performing the same gesture. These variations include: body dimensions, distance of signer from the camera and position of the signer within the video frame. In order to test the performance of the algorithms constructed to cater for these variations within our normalising method, we conducted three separate tests. The overview for these test is listed below:

- (Test A) Test the performance of the normalising method with signers with varying body dimensions,
- (Test B) Test the performance of the normalising method with signers at varying distances from the camera, and
- (Test C) Test the performance of the normalising method with signers at varying positions within the video frame.

6.2.3.1 Test A

Recall Chapter 3 where a grid is constructed around the signer's head. The grid dimensions are then halved and an additional grid is added to the image, to increase the resolution of the grid. For Test A, we test if the resolution of the grid affects the performance of the normalising method for signers of varying body dimensions. Figure 6.11 shows 3 examples of different signers with varying body dimensions used in this experiment. In addition, for this test, 3 grid resolutions are used. First, the grid resolution is determined by the width and height of the signer's head, shown in Figure 6.12(a). Second, the grid resolution is determined by half the width and height of the signer's head, shown in Figure 6.12(b). Third, the grid resolution is determined by a quarter of the width and height of the signer's head, shown in Figure 6.12(c).

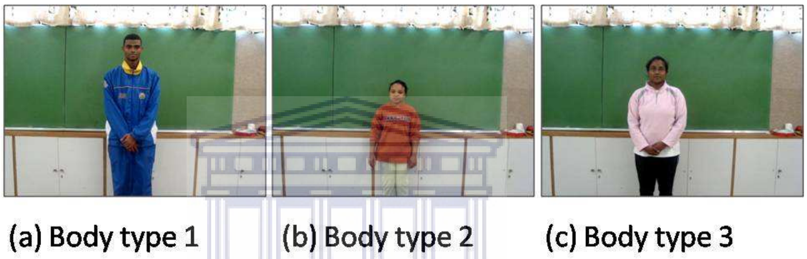


FIGURE 6.11: Signers with different body types.

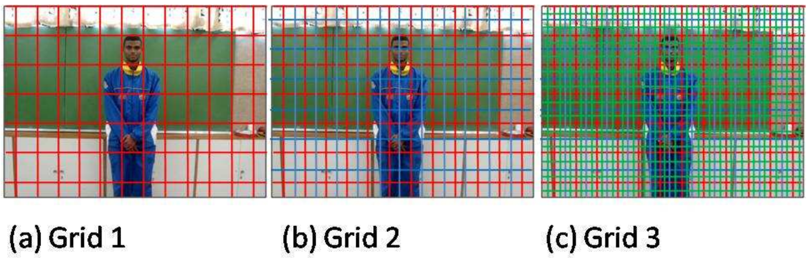


FIGURE 6.12: Different grid sizes used to test the performance of the normalising method on signers with varying body types.

The same training data as used for training the HMMs on seen data is used for training the models in this experiment. An overview of the training sample videos for each HMM is shown in Table 6.11. The same sample set used for training is used for testing. The results for each of the three different grid resolutions are shown in Table 6.12.

Training and Testing Sample Set for Test A		
Sample Size per SASL Gesture	**Native SASL Signers**	**UWC Students**
42	10	32

Table 6.11: Training and Testing sample set for Test A.

Results for Test A			
	Grid 1	**Grid 2**	**Grid 3**
States	28	110	448
Log Probability	-67.325	-38.065	-39.351
Accuracy	54.25%	73.38%	71.42%

Table 6.12: **Testing Performance:** The table shows the test results for Test A.

The test on Grid 1 in Figure 6.12 performed the worst, achieving a classification accuracy of 54.25%. Only 28 unique states were identified during training, which means that the HMMs only has 28 states to discriminate one SASL gesture from another. The low resolution of the grid thus negatively impacted the performance of our system. The test on Grid 2 performed the best, and achieved an overall accuracy of 73.38%, which is identical to the results of the test on seen data in Table 6.2. This was expected as the same grid resolution was used in both tests. The test on Grid 3 performed similar to the test Grid 2, but used more than 3 times more states than the test on Grid 2, drastically slowing down the training and testing process. These results confirm that using the grid resolution determined by half the width and height of the signers head, produce the optimal results.

6.2.3.2 Test B

For this test we validate that our system correctly classifies gestures when signers stand at various distances from the camera. We could not use the training set for testing as all the signers stood at a similar distance from the camera. Due to constraints of collecting a new testing sample for this test, we manually manipulate each video in our testing set by enlarging and cropping each frame in the video. For the testing samples we modified each video by scaling each frame and cropping the images to fit a 320 × 240 resolution. This process was performed twice on each video, zooming in on each frame in each video at 125% and 150% respectively. The two new videos produced, give an illusion that the signer is standing at different distances from the camera, when compared to the original video. An example of the image manipulation that was done is shown in Figure 6.13.

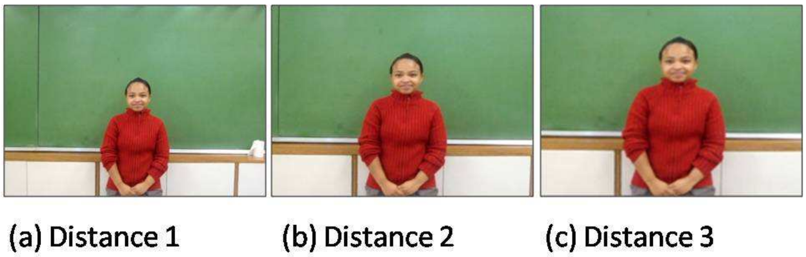


FIGURE 6.13: Image (a) shows the original frame in a video with Image (b) and (c) showing the manipulated frames, appear as if the signer is closer to the camera.

Figure 6.13(a) shows an original frame of a video, with 6.13(b) and 6.13(c) showing the modified frames. Figure 6.13(b) and 6.13(c) clearly gives the illusion that the signer is standing closer to the camera. The two new videos for each SASL gesture was added to the training set. The testing videos were divided into three subsets:

- (Distance A) The original set of testing videos of 21 different signers performing the 20 SASL gestures twice,
- (Distance B) The manipulated set of videos of all 21 signers scaled to 125% and resized to fit a 320 × 240 resolution, and
- (Distance C) The manipulated set of videos of all 21 signers scaled to 150% and resized to fit a 320 × 240 resolution.

Each subset of this testing sample was individually used to test the performance of our normalising method. An overview of the training and testing sample videos for each HMM are shown in Tables 6.13 and 6.14.

Training Sample Set for Test B		
Sample Size per SASL Gesture	**Native SASL Signers**	**UWC Students**
42	10	32

TABLE 6.13: Training sample set for Test B.

Testing Sample Set for Test B		
	Sample Size per SASL Gesture	**(Modified/Scaled)**
Distance A	42	Original Videos
Distance B	42	125%
Distance C	42	150%

TABLE 6.14: Testing sample set for Test B.

If the test results produced by each individual subset are substantially different, this would indicate that our system has failed to cater for signers that stand at varying distances from the camera. However, if the test results are similar or identical, it would indicate that our normalising method has successfully catered for signers that stand at varying distances from the camera. The results for this test are shown in Table 6.15

Results for Test B			
	Distance 1	**Distance 2**	**Distance 3**
States	110	110	110
Log Probability	-34.628	-34.628	-34.628
Accuracy	71.42%	71.42%	71.42%

Table 6.15: **Testing Performance:** The table shows the test results of Test B.

The results show that our system is able to cater accurately for signers who stand at varying distances from the camera. Each subset of our testing sample produced exactly the same result, with an average classification accuracy of 71.42%. This result confirms that our algorithm accurately normalises videos of signers who stand at varying distances from the camera.

6.2.3.3 Test C

Due to the constraint of obtaining cellphone recordings of some of the signers used to train our system on seen data, we were only able to use 120 recordings of gestures made using the camera on a mobile phone. For the remaining 720 testing samples needed, we simulated the movement of the camera by randomly shifting the signer within each frame in the video—ensuring that hand regions do not fall outside of the image frame. This was to ensure that the position of the signer within each video frame would not be static. Figure 6.14 shows some frames of a video recorded by the mobile phone. Movement of the camera can be seen in Figure 6.14(b), where the frame has shifted up and to the left. Figure 6.14(c) shows the frame has shifted down. Some of the manipulated video frames are shown in Figure 6.15.

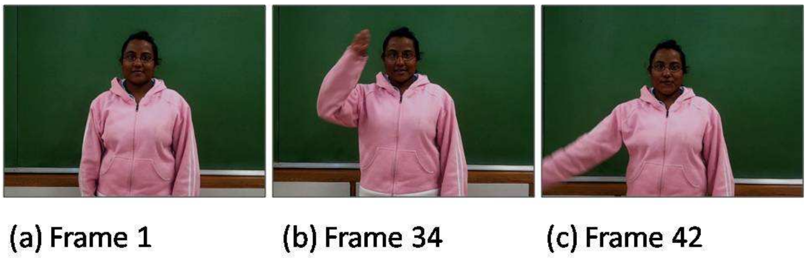


Figure 6.14: An example of video frames of a signer performing the *Hello* SASL gesture taken by a cellphone camera.

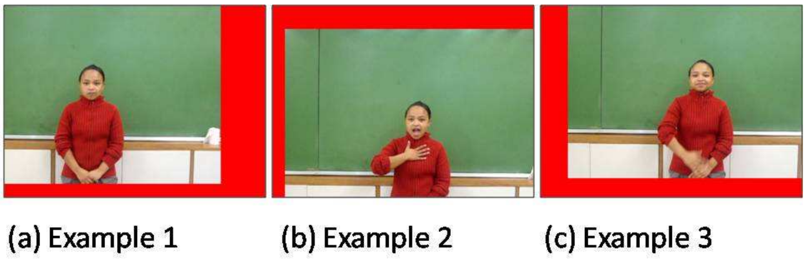


FIGURE 6.15: An example of video frames of various signers, manipulated to simulate the movement of a camera.

Figure 6.15, shows how we simulated the camera's movement by shifting the singer within each video. We combine the cellphone recordings and the image manipulated videos to use as test data.

For Test C we needed to test the performance of the normalising method with signers at varying positions within the video frame. To this end, we performed four tests:

- (**Test C**1) For training, the same training set used in the test on seen data is used. For testing the models, we used the stabilised cellphone recordings combined with the stabilised image manipulated videos,
- (**Test C**2) For this test we disable the stabilising algorithm in the normalising method. The same test and training set as used in Test C1 is used,
- (**Test C**3) For training, the unstabilised cellphone recordings combined with the unstabilised image manipulated videos is used. For testing the same, set of unstabilised cellphone recordings combined with the unstabilised image manipulated videos is used, and
- (**Test C**4) For training, the unstabilised cellphone recordings combined with the unstabilised image manipulated videos is used. For testing we again, randomly shifted the cellphone recordings and manipulated videos. This was to ensure that the testing set is different to the training set.

For the first test, we recorded the results and compared them to our original test on seen data with the results shown in Table 6.2. Similar results would indicate that our system is able to cater for videos where the camera has moved i.e. stabilising the shaking of video when taken by a mobile phone. The comparison is shown in Table 6.16.

Comparing the two tests we see that they yield similar results, with the test on seen data classifying 73.38% of SASL gestures correctly and the test using the cellphone and

Results for First Test on Test C		
Testing Set	**Seen Data**	**Cellphone & Manipulated Videos**
States	110	110
Log Probability	-38.065	-45.167
Accuracy	73.38%	72.02%

Table 6.16: **Testing Performance:** The table shows the test results for Test C1.

manipulated videos classifying 72.02% of SASL gestures correctly. This confirms that our system is able to cater for videos where the camera has moved. We have thus shown that variations of the position of a signer within a video do not negatively impact the performance of our system.

For the second test, we disabled the stabilising method and ran the test again. We expected a weak performance from the models when the stabilising method is enabled because of the moving of the signer within each video. The results are shown in Table 6.17.

Results for Second Test on Test C		
Normalising method	**Enabled**	**Disabled**
States	110	110
Log Probability	-45.167	$-\infty$
Accuracy	72.02%	16.67%

Table 6.17: **Testing Performance:** The table shows the test results for Test C2.

As per our expectation the test performed poorly, obtaining an average classification rate of 16.67%. The poor results obtained from the cellphone video samples and manipulated videos when the stabilising method is disabled, can be attributed to the vast differences of the position of the signer within each frame of the testing set to that of the signers static positions in the training set.

For the third test, we trained and tested the models on the video samples obtained from the cellphone as well as the image manipulated videos. We first ran the test with the stabilising method enabled and then disabled. We expected the models to perform well when the normalising method, which centralises the location of the signer, is disabled. This is due to testing the models on the same, seen, un-normalised data that we trained on. The results for this test are shown in Table 6.18

The test performed as we anticipated, with the HMMs achieving a classification rate of 72.02% when the normalising method was disable. This is because the same seen data was used for training and testing. However when the normalising method was enabled the models were unable to accurately discriminate between the gestures.

Results for Third Test on Test C		
Normalising method	**Enabled**	**Disabled**
States	110	110
Log Probability	$-\infty$	-45.167
Accuracy	19.04%	72.02%

TABLE 6.18: **Testing Performance:** The table shows the test results for Test C3.

For the fourth test, we test the performance of the normalising method, with signers at various locations within the frame, on unseen data. For training, the cellphone recordings combined with the image manipulated videos are used. For testing we again, randomly shifted the cellphone recordings and manipulated videos. The results for this test are shown in Table 6.19.

Results for Fourth Test on Test C		
Normalising method	**Enabled**	**Disabled**
States	110	110
Log Probability	$-\infty$	$-\infty$
Accuracy	3.69%	1.79%

TABLE 6.19: **Testing Performance:** The table shows the test results for Test C4.

The test performed poorly for when the stabilising method was enabled and disabled. This is due to the vast differences in the position of the signer in the training and testing set. This makes HMM discrimination almost impossible. To confirm this, we compare the feature vector produced from the SASL gesture *Goodbye* when a static camera position is used and when one is not used. The feature vectors produced are shown in Figure 6.16. Feature vector 1 was produced by the system when the camera position was static. Feature vector 2 was produced by using sample video recorded using a cellphone camera when the normalising method was enabled and feature vector 3 is produced when the normalising method was disabled.

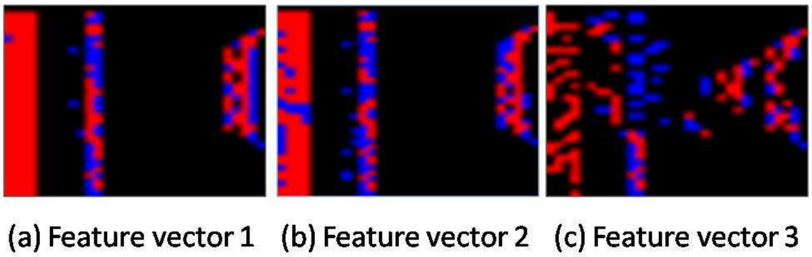

(a) Feature vector 1 (b) Feature vector 2 (c) Feature vector 3

FIGURE 6.16: Feature vectors for the SASL gesture *Goodbye*. Feature vector 1 is produced when a static camera position is used, feature vector 2 is produced when a shaking cellphone recording is normalised and feature vector 3 is produced when a shaking cellphone recording is not normalised.

Recall Chapter 3, where red indicates a region of strong skin value, and blue indicates a region of weak skin value. By comparing the location of skin values in the three feature vectors we can see that vector 3 shows scattered skin regions. This is due to the shifting of the signer within each frame of the video. Feature vector 1 and 2 show a more organised distribution of skin regions. It can be seen that feature vectors 1 and 2 look similar, this is due to the normalising method centring the signer within each video frame in feature vector 2. The result is that the HMM is able to discriminate and accurately classify this gesture. The scattered skin regions in feature vector 3 make HMM discrimination extremely difficult when training of the models was done using gestures which where recorded using a static camera position. The poor performance of this test can be attributed to the HMM's inability to classify a SASL gesture correctly when the position of the signer is not static in the video frame.

6.2.3.4 Further Testing

The above tests have focused on the three individual components of our Normalising method. We now test the performance of the method as a whole on all twenty SASL gestures. For testing our method, we use the same unseen gestures as were used in Table 6.10, however we randomly shift the signer within each frame in all of the test videos — making sure that hand regions do not fall outside of the image frame. In addition we simulated signers standing at varying distances from the camera by manually manipulating each frame by scaling each frame and cropping the images to fit a 320×240 resolution. Examples of frames that have been modified are shown in Figure 6.17. We plot the results of this experiment on a graph against the results in Table 6.10 for comparison. The graph is shown in Figure 6.18.

To evaluate how well these test results fit each other, we use the Pearson product moment correlation. The Pearson product moment correlation produces a high correlation of 0.99, indicating a near perfect positive linear relationship. The high correlation between these two experiments confirms that our normalising method performs well, normalising the three variations, listed below:

- Variations in signers with varying body dimensions,
- Variations in distances of the camera from the signer, and
- Variations in the position of the signer within the video frame.

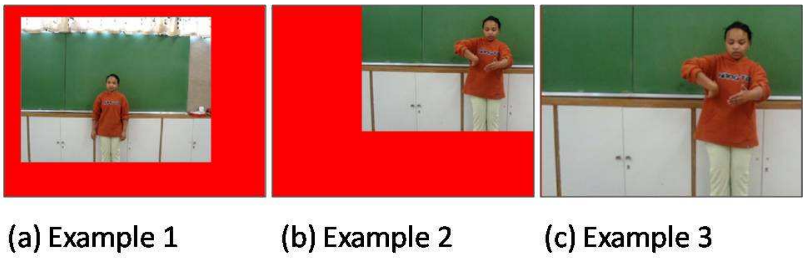


FIGURE 6.17: An example of video frames of various signers, manipulated to simulate the movement of a camera and signers standing at varying distances from the camera.

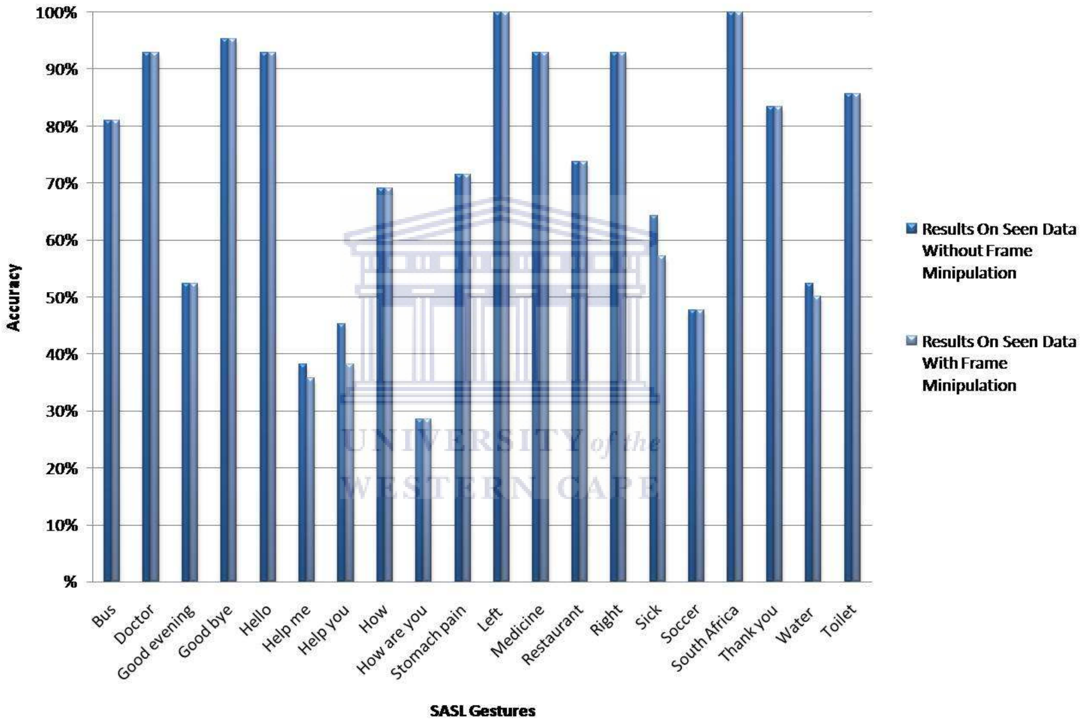


FIGURE 6.18: **Testing Performance:** The graph shows the results on seen data from Table 6.2 with the result of the manipulated data.

6.3 Summary

In this chapter we have presented our results and contributions to GR in the SASL project. We have tested several aspects of our system. First, we tested the performance of the trained HMMs on both seen and unseen data. While the system only achieved a classification rate of 69% on unseen gestures, the result can be attributed to our

HMMs inability to accurately discriminate between SASL gestures with similar hand movements.

Second, we tested each algorithm within our normalising method individually for variances that may occur when recording SASL gestures across different signers. All three algorithms perform exceptionally well.

Finally, we tested normalising as a whole by applying it on unseen data that were manipulated. The results were then compared to that of the un-manipulated testing set. The results of both test were similar, proving that our method adequately caters for variations that occur in video recordings of gestures across different signers.

Chapter 7

Conclusion and Direction for Further Research

In this thesis we have developed several methods which have contributed to the SASL objective of creating a full translation system for the deaf community. First, we have constructed a dynamic preprocessing method. This method consists of several important processes listed below:

1. Skin segmentation in the HSI colour space,
2. Background modelling using a non-adaptive approach,
3. Centring the signer in each video frame by locating the position of the signer's head. This method is part of the normalising method,
4. By using artist's techniques for drawing the human body, we are able to normalise input for HMM classification across different signers. This method is part of the normalising method, and
5. Using features found in each frame of the video, we construct a feature vector containing hand movement and temporal information for HMM classification.

7.1 Skin Segmentation

Accurate skin segmentation is paramount to the success of our system. Since we do not use any extra hardware such as datagloves or coloured markers, skin colour information in SASL gestures is used to classify gestures. To maximise the performance of our skin segmentation methods, a suitable colour space was required. In this thesis, we have

have shown the effectiveness of using the HSI colour space over the RGB colour space for skin segmentation in various lighting conditions. This has enabled our system to identify skin regions accurately and ultimately classify SASL gestures.

7.2 Background Modelling

By using a non-adaptive approach for background modelling, we achieve a quicker classification of the background and foreground. For the purpose of this thesis, the non-adaptive approach proved to be a more suitable means for background modelling. We illustrated that by combining several frame differencing techniques, we are able to eliminate background information which is not needed.

7.3 Normalising method

By centring the signer within each video frame, we have shown that the performance of the system is not adversely affected when recording a video from a moving device, such as a cellphone. Our research has demonstrated that by normalising an input video of a signed gesture, we are able to achieve comparable recognition rates of seen and unseen data.

7.4 Feature Vector

Using skin segmentation we are able to extract the location of a signer's hands with respect to the signer's body in consecutive frames in a video. This information, is used to construct a feature vector containing hand position information at any point in time during the signed gesture. This feature vector is used to classify a SASL gesture using HMMs. We have thus addressed our first, third and fourth research questions:

1. What are the appropriate features that will allow us to classify a signed gesture accurately and how will these be extracted?
2. What is an appropriate manner for describing hand movement in a video that can be used to train and test the system?
3. Given that no two signers have the same physical body structure, how do we dynamically normalise the features that would need to be extracted?

Second, we have successfully implemented HMMs which are able to discriminate between signed gestures with different hand movements. We have thus addressed our second research question: is it possible to interpret SASL gestures by only using hand gestures? Further we have illustrated that in cases where hand movements are unique in a SASL gesture, the hands' gestures contain sufficient information to interpret signs. In cases where hand movements are similar or the same in SASL gestures, our system is unable to classify them accurately. In these cases, the sole use of the pattern of hand motions to classify SASL gestures does not contain enough information for accurate classification.

7.5 Future Work

In order to differentiate between signed gestures with similar hand movements, future research can focus on extracting hand articulation and classifying it. This information can then be added to the feature vector. In addition facial expression information can also be included in the feature vector. This will provide a richer set of attributes which will make HMM discrimination of a signed gesture more refined.

While developing each of these steps no attempt was made to optimise these processes as our primary focus for this thesis was to attempt to answer our research questions and assess the feasibility of our research. Future research can therefore focus on the parallelising of these processes, which can greatly improve the overall time complexity of the system.

Finally, we (together with Mehrdad Ghaziasgar) have constructed the first working prototype translation system in the SASL project. The iSign system which has achieved recognition by winning the national Microsoft Imagine Cup competition in 2008 and representing South Africa in the international leg of the competition in 2009, can readily form the framework for future research in the SASL project at UWC. The merging of work such as that of Whitehill [4] and Segers [3] with the iSign system will push the SASL project ever closer to its goal of creating a full translation system and ultimately bridge the communication divide between the deaf and hearing communities.

Conducting this research for the SASL project at the University of The Western Cape has been enormously rewarding for this researcher; we hope that future researchers in the SASL project will strive to achieve its mandate and be equally rewarded.

Bibliography

[1] M. Turk and R. Cutler. View-Based Interpretation of Real-Time Optical Flow for Gesture Recognition. In *Proceedings of the International Third IEEE Conference on Face and Gesture Recognition*, pages 416–421, 1998.

[2] G. Rigoll, S. Eickler, and A. Kosmala. Hidden Markov Model Based Continuous Online Gesture Recognition. In *IEEE International Conference on Pattern Recognition*, pages 1206–1208, 1998.

[3] V. Segers. The Efficacy of the Eigenvector Approach to South African Sign Language Identification. In *SATNAC*, pages 363–366, 2009.

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

[5] W. C. Stokoe. *Sign Language Structure: An Outline of Visual Communication Systems of the American Deaf.* Linstok Pr, 1978.

[6] K. S. Liddell and R. E. Johnson. American Sign Language: The Phonological Base. In *Sign Language Studies*, pages 195–277, 1989.

[7] M. Lang and P. Morguet. Feature Extraction Methods for Consistent Spatio-Temporal Image Sequence Classification using Hidden Markov Models. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 4(2893), 1997.

[8] J. Brand and J. Mason. A Comparative Assesment of Three Approaches to Pixel Level Human Skin Detection. *International Conference on Pattern Recognition*, 1: 1056–1059, 2000.

[9] K. J. Han and A. H. Tewfik. Eigen-Image Based Video Segmentation and Indexing. In *ICIP*, pages 538–541, 1997.

[10] P. Hong, M. Turk, and T. S. Huang. Gesture Modeling and Recognition using Finite State Machines. In *Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition*, pages 410–415, 2000.

[11] S. Eickeler, A. Kosmala, and G. Rigoll. Hidden Markov Model Based Continuous Online Gesture Recognition. In *Int. Conference on Pattern Recognition*, pages 1206–1208, 1998.

[12] Z. J. Xiaojin. Segmenting Hands of Arbitrary Color, 2000.

[13] T. Starner, J. Weaver, and A. Pentland. A Wearable Computer Based American Sign Language Recognizer. *Lecture Notes in Computer Science*, 1458:84–90, 1998.

[14] Christian Vogler and Dimitris Metaxas. Adapting Hidden Markov Models for ASL Recognition by using Three-Dimensional Computer Vision Methods. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pages 156–161, 1997.

[15] C. Vogler and D. Metaxas. Toward Scalability in ASL Recognition: Breaking Down Signs into Phonemes, 1999.

[16] R. H. Liang and M. Ouhyoung. A Real-Time Continuous Gesture Recognition System for Sign Language. In *AFGR98*, pages 558–567, 1998.

[17] C. Rajah. Chereme-Based Recognition of Isolated, Dynamic Gestures from South African Sign Language with Hidden Markov Models. Master's thesis, University of The Western Cape, 2006.

[18] G. R. Coulter. Current Issues in ASL Phonology. *Phonetics and Phonology*, 3: 17–29, 1993.

[19] Y. Nam, T. Korea, and K. Wohn. Recognition of Space-Time Hand-Gestures using Hidden Markov Model, 1996.

[20] S. Kim and M. B. Waldron. Isolated ASL Sign Recognition System for Deaf Persons. In *IEEE Transactions on Rehabilitation Engineering*, pages 261–271, 1995.

[21] Q. Yang, R. Yang, J. Davis, and D. Nister. Spatial-Depth Super Resolution for Range Images. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[22] K. Fujimura and X. Liu. Sign Recognition using Depth Image Streams. pages 381–386, 2006.

[23] M. Assam and K. Grobel. Isolated Sign Language Recognition using Hidden Markov Models. In *IEEE Int. Conference on Systems, Man and Cybernetics*, pages 162–167, 1997.

[24] A. Pentland and T. Starner. Real-Time American Sign Language Recognition from Video using Hidden Markov Models, 1997.

[25] F. K. H. Quek. Toward a Vision-Based Hand Gesture Interface. *Proceedings of the International Virtual Reality System Technology Conf.*, pages 17–29, 1994.

[26] T. S. Huang and V.I. Pavlovic. Hand Gesture Modeling, Analysis and Synthesis. *Proceedings of the International Workshop on Automatic Face and Gesture Recognition*, pages 73–79, 1995.

[27] V. I. Pavlovic, R. Sharma, and T. S. Huang. Visual Interpretation of Hand Gestures for Human-Computer Interaction. *IEEE Trans. Pattern Analysis*, 19:677–695, 1997.

[28] J. A. Adam. Virtual Reality. *IEEE Spectrum*, 30(10):22–29, 1993.

[29] F. K. H. Quek, T. Mysliwiec, and M. Zhao. Finger Mouse: A Free Hand Pointing Interface. *Proceedings of the International Workshop on Automatic Face and Gesture Recognition*, pages 372–377, 1995.

[30] B. Dorner. Hand Shape Identification and Tracking for Sign language Interpretation. *In IJCAI Workshop on Looking at People*, 1993.

[31] A. Kirillov. Hand Gesture Recognition. *The Code Project*, 2008.

[32] R. Bowden, D. Windridge, T. Kadir, A. Zisserman, and M. Brady. A Linguistic Feature Vector for the Visual Interpretation of Sign Language. pages 391–401. Springer-Verlag, 2004.

[33] T. Starner and A. Pentland. Visual Recognition of American Sign Language using Hidden Markov Models. In *International Workshop on Automatic Face and Gesture Recognition*, pages 189–194, 1995.

[34] M. W. Kadous. Machine Recognition of Auslan Signs using Powergloves: Towards Large-Lexicon Recognition of Sign Language. *In Int. Workshop on the Integration of Gesture in Language and Speech*, pages 165–174, 1996.

[35] H. Rheingold. *Virtual Reality.* Touchstone Books, 1991.

[36] J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[37] C. Wang, W. Gao, and J. Ma. A Real-Time Large Vocabulary Recognition System for Chinese Sign Language, 2001.

[38] C. Wang, X. Chen, and W. Gao. Expanding Training Set for Chinese Sign Language Recognition. In *FGR '06: Proceedings of the 7th International Conference on Automatic Face and Gesture Recognition*, pages 323–328, Washington, DC, USA, 2006. IEEE Computer Society. doi: http://dx.doi.org/10.1109/FGR.2006.39.

[39] P. Vamplew and A. Adams. Recognition of Sign Language Gestures using Neural Networks. *Australian Journal of Intelligent Information Processing Systems*, 5(2): 94–102, 1998.

[40] S. Akyol and U. Canzler. An Information Terminal using Vision Based Sign Language Recognition. *ITEA Workship on Virtual Home Environments*, pages 61–68, 2002.

[41] R. Hassanpour, A. Shahbahrami, and S. Wong. Adaptive Gaussian Mixture Model for Skin Color Segmentation. *PWASET*, 31, July 2008.

[42] A. A. Argyros and M. I. A. Lourakis. Three-Dimensional Tracking of Multiple Skin-Colored Regions by a Moving Stereoscopic System. *Applied Optics*, 43(2):366–378, January 2004.

[43] P. Perez, C. Hue, J. Vermaak, and M. Cangnet. Color-Based Probabilistic Tracking. *European Conference on Computer Vision*, pages 661–675, 2002.

[44] A. A. Argyros and M. I. A. Lourakis. Real-Time Tracking of Multiple Skin-Colored Objects with a Possibly Moving Camera. *European Conference on Computer Vision*, 3:368–379, 2004.

[45] V. Vezhnevets and A. Andreeva. A Comparative Assessment of Pixel-Based Skin Detection Methods. In *Technical Report 375, MIT Media Laboratory*, 2005.

[46] B. Jedynak, H. Zheng, M. Daoudi, and D. Barret. Maximum Entropy Models for Skin Detection. In *Proceedings Third Indian Conference on Computer Vision, Graphics and Image Processing*, pages 276–281, 2002.

[47] H. Kruppa, M. A. Bauer, and B. Schiele. Skin Patch Detection in Real-World Images. In *Annual Symposium for Pattern Recognition of the DAGM, Springer LNCS 2449*, pages 109–117, 2002.

[48] A. Diplaros, T. Gevers, and N. Vlassis. Skin Detection using the EM Algorithm with Spatial Constraints. In *Systems, IEEE International Conference on Man and Cybernetics*, volume 4, pages 3071–3075, 2004.

[49] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*, volume 1. Prentice-Hall Inc, second edition, 2002.

[50] F. Porikli and O. Tuzel. Human Body Tracking by Adaptive Background Models and Mean-Shift Analysis. In *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, 2003.

[51] C. Stauffer and W. E. L. Grimson. Adaptive Background Mixture Models for Real-Time Tracking. volume 2, pages 2246–2252, 1999.

[52] P. Viola and M. Jones. Rapid Object Detection using a Boosted Cascade of Simple Features. pages 511–518, 2001.

[53] P. Viola and M. Jones. Robust Real-Time Object Detection. In *International Journal of Computer Vision*, number 2, pages 137–154, 2001.

[54] OpenCV Wiki, January 2009. URL `http://opencv.willowgarage.com/wiki`.

[55] OpenCV on Sourceforge, January 2009. URL `http://sourceforge.net/projects/opencvlibrary`.

[56] Official OpenCV Usergroup, January 2009. URL `http://tech.groups.yahoo.com/group/OpenCV`.

[57] D. Marr and E. Hildreth. Theory of Edge Detection. *Proceedings of the Royal Society of London. Series B, Biological Sciences*, 207:187–217, 1980.

[58] R. Stemp. *The Secret Language of the Renaissance.* Duncan Baird Publishers, 2006.

[59] The Vitruvian Man, Febuary 2009. URL `http://www.world-mysteries.com/sci_17_vm.htm`.

[60] The Vitruvian Man, Febuary 2009. URL `http://www.associatedcontent.com/article/2107179`.

[61] The Vitruvian Man, Febuary 2009. URL `http://en.wikipedia.org/wiki/Vitruvian_Man`.

[62] R. L. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. In *Proceedings of the IEEE*, volume 77, pages 257–286, 1989.

[63] D. O. Tanguay. Hidden Markov Models for Gesture Recognition. Master's thesis, MIT, 1993.

[64] Viterbi Algorithm, April 2008. URL `http://en.wikipedia.org/wiki/Viterbi_algorithm`.