

UNIVERSITY OF THE WESTERN CAPE

**Robust South African Sign Language
gesture recognition using hand motion
and shape**



A thesis submitted in fulfillment for the
degree of Master of Science

in the
Faculty of Science
Department of Computer Science

Supervisor: Mehrdad Ghaziasgar
Co-supervisor: James Connan

February 2014

Declaration



I, Ibraheem Frieslaar, declare that this thesis “Robust South African Sign Language gesture recognition using hand motion and shape” is my own work, that it has not been submitted before for any degree or assessment at any other university, and that all the sources I have used or quoted have been indicated and acknowledged by means of complete references.

Signature:

Date:

“Seek knowledge from the cradle to the grave.”



Abstract

Research has shown that five fundamental parameters are required to recognize any sign language gesture: hand shape, hand motion, hand location, hand orientation and facial expressions. The South African Sign Language (SASL) research group at the University of the Western Cape (UWC) has created several systems to recognize sign language gestures using single parameters. These systems are, however, limited to a vocabulary size of 20 – 23 signs, beyond which the recognition accuracy is expected to decrease. The first aim of this research is to investigate the use of two parameters – hand motion and hand shape – to recognise a larger vocabulary of SASL gestures at a high accuracy. Also, the majority of related work in the field of sign language gesture recognition using these two parameters makes use of Hidden Markov Models (HMMs) to classify gestures. Hidden Markov Support Vector Machines (HM-SVMs) are a relatively new technique that make use of Support Vector Machines (SVMs) to simulate the functions of HMMs. Research indicates that HM-SVMs may perform better than HMMs in some applications. To our knowledge, they have not been applied to the field of sign language gesture recognition. This research compares the use of these two techniques in the context of SASL gesture recognition. The results indicate that, using two parameters results in a 15% increase in accuracy over the use of a single parameter. Also, it is shown that HM-SVMs are a more accurate technique than HMMs, generally performing better or at least as good as HMMs.

Keywords

Hidden Markov Models, Support Vector Machines, Hidden Markov Support Vector Machine, Face Detection, Skin Detection, Background Subtraction, Hand Shape Recognition, Hand Motion

Acknowledgements

First and foremost, all praise is given to Allah, the most merciful and most gracious. Thank you for giving me a healthy body and allowing me the ability to seek knowledge.

I would like to thank my parents for ensuring that I receive a proper education and having the patience to allow me to further my studies.


My sincerest appreciation is given to Mr James Connan for having given me the opportunity to work for him or, as he would say, pretend to work. If it wasn't for you Sir, I would probably be standing in some clothing store wasting my life away.

I would like to thank Mr Mehrdad Ghaziasgar for reminding me that we are value adders and not value seekers. Thank you for always pushing me to not achieve the best, but to strive for something greater. It has been a pleasure to have you as a supervisor and receive your words of wisdom.

A personal thank you goes to Dane Brown for all the years of friendship. You are not just a friend, but I see you as a brother. Thank you for the endless advice, even though it seems I never listen to you, and thank you for putting up with all my drama and problems. To my brother from another mother, Gary Jansen, thank you for reminding me that there is life out there, and for all the crazy and epic moments in the Jungle.

I would also like to thank my lab mates Jihad Mohammad, Warren Nel, Diego Mushfieldt, Kenzo Abrahams, Imran Achmed and Roland Foster for all the moments in the lab, especially being trolled every day. It has been an honour serving with you guys.

Contents

Declaration of Authorship	i
Abstract	iii
Keywords	iii
Acknowledgements	iv
List of Figures	viii
List of Tables	x
Abbreviations	xi
 <p>UNIVERSITY <i>of the</i> WESTERN CAPE</p>	
1 Introduction	1
1.1 Background and Motivation	1
1.2 Research Question	4
1.3 Research Objectives	4
1.4 Premises	5
1.5 Thesis Outline	5
2 Related Work	7
2.1 Articulated Object Recognition	7
2.1.1 Template Matching Techniques	8
2.1.2 Machine Learning Techniques	12
2.1.3 Summary of Articulated Object Recognition	15
2.2 Gesture Recognition using Hidden Markov Models	15
2.2.1 Glove-Based Systems	16
2.2.2 Hardware-Based Systems	18
2.2.3 Vision-Based Systems	19
2.3 Hidden Markov Support Vector Machines (HM-SVMs)	21
2.4 Summary of the Related Work	23
3 Image Processing in Gesture Recognition	24

3.1	Face Detection	24
3.1.1	Haar-like Wavelet Detection	25
3.1.2	Integral Image	26
3.1.3	AdaBoost	26
3.1.4	Construction of a Rejection Cascade of Weak Classifier Nodes	27
3.1.5	Testing and Results on the Face Detection Method	28
3.2	Skin Detection	28
3.2.1	RGB Colour Space	29
3.2.2	Normalised RGB Colour Space	29
3.2.3	HSV Colour Space	30
3.2.4	YCbCr colour space	31
3.2.5	Selecting An Appropriate Colour Space for Skin Detection	32
3.2.6	Skin Model	33
3.3	Background Subtraction	34
3.3.1	Static Background Subtraction	35
3.3.2	Frame Differencing	35
3.3.3	Statistical Background Subtraction	36
3.3.4	A Comparison of the Background Subtraction Techniques	37
3.4	Image Smoothing Using Gaussian Blur	38
3.5	Feature Normalisation by Applying a Minimum Bounding Rectangle	40
3.6	Summary and Conclusion	41
4	Support Vector Machines, Hidden Markov Models and a Hybrid Approach	43
4.1	Support Vector Machines	43
4.1.1	Kernel Functions	48
4.1.2	SVM Multi-Class Classification	49
4.1.2.1	One-vs-Others	49
4.1.2.2	One-vs-One	50
4.1.2.3	Directed Acyclic Graph (DAG) Technique	50
4.2	Hidden Markov Models (HMMs)	51
4.2.1	Markov Chain	51
4.2.2	Formulation of HMMs Based on Markov Models	54
4.2.3	The Three Basic Problems for HMMs	56
4.2.3.1	A Solution to the Evaluation Problem	57
4.2.3.2	A Solution to the Decoding Problem	57
4.2.3.3	A Solution to the Learning Problem	59
4.3	Hidden Markov Support Vector Machines (HM-SVMs)	60
4.3.1	Input and Output Mappings via Joint Feature Functions	60
4.3.2	Hidden Markov Chain Discriminants	62
4.3.3	Hidden Markov Perceptron Learning	63
4.3.4	Hidden Markov Support Vector Machine	64
4.4	Summary	65
5	Design and Implementation of the Gesture Recognition System	66
5.1	Feature Extraction	66
5.1.1	Skin Detection	68

5.1.2	The Locating Procedure	69
5.1.3	The Correction Procedure	71
5.1.4	Hand Shape Recognition	71
5.1.5	Hand Motion Extraction	73
5.2	Feature Vector	74
5.3	Conclusion	75
6	Experimental Results and Analysis	76
6.1	Experimental Setup	77
6.2	Testing of the Hand Tracking Component	77
6.2.1	Hand Tracking Experimental Procedure	77
6.2.2	Results and Analysis	78
6.3	Training and Testing of the Hand Shape Recognition Component	79
6.3.1	Training Set	79
6.3.2	Training and Optimisation of the Radial Basis Function Kernel Parameters	80
6.3.3	Hand Shape Recognition Experimental Procedure	81
6.3.4	Results and Analysis	82
6.4	Testing of the Proposed SASL Gesture Recognition System	83
6.4.1	Data Set	84
6.4.2	Feature Vector Comparison Experiment	86
6.4.2.1	Feature Vector Comparison Experimental Procedure	86
6.4.2.2	Results and Analysis	87
6.4.3	HMM versus HM-SVM Comparison Experiment	90
6.4.3.1	HMM versus HM-SVM Comparison Experimental Pro- cedure	90
6.4.3.2	Results and Analysis	91
6.5	Conclusion	94
7	Conclusion	96
7.1	Directions for Future Work	97
7.1.1	Optimising HM-SVMs	97
7.1.2	Parallel HMMs	97
7.1.3	Including Additional Sign Language Parameters	97
7.2	Concluding Remarks	98
A	Additional Test Results	99
	Bibliography	102

List of Figures

2.1	The skin image of the hand with the green boxes representing the matched features. [99].	9
2.2	An overview of Mohr and Zachmann’s system [48].	10
2.3	Silhouettes approximated by a set of rectangles. The top row depicts approximating rectangles inscribing the hand shape and the bottom row depicts approximating rectangles circumscribing the hand shape [48].	10
2.4	Shimada <i>et al.</i> ’s hand shape estimation by using distance measurement [69].	11
2.5	Schreer <i>et al.</i> ’s approach to recognising the hand shapes [65].	12
2.6	The visual results of Schreer <i>et al.</i> ’s system correctly detecting the hands [65].	12
2.7	Stergiopoulou and Papamarkos’ hand shape recognition system using a SGONG [74].	13
2.8	Sato <i>et al.</i> ’s process to obtain normalised image of the hand [93].	14
2.9	The 10 hand shapes recognized by Li’s system [44].	14
2.10	Coloured glove used by Sandjaja and Marcos [63].	16
2.11	Zhang <i>et al.</i> ’s system demonstrating: (Left column) Background subtraction and (Right column) locating the hands [97].	17
2.12	The tracking system to locate and track the hands in 3D from a front, side and top view using multiple 3D cameras [90].	18
2.13	The generated HMM by Matsuo <i>et al.</i> ’s system for the word “winter clothing” in an unspecified sign language [47].	20
2.14	HMM model for gesture “Alexandria” by Youssif <i>et al.</i> [95].	21
3.1	The four types of Haar-like wavelet features [88].	25
3.2	Haar-like wavelet feature detection used to detect if a feature is present [88].	25
3.3	Haar-like wavelet feature detection [88].	26
3.4	Possible features selected by AdaBoost [88].	27
3.5	The rejection cascade of weak classifier nodes [88].	27
3.6	Rejection cascades used to determine face correctly.	28
3.7	A visual representation of the RGB colour space cube [62].	29
3.8	Conversion from the default RGB colour space to a normalised image.	30
3.9	Graphical depiction of the cylindrical colour space representation of the HSV colour space [62].	31
3.10	Graphic representation of a skin histogram.	33
3.11	Results of the skin detection algorithm.	34
3.12	The process of determining the minimum bounding rectangle of the hand contour [44].	40

4.1	The optimal solution hyperplane and other solution hyperplanes to separate the two data sets S^- and S^+ [13].	45
4.2	The optimal hyperplane with the maximum margin [13].	45
4.3	A 4-class Directed Acyclic Graph (DAG) with a rejection class at each node.	51
4.4	A Markov chain with states S_1, S_2, \dots, S_5 [57].	52
4.5	Three possible solutions to the coin toss modelling problem [57].	54
4.6	Illustration of the natural language parsing model [50].	61
4.7	The interaction between neighbouring label states [50].	62
4.8	Dual perceptron algorithm [3].	63
5.1	Broad overview of the proposed gesture recognition system.	66
5.2	Overview of the feature extraction component.	67
5.3	The skin image.	69
5.4	The skin image after Gaussian smooth has been applied.	69
5.5	The moving skin image.	70
5.6	The side of the hand located.	70
5.7	The right side of the hand located.	70
5.8	The extent of the search area to locate the top and bottom of the hand.	70
5.9	Various hand shapes that are correctly isolated by a dynamic box.	71
5.10	The correction procedure.	72
5.11	Normalising the hand contours with a minimum bound box.	72
5.12	A scaled binarised hand contour image.	73
5.13	A dynamic grid centred on and proportional to the face used to represent the location of the hands.	73
6.1	Experimental setup.	77
6.2	The hand motion sequence performed by test subjects in the hand tracking experiment.	78
6.3	The 10 unique hand shapes in the 35 SASL gestures (Top row, left to right: shapes 1 to 5) (Bottom row, left to right: shapes 6 to 10).	80
6.4	The five subjects used to train the hand shape recognition component.	80
6.5	The 15 subjects for gesture recognition (Top row, left to right: Subjects 1 – 5) (Middle row, left to right: Subjects 6 – 10) (Bottom row, left to right: Subjects 11 – 15).	84
6.6	Graph of gesture recognition accuracy of HMM with only motion versus HMM with motion and hand shape.	89
6.7	Graph of gesture recognition accuracy of HMM vs HM-SVM.	93

List of Tables

5.1	A snippet of the motion feature vector.	75
6.1	Hand tracking accuracy results.	79
6.2	C and γ optimisation results and accuracy for various hand contour resolutions.	81
6.3	Hand shape recognition accuracy results.	82
6.4	Hand shape recognition accuracy results per subject.	83
6.5	Description of the selected gestures.	85
6.6	Average accuracy of the HMM with motion and the HMM with motion and hand shape per gesture.	88
6.7	Average accuracy of the HMM with motion and the HMM with motion and hand shape per subject.	90
6.8	Average accuracy between the HMM with motion and hand shape and the HM-SVM per gesture.	92
6.9	Overall average accuracy per subject.	93
A.1	Number of correctly recognised videos by the HMM using only motion and the HMM using motion and hand shape, per gesture.	100
A.2	Number of correctly recognised videos by the HMM and HM-SVM, both using hand motion and hand shape, per gesture.	101

Abbreviations

2D	Two Dimensional
3D	Three Dimensional
ASL	American Sign Language
AU	Action Unit
BSL	British Sign Language
CSL	Chinese Sign language
CCA	Connected Component Analysis
DAG	Directed Acyclic Graph
FSL	Filipino Sign Language
FPS	Frames Per Second
GMM	Gaussian Mixture Models
HMMs	Hidden Markov Models
HM-SVM	Hidden Markov Support Vector Machine
HSV	Hue Saturation Value
LibSVM	Library of Support Vector Machines
NN	Neural Networks
PaHMM	Parallel Hidden Markov Models
RAM	Random Access Memory
RBF	Radial Basis Function
SASL	South African Sign Language
SL	Sign Language
SVM	Support Vector Machines

Chapter 1

Introduction

1.1 Background and Motivation

Imagine a scenario in which a hearing person attempts to interact with a deaf person. At present, this is a difficult undertaking since the hearing person does not have any knowledge on how to communicate in sign language (SL). Millions of Deaf people, not just in South Africa, but across the globe, find the task of interacting and communicating with hearing people extremely difficult. The reverse is also true. This leaves deaf people demoralised and feeling unwanted in society [9].

A common misconception amongst the hearing is that the Deaf use a universal Sign Language to communicate with each other. However, different countries have different sign languages such as American Sign Language (ASL), British Sign Language (BSL) and South African Sign language (SASL), amongst others. These sign languages are all completely different from each other [75]. The second incorrect assumption amongst the hearing is that the Deaf are able to read and write in spoken languages. Sign languages have an entirely different set of grammatical rules and syntactic structure to spoken languages [46]. As a result, the Deaf are unable to read and write in spoken languages. Thus, the question arises: how can the gap in communication between hearing people and the Deaf be bridged?

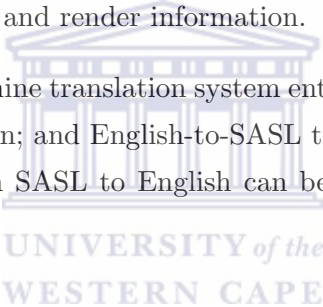
The solution to bridging the gap up until now has been the use of skilled sign language translators. However, SASL translators are rare. This results in very expensive rates for translation services. The South African Deaf are mostly unable to afford such services [28]. A further issue with this solution is that, in some cases, the Deaf may feel that the use of a translator is invasive of their privacy such as in a medical or psychological consultation. As such, this solution to the gap in communication cannot be used as a

day-to-day and common place solution, but more for very specific formal occasions such as in meetings involving officials of Deaf organisations.

The “Integration of Signed and Verbal Communication: South African Sign Language Recognition and Animation” project at the department of Computer Science at the University of the Western Cape (UWC) [28], which is also called the South African Sign Language (SASL) group, of which this research is part, aims to develop an automatic natural machine translation system that can translate between English and SASL. This aims to not only aid in the communication between the deaf and hearing, but also aid in deaf-to-deaf communication.

In order to achieve a form of interaction that is as natural to the Deaf as possible, one of the key aims of the SASL group is to make use of equipment that is not cumbersome. As such, the group aims to avoid solutions that require the Deaf to wear specialised equipment such as Data Gloves or gloves with colour-coded markers when capturing sign language gestures. Where possible, commodity hardware – in most cases a mobile phone – is used to capture and render information.

The conceptual SASL machine translation system entails carrying out two distinct tasks: SASL-to-English translation; and English-to-SASL translation. A basic overview of the translation procedure from SASL to English can be described briefly in the following steps:

- 
1. A video recording is made of a person performing SASL.
 2. Image processing techniques are used to extract vital SASL semantic information from the recorded video.
 3. Machine learning techniques are used to interpret this information, converting it to English text.
 4. A text-to-speech engine is used to synthesise the English text into English audio.

This procedure enables the hearing person to understand what the Deaf person says. The procedure involved in translating English to SASL can be described in the following steps:

1. An audio recording of the person speaking English is captured.
2. Speech recognition technology is used to convert the audio into English text.
3. Machine learning techniques are used to interpret the English text, converting it into SASL semantic information.

4. The semantic information is used to synthesise SASL by means of a 3D avatar performing the required SASL phrase.

This allows the hearing person to respond to the deaf person. The SASL group has, to date, developed the first prototype system in the world that can translate from English to SASL and SASL back to English, with mobile phones as the delivery mechanism of the system [28].

This research involves the vital sub-task of SASL-to-English translation involving the use of image processing techniques to extract semantic information from SASL videos. There are five parameters that define any sign language (SL) gesture which need to be extracted from SASL videos. These parameters are: facial expressions, hand shape, hand motion, hand orientation and hand location [32, 44]. Research has been conducted into the recognition of these five parameters by the SASL group. Whitehill developed a robust facial expression recognition system [92]. Li developed a hand shape recognition and estimation system [44]. Naidoo [49] and Rajah [58] developed hand motion recognition systems. Achmed [1] and Brown [9] developed hand location recognition systems.

All of these systems focused on the recognition of only a single SL parameter from SASL videos. Naidoo and Rajah's systems additionally attempted to use the extracted SASL parameter - in their case motion - to characterise and recognise SASL phrases. Both systems made use of Hidden Markov Models (HMMs) to recognise motion sequences in order to classify gestures. Rajah's system recognised 23 SASL phrases at an average accuracy of 68%. Naidoo's system recognised a similar number of SASL gestures, 20 in total, at an average recognition accuracy of 69%.

While both of these systems proved successful as proofs-of-concept, the use of a single parameter to characterise SASL gestures is expected to limit the size of the vocabulary that can be recognised at a high accuracy. The first aim of this research is to pioneer research into the use of two parameters - hand shape and hand motion - to recognise SASL gestures. The use of two parameters is expected to enable the recognition of a greater number of SASL phrases at a high accuracy.

The majority of the related work in the field of sign language gesture recognition using these two parameters makes use of Hidden Markov Models (HMMs) to classify gestures. This is due to the suitability of HMMs to temporal pattern matching problems [55, 56]. A relatively new machine learning technique known as the Hidden Markov Support Vector Machine (HM-SVM) combines the use of HMMs and Support Vector Machines (SVMs) to form a hybrid technique. The new technique appears to provide the benefits of both HMMs, i.e. suitability to temporal pattern matching, and SVMs, i.e. kernel-based learning and margin maximisation, in one technique. Research indicates that HM-SVMs

may perform better than HMMs in some applications [3, 85]. To our knowledge, they have not been applied to the field of sign language gesture recognition. Therefore, the second aim of this research is to compare the use of these two techniques in the context of SASL gesture recognition.

1.2 Research Question

The following research questions can be specified based on the previous section:

1. How should a combination of hand shape and hand motion be used to recognise more SASL phrases with a higher accuracy than hand motion alone?
2. How do HMMs and HM-SVMs compare in terms of accuracy when applied to SASL gesture recognition using hand shape and motion as gesture descriptors?

1.3 Research Objectives

1. Implement a hand tracking strategy that locates and tracks both hands simultaneously.
2. Implement a hand shape recognition strategy to recognise hand shapes in real-time.
3. Train a set of HMMs to recognise a set of SASL gestures using both hand shape and motion as a feature vector.
4. Train a set of HMMs to recognise the set of SASL gestures using only hand motion as a feature vector.
5. Compare the accuracy of the two gesture-descriptor approaches.
6. Train a set of HM-SVMs to recognise the set of SASL gestures using both hand shape and motion as a feature vector.
7. Compare the accuracy of the HMM and HM-SVM-based strategies to determine which technique produces the best result in the context of SASL gesture recognition.

1.4 Premises

Hand tracking is a complex problem [68] which has, on its own, been the subject of numerous research studies. The main obstacle to robust hand tracking in a monocular video feed is occlusions of the two hands. The challenge is to determine which hand is the left hand and which is the right hand after the hands emerge from an occluded state. This challenge may be alternatively framed as determining whether the hands have returned to their respective sides or crossed after an occlusion. Achieving such robust hand tracking is beyond and outside the scope of this research, which focuses on the gesture recognition aspect. This research, therefore, does not consider SASL gestures in which the hands cross over. However, gestures in which the hands occlude and subsequently move back to their respective sides of the body are considered.

A further challenge is presented when one or both hands occlude the face. The challenge is to separate the pixels representing the face and those representing the hand(s). This makes it difficult to carry out hand shape recognition towards the objective of this research: gesture recognition. Therefore, gestures in which one or both hands occlude the face are not considered in this research, and are considered beyond the scope of the research.

The SASL dictionary of the Fulton School for the Deaf [32] was analysed to determine the percentage of SASL signs in which the hands cross over or occlude the face. The dictionary contains 732 of the most common SASL phrases. The analysis revealed that 72% of these signs are performed with the hand(s) on their respective sides of the body. An additional 22% of these signs are performed with the hands occluded in the centre of the body. Only 3% of the signs are performed with one or both hands crossed over to the other side of the body. An additional 3% of signs are performed with the hand(s) in front of the face. This allows for the formulation of a strategic simplification to the problem as follows: The assumption that there will only be one hand on either side of the image at any time and the hand(s) will not occlude the face is justified and sufficient to represent the majority of SASL phrases. This assumption still makes it possible to recognise 94% of the most common SASL phrases and only excludes 6% of these phrases.

1.5 Thesis Outline

The remainder of the thesis is arranged as follows:

Chapter 2: *Related Work*: This chapter reviews existing literature in the fields of articulated object recognition and gesture recognition using Hidden Markov Models (HMM).

It examines the various approaches and algorithms used for articulated object recognition and gesture recognition. The advantages and disadvantages of these approaches are discussed.

Chapter 3: *Image Processing in Gesture Recognition:* This chapter discusses the components that form part of the proposed gesture recognition system in detail. This discussion forms the basis of the description of the proposed implementation and the optimisation thereof in subsequent chapters.

Chapter 4: *Support Vector Machines, Hidden Markov Models and a Hybrid Approach:* This chapter discusses the theory behind SVMs and HMMs in detail. The strategy used by each approach to achieve data classification is explained. This is used as a basis for the description of a hybrid approach that combines elements of both methods into a Hidden Markov Support Vector Machine (HM-SVM) method which is explained in detail.

Chapter 5: *Design and Implementation of the Gesture Recognition System:* This chapter discusses the design of the gesture recognition system proposed in this research. The methods used to extract hand motion and hand shape features to form a feature vector are described.

Chapter 6: *Experimental Results and Analysis:* This chapter describes the experiments carried out in order to answer the research questions posed in this chapter.

Chapter 7: *Conclusion:* This chapter concludes the thesis, highlights the contributions made towards the research and provides directions for future work.

Chapter 2

Related Work

This chapter reviews the previous research done on articulated object recognition, gesture recognition using Hidden Markov Models (HMMs) and the application of Hidden Markov Support Vector Machines (HM-SVMs) to research. The chapter motivates the use of a suitable hand shape recognition strategy as a pre-cursor to the gesture recognition system proposed in this research. It details other gesture recognition systems that use HMMs and demonstrates that HMMs are a suitable and high accuracy technique for this task. It also reviews previous research that has applied Hidden Markov Support Vector Machines (HM-SVMs). The chapter details studies that have used HM-SVMs for other purposes and demonstrates that they can perform better than HMMs in some contexts.

The chapter is organised as follows: Section 2.1 discusses previous work done to address the problem of articulated object recognition. It concludes with the selection of an appropriate hand shape recognition strategy for the proposed gesture recognition system. Section 2.2 details the previous work done to recognise gestures using Hidden Markov Models (HMMs). Section 2.3 reviews discusses those studies that have used Hidden Markov Support Vector Machines (HM-SVMs). To our knowledge, HM-SVMs have not been applied to gesture recognition. They have, however, been adopted in other systems which, will is discussed in 2.3.

2.1 Articulated Object Recognition

Articulated objects are those objects that have components attached via joints and these components can move with respect to one another, such as with the hand and fingers. Articulated object recognition attempts to determine the shape of an articulated object.

The methods that can be used to recognise the shape of articulated objects can broadly be divided into those that use template matching and those that use machine learning techniques.

Template matching techniques use pre-constructed example-based images of the hand to compare and match with an observed image from a camera. The match between the examples and the observed image is carried out using a data-driven metric.

Machine learning techniques use trained classifiers which are constructed using a predefined feature vector [13]. Various machine learning techniques can be used for classification such as Support Vector Machines (SVMs) and Neural Networks (NNs).

Section 2.1.1 discusses the various systems that recognise hand shapes using template based approach and Section 2.1.2 focuses on the systems that use machine learning techniques for this purpose.

2.1.1 Template Matching Techniques

Template matching techniques are techniques used in image processing to find small parts of an image that matches a template image [10]. They are data-driven methods that use a matching heuristic to determine a match between the template and query image.

Zhou and Huang [99] proposed a technique to recognise articulated objects using the Okapi-Chamfer matching algorithm based on the inverted index technique. The inverted index technique was originally used to organise collections of text documents. Using this method, only documents that contain query terms are accessed and used for matching.

To be able to use this approach in image processing, the system makes use of an image database based on a lexicon of local visual features by clustering the features extracted from the training images. Features are extracted from the query image and quantised based on the lexicon. The inverted index is used to identify the subset of the trained images using a non-zero matching score. The scores are evaluated by using a modified Okapi weighting formula with the Chamfer distance.

Their feature extraction technique comprised of four steps:

1. The hand region is segmented based on the skin colour histogram.
2. Discriminative patches are computed along the boundary between the background and hand region.

3. Local orientation histogram features are extracted from each patch.
4. Each feature is quantised based on a pre-trained lexicon.

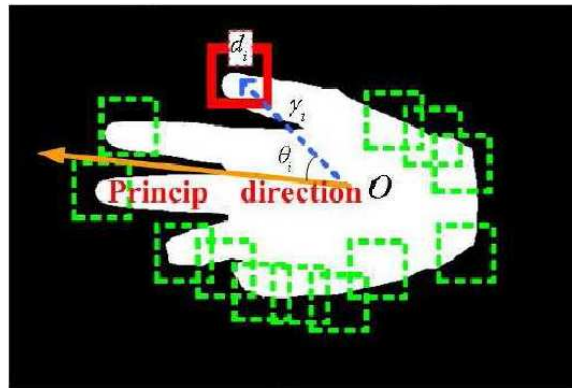


FIGURE 2.1: The skin image of the hand with the green boxes representing the matched features. [99].

Figure 2.1 illustrates the skin image of the hand with the green boxes representing the matched features.

The system was tested on a database of 16384 images of hands in different postures. The images were of two types. The first type of images were generated by a 3D kinematic hand model from several view points and rotation angles. This resulted in 500 synthesised query images in which the finger configurations were randomly sampled from 1024 possible hand shapes in 16 different viewing angles. The second type of images consisted of real world images with the finger configurations manually labelled and controlled.

The system was shown to be able to recognise hand shapes at a computational speed of 3 seconds per query in a database of 16384 training images. The researchers stated that the focus of the research was computational speed and not recognition accuracy and, as such, no indication of the accuracy was provided.

Mohr and Zachmann [48] proposed a novel technique for adaptive template matching based on the silhouette area of the articulated object. The silhouette area is approximated by using a small set of axis-aligned rectangles. Figure 2.2 illustrates an overview of the system.

The pre-processing procedure consists of the following. The skin pixels within the image are detected using a static range function. The image is converted to grayscale and binarised to obtain the hand contour. A set of axis-aligned rectangles are used to approximate the shape of the resulting image. Two sets of rectangles are drawn. The first set inscribes the hand shape with rectangles as seen in the first row in Figure 2.3 and the second set circumscribes the hand shape as seen in the second row of the figure.

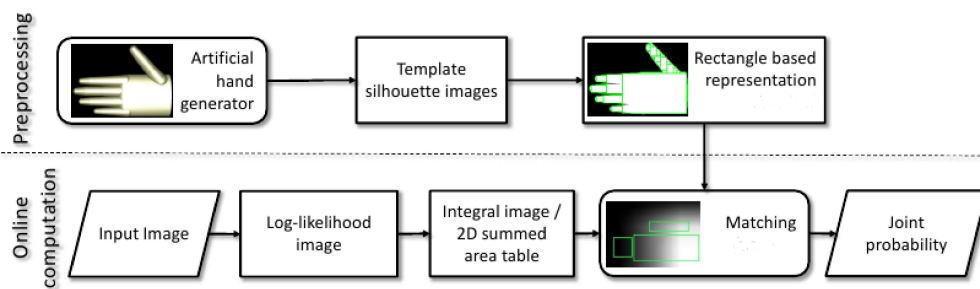


FIGURE 2.2: An overview of Mohr and Zachmann's system [48].

The result is compared with a database of pre-generated synthetic hand images that are pre-processed, inscribed and circumscribed in the same manner, to identify a match.

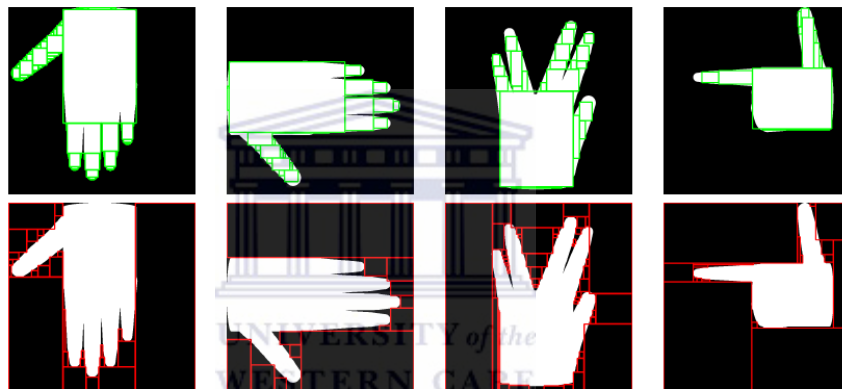


FIGURE 2.3: Silhouettes approximated by a set of rectangles. The top row depicts approximating rectangles inscribing the hand shape and the bottom row depicts approximating rectangles circumscribing the hand shape [48].

An experiment was carried out to determine the accuracy of the system in recognising three hand shape classes: open hand, pointing hand and open hand with abducting fingers. Three data sets representing each class were used in experimentation. Data set 1 consisted of 1536 templates containing an open hand at different rotation angles. Data set 2 consisted of a pointing hand rendered at the same rotation angles as data set 1. Data set 3 consisted of 3072 templates with an open hand and abducting fingers. For evaluation, an input image resolution of 256×256 was used and each template at 5 different scaling's from 70×70 pixels up to 200×200 pixels were used and compared. The approach was shown to always find the correct location of the hand in the input image.

It was shown that, the higher the rectangle approximation accuracy, the higher the matching quality of images with the open hand ranging from 91% to 98% accuracy. A tradeoff to a higher rectangle approximation accuracy is a reduction in computational

speed. A similar trend was observed for the open hand with abducting fingers hand shape. It was, however, noted that the matching quality in the pointing hand data set is very high even at low rectangle approximation accuracy, centred at around 98%. The computational speed of the system is determined to be between 1–10 Frames Per Second (FPS) – less than real-time.

Shimada *et al.* proposed a technique to recognise hand shapes using a distance-to-edge method [69]. The method consists of placing 256 points located on the binary silhouette of the hand as seen in Figure 2.4. The distance of each point to the centre of gravity of the silhouette is computed. A graph of these distances is generated as seen in Figure 2.4. For every graph generated, a unique hand shape can be assigned to it. To obtain scale invariance, the plotted distances are shifted to start with the most significant peak. It is important that the points are consistently sampled in either a clockwise or counter-clockwise fashion. The system was tested on a 6 node cluster and achieved a real-time performance accuracy of 30 Frames Per Second (FPS). The recognition accuracy is not clear from the literature. However, the researchers comment that the approach struggles to distinguish between hand shapes.

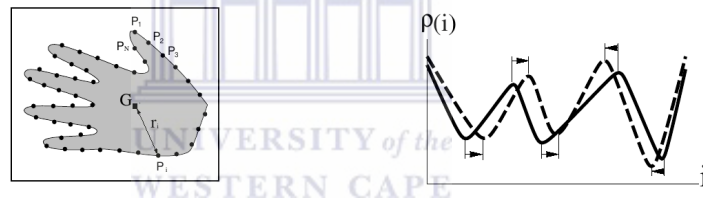
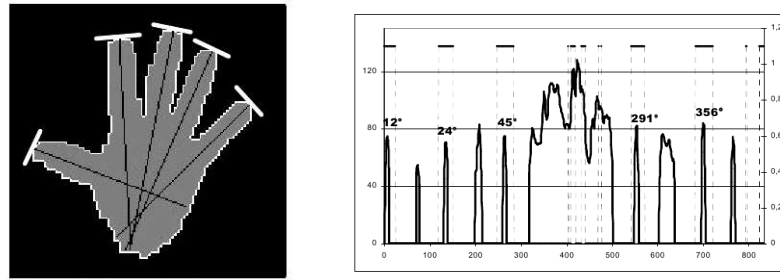


FIGURE 2.4: Shimada *et al.*'s hand shape estimation by using distance measurement [69].

Schreer *et al.*'s approach was similar [65] to Shimada *et al.*'s approach. However, a different distance function was used. At each point on the hand contour, the normal to the tangent of the contour at that point is determined. Following the path of the normal, the distance to the contour opposite the point is computed and plotted on a graph.

Figure 2.5 (a) illustrates a hand shape with tangent lines drawn at the finger tips and Figure 2.5 (b) illustrates the graph of the distances produced from that hand shape. Using this method, they are able to classify 13 different hand shapes. The system was tested on a desktop computer using a video camera with a resolution of 640×480 pixels as input. The system operates at less than real time, at a speed of 2 FPS. However, the recognition accuracy is not provided. Visual results are presented by the researchers to illustrate that the system has a high recognition accuracy. The visual results are provided in Figure 2.6.



(a) Normal to the tangent along the contour for the fingers. (b) Distance function from the silhouette generated from the hand

FIGURE 2.5: Schreer *et al.*'s approach to recognise the hand shapes [65].



FIGURE 2.6: The visual results of Schreer *et al.*'s system correctly detecting the hands [65].

2.1.2 Machine Learning Techniques

Machine learning is defined as the field of study that gives computers the ability to learn without being explicitly programmed [72]. Thus it falls under the branch of artificial intelligence. The key factor of machine learning is that it is able to classify unseen data given a training data set.

An extended neural network-based hand shape recognition system was proposed by Stergiopoulou and Papamarkos [74]. Their system uses a skin colour distribution map in the YCbCr colour space [12] to locate the hands. Once the hands are isolated, they propose the use of a new technique known as the Self-Growing and Self-Organised Neural Gas (SGONG) neural network [5]. This innovative technique starts with a small number

of neurons and grows according to the structure of the hand to approximate its shape in a very robust way.

Figure 2.7 illustrates this procedure: Figure 2.7 (a) depicts the SGONG starting with only two neurons; the neurons increase, as seen in Figure 2.7 (b) and in Figure 2.7 (c) the grid of the output neurons has taken the shape of the hand.

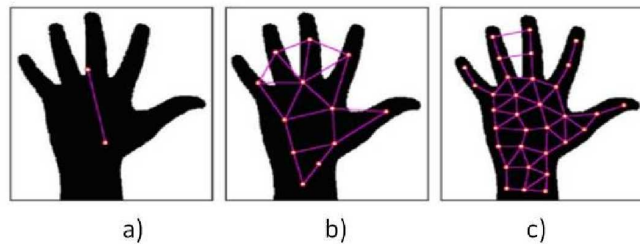


FIGURE 2.7: Stergiopoulou and Papamarkos' hand shape recognition system using a SGONG [74].

The system is able to detect the finger tips. Root neurons are taken as the neurons which have a small number of links with other neurons. This is used to locate the finger tips since the finger tips are in most cases root neurons as seen in Figure 2.7 (c). The proposed hand shape recognition system was trained on 31 hand gestures that consisted of combinations of raised and non-raised fingers. The test set consisted of 180 hand shapes repeated 10 times, giving a total of 1800 hand shape images. The images consisted of clear hands on a simple background in a controlled lighting environment. The system achieved a high accuracy of 90.45%. However, it is stated that the system processes at a speed of less than 1 FPS, much lower than real-time.

Sato *et al.* also used neural networks. However, a different approach was used for hand shape recognition [93]. The neural network classifier was trained on six hand shapes based on a number of examples for each hand shape. To achieve a high accuracy, the images were normalised as follows: Once the hand region is located, the region is rotated based on the orientation of the principal axis of the region so that the axis is aligned with one of the image axes. Once the hand region is rotated it is then down-sampled to a resolution of 12×12 pixels. The normalised hand region is sent to the neural network. This normalisation procedure ensures that the system is invariant to variations in hand alignment in the image. Figure 2.8 illustrates the normalisation procedure.

The system was tested on a Pentium II 450 MHz PC with 384 MB memory using two cameras to observe the hand motion. The system achieved a recognition accuracy of 85% and was able to process at a rate greater than 20 FPS. The transitions from one hand shape to another led to classification errors which was expected as the neural network was not trained to recognise intermediate hand shapes.



FIGURE 2.8: Sato *et al.*'s process to obtain normalised image of the hand [93].

Li created a novel system which recognises sign language hand shapes from 2D video input and estimates the output in 3D [44], as part of the South African Sign Language (SASL) research project. It is able to track the hands and recognise hand shapes using a Support Vector Machine (SVM) at a very high accuracy on simple as well as complex backgrounds.

The image pre-processing procedure was as follows. Haar classifiers are used to locate the position of the face [88]. The skin colour distribution of the user is extracted from a 10×10 pixel area around the nose region. This is used to segment the skin pixels of the user in the image. The system makes use of two cues: motion and skin cues. Gaussian Mixture Models (GMMs) are used to obtain the motion cue and the skin cue is determined by the previous mentioned method. These two cues are combined to form a moving skin image. Hierarchical Chamfer matching is used once to locate the hand in the image. Once the hand is obtained CAMShift is used to continuously track the hand.

In order to recognise the hand in the image, connected component analysis (CCA) is used to obtain the biggest contour, which is regarded as the hand. A minimum bounding box is computed around the hand contour. The principal axis of the bounding box is aligned with the y -axis to prevent misalignment invariance. The resulting contour image is down-sampled to a 20×30 pixel resolution. The resulting image is used as a hand shape feature vector to train and test the SVM.



FIGURE 2.9: The 10 hand shapes recognized by Li's system [44].

The system was trained on 40 images for each of 10 hand shapes, seen in Figure 2.9, performed by three subjects with different skin colours. To collect a testing data set,

six different subjects were asked to perform a transition between the 10 hand shapes resulting in six videos. Each video consisted of no less than 500 frames. 30 images of each hand shape performed by each subject were randomly chosen from the videos and used for testing. The system achieved an average recognition accuracy of 88.5%. It was shown to be robust to a complex background and variations in skin colour.

2.1.3 Summary of Articulated Object Recognition

This section summarised the two different approaches – template matching and machine learning techniques – used to recognise articulated objects. Template matching has generally been shown to either have a low accuracy recognition at a high computational speed or a low computational speed at a high recognition accuracy. Additionally, to extract complete and clear hand contours, a simple background needs to be used. It was shown that most such studies use a simple or static background.

Machine learning techniques are more promising. They make use of a training model which is used to recognise hand shapes. The model is trained on various examples. Machine learning techniques have been shown to be able to achieve a high recognition accuracy at high computational speeds. They can also be robust in complex and simple environments. They become more robust as more diverse examples are used. Therefore machine learning techniques are more suitable to the proposed application than template matching. Li's system [44] was shown to be especially suitable because it can operate on simple as well as complex backgrounds and on multiple skin colours as required by the proposed gesture recognition system. Thus, for this research, Li's hand shape recognition approach will be adopted as a pre-cursor to gesture recognition.

2.2 Gesture Recognition using Hidden Markov Models

This section discusses the previous work done in the field of gesture recognition using HMMs. HMMs are statistical models of sequential data [6]. HMMs compensate for the time and amplitude variance of signals. This has proven useful in speech and character recognition applications [27, 36, 57]. They have also been shown to be effective in a large number of gesture recognition systems [60, 94]. This section focuses on gesture recognition systems that have used HMMs for gesture classification.

These systems can broadly be sub-divided into three categories: those that use gloves with colour-markers to simplify the hand segmentation and feature extraction procedure, henceforth referred to as glove-based systems; those that use specialised equipment such

as stereo cameras or data gloves to simplify the hand segmentation and feature extraction procedure referred to as hardware-based systems; and those that use a single camera and computer vision techniques only, referred to as vision-based systems. Sub-sections 2.2.1–2.2.3 discuss related work in each of these categories.

2.2.1 Glove-Based Systems

Glove-based systems are systems that require the signer to wear gloves with pre-defined colour-markers. This allows for the system to easily locate and track the hands, as well as to extract relevant features pertaining to gestures. In a sign language context this could include hand shape, hand orientation etc. Figure 2.10 illustrates the signer wearing a white glove with different coloured fingertips [63] and Figure 2.11 depicts the signer wearing two different coloured gloves [97].



FIGURE 2.10: Coloured glove used by Sandjaja and Marcos [63].

Sandjaja and Marcos proposed a sign language number recognition system to recognise Filipino Sign Language (FSL) numbers using HMMs [63]. The signer is required to wear colour-coded gloves as seen in Figure 2.10. Video input from a camera is analysed using computer vision techniques to locate the pre-defined marker points and extract important features from the video to construct a feature vector. The feature vector consists of the following: the position of the face; the location, area, and orientation of the dominant and non-dominant hands and fingers. This feature vector is used by HMMs to predict the recognised FSL number. The system uses a 10-state HMM to represent the signing space.

A set of 5000 FSL number videos were used to test the accuracy of the system in recognising 172 FSL number gestures. A five-fold cross-validation procedure was used resulting in 5 test sets in which 4000 videos were used in training the system and 1000 were used to test the system accuracy. On average, the system was shown to achieve a recognition accuracy of 85%.

Zhang *et al.* [97] proposed a system to recognise Chinese Sign language (CSL) gestures using HMMs. Similar to the previous approach, the signer is required to wear colour-coded gloves as seen in Figure 2.11. One of the gloves is purple in colour and is set to be worn by the non-dominant hand. The other glove is multi-coloured, with the finger tips, the palm and the back of the hand each being a unique colour. This glove is worn by the dominant hand. Video input is analysed to extract relevant features. The proposed gesture recognition approach is separated into two stages. Stage one is the pre-processing stage which involves hand detection, background subtraction and pupil detection, all with the assistance of the coloured gloves. The system uses a custom-made algorithm to detect the two pupils of the user. This serves as a reference point on which to centre the images. Background subtraction is applied resulting in images illustrated in the left column of Figure 2.11 and colour and hand shape geometry information is used to locate and detect the hands as illustrated in the right column of Figure 2.11.

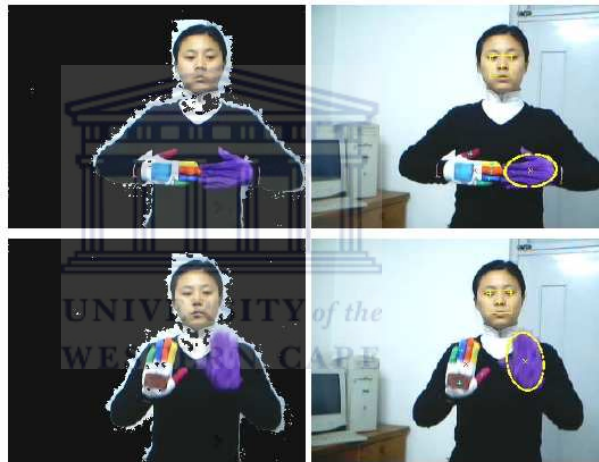


FIGURE 2.11: Zhang *et al.*'s system demonstrating: (Left column) Background subtraction and (Right column) locating the hands [97].

Once the hands have been acquired, relevant features pertaining to the gesture are extracted from the hands. The features pertain to hand location, motion, orientation and shape. These features are used to construct a 35-dimensional feature vector.

Stage two is the recognition procedure in which the system uses Tied-Mixture Density Hidden Markov Models (TMDHMMs) to recognise CSL gestures. The system was trained to recognise 439 CSL signs comprising of 223 two-handed and 216 one-handed signs. These signs included different word types such as nouns, verbs and adjectives. A single signer was used to collect a training and testing set. Four videos per sign were collected. This resulted in a total of 1756 videos to be used of which 439 were used for testing and the rest for training. The system achieved an average sign recognition accuracy of 92.5%. The researchers state that the system is signer-dependent as it was only tested on a single signer.

2.2.2 Hardware-Based Systems

Hardware-based systems use specialised hardware to capture the signer in 3D either using sensors placed on the signer hands and body or using multiple-camera configurations.

Vogler and Metaxas presented a novel approach for American Sign Language (ASL) recognition [90]. The system uses the Ascension Technologies MotionStarTM3D tracking system to locate and track the hands in 3D from a front, side and top view using multiple 3D cameras as seen in Figure 2.12. This configuration makes it possible to easily extract gesture information from images. An 8-dimensional feature vector is generated for each hand. These features consist of the 3D position and velocities of the hand in each direction, relative to the base of the signer's spine.

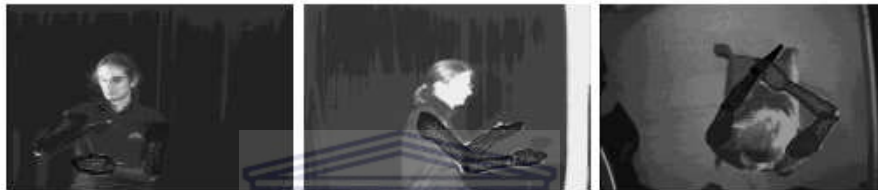


FIGURE 2.12: The tracking system to locate and track the hands in 3D from a front, side and top view using multiple 3D cameras [90].

The system uses parallel HMMs (PaHMMs) to carry out gesture recognition. PaHMMs are known to be able to model parallel processes independently and efficiently. Two sets of PaHMMs are used, one for each hand. A significant advantage of this approach is that gestures of each hand can be trained independently, and consideration of the different hand gesture combinations are required at training time.

A total of 140 HMMs were trained: 89 unique HMMs for the right hand and 51 HMMs for the left hand. The system was trained on 400 ASL sentences constructed from a vocabulary of 22 ASL signs. The system was tested on 99 ASL sentences constructed from the same sign vocabulary. The system achieved an average recognition accuracy of 80.81% for sentences using ordinary HMMs and a slightly improved accuracy of 84.85% using PaHMMs.

Kong proposed a system to extract phonemes from American Sign Language (ASL) sentences [43]. A Polhemus FASTRACK electromagnetic motion tracking system was used to capture the 3-D trajectories of a signer's hands. The signers were instructed to equip trackers on their body to provide reference values. Two trackers were attached to both hands and two additional trackers were attached to the waist and the head. The system applies a rule-based segmentation algorithm to segment the hand motion trajectories in a signed sentence. Principal component analysis (PCA) is used to the represent the

feature segments. The system uses a feature vector consisting of 3D trajectories, 3D motion and a 3D hand shape as input to HMMs which are trained to recognise ASL phonemes that result from the feature segmentation procedure.

The HMMs are trained to recognise ASL phonemes. Each phoneme had between three and five states and each state was represented as a single Gaussian matrix. Five-fold cross-validation was used to determine the recognition accuracy of the system in which the data set was divided into 5-part subsets, with the 4 parts of each subset used to train the system and the 1 remaining part used to test the system. The data set consisted of 25 ASL sentences repeated five times by a deaf signer.

The system was shown to obtain 165 segments and 58 phonemes from the 25 ASL sentences. The recognition error rate was 11.4%. This was compared with a manual transcription of the data set by an ASL linguist who was shown to obtain 173 segments and 57 phonemes, but with an error rate of 19.5%.

2.2.3 Vision-Based Systems

Vision-based systems focus on using image processing techniques to determine the actions of the signer while keeping the system as natural as possible by not using coloured gloves or specialised equipment. Typically, only a single web camera is used.

Matsuo *et al.* proposed a method to recognise sign language words using an automatic transitional HMM generation structure [47]. It is not mentioned which specific sign language the method focuses on recognising. The system consists of segmenting the motion of sign language words and creating a HMM topology that recognises each word.

To simplify the problem of detecting the skin pixels in the image, the following assumptions are made: a simple static background is assumed; the signer is expected to wear black long-sleeved clothing; signers are assumed to have the same skin colour tone; and the starting location of the hands is assumed to be at the waist region. Based on these assumptions it is possible to locate and track the hands using a static skin range function.

The direction and velocity of the hands are determined. Frames are grouped into three segment categories based on the hand motion in those frames. Stationary frames are grouped together and labelled as a stationary segment. Moving frames in which the hands display motion in a straight line are labelled as a straight segment. Moving frames in which the direction of the hand motion changes in a short time are labelled as a vibrating segment. These segment types are proposed to be basic phonemes that can represent sign language signs. The segment types are used to generate a specific

HMM based on the word in which the segments become HMM states. Figure 2.13 illustrates an example of a dynamically generated HMM for the word “winter clothing” in an unspecified sign language.

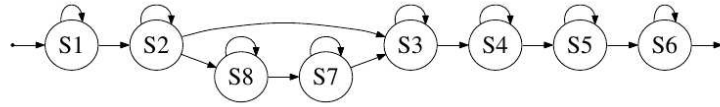


FIGURE 2.13: The generated HMM by Matsuo *et al.*'s system for the word “winter clothing” in an unspecified sign language [47].

Two subjects were asked to perform 43 words in the unspecified sign language three times. The first sample of each word was used for testing and the rest for training the system. Thus, a total of 86 words were used for training and 43 for testing. Each word was selected such that various hand motions were represented. The system achieved an overall average recognition accuracy of 87.6%.

Youssif *et al.* proposed an automatic Arabic Sign Language (ArSL) recognition system using HMMs [95]. The system begins by detecting the signer's skin pixels. The procedure to obtain the skin pixels is as follows. The image is converted from the RGB to the HSV colour space. A static range in the Hue and Saturation channels from the HSV colour space are used to represent skin pixels in the image. This results in a skin image containing three skin blobs. The biggest blob is regarded as the head, whilst the two remaining blobs are regarded as the hands. These two blobs are located and tracked using optical flow analysis.

The features pertaining to ArSL gestures that are extracted from the hand blobs include: the position of the head; coordinates of the centres of the hands; the direction of the movement of the hands; and the shape of the hand. These features are used as a feature vector in training HMMs to recognise ArSL gestures. Figure 2.14 illustrates the HMM produced by the system to recognise the ArSL word “Alexandria”.

The system was trained to recognise 20 ArSL words, resulting in a total of 20 HMMs, one HMM per word. A single database of 360 ArSL words, 18 videos for each of the 20 ArSL words, was used to train and test the system. A number of different HMM configurations consisting of between 3 and 6 states and 4 and 10 mixtures were considered and compared. Varied configurations were found to be ideal for each of the ArSL words. For example, the word “French” was best represented by 6 states and 4 mixtures, whereas the word “sleep” was best represented by 3 states and 10 mixtures. An overall recognition accuracy of 82.22% was achieved across all words.

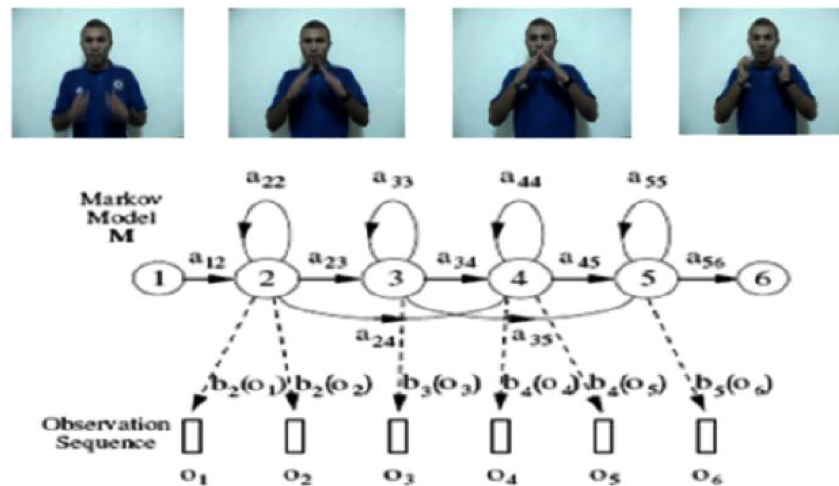


FIGURE 2.14: HMM model for gesture “Alexandria” by Youssif *et al.* [95].

2.3 Hidden Markov Support Vector Machines (HM-SVMs)

HM-SVMs are a new technique that combine the architecture of HMMs and SVMs into a hybrid approach. For the last two decades HMMs and variations thereof have been the dominant technique for modelling and predicting label sequences [37]. Despite their success, HMMs have at least three major limitations [3]:

1. HMMs are trained non-discriminatively.
2. The conditional independence assumptions are often too restrictive.
3. They are unable to make use of kernel-based methods.

SVMs are easy to train and scales relatively well to high dimensional data [13]. However, they are also known to have the following limitations[66]:

1. They are unable to directly provide probability estimates.
2. Selecting the optimal kernel for the application.

HM-SVMs combine HMMs and SVMs to overcome these challenges to an extent. HM-SVMs are a relatively new technique and, to our knowledge, research that has applied HM-SVMs is limited. Furthermore, there has been no work done in applying HM-SVMs to the task of sign language gesture recognition. Systems that have applied HM-SVMs are described below.

Altun *et al.* proposed a novel technique for label sequence learning using a HM-SVM comprising of an innovative combination of SVMs and HMMs [3]. The researchers call the novel architecture a Hidden Markov SVM (HM-SVM). They remark that HM-SVMs address the shortcomings of conventional HMMs while retaining some of the key advantages of HMMs such as the Markov chain dependency structure between labels. The two important factors that are added to HMMs by the HM-SVM hybrid are the maximum margin principle and the inheritance of the kernel-centric approach of SVMs to learning non-linear discriminant functions.

A test was conducted using a Spanish news article that consisted of 300 sentences. A HMM and a HM-SVM were used to recognise the different types of expressions that were used in these sentences, and the results were then compared. The expression types in this corpus were limited to person names, organisations, locations and miscellaneous names, resulting in nine different labels. The HMM was shown to achieve a recognition error rate of 9.36% while the HM-SVM had a lower recognition error rate of 5.08%. The experiment indicates that the HM-SVM can perform better than a conventional HMM.

Valstar and Pantic proposed two methods for improving their base system of facial action unit (AU) phase recognition using an SVM [85]. The first method uses polynomial parametrisation. However, the second technique, which used a HM-SVM classifier, is of interest to this research. The system attempts to recognise the temporal phase of 20 key facial AUs. The temporal phase of each feature is categorised into four types: the neutral, onset, apex and offset phase. The recognition accuracy of the original SVM-based system was compared with that of a new HM-SVM-based system.

The system used a modified Viola-Jones face detector which uses a bootstrapping classifier known as GentleBoost in place of the conventional AdaBoost to locate and track the face. Using a combination of heuristic techniques, the system was able to locate the irises and the centre of the mouth based on the analysis of the vertical and horizontal histograms of the face region. The location of these three points were combined with anthropomorphic rules of the face to locate 20 key facial feature points located on regions hosting AUs of interest. A grayscale image of a 13×13 patch surrounding each feature point and the responses to 48 Gabor filters were used to form a feature vector for each point.

The system was tested on 196 videos selected from the MMI-Facial Expression Database [52], containing videos of 23 AUs. The original SVM-based system achieved an overall recognition accuracy of 60.3%, whereas the HM-SVM achieved an overall recognition accuracy of 66.0%. This result shows that the HM-SVM can achieve a higher recognition accuracy than that of a conventional SVM.

Zhong-Hua and Hui proposed a system using HM-SVMs to solve the Chinese chunk problem [98]. The Chinese chunk problem is the problem of determining which parts of Chinese sentence are verbs, nouns, adjectives etc., that is, recognising the chunks of a Chinese sentence. Chinese chunking provides important and useful syntactic information for applications such as text mining, text categorisation and machine translation. A comparison was carried out between a SVM-based system and a HM-SVM-based system applied to this task. Ten thousand sentences of the China Daily newspaper were used to train and test the two systems. Eighth thousand sentences were used as training data and 2000 sentences are used as test data. The SVM-based system achieved an average accuracy of 86.95% while the HM-SVM-based system achieved an improved overall average accuracy of 89.62%, once again demonstrating that HM-SVM hybrids can perform better than each original learning algorithm.

2.4 Summary of the Related Work

This chapter summarises the related work in the fields of articulated object recognition, gesture recognition using HMMs, and relatively new research in the field of HM-SVMs applied to classification.

Articulated object recognition is sub-divided into two main categories: template matching and machine learning techniques. It was shown that machine learning techniques out-perform template matching techniques in both accuracy and computational speed. Li's system [44] was shown to be particularly robust to various skin colour and in simple and complex environments. The approach uses SVMs to predict the correct hand shape in real-time. The approach was selected as the hand shape recognition strategy in the proposed gesture recognition system.

Gesture recognition using HMMs was divided into three broad system categories: glove-based systems; hardware-based systems; and vision-based systems. Details were provided on the various studies that attempted to recognise gestures using HMMs. It was shown that various systems use different techniques and approaches but all of these systems achieved a high recognition accuracy. Thus, it was shown that HMMs are able to recognise gestures at a high accuracy.

HM-SVMs are hybrid systems based on the combination of HMMs and SVMs. It was shown that HM-SVMs out-perform conventional HMMs or SVMs in terms of accuracy. This new approach has not been applied to sign language gesture recognition and this research is thus novel in this regard.

Chapter 3

Image Processing in Gesture Recognition

In this chapter the key image processing techniques used in the proposed gesture recognition system to extract relevant gesture feature descriptors will be discussed. The components used for the feature extraction process such as: face detection, skin detection, background subtraction, and image smoothing using Gaussian blur and minimum bounding box generation will be discussed in separate sections.

UNIVERSITY of the
WESTERN CAPE

3.1 Face Detection

Face detection serves as the foundation for computation in many learning-based gesture recognition systems for three reasons [9, 49]:

1. It identifies whether or not an individual is in the frame.
2. It serves as a base reference on which to normalise the image sequence by centring the body in each frame.
3. It can be used as a reference point to find other points of interest on the human body such as the shoulders and the torso.

The Viola-Jones object detection framework is a world-renowned face detection technique. It achieves a high detection rate using Haar classifiers to build a boosted rejection cascade of nodes. AdaBoost is used at every node in the cascade using a rejection multi-tree classifier. Their framework can be described in a four step procedure [7]:

1. Haar-like features are computed.
2. An integral image is computed and used to accelerate the detection of these features.
3. A statistical boosting algorithm based on AdaBoost is used to create weak binary classification nodes.
4. The weak classifier nodes are organised as a rejection cascade.

These four steps will be discussed in the following subsections.

3.1.1 Haar-like Wavelet Detection

A Haar-like wavelet is a single wavelength square wave that has one high and low interval [88]. It consists of pairs of rectangles that are identical in shape and size. Each rectangle is either light or dark and is either vertically or horizontally adjacent to the other rectangle or rectangles in the wavelet. These wavelets are of three distinct forms: two-rectangle, three-rectangle or four-rectangle features. Figure 3.1 (d) depicts the four types of Haar-like wavelet features.

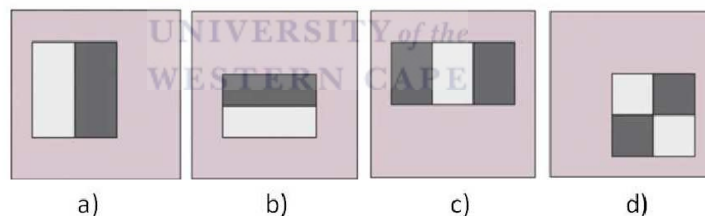


FIGURE 3.1: The four types of Haar-like wavelet features [88].



FIGURE 3.2: Haar-like wavelet feature detection used to detect if a feature is present [88].

The Haar-like wavelet feature detection procedure commences by converting the image into a grayscale image. This is done to define the image as a set of intensity values from 0 to 255. Large-scale rectangular Haar-like wavelet features are detected using these values. The presence of a Haar-like feature is determined by subtracting the sum of the pixel values in the dark-region from the sum of the pixel values in the light-region.

If the difference is greater than a pre-determined threshold value, then that feature is determined to be present. This procedure is carried out over multiple locations and scales for each of the four wavelets as illustrated in Figure 3.2.

3.1.2 Integral Image

An integral image is a data structure that allows for the rapid summing of pixel values in image sub-regions. This data structure is used by the Viola-Jones face detector to increase the computational speed of Haar-like wavelet detection.

The integral image is an intermediate representation of the image which progressively adds together small units of a region. The original image is converted to an integral image by taking the sum of all the pixels to the left and above each pixel in the original image, starting at the top left pixel of image I , proceeding row by row, and traversing to the right and down. Each integral pixel value $I'(x, y)$ in the integral image I' is computed recursively by the formula [88]:

$$I'(x, y) = I(x, y) + I'(x - 1, y) + I'(x, y - 1) - I'(x - 1, y - 1) \quad (3.1)$$

Figure 3.3 illustrates an example of an image that is converted to an integral image.

1	2	5	1	2
2	20	50	20	5
5	50	100	50	2
2	20	50	20	1
1	5	25	1	2
5	2	25	2	5
2	1	5	2	1

a) Original image

1	3	8	9	11
3	25	80	101	108
8	80	235	306	315
10	102	307	398	408
11	108	338	430	442
16	115	370	464	481
18	118	378	474	492

b) Integral image

FIGURE 3.3: Haar-like wavelet feature detection [88].

3.1.3 AdaBoost

A modified AdaBoost algorithm was used in the Viola and Jones [7, 88] face detection method. This algorithm selects a small set of features and trains the classifier. Many weak classifiers are combined to form a strong classifier. To minimise training errors, weights are assigned to each weak classifier, a process known as boosting. The resulting classifier is a strong classifier consisting of a weighted combination of weak classifiers.

Figure 3.4 depicts features that may be selected by AdaBoost in the creation of a strong classifier.

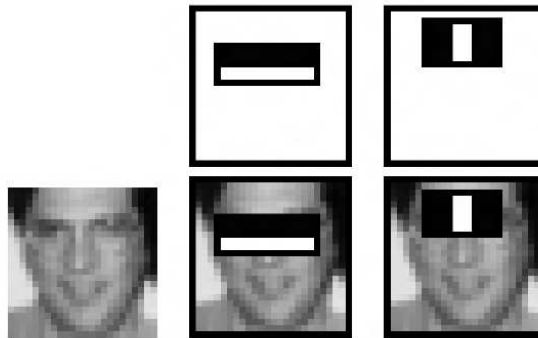


FIGURE 3.4: Possible features selected by AdaBoost [88].

3.1.4 Construction of a Rejection Cascade of Weak Classifier Nodes

Computational time is substantially reduced by using a rejection cascade. This structure is a degenerate decision tree. The cascade contains the weak classifiers with heavier weighted classifiers earlier on in the cascade to achieve a faster elimination process. Each classifier is only computed if all of the previous – and more significant – nodes have returned a positive result. A predefined search window scans across the image and checks each feature location for an object.

A large cluster of regions are detected around the face. If a negative result is obtained any classifier, the processing is immediately halted. The process is continued as long as no negative result is returned. The algorithm will only recognise a region as a face when all classifier nodes return a positive result. This significantly enhances the speed of the algorithm. The algorithm is illustrated in Figure 3.5 and Figure 3.6 illustrates an example of its use.

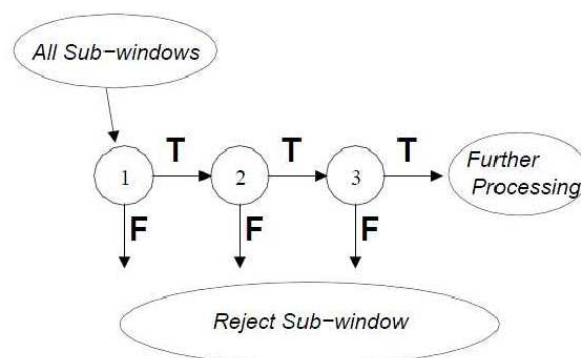
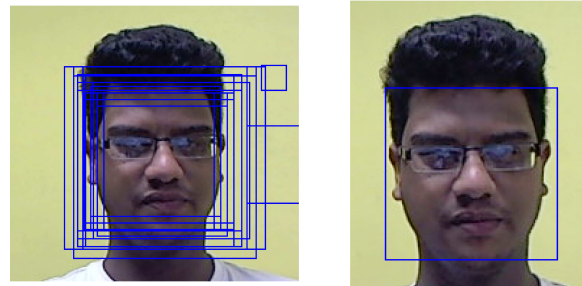


FIGURE 3.5: The rejection cascade of weak classifier nodes [88].



(a) Large clusters of regions detected (b) Correct object detected as face

FIGURE 3.6: Rejection cascades used to determine face correctly.

3.1.5 Testing and Results on the Face Detection Method

Viola and Jones performed a test to determine the accuracy of the approach on the CMU-MIT frontal data set which is made up of randomly selected images from the Internet [88]. The images have varying background complexities and camera properties. It was shown that the Viola-Jones face detector achieves a high average accuracy of 88.9%. Castrilln *et al.* [11] tested the Viola-Jones face detector on the CMU dataset [64] which contains a collection of heterogeneous frontal face images. They achieved a high average recognition of 89.07%.

The Viola-Jones face detector is a suitable option as basis of the feature extraction procedure since it is very robust and achieves a high recognition accuracy.

3.2 Skin Detection

Skin detection is the process of automatically distinguishing between skin and non-skin pixels in an image [8]. Many applications involving the detection of the human body as well as hand detection and tracking use skin detection to segment the relevant body parts in the image [35, 44]. Skin detection is very robust to partial occlusions, rotations and scaling of body parts [40]. Skin is distinct in colour from most other objects. However, factors such as illumination and viewing angle of the camera can hinder the detection of skin pixels. The following three steps are used to create a skin filter [9, 40, 87]:

1. A relevant colour space needs to be selected to represent the pixels in the image.
2. The skin pixels need to be modelled using an applicable classification algorithm.
3. Each pixel needs to be classified as either being a skin or non-skin pixel; i.e: setting skin pixels to 1 and non-skin pixels to 0.

Colour spaces can be mathematically represented in numerous ways. There are a wide range of colour spaces which have common properties. In Subsections 3.2.1 - 3.2.4 the most renowned colour spaces will be discussed.

3.2.1 RGB Colour Space

RGB is short for Red-Green-Blue and is the most basic colour space used [87]. A single colour pixel is represented by the fusion of red, green and blue pixel values. To achieve other colour spaces, linear and non-linear transformations are performed on the RGB colour space. A visual representation of this colour space is a cube consisting of red, green and blue on the three perpendicular axes, respectively. A specific colour is a point in the cube. Figure 3.7 illustrates this cube.

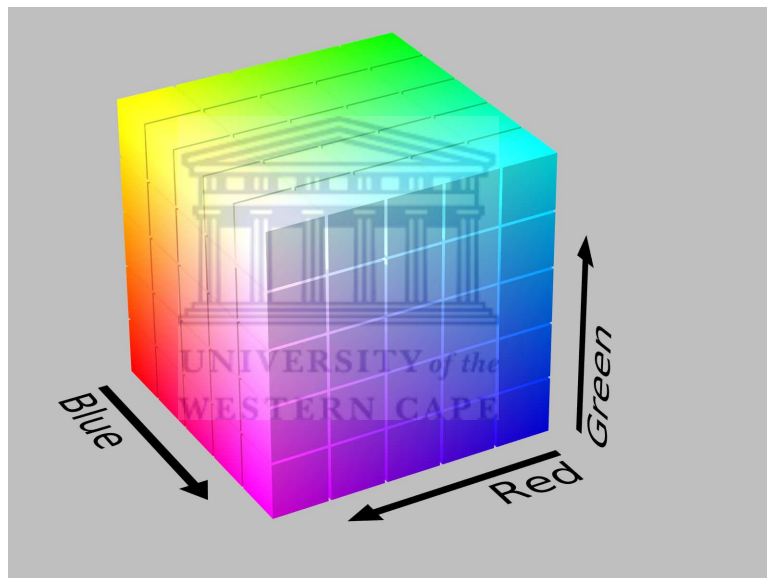


FIGURE 3.7: A visual representation of the RGB colour space cube [62].

This colour space is very simple to use. However, the colours that humans perceive do not correspond to the colour value in this representation [87, 96]. Luminance and chrominance data are not separated since these three colours are highly interconnected with each other. Thus, colour-based recognition algorithms will not be robust to illumination changes if this colour space is used.

3.2.2 Normalised RGB Colour Space

A linear normalisation formula is applied to the RGB colour space to achieve the normalised RGB colour space representation. The equation is as follows:

$$r = \frac{R}{R + G + B}, \quad g = \frac{G}{R + G + B}, \quad b = \frac{B}{R + G + B} \quad (3.2)$$

the variables r, g and b is denoted as the normalised red, green and blue pixels respectively. R, G and B are the RGB-colour space red, green and blue components. The sum of the three normalised components r, g and b is equal to 1 as:

$$r + g + b = 1 \quad (3.3)$$

The space dimensionality of the colour space can therefore be reduced by omitting the third component which can be inferred [40, 87]. Typically, r and g remain which are less sensitive to illumination in the normalised RGB colour space than those values in the original RGB colour space. Figure 5.1.4 (a) illustrates an image using the RGB colour space and Figure 5.1.4 (b) depicts when the image is normalised.

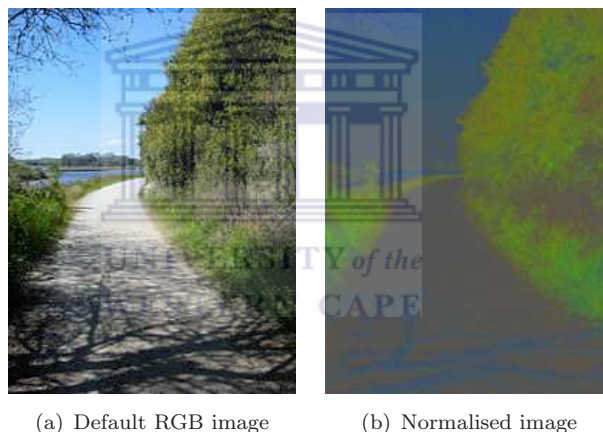


FIGURE 3.8: Conversion from the default RGB colour space to a normalised image.

3.2.3 HSV Colour Space

HSV stands for Hue-Saturation-Value. It is also referred to as the HSI (Hue-Saturation-Intensity), HSB (Hue-Saturation-Brightness) and HSL (Hue-Saturation-Luminance) colour space. The HSV colour space is used in image analysis and computer vision and has been proven to make skin detection methods robust to variations in illumination [42, 71, 96].

HSV rearranges the geometric RGB cube into a cylindrical-coordinate representation that is more intuitive and perceptually relevant. Figure 3.9 illustrates the HSV colour space representation. In the HSV colour space, the Hue component is defined as the dominant colour of an area. The Saturation relates to the dominant area corresponding

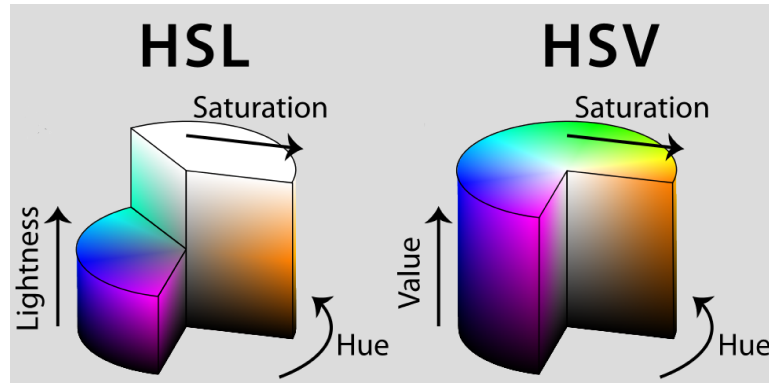


FIGURE 3.9: Graphical depiction of the cylindrical colour space representation of the HSV colour space [62].

to its illumination. The brightness information of a colour is stored in the Value component. To convert from RGB colour space to HSV colour space the following non-linear transformation is applied [87]:

$$V = \max_{r,g,b} \quad (3.4a)$$

$$S = \frac{\max_{r,g,b} - \min_{r,g,b}}{V} \quad (3.4b)$$

$$H = \begin{cases} \frac{g - b}{6(\max_{r,g,b} - \min_{r,g,b})}, & \text{if } V = r \\ \frac{2 - r + b}{6(\max_{r,g,b} - \min_{r,g,b})}, & \text{if } V = g \\ \frac{4 - g + r}{6(\max_{r,g,b} - \min_{r,g,b})}, & \text{if } V = b \end{cases} \quad (3.4c)$$

where H , S and V are the Hue, Saturation and Value components, respectively, with r , g and b being the normalised red, green and blue pixel values, respectively. The maximum and minimum between the normalised red, green and blue pixel values are $\max_{r,g,b}$ and $\min_{r,g,b}$ respectively. The Hue component has a unique feature in that it is unaffected by illumination changes [73, 96].

3.2.4 YCbCr colour space

The YCbCr colour space is mainly used in image compression [87]. To obtain the YCbCr colour space a linear transformation is applied to the default RGB colour space. The luminance component Y is computed by taking a weighted sum of the RGB pixels. The chrominance components Cr and Cb are computed by subtracting the newly obtained

luminance component Y from the red and blue RGB colour space values, respectively. This is expressed mathematically as:

$$Y = 0.299R + 0.587G + 0.114B \quad (3.5a)$$

$$Cr = R - Y \quad (3.5b)$$

$$Cb = B - Y \quad (3.5c)$$

3.2.5 Selecting An Appropriate Colour Space for Skin Detection

When selecting an appropriate colour space for skin detection it is important to take into consideration two factors.

1. The colour space must support the separation of skin and non-skin pixel values.
2. The colour space must be robust to variations in lighting conditions.

Various researchers [40, 70, 87, 96] have performed studies to determine the effect of different colour spaces on skin detection algorithms. Kakumanu *et al.* [40] and Shin *et al.* [70] demonstrated that there is no noteworthy improvement to the skin detection process using either an RGB or non-RGB colour space. They also conclude that the discrimination between skin and non-skin pixels is not affected when removing the luminance component from the chrominance component. However, they suggest that removing the luminance component may benefit the training data used in classification.

Zarit *et al.* [96] and Vezhnevets *et al.* [87] recommend that a suitable colour space should be selected based on the requirements of the particular application. In their research, the HSV colour space was confirmed to benefit the skin detection process.

From the above research, it is undecided whether an RGB or non-RGB colour space should be used for skin detection. It appears that it only depends on the application. However, many researchers concur with Forsyth and Fleck [24] that the human skin colour can be successfully characterised by the HSV component.

The chief factor in human skin colour is Melanin [83]. This can be divided into two types: Pheomelanin and Eumelanin, which are red and dark brown in colour respectively. A mixture of these two colours can be easily represented by the Hue component in HSV [20, 21].

Determining the ideal colour space for skin detection is outside the scope of this research. However, it is possible to conclude that the skin colour of humans can be represented by the Hue component.

3.2.6 Skin Model

Static skin models are used by many researchers. Such models define skin as a range in a specific colour space. However, when applied to assorted skin tones, these techniques often fail [2, 53]. South Africa and other Southern African countries have been shown to have the most diverse skin tones in the world [59]. Therefore, in the context of South African Sign Language recognition, a static skin model cannot be used. An adaptive and dynamic skin model is required.

A dynamic solution that makes use of an individual's nose colour to recognise and continuously update the skin colour distribution, even when illumination changes occur was proposed by Achmed [1]. The centre of the facial frame is almost always the nose region which is generally well exposed and unoccluded by facial hair, shadows, objects such as glasses etc. Achmed proposed the use of a 10×10 pixel area in the nose region, from which to extract a skin distribution model. The Hue values in this region are used to compute a histogram. The histogram serves as a look-up table for the skin pixel values of the individual. Figure 3.10 is an example of such a histogram. Brown [9] determined that using 8 bins in the histogram – as opposed to the default 16 bins originally used by Achmed [1] – achieved the optimal skin distribution histogram.

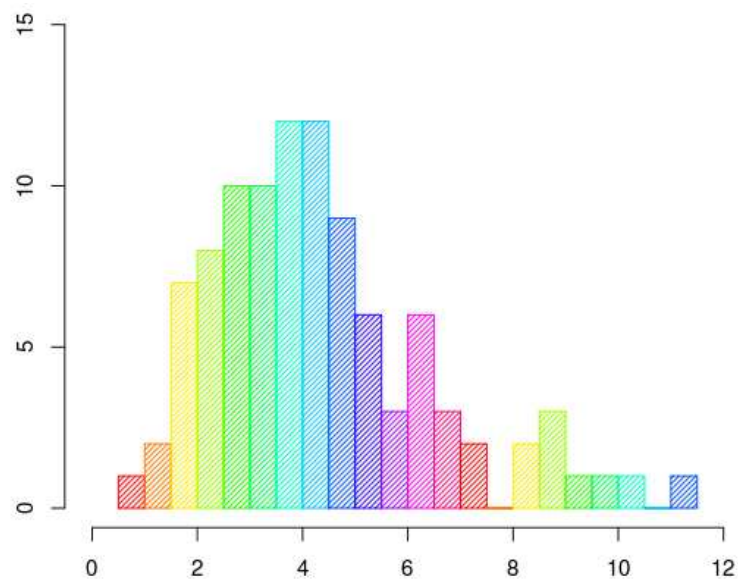


FIGURE 3.10: Graphic representation of a skin histogram.

A new grayscale image is formed by back-projecting the Hue histogram onto the original image. The new image has intensity values ranging between 0 to 255. The highest probability that a pixel is a non-skin pixel is indicated by the value 0 at that location in the new image, and the highest probability that a pixel is a skin pixel is indicated by the value 255 in that pixel location in the new image.

Given that a colour pixel C and the probability that the pixel is skin F , the probability of plotting that colour onto the histogram when the pixel is actually skin is $P(C|F)$. Then $P(F|C)$ is the probability that the pixel is skin given its colour, given by:

$$P(F|C) = \frac{P(F)}{P(C)}P(C|F) \quad (3.6)$$

The image is converted into binary by applying a threshold. In the resulting image, a value of 0 represents the non-skin class and 1, the skin class. Achmed [1] and Li [44] determined that the optimal threshold to be used to obtain the best skin detection results is a value of 60. Figure 3.11 (b) illustrates the skin detection result applied to the image in Figure 3.11 (a).

In some cases, certain objects may have the same colour as the human skin and will appear as noise in the resulting skin image. This noise can be reduced by merging the skin image with a background subtraction image. The next section discusses techniques that can be used to achieve background subtraction.

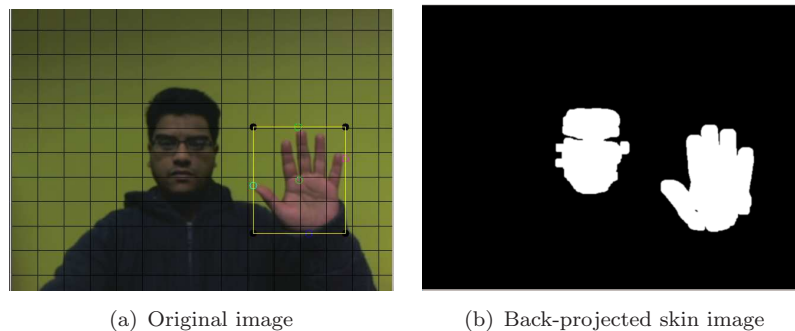


FIGURE 3.11: Results of the skin detection algorithm.

3.3 Background Subtraction

In a sequence of images, background subtraction aims to detach the background from a desired foreground. Most background subtraction techniques regard any moving objects as the foreground and stationary objects as the background. In this research the foreground image consists of both hands of the signer performing South African Sign

Language. In order to effectively capture these objects of interest three, a technique has to be selected that meets the following conditions:

1. It should be robust under dynamic lighting conditions.
2. Moving environmental objects in the background should not be detected as the foreground image
3. It should be robust to sudden changes in the scene.

The subsections below discuss the three most popular background subtraction techniques.

3.3.1 Static Background Subtraction

Static background subtraction uses a pre-defined reference image. This is commonly the first image in the sequence. Every new frame is subtracted from the reference image and the resulting image is then thresholded. Applying this threshold separates the stationary background from the moving foreground resulting in only two classes. This approach can be described by the following thresholding function applied to each pixel (x, y) in the current image I_t to obtain the binary foreground image F_t at the current time t in which a 0 indicates a stationary pixel and a 1 indicates a moving pixel:

$$F_t(x, y) = \begin{cases} 1 & |I_t(x, y) - B(x, y)| > T \\ -1 & |I_t(x, y) - B(x, y)| \leq T \end{cases} \quad (3.7)$$

where $I_t(x, y)$ and $B(x, y)$ are the pixel intensities at position (x, y) in the current image I_t and the reference image B and T is a motion threshold value that is determined empirically [14].

The main disadvantage of this method is that it is easily affected by small changes in illumination or camera movements.

3.3.2 Frame Differencing

Frame differencing can solve the problems faced by static background subtraction. This is achieved by continuously updating the reference image in the image sequence with more recent frames. This is also known as an adaptive background subtraction approach.

It is mathematically expressed as equation 3.7 with the exception that the reference image B is updated in time, typically given by:

$$B(x, y) = I_{t-k}(x, y) \quad (3.8)$$

where k is a frame offset applied to obtain longer-term or shorter-term updates of the foreground image with larger and smaller values of k , respectively. Other schemes may also be used, such as taking an average of N previous frames as the reference image. This approach is more robust to illumination and changes in environment than the static background subtraction technique.

3.3.3 Statistical Background Subtraction

Statistical background subtraction attempts to cluster objects in the frame sequence to generate a converged background model. The background model is robust to illumination changes by only detecting highly dynamic objects of interest. A popular approach is to model the background as a mixture of Gaussians. This resulting model is known as a Gaussian Mixture Model (GMM). The history of a pixel (i, j) at time t across image sequence I can be formulated as:

$$\{I_1, \dots, I_t\} = \{I(i, j, x) : 1 \leq x \leq t\} \quad (3.9)$$

Each pixel can be modelled by k Gaussian distributions. At the time t , the probability that the pixel may contain the value I_t is given by the following expression:

$$P(I_t) = \sum_{n=1}^k W_{n,t} \times \eta(I_t, \mu_{n,t}, \Sigma_{n,t}) \quad (3.10)$$

where $W_{n,t}$ is the estimated weight parameter of the n -th Gaussian component. The normal distribution of the n -th Gaussian component is $\eta(I_t, \mu_{n,t}, \Sigma_{n,t})$ given by:

$$\eta(I_t, \mu_{x,t}, \Sigma_{x,t}) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma_{x,t}|^{\frac{1}{2}}} e^{-\frac{1}{2}(I_t - \mu_{x,t})^T \Sigma_{x,t}^{-1} (I_t - \mu_{x,t})} \quad (3.11)$$

where $\mu_{x,t}$ is the mean and $\Sigma_{k,t} = \sigma_{k,t}^2 \mathbf{I}$ is the covariance of the k -th Gaussian component and \mathbf{I} is the identity matrix. The fitness value $\frac{W_{x,t}}{\sigma_{x,t}}$ is used to order the number of distributions k and the first M distributions is used to model the background where M is estimated as:

$$M = \operatorname{argmin}_m \left(\sum_x^m W_{x,t} > T \right) \quad (3.12)$$

where the minimum segment of the background model is the threshold T . Upon updating the background, the foreground is calculated by labelling any pixel that is more than 2.5 standard deviations away from any one of the M distributions as foreground. If the test value matches the x -th Gaussian component, then it is updated as follows:

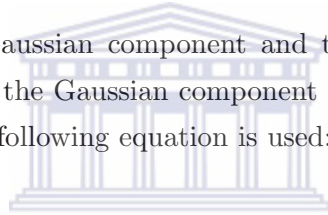
$$W_{x,t} = W_{x,t-1} \quad (3.13a)$$

$$\mu_{x,t} = (1 - \rho)\mu_{x,t-1} + \rho I_t \quad (3.13b)$$

$$\sigma_{x,t}^2 = (1 - \rho)\sigma_{x,t-1}^2 + \rho(I_t - \mu_{x,t})^T(I_t - \mu_{x,t}) \quad (3.13c)$$

$$\rho = \alpha \eta(I_t | \mu_k, \Sigma_k) \quad (3.13d)$$

where $W_{x,t}$ is the x -th Gaussian component and the time constant that determines change is defined by $\frac{1}{\alpha}$. If the Gaussian component and test value does not correspond with each other, then the following equation is used:



UNIVERSITY of the

WEST INDIES

$$W_{x,t} = (1 - \alpha)W_{x,t-1} \quad (3.14a)$$

$$\mu_{x,t} = \mu_{x,t-1} \quad (3.14b)$$

$$\sigma_{x,t}^2 = \sigma_{x,t-1}^2 \quad (3.14c)$$

If none of the components match the test value, a new value with a low weight parameter, a high variance and the current value as its mean replaces the lowest probability component. The Gaussian distributions are determined, if there is no match in some pixels, those pixels are determined to belong to the foreground and grouped using 2D connected component analysis.

3.3.4 A Comparison of the Background Subtraction Techniques

The different background subtraction techniques have relative advantages and disadvantages. Depending on the application of use, the most effective background subtraction method to achieve a high accuracy is selected. Currently, there is a lack of studies that have compared the different background subtraction techniques. This is more so on a per-context basis.

In scenarios in which perfectly static conditions are observed i.e. a background that is simple and static and a camera that is stationary, static background subtraction is a suitable technique. The technique offers a high processing speed and algorithmic simplicity. However, when the environment is not stationary, the technique fails. GMMs are low more computationally expensive compared to the other two techniques. However, they can robustly adapt to a changing background and generally provide accurate motion information about objects in a scene. Frame differencing operates at a high speed with a low algorithmic complexity comparable to static background subtraction, is more robust to a non-static background and can adapt quickly to changes in illumination.

Achmed's original upper body pose estimation system used GMMs as a motion cue [1]. Brown showed that frame differencing was a better choice since it was sufficiently accurate but was a much faster alternative [9]. For this research, frame differencing is chosen as the background subtraction technique since it operates at high speed and can adapt quickly to changes in illumination.

3.4 Image Smoothing Using Gaussian Blur

Image smoothing, also known as image blurring, is used to reduce the effect of noise in an image [7]. There are several prominent techniques that can be used to this effect including: the simple blur, median blur, Gaussian blur and the bilateral filter. The most popular and useful of these techniques is the Gaussian blur. Some of the advantages of Gaussian blur over other smoothing techniques are: they are not complex; they are computationally efficient; they are linear low-pass filters; they preserve edge features; and the degree of smoothing can be controlled using the parameter σ to the function [7, 91]. Therefore, this technique is selected for use in reducing noise in the images. Only this technique is discussed below but the interested reader is referred to [4, 7, 38] for information about the other image smoothing techniques.

Gaussian blur operates on a pixel in the image by summing weighted grayscales of the pixels in the neighbourhood to the pixel. The centre pixel has the biggest weight and the neighbouring pixels have progressively lower weights as the distance from the centre increases to each of those neighbouring pixels. A Gaussian function G is used to produce a table of weights, better known as a Gaussian kernel, on a $n \times n$ neighbourhood as follows:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}; \quad -(n-2) \geq x \leq (n-2); \quad -(n-2) \geq y \leq (n-2) \quad (3.15)$$

where x and y are the coordinates of each pixel relative to the centre pixel and σ is the standard deviation of the distribution. The level of blurring in a two dimensional image is controlled by adjusting the standard deviation σ . Equation 3.16 illustrates an example of a 3×3 grayscale image matrix I .

$$I = \begin{bmatrix} 200 & 200 & 100 \\ 130 & 200 & 120 \\ 30 & 130 & 30 \end{bmatrix} \quad (3.16)$$

The procedure involved in applying a Gaussian blur to the centre pixel of the matrix is explained. The Gaussian kernel for a 3×3 neighbourhood when $\sigma = 1$ is calculated as:

$$I = \begin{bmatrix} G(-1, -1) = 0.0585498 & G(0, -1) = 0.0965324 & G(1, -1) = 0.0585498 \\ G(-1, 0) = 0.0965324 & G(0, 0) = 0.159155 & G(1, 0) = 0.0965324 \\ G(-1, 1) = 0.0585498 & G(0, 1) = 0.0965324 & G(1, 1) = 0.0585498 \end{bmatrix} \quad (3.17)$$

The kernel is constant for a given neighbourhood size and σ . Thus, it is only necessary to calculate the kernel once. It should be noted that there are values in the Gaussian kernel that are the same. This is because the Gaussian function is circularly symmetric.

The next step in the procedure is to sum the products of each $G(x, y)$ value and the original grayscale pixel values as in:

$$\begin{aligned} P_c &= 200 \cdot (-1, -1) + 200 \cdot (0, -1) + 100 \cdot (1, -1) \\ &\quad + 130 \cdot (-1, 0) + 200 \cdot (0, 0) + 30 \cdot (1, 1) \\ &= 108 \end{aligned} \quad (3.18)$$

where P_c is the new value of the pixel at the centre which is thus determined. The original centre pixel is affected the most by this function since it contains the biggest weight. Applied to an entire image, this operation is repeated for each pixel in order to achieve a blur/smooth of the entire image.

The Gaussian blur has a property that allows for very fast computation: the filter can be applied in both dimensions independently. This is faster since it is not a $n \times n$ matrix computation but a one-dimensional array computation. Therefore, it can be expressed as a one-dimensional filter that operates vertically, and another one-dimensional filter that operates horizontally.

The Equation 3.16 can be separated into two steps. The one-dimensional $G(x)$ is applied first and then a second one-dimensional $G(y)$ is applied on the result. The one dimensional function is written as:

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}} \quad (3.19)$$

The example discussed the use of a 3×3 neighbourhood to achieve Gaussian blur. However, larger neighbourhood sizes such as 5×5 , 7×7 etc. can be used to achieve larger-scale smoothing effects.

3.5 Feature Normalisation by Applying a Minimum Bounding Rectangle



FIGURE 3.12: The process of determining the minimum bounding rectangle of the hand contour [44].

Given an object contour, an oriented minimum bounding box can be used to segment the object using a computational geometry algorithm. The minimum bounding box will completely enclose the vertices of the contour. Freeman and Shapira's theorem [26] is used in this algorithm, which is given as: the minimum area rectangle enclosing a convex polygon has a side collinear with an edge of the polygon. The vertices of the input, which is assumed to be a convex polygon, are given in clockwise order. Figure 3.12 illustrates the contour being encased by the minimum bounding box, with the image on the extreme right of Figure 3.12 illustrating that a minimum bounding box has a side collinear with the object contour. The rotating calliper algorithm is applied to the contour in order to determine the minimum bounding rectangle as follows:

1. A closed curve or convex polygon is defined by the chain of vertices a_1, a_2, \dots, a_n of the contour p .

2. A bounding rectangle that encloses the given curve is determined. This rectangle is formed by the following lines:

$$\begin{aligned}
 x_1 &= W_{min}; y_1 = H_{min}; x_2 = W_{max}; y_2 = H_{max}; \\
 W_{min} &= \min_j \sum_{i=0}^j a_{ix}, H_{min} = \min_j \sum_{i=0}^j a_{iy}; \\
 W_{max} &= \max_j \sum_{i=0}^j a_{ix}; H_{max} = \max_j \sum_{i=0}^j a_{iy};
 \end{aligned} \tag{3.20}$$

where $j = 0 \dots n$; with a_{ix} and a_{iy} are the x and y components of the vector denoted by a_i .

3. The given curve is encased by rotating the lines clockwise. A search is carried out for the coordinate parameters (x, y) of all the chain points lying on these lines. If more than two chain points lay on one line, only the first and last chain points are captured in the vertex list. The selected chain point is labelled with the number of the chain link and it is kept in an ordered vertex list.
4. Two set of callipers are formed by using the current bounding rectangle and the previous bounding rectangle, the area of the current bounding rectangle is computed and compared to the previous one.
5. The current minimum bounding rectangle is updated, if necessary, based on the new bounding rectangle.
6. This procedure – steps 3–5 – is repeated until all pairs of vertices have been successfully computed.
7. The minimum bounding rectangle is obtained.

For more details about the geometric computation of this algorithm, the interested reader is referred to [26].

3.6 Summary and Conclusion

This chapter discussed the components used in the proposed gesture recognition system used in this research.

The various techniques used in the feature extraction process were discussed in detail. The Viola-Jones face detection algorithm was explained and shown to be robust and accurate. A discussion of the various colour spaces, starting from RGB, it was

demonstrated how other colour spaces such as the HSV and the YCbCr are obtained by applying mathematical transformations to this colour space. Skin detection techniques were discussed and the skin detection method used in the proposed system was detailed.

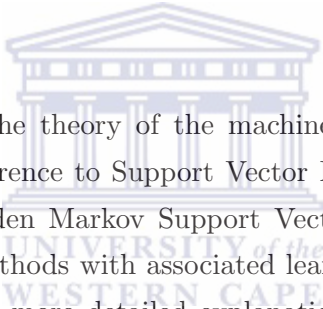
A detailed breakdown on prominent background subtraction techniques was provided. Each technique was shown to have its relative advantages and disadvantages. For this research frame difference was selected as the background subtraction technique and its use justified.

An image smoothing technique known as Gaussian blur was discussed. The technique helps reduce noise in an image. Lastly, the use of a minimum bounding box to isolate an object using its contour was discussed. Chapter 5 describes the use of these techniques in the proposed gesture recognition framework of this research.



Chapter 4

Support Vector Machines, Hidden Markov Models and a Hybrid Approach



This chapter focuses on the theory of the machine learning techniques used in this research, with specific reference to Support Vector Machines (SVMs), Hidden Markov Models (HMMs) and Hidden Markov Support Vector Machines (HM-SVMs). SVMs are supervised learning methods with associated learning algorithms that analyse data and recognise patterns. A more detailed explanation of SVMs is provided in Section 4.1. HMMs are statistical models in which the system being modelled is assumed to be a Markov process with hidden states. HMMs are discussed further in Section 4.2. HM-SVMs are a combination of HMMs and SVMs and are further discussed in Section 4.3.

4.1 Support Vector Machines

SVMs make use of statistical learning theory. Originally they were created to deal with binary classification problems. However, they have evolved to deal with multiple class classification [33, 86]. The use of SVMs has become widespread in image pattern recognition problems due to the abundant advantageous qualities that they possess [18, 86].

SVMs have a variety of advantages over other classifiers [92]. Two of the main advantages are that the dimensionality of feature vectors from a large image has no effect on training time and kernel functions are used which allow for non-linear problems to be solved

linearly. SVMs use a linear kernel by default. However, it is possible to use other kernels such as the radial basis function (RBF), polynomial and sigmoid kernels, each of which may be better suited than other kernels to specific classification problems. These kernels are explained in Subsection 4.1.1.

More formally, given a set of data points, a SVM learning attempts to maximise a mathematical function based on these points. The data points are assumed to belong to two sets or classes. Determining the boundary between these sets will allow for separation. Consider P training points in a set $S = (x_1, y_1), (x_1, y_2) \dots (x_P, y_P)$. Assuming the subscript $i \in 1, 2, 3 \dots P$, each x_i is a data point in \mathbb{R}^n and each $y_i \in -1, 1$ is a label that is assigned to each data point to assign the point into one of the two classes S^+ or S^- .

The two classes are considered to be linearly separable in R^n resulting in at least one boundary that can be formed between the sets. This is known as the decision boundary. The decision boundary takes the shape of a plane in a higher-dimensional space and is regarded as the decision hyperplane which can be defined by the equation:

$$f(x) = \mathbf{w} \cdot x + b = 0; \mathbf{w} \in \mathbb{R}^n, b \in \mathbb{R} \quad (4.1)$$

where b is the provisional term with $b \in \mathbb{R}$ and \mathbf{w} is the normal vector. A linear combination of x_i with weights α_i is used to define vector \mathbf{w} of the decision hyperplane. The equation is as follows:

$$\mathbf{w} = \sum_{i=1}^M \alpha_{\alpha_i} x_i y_i \quad (4.2)$$

A solution to the classification problem is achieved by using the maximum margin classifier which attempts to achieve maximum separation between the two data sets S^- and S^+ . Figure 4.1 illustrates the many hyperplanes that can be drawn to separate the two distinct sets. It is clear, however, that only the bold green line achieves maximum separation between the two classes. The bold green line is known as the maximum margin hyperplane or the optimal hyperplane.

The maximum margin hyperplane is the hyperplane that passes through the midpoint between the boundaries of the two distinct data sets. Figure 4.2 depicts the use of the maximum margin which determines the optimal hyperplane that passes through the midpoint between sets S^+ and S^- . The training vectors that lie on the data set

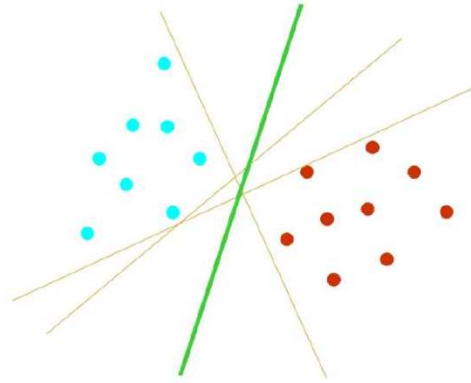


FIGURE 4.1: The optimal solution hyperplane and other solution hyperplanes to separate the two data sets S^- and S^+ [13].

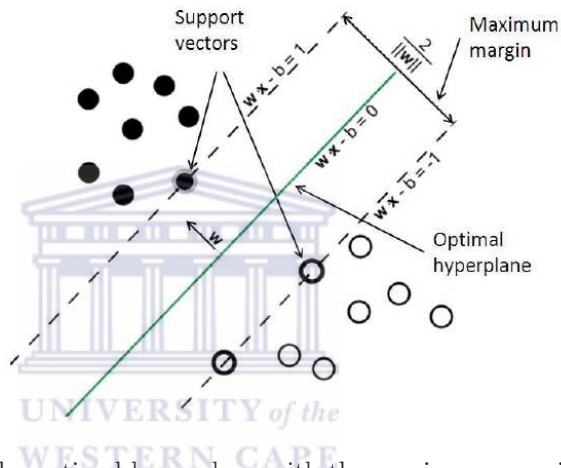


FIGURE 4.2: The optimal hyperplane with the maximum margin [13].

boundaries are called support vectors. A simple rescale can be performed for all x_i which are support vectors in w as follows:

$$w \cdot x_i + b = 1 \quad (4.3a)$$

$$w \cdot x_i + b = -1 \quad (4.3b)$$

The difference in distance d of the decision boundary and the margin can be expressed as:

$$d = \frac{2}{\|w\|} \quad (4.4)$$

Two important qualities are possessed by the optimal hyperplane:

1. It is able to separate the two data sets.
2. It has the maximum distance to the nearest data point of each of the two sets, which are also the support vectors.

The first quality ensures that all data points are classified correctly [84]. Therefore, an estimation of the parameters b and \mathbf{w} is carried out as follows:

$$y_i(\mathbf{w} \cdot x_i + b) \geq 1 \text{ for } y_i = 1 \quad (4.5)$$

and

$$y_i(\mathbf{w} \cdot x_i + b) \leq -1 \text{ for } y_i = -1 \quad (4.6)$$

A combination of these equations results in:

$$y_i(\mathbf{w} \cdot x_i + b) - 1 \geq 0, \forall i = 0, 1, 2, \dots, N \quad (4.7)$$

The second quality of the optimal hyperplane ensure that a maximum margin is obtained. This amounts to maximising the distance equation 4.4 which, in turn, amounts to the following minimisation problem subject to Equation 4.7:

$$\text{Minimise } \frac{1}{2} \|\mathbf{w}\|^2 \quad (4.8)$$

The Lagrange multipliers $\alpha_1, \alpha_2, \dots, \alpha_N \geq 0$ and the saddle point of the Lagrange function can be used to solve this problem as follows:

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^N \alpha_i (y_i(\mathbf{w} \cdot x_i + b) - 1) \quad (4.9)$$

Thus, the optimisation problem can be formulated as:

$$\text{Maximise } \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i,j=1}^N \alpha_i \alpha_j y_i y_j (x_i, x_j) \quad (4.10)$$

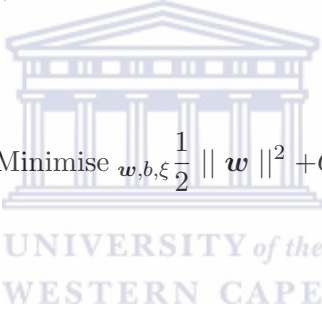
subject to

$$\alpha \geq 0, i = 0, 1, 2, \dots, N \text{ and } \sum_{i=1}^N \alpha_i y_i = 0 \quad (4.11)$$

Letting S be the subset of support vectors that correspond to positive Lagrange multipliers allows for a formulation of the discriminant function of the optimal hyperplane, given by:

$$f(x) = \sum_{i \in S} \alpha_i y_i (x_i x) + b \quad (4.12)$$

To determine hyperplanes for classification problems that are not linearly separable, a more complex structure is required. In such cases Equation 4.12 cannot be solved. A solution to this problem is a cost function which combines the maximisation and minimisation of the margin. Slack variables ξ_i , which measure the degree to which the data x_i is misclassified, are used to achieve a solution. This cost function can be formulated as:



Minimise $w, b, \xi \frac{1}{2} \|w\|^2 + C \cdot \sum_{i=1}^M \xi_i$ (4.13)

subject to

$$y_i (w \cdot x_i + b) \geq 1 - \xi_i \quad (4.14)$$

where C and slack variable $\xi_i \geq 0$ are constants.

The tradeoff between the degree of margin maximisation and the tolerable error is determined by the parameter C . In the mapping space, Mercer's theorem [80] is used so that the dot product of the vectors can be equally formed as a function of dot products of the corresponding vectors in the current space [51]. This can be formulated as:

$$\begin{aligned} K(x_i, x_j) &= \phi(x_i) \cdot \phi(x_j) \\ &= (x_i, x_i^2) \cdot (x_j, x_j^2) \\ &= x_i x_j + x_i^2 x_j^2 \\ &= x_i \cdot x_j + (x_i, x_j)^2 \end{aligned} \quad (4.15)$$

where $K(x_i, x_j)$ is a suitable kernel function. This equation only holds true if and only if the following expression also holds true for an arbitrary function h :

$$\int h(x)^2 dx \text{ is finite} \implies \int K(x, y)h(x)h(y) dx dy \geq 0 \quad (4.16)$$

It is thus possible to achieve linear separability in the higher dimensional space using a suitable kernel function, without an explicit knowledge of the form of ϕ . The optimisation problem can now be expressed as:

$$\text{Maximise } \sum_{i=1}^M \alpha_i - \frac{1}{2} \sum_{i,j=1}^M \alpha_i \alpha_j y_i y_j (x_i, x_j) \quad (4.17)$$

subject to

$$\alpha_i \geq 0 \text{ and } \sum_{i=1}^M \alpha_i y_i = 0 \quad (4.18)$$

The use of a complex curve to separate the data is not a suitable solution. Rather, an optimal hyperplane is drawn in the feature space that separates the data clearly. This leads to a decision function, in which S is the set of support vectors, given by:

$$f(x) = \sum_{i \in S} \alpha_i y_i (x_i, x) + b \quad (4.19)$$

4.1.1 Kernel Functions

An appropriate hyperplane is needed to separate data that is not linearly separable. In order to achieve this, the data is mapped from the current space onto a higher-dimensional feature space using a kernel function. Based on Mercer's theorem, the following kernels are used by the SVM for training and classification:

- Linear kernel: $K(x_i, x_j) = (x_i)^T \cdot (x_j)$
- Sigmoid kernel: $K(x_i, x_j) = \tanh(\gamma \cdot (x_i)^T \cdot (x_j) + r)$
- Radial Basis Function kernel: $K(x_i, x_j) = \exp(-\gamma \cdot \|x_i - x_j\|^2)$
- Polynomial kernel: $K(x_i, x_j) = (\gamma(x_i)^T(x_j) + r)^d$

where r, d and γ are kernel parameters and $\gamma > 0$ [33]. Choosing a kernel that is appropriate is vital as it directly affects the classification and prediction accuracy [18]. There is currently no criterion on the selection of an ideal kernel. The best kernel is commonly determined experimentally on a per-application basis.

Keerthi and Lin [41] demonstrated that problems that are non-linear in nature between attributes and multi-classes are best solved using the radial basis function (RBF) kernel. In comparison with the polynomial kernel, the RBF kernel has a reduced complexity due to the fact that the polynomial kernel has more parameters than the RBF Kernel [45]. The sigmoid kernel with certain parameters behaves similar to that of the RBF kernel. They also determined that the RBF kernel with parameters and linear kernel can achieve comparable accuracy. Li determined that the RBF kernel was the most suitable kernel for hand shape recognition [44].

In light of the above, the RBF kernel is selected for use in the proposed system. The RBF kernel has two parameters C and γ that can be optimised. The best C and γ values should be determined in order to achieve accurate prediction results. LibSVM provides a grid-search function which uses cross-validation on the training set to systematically and automatically determine the optimal values of C and γ . This procedure is used in the training phase of the proposed system detailed in a subsequent chapter.

4.1.2 SVM Multi-Class Classification

SVMs are generally intended towards problems involving two classes since the SVM is a binary classifier. This is evidenced by the derivation in the previous section that focused on separated two data sets S^+ and S^- . However, it is possible to use SVMs to separate more than two classes. Hsu and Lin [34] proposed various techniques to achieve this. Binary classifiers and decision strategies are combined to form these techniques. Subsections 4.1.2.1 – 4.1.2.3 discuss these common techniques.

4.1.2.1 One-vs-Others

Consider a K -class problem. The data points of each class k is separated from the data points of every other class. This is achieved by assuming two classes on the part of the data set for every $k \in \{1, 2, \dots, K\}$: one class representing k and one class representing all other classes. This results in K binary classifiers, each of which has a label for class $k \in \{1, 2, \dots, K\}$ and a label for the remaining classes.

The test pattern is presented to all K classifiers in the testing phase. The predicted label is taken to be the class with the maximum output value. This technique can result

in a longer computational time for both training and testing due to the possible large number of data points in each combination pair of classes.

4.1.2.2 One-vs-One

This technique groups each binary pair-wise combination of K classes in a single classifier that separates the two classes, resulting in $\frac{K(K-1)}{2}$ binary classifiers. The data points of the two classes involved in each classifier are used as negative and positive examples.

A test pattern is presented to all the classifiers. The results of the classifiers are combined using the Max-Wins algorithm [22]. This involves determining the class which has the maximum number of votes given the test pattern, which is taken as the predicted label. Compared to the One-vs-Others technique, this technique can have a shorter training time due to number of training examples that are involved in each of the classifiers. However, due to the large number of classifiers, the technique can have longer testing times.

4.1.2.3 Directed Acyclic Graph (DAG) Technique

The Directed Acyclic Graph (DAG) was introduced by Platt *et al.* [54]. It involves a similar training procedure to the One-vs-One technique in which $\frac{K(K-1)}{2}$ binary classifiers are trained using every binary pair-wise combination of the K classes. These classifiers are used to construct a rooted binary DAG that consists of $\frac{K(K-1)}{2}$ internal nodes and K leaves.

The DAG constructed for the case where $K = 4$ classes are to be separated is illustrated in Figure 4.3. Starting in the root node class, a comparison of classes 1 and 4 is carried out first. Assume that the class 1 is rejected. Thus, class 4 is provisionally preferred. From this point onwards, it is not necessary to classify against class 1. Comparisons against classes 2 and 3 take place in a similar manner down the graph. This implies that only $K-1 = 3$ steps are required to remain with a single predicted class.

Compared to the One-vs-Rest technique, this technique can have a shorter training and testing time. While the training time of the technique is longer than the One-vs-One technique, it has shorter testing times. It is an appropriate technique for multi-class SVMs.

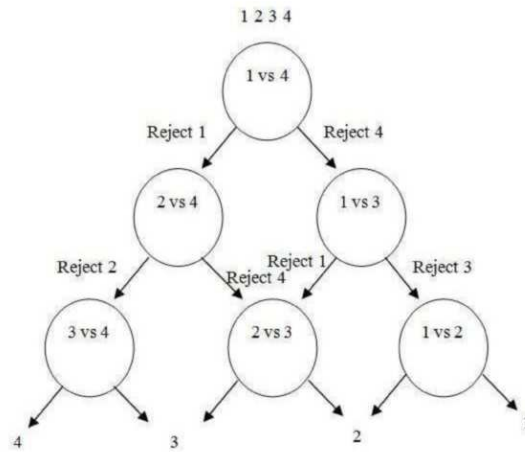


FIGURE 4.3: A 4-class Directed Acyclic Graph (DAG) with a rejection class at each node.

4.2 Hidden Markov Models (HMMs)

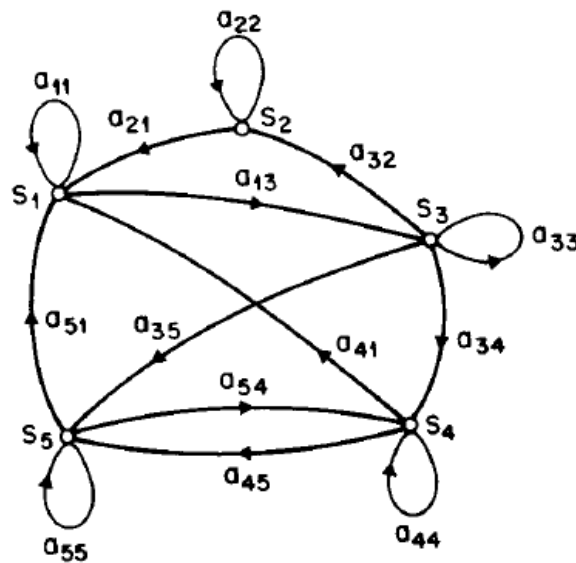
This section is organised as follows: Subsection 4.2.1 reviews the theory of Markov chains with Subsection 4.2.2 explaining the concept of hidden states by using a simple example: the coin toss. Subsection 4.2.3 discusses the problems faced by HMMs and how to overcome these problems.

4.2.1 Markov Chain

Consider a system where, at any given time, the system can only be in a single state S out of a total of N distinct states. Figure 4.4 illustrates the state machine consisting of states S_1, S_2, \dots, S_n where $n = 5$. The system can move from any state to another state based on the probabilities associated with each state. The time that it takes to move from one state to another is denoted as $t = 1, 2, 3, \dots, m$. The actual state time at t is denoted as $t = q_t$. A full description of the probability of this system would require information about the current states as well as all the previous states. This is known as a first-order Markov chain [57]. The probabilistic description is truncated to just the current and previous state as follows:

$$P[q_t = S_j | q_{t-1} = S_i, q_{t-2} = S_k \dots] = P[q_t = S_j | q_{t-1} = S_i] \quad (4.20)$$

The process on the right hand side of Equation 4.20 is only considered when it is independent of time. Therefore this leads to a set of state transition probability coefficients a_{ij} , represented as:

FIGURE 4.4: A Markov chain with states S_1, S_2, \dots, S_5 [57].

$$a_{ij} = P[q_t = S_j | q_{t-1} = S_i], 1 \leq i, j \leq N \quad (4.21)$$

with the state transition coefficients having the following properties:

UNIVERSITY of the
WESTERN CAPE

$$a_{ij} \geq 0 \quad (4.22a)$$

$$\sum_{j=1}^N a_{ij} = 1 \quad (4.22b)$$

The above stochastic process is known as an observable Markov model. An example of a 3-state Markov model of the weather is explained. Once a day, the weather is observed, with the possible weather states S being:

State S_1 = Rain; State S_2 = Cloudy; State S_3 = Sunny.


Given this information, a matrix M , which is the cross product $S \times S$, can be populated with the state transition probability. The matrix summarises each probability $P(S_i | S_j)$ that state S_i will be observed next if state S_j is currently observed, of the following form:

$$M = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \quad (4.23)$$

These probabilities may be determined empirically. Assume that matrix M is determined to be:

$$M = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{bmatrix} \quad (4.24)$$

Given that the weather on day 1 with $t = 1$ is sunny (S_3), a question can be formulated: “What is the probability that the weather sequence is going to be ‘sun-sun-rain-rain-sun-cloudy-sun’ for the next 7 days?”. This can be rewritten mathematically as the observation sequence $O = S_3, S_3, S_3, S_1, S_1, S_3, S_2, S_3$ for time $t = 1, 2, 3, \dots, 8$. The question can be expressed as determining the probability of O based on the model. The probability can be expressed as:



$$\begin{aligned} P(O|\text{Model}) &= P[S_3, S_3, S_3, S_1, S_1, S_3, S_2, S_3|\text{Model}] \\ &= P[S_3] \cdot P[S_3|S_3] \cdot P[S_3|S_3] \cdot P[S_1|S_3] \\ &\quad \cdot P[S_1|S_1] \cdot P[S_3|S_1] \cdot P[S_2|S_3] \cdot P[S_3|S_2] \\ &= \pi_3 \cdot a_{33} \cdot a_{33} \cdot a_{31} \cdot a_{11} \cdot a_{13} \cdot a_{32} \cdot a_{23} \\ &= 1 \cdot (0.8)(0.8)(0.1)(0.4)(0.3)(0.1)(0.2) \\ &= 1.536 \times 10^{-4} \end{aligned} \quad (4.25)$$

where π_i is the initial state probability of state i given by:

$$\pi_i = P[q_1 = S_j], 1 \leq i \leq N \quad (4.26)$$

Up to this point, a Markov model was considered where each state in the model corresponds to a given observed event. However, this model is very restrictive and may not be applicable to real world problems. The next subsection discusses the extension of the concept of Markov models to include cases in which the observation is a probabilistic function of the state.

4.2.2 Formulation of HMMs Based on Markov Models

The Hidden Markov Model (HMM) consists of a doubly-embedded stochastic process which has a hidden underlying stochastic process which can only be observed by an additional set of stochastic processes that produce the sequence of observations [57]. A simple coin toss example will be discussed which demonstrates how HMMs work.

Consider a person standing in a room with a barrier which prevents the person from seeing the other side. On the other side of the barrier is a second person who is repeatedly tossing a coin and reveals the result of each toss. A sequence of hidden coin tosses are performed with the observation $O = O_1, O_2, O_3, \dots, O_T$ with $O = H, H, T, T, T, H, T, T, H, \dots, H$ where H is heads and T is tails.

Given the above setup, a question arises as to how an HMM can model the observed sequences of heads and tails. The first problem that is encountered is determining what the states in the model correspond to and how many states there are in the model.

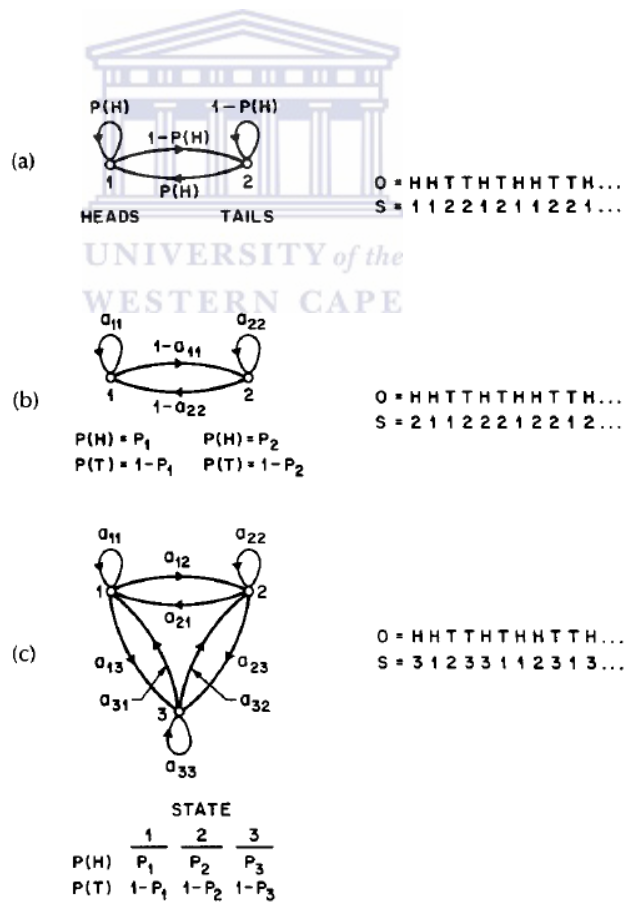


FIGURE 4.5: Three possible solutions to the coin toss modelling problem [57].

It can be assumed that a single biased coin was tossed. In this case, the problem can be modelled by 2 states that correspond to heads and tails. This interpretation results

in an observable Markov model. The only remaining concern would be to determining an appropriate value for the bias of the coin. Figure 4.5 (a) illustrates the model for this problem. Another possible solution to the problem is depicted in Figure 4.5 (b). Referring to the Figure, there are 2 states in the model with each state corresponding to a different biased coin. The probability distribution of each state is set to heads or tails. A state transition matrix is defined to determine the transitions between states. A third solution to this problem is to assume the use of 3 biased coins and choosing from the 3 states based on a probabilistic event. Figure 4.5 (c) illustrates this proposed solution.

The question thus arises as to which model would best predict the correct observation. The solution in Figure 4.5 (a) only has one unknown parameter, the one in Figure 4.5 (b) has four unknown parameters, and the one in Figure 4.5 (c) has nine unknown parameters. As the degrees of freedom increase, the HMM grows larger, consisting of more probabilities. It appears that using larger HMMs can result in better modelling results. However, it is shown in a subsequent section that larger solutions may not be practical.

The above example provides an overview of what an HMM is and how it can be applied. Formally, an HMM is characterised as follows:

1. The number of states in the model is N . Although the states are hidden there is valuable information attached to these states such as in the coin toss experiment where each state represented heads and tails. The individual states are denoted as $S = S_1, S_2, \dots, S_N$ with the state time at time t denoted q_t .
2. The number of distinct observation symbols that each state can have is denoted as M . The individual symbols are denoted by V with $V = V_1, V_2, \dots, V_M$.
3. The state transition probability distribution is defined as $A = a_{ij}$ where: $a_{ij} = P[q_{t+1} = S_j | q_t = S_i], 1 \leq i; j \leq N$.
4. The observation symbol probability distribution in state j is $B = b_j(k)$ where: $b_j(k) = P[V_k | q_t = S_j], 1 \leq j \leq N; 1 \leq k \leq M$.
5. The initial state distribution is $\pi = \pi_i$ where: $\pi_i = P[q_1 = S_i], 1 \leq i \leq N$.

Given each observation O_t is one of the symbols from V and the number of observations in the sequence is T , an observation sequence $O = O_1, O_2, O_3, \dots, O_T$ can be generated by an HMM with parameters N, M, A, B and π as follows:

1. Use the initial state distribution π to choose an initial state $q_1 = S_i$.

2. set $t = 1$.
3. Based on the symbol probability distribution $b_i(k)$ in S_i , choose $O_t = V_k$.
4. Based on the state transition probability a_{ij} of state S_i , transit to a new state $q_{t+1} = S_j$.
5. Set $t = t + 1$. Return to step 3 if $t < T$ otherwise exit.

The above procedure can be used to generate observations and/or as a model of how a given observation was generated by a HMM. From the discussion in this section it can be noted that a HMM requires two model parameters N and M , a set of observation symbols and three probability measures A, B and π to be fully defined. This can be summarised to define an HMM λ as:

$$\lambda = (A, B, \pi) \quad (4.27)$$

4.2.3 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. These problems take the form of the following three questions [57]:

1. Given the observation sequence $O = O_1, O_2, \dots, O_T$, where T is the number of observations, and a model $\lambda = (A, B, \pi)$, how can the probability of the observation sequence $P(O|\lambda)$ be computed efficiently?
2. Given the same observation sequence O and model λ , how can a corresponding state sequence $Q = q_1, q_2, \dots, q_T$ be chosen to best explain the observations?
3. How can the model parameters $\lambda = (A, B, \pi)$ be optimised to maximise the required probability $P(O|\lambda)$?

Problem 1 is known as the **evaluation problem**. Given, a model and a sequence, problem 1 is to determine how the probability that the observed sequence was generated from the model can be computed. Problem 2 is known as the **decoding problem**. Given a model and a sequence, problem 2 is to find the correct state sequence. Problem 3 is known as the **learning problem** or the **training problem**. Problem 3 is to optimise the model parameters in order to best describe how the observation sequence came out of the HMM. The following subsections explain how these problems are solved.

4.2.3.1 A Solution to the Evaluation Problem

Consider an observation sequence $O = O_1, O_2, \dots, O_T$, where T is the number of observations, and a model $\lambda = (A, B, \pi)$. $P(O|\lambda)$ needs to be calculated. The number of operations needed to compute $P(O|\lambda)$ is proportional to $T\dot{N}^T$, where N is the number of states. This is exponential and very computationally expensive, even for moderate values of T and N .

An alternative method is used to reduce the computational time. This method is the forward-backward algorithm. The algorithm makes use of two supporting variables known as the forward and backward variables. To evaluate this problem, only the forward variable is used. The forward variable, $\alpha_t(i)$, is defined as the probability of the partial observation sequence $O = O_1, O_2, \dots, O_T$ until termination at state S_i at time t . This is defined as:

$$\alpha_t(i) = P(O = O_1, O_2, \dots, O_T, q_t = S_i | \lambda) \quad (4.28)$$

Using induction, the previous equation can be solved for $\alpha_t(i)$ as follows:

1. Initialisation:

$$\alpha_t(i) = \pi_i b_i(O_1), 1 \leq i \leq N \quad (4.29)$$

2. Recursion:

$$\alpha_{t+1} = \left[\sum_{i=1}^N \alpha_t(i) a_{ij} \right] b_j(O_{t+1}), 1 \leq t \leq T - 1, 1 \leq j \leq N, \quad (4.30)$$

3. Termination:

$$P(O|\lambda) = \sum_{i=1}^N \alpha_T(i) \quad (4.31)$$

The time complexity for both the forward and backward section of this algorithm is proportional to $N^2\dot{T}$ which is considerably less complex, faster and of a more feasible computational time than N^T .

4.2.3.2 A Solution to the Decoding Problem

A solution to the decoding problem requires an optimal sequence of states for an observation sequence $O = O_1, O_2, \dots, O_T$. Various criteria can be used to determine an optimal solution. However, in many cases, such solutions can result in invalid states

sequences. For example, a criterion of achieving a high probability at each individual state results in the most likely state at every instant, without regard to the probability of occurrence of the sequence of states [57].

The Viterbi algorithm is the best solution to the decoding problem. The Viterbi algorithm was originally used for error reduction in noisy communication links [25]. The algorithm attempts to find the entire state sequence which maximises the probability of observing each state. Given an observation, the most correct sequence of states is found dynamically. This sequence is known as the Viterbi path. To find the Viterbi path or the most correct sequence of states $Q = q_1, q_2, \dots, q_T$ with observations $O = O_1, O_2, \dots, O_T$ with an auxiliary variable $\delta_t(i)$, we define the equation:

$$\delta_t(i) = \max_{q_1, q_2, \dots, q_{t-1}} P[q_1, q_2, \dots, q_t = i, O_1, O_2, \dots, O_t | \lambda] \quad (4.32)$$

The Viterbi path with the highest probability of observing the first t observations is $\delta_t(i)$. Given $\delta_t(i)$, the following recursive relationship will hold:

$$\delta_{t+1}(j) = \left[\max_{1 \leq i \leq N} \delta_t(i) a_{ij} \right], 1 \leq N, 1 \leq t \leq T - 1 \quad (4.33)$$

where:

$$\delta_1(i) = \pi_i b_i(O_1), 1 \leq i \leq N \quad (4.34)$$

The algorithm needs to keep track of each computed state to have the highest probability in order to retrieve the Viterbi path. An array $\psi_t(j)$ is used and the procedure can be solved as:

1. Initialisation:

$$\delta_1(i) = \pi_i b_i(O_1), 1 \leq i \leq N \quad (4.35a)$$

$$\psi_1(i) = 0 \quad (4.35b)$$

2. Recursion:

$$\delta_t(j) = [\delta_{t-1}(i) a_{ij}] b_j(O_t), 12 \leq t \leq T, 1 \leq j \leq N \quad (4.36a)$$

$$\psi_t(j) = [\delta_{t-1}(i) a_{ij}] b_j(O_t), 12 \leq t \leq T, 1 \leq j \leq N \quad (4.36b)$$

3. Termination:

$$P^* = [\delta_T(i)]; \quad (4.37)$$

4. Path backtracking:

$$q_t^* = \psi_{t+1}(q_{t+1}^*), t = T - 1, T - 2, \dots, 1 \quad (4.38)$$

4.2.3.3 A Solution to the Learning Problem

The parameters (A, B, π) can affect the performance of an HMM model. A series of observations can be used to optimise these parameters. These observations are called the training sequence. The learning problem aims to find a method that can adjust the parameters of a model (A, B, π) given a set of training sequences. However, with a finite number of training sequences, there is currently no way of estimating the models parameters optimally [57].

A solution to this problem is to select $\lambda = (A, B, \pi)$ such that $P(O|\lambda)$ is iteratively locally maximised. This technique is known as the Baum-Welch method [57]. The method is a generalised expectation maximisation method. This means that it is able compute the transition and emission parameters of a model given a set of training sequences. The probability of being in state S_i at time t and state S_j at time $t + 1$ with a given training sequence and a HMM model $\lambda = (A, B, \pi)$ is defined as $\varepsilon_t(i, j)$ and used to optimise the model using the Baum-Welch algorithm as follows:

$$\varepsilon_t(i, j) = P(q_t = S_i, q_{t+1} = S_j | O, \lambda) \quad (4.39)$$

The probability $\gamma_t(i)$ of being in state S_i at time t is defined as:

$$\gamma_t(i) = \sum_{j=1}^N \varepsilon_t(i, j) \quad (4.40)$$

The re-estimation formula for (A, B, π) is given by:

$$\bar{\pi}_i = \gamma_1(i) \quad (4.41)$$

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \varepsilon_t(ij)}{\sum_{t=1}^{T-1} \gamma_t(i)} \quad (4.42a)$$

$$\bar{b}_j(k) = \frac{\sum_{t=1, O_t=v_k}^T \gamma_t(j)}{\sum_{t=1}^T \gamma_t(j)} \quad (4.42b)$$

$\bar{\lambda} = (\bar{A}, \bar{B}, \bar{\pi})$ denotes the re-estimation formula. The variable \bar{A} is the probability that a transition will occur from state i to state j . The variable \bar{B} is the ratio of observing symbol O_k while in state j . Lastly, the variable $\bar{\pi}$ is the probability of being in state i at time t .

4.3 Hidden Markov Support Vector Machines (HM-SVMs)

As explained in the previous section, HMMs label a sequence as a Markov chain. This avoids direct dependencies between subsequent observations. Despite their success, however, HMMs have at least three major limitations: They are trained non-discriminatively; the conditional independence assumptions are restrictive; and they are unable to make use of kernel based methods [3].

HM-SVMs are able to address the shortcomings of HMMs, while holding onto some of the key strengths of HMMs, namely, the Markov chain dependency structure between labels. The two important factors added to HMMs by HM-SVMs are the maximum margin principle and a kernel-centric approach to learning non-linear discriminant functions. These two qualities are inherited from SVMs. Many approaches have been proposed to achieve hybrids between HMMs and SVMs. However, the approach by Altun *et al.* is the most prominent technique that has been the basis for many other similar approaches [15, 17, 23, 30, 39, 61, 67, 76–79, 81, 82, 89]. The technique was discussed in Chapter 2 [3]. It is selected for use in the comparison proposed in this research. In future, other HM-SVM techniques can be investigated and a comparison of these techniques, carried out. Subsections 4.3.1 – 4.3.4 discuss the framework of the HM-SVM hybrid proposed by Altun *et al.* [3].

4.3.1 Input and Output Mappings via Joint Feature Functions

Prior to describing learning approach, a general framework is proposed which allows for mappings to discrete output spaces. This framework absorbs many issues such as binary, multi-class and multi-label classification and label sequence learning [29]. A general approach is to learn a w -parametrised discriminant function F given by $F : X \times Y \rightarrow \mathbb{R}$ with pairs of input/output that can be maximised with a response variable to produce a prediction. Therefore, the function f is given by:

$$f(x) = \arg \max_{y \in Y} F(x, y; w) \quad (4.43)$$

The objective of interest is a situation in which F is linear with a specific combined input and output feature representation defined as $\Phi(x, y)$ formulated as:

$$F(x, y; w) = \langle w, \Phi(x, y) \rangle \tag{4.44}$$

The example of natural language parsing will be used to discuss the previous mentioned equation. Natural language parsing is used to predict the parse tree y that generates a given input sentence $x = (x_1, x_2, \dots, x_n)$. Each node in the tree y is generated by a rule of a weighted context-free grammar assumed to be in Chomsky normal form [31]. Figure 4.6 illustrates the example that is to be explained.

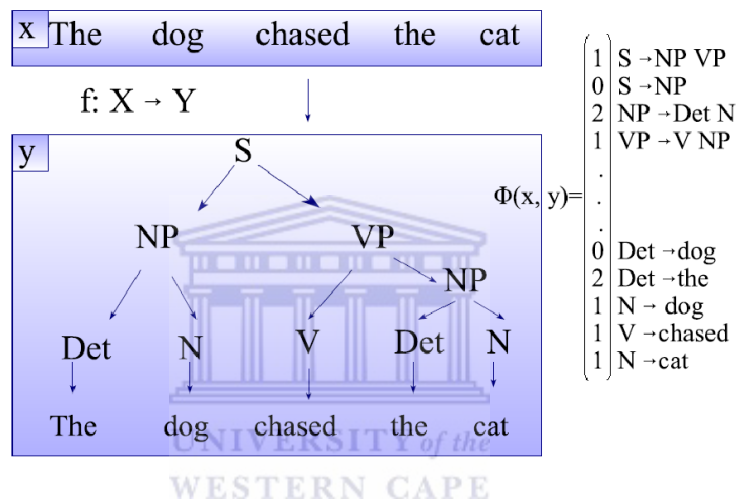


FIGURE 4.6: Illustration of the natural language parsing model [50].

Referring to Figure 4.6, the input set $x = (The, dog, chased, the, cat)$ with y being a grammar-based parse tree with rules $g_j = (S \Rightarrow NP, VP)$. Function f maps a given input x to a parse tree y . The number of all possible solutions d can be denoted as \mathbb{R}^d . Mapping $(\Phi(x, y) = \Phi_1(x, y) + \Phi_2(x, y))$ represents the inter-dependencies between labels and the nodes of the tree. $\Phi(x, y)$ is a histogram vector that counts how often each grammar rule g_j occurs in the tree y . Thus $f(x; w)$ can be computed by obtaining the structure $y \in Y$ that maximises $f(x, y; w)$

To avoid performing an explicit mapping Φ , the kernel functions of the SVM are applied. This is possible since F is linear if a kernel K is assumed over the joint input and output space as:

$$K((x, y), (\bar{x}, \bar{y})) = \Phi(x, y), \langle \Phi(\bar{x}, \bar{y}) \rangle \tag{4.45}$$

and assuming that F has a dual representation over a set of limited samples $(\tilde{x}_1, \tilde{y}_1), \dots, (\tilde{x}_n, \tilde{y}_n)$ in terms of an expansion:

$$F(x, y) = \sum_{i=1}^m \alpha_i K((\tilde{x}_i, \tilde{y}_i), (x, y)) \quad (4.46)$$

The basis of this method is to obtain features from, both, input patterns as is the case in binary classification, as well as jointly from input/output pairs.

4.3.2 Hidden Markov Chain Discriminants

The objective of label sequence learning is to learn a mapping f from a set of observation sequences $x = (x^1, x^2 \dots x^n)$ to the set of labelled sequences $y = (y^1, y^2 \dots y^n)$, where the labels belong to a label set Σ as in $y^t \in \Sigma$. The range of f is finite for every x because only label sequences of the same length as each observation sequence are considered.

The output space Y consists of all possible labelled sequences. In order to define a suitable parametric discriminant function F , a mapping Φ is required that extracts features from a pair (x, y) of observations and labels.

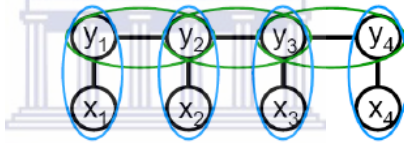


FIGURE 4.7: The interaction between neighbouring label states [50].

Two feature types are defined: the interactions between each label and the attributes of the observation vectors; and the interactions between labels adjacent in the chain. Figure 4.7 illustrates these interactions. The blue ellipses represent interactions between labels and observation vector attributes while the green ellipses represent the interaction between adjacent labels. The main objective is to achieve Viterbi-like decoding by defining Φ such that f can be computed from F efficiently. This requires a restriction of inter-label interactions to nearest neighbours, while also making use of interactions between observations and labels.

All of the features extracted at position t are simply stacked together into function $\Phi(x, y; t)$. This feature map is accumulated over T sequences (x, y) as follows:

$$\Phi(x, y) = \sum_{t=1}^T \Phi(x, y; t) \quad (4.47)$$

4.3.3 Hidden Markov Perceptron Learning

An online learning-based approach to label sequencing was proposed by Collins *et al.* [16]. The algorithm is defined as: A given sequence x_i and the optimal decoding $f(x_i)$ is computed. This is equivalent Viterbi decoding. If the correct output sequence is predicted i.e. $y_i = \hat{y}$ then no update is performed. However, if the sequence is incorrect, a weight vector w is updated based on the difference vector ($\Delta\Phi = \Phi(x_i, y_i) - \Phi(x_i, \hat{y})$). This can be also written as ($w^{updated} \leftarrow w^{previous} + \Delta\Phi$).

A new dual formulation of the perceptron algorithm is derived to avoid the evaluation of the feature map and exact specification of the discriminant function given by:

$$F(x, y) = \sum_i \sum_{\bar{y}} \alpha_i(\bar{y}) \langle \phi(x_i, \bar{y}), \phi(x, y) \rangle \quad (4.48)$$

If an update is required for training sequence (x_i, y_i) and \hat{y} that has been incorrectly decoded, $\alpha_i(y_i)$ is incremented and $\alpha_i(\hat{y})$ is decremented by 1, respectively. The above formulation is valid for any joint feature function Φ on label sequences.

In order to perform the Viterbi decoding, the transition and observation cost matrix M_i for the i -th sequence needs to be computed. This is derived as:

$$H_i^{s\sigma} = \sum_j \sum_t \beta(j, t, \sigma) k(x_i^s, x_j^t) \quad (4.49)$$

where the values $\delta(\sigma, t)$ are the coefficients of the transition matrix. Once this calculation is completed, the Viterbi decodes the best path to take at each position in the sequence. The algorithm in Figure 4.8 is the dual perceptron algorithm for learning via joint feature functions used to achieve Viterbi decoding.

```

1: initialize all  $\alpha_i(\mathbf{y}) = 0$ 
2: repeat
3:   for all training patterns  $\mathbf{x}_i$  do
4:     compute  $\hat{\mathbf{y}}_i = \arg \max_{\mathbf{y} \in \mathcal{Y}} F(\mathbf{x}_i, \mathbf{y})$ , where
        $F(\mathbf{x}_i, \mathbf{y}) = \sum_j \sum_{\bar{\mathbf{y}}} \alpha_j(\bar{\mathbf{y}}) \langle \Phi(\mathbf{x}_i, \mathbf{y}), \Phi(\mathbf{x}_j, \bar{\mathbf{y}}) \rangle$ 
5:     if  $\mathbf{y}_i \neq \hat{\mathbf{y}}_i$  then
6:        $\alpha_i(\mathbf{y}_i) \leftarrow \alpha_i(\mathbf{y}_i) + 1$ 
7:        $\alpha_i(\hat{\mathbf{y}}_i) \leftarrow \alpha_i(\hat{\mathbf{y}}_i) - 1$ 
8:     end if
9:   end for
10: until no more errors

```

FIGURE 4.8: Dual perceptron algorithm [3].

4.3.4 Hidden Markov Support Vector Machine

The section focuses on deriving a maximum margin expression for the joint kernel learning setting. A general notion of a separation margin is defined by a margin of a training example with respect to a discriminant function F , expressed as:

$$\gamma_i = F(x_i, y_i) - \max_{y \neq y_i} F(x_i, y) \quad (4.50)$$

Therefore the maximum margin problem can be thought of as finding w , a weight vector that maximises $(\min_i \gamma_i)$. Assuming a fixed functional margin as $(\max_i \gamma \geq 1)$, as is a standard method in maximum margin classification with binary labels, results in the following quadratic optimisation problem:

$$\min \frac{1}{2} \|w\|^2 \text{ subject to } F(x_i, y_i) - \max_{y \neq y_i} F(x_i, y) \geq 1, \forall i \quad (4.51)$$

A set of linear constraints can be used to replace every non-linear constraint in Equation 4.51 as follows:

$$F(x_i, y_i) - F(x_i, y) \geq 1, \forall_i \text{ and } \forall_y \neq y_i \quad (4.52)$$

The equation can be rewritten by adding a threshold θ_i for every example:

$$z_i(y)(F(x_i, y) + \theta_i) \geq \frac{1}{2}, z_i(y) = \begin{cases} 1 & \text{if } y = y_i \\ -1 & \text{otherwise} \end{cases} \quad (4.53)$$

The equation is expressed in terms of function z_i to emphasise that a binary classification problem has been obtained. As per standard procedure, as shown in Section 4.1, the Lagrangian function is used to obtain the following Lagrangian dual:

$$\max W(\alpha) = -\frac{1}{2} \sum_{i,y} \sum_{j,\bar{y}} \alpha_i(y) \alpha_j(\bar{y}) z_i(y) z_j(\bar{y}) k_{I,j}(y, \bar{y}) + \sum_{i,y} \alpha_i(y) \quad (4.54)$$

subject to:

$$\begin{aligned} \alpha_i(y) &\geq 0, \forall_i = 1, 2, \dots, n \forall y \in Y \\ \sum_{y \in Y} z_i(y) \alpha_i(y) &= 0, \forall_i = 1, 2, \dots, n \end{aligned} \quad (4.55)$$

where:

$$k_{i,j}(y, \bar{y}) = \langle \phi(x_i, y), \phi(x_j, \bar{y}) \rangle. \quad (4.56)$$

4.4 Summary

This chapter discussed the three machine learning techniques that are used in the proposed SASL gesture recognition system of this research. A detailed description of the classification approach used by Support Vector Machines, Hidden Markov Models and Hidden Markov Support Vector Machines was provided. The next chapter describes the use of these techniques in the implementation of the proposed SASL gesture recognition system.



Chapter 5

Design and Implementation of the Gesture Recognition System

This chapter discusses the design of the gesture recognition system proposed in this research. The system can be viewed, in the most basic form, as having two stages illustrated in Figure 5.1. The first stage comprises of image processing techniques used to extract features from a SASL video. Details of this stage are provided in Section 5.1. The second stage entails classifying the given features into specific gesture classes using HMMs and HM-SVMs. Section 5.2 discusses the feature vector used to characterise gestures, as well as the training of the HMMs and HM-SVMs used to achieve classification.

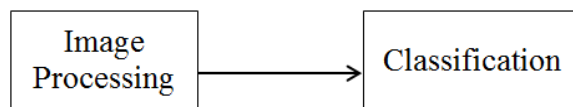


FIGURE 5.1: Broad overview of the proposed gesture recognition system.

5.1 Feature Extraction

Figure 5.2 is an overview of the proposed feature extraction procedure. The procedure consists of the following main segments: skin detection; the locating procedure; the correction procedure; hand shape recognition; and hand motion extraction. The skin detection procedure locates the user's face and nose region to determine the skin distribution of the user resulting in a skin image. The first time that the system runs, the

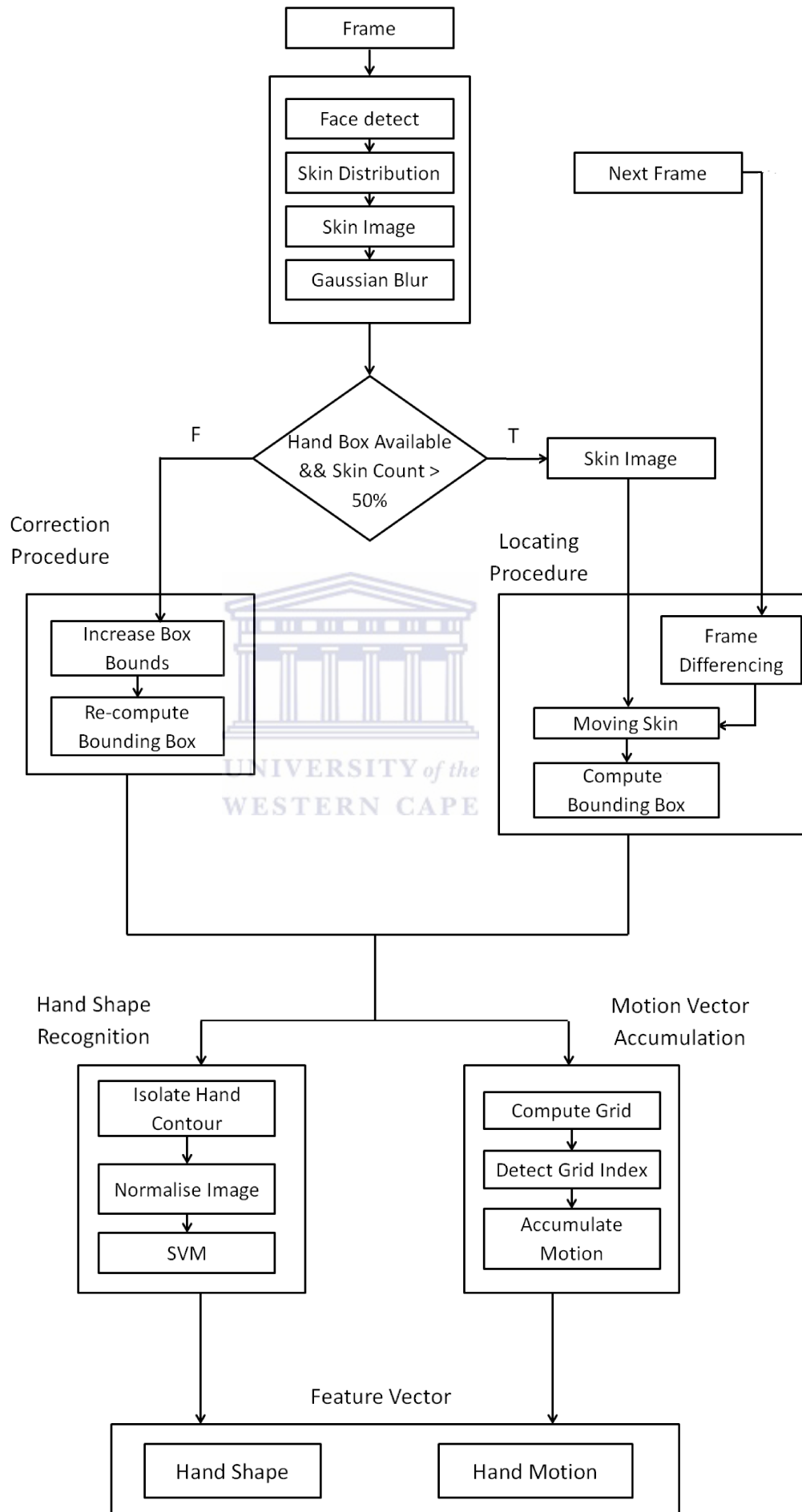


FIGURE 5.2: Overview of the feature extraction component.

locating procedure is run after skin detection in order to detect the position of the hands and draw initial tracking windows around them. From this point onwards, after skin detection, a check is carried out on the amount of skin in either of the tracking windows.

If there is less than 50% skin in either tracking window, this indicates that a large displacement of that hand has taken place. The locating procedure is carried out again on that tracking window. However, if the amount of skin in each tracking window is greater than 50%, this is taken as an indication that the hand in that window has moved slightly, if at all. The correction procedure is carried out. The correction procedure is less computationally expensive than the locating procedure and attempts to correct the location of the window with the new location of the hand. This methodology makes it possible to track the hands in real-time.

Thereafter, the location and contours of the hands are used to extract the two required features: hand motion and hand shape. The following subsections 5.1.1 – 5.1.4 discuss each of these segments in further detail.

5.1.1 Skin Detection

The system uses the Viola-Jones face detection method [88] to acquire the position of the user's face in the image. The face detection component is the basis of the image pre-processing procedure. The resulting frame is used to determine the centre of the facial frame. The centre of the face is used to achieve two objectives [1]:

- To compute a skin colour distribution that represents the individual's skin tone.
- To normalise every image in the image sequence by repositioning the image such that the person is in the centre of the frame.

The area of the nose is located in the centre of the facial frame [1]. A 10×10 pixel area around the nose region is extracted to compute the skin colour distribution, as explained in a previous chapter. Brown [9] determined that only using the hue component as opposed to Li [44] who used the Hue and Saturation yields better skin detection results. Thus, the Hue component of this area is used and represented in a histogram which serves as a lookup table for skin pixels. This is back-projected onto the original image to achieve a skin image. The resulting image is binarised by applying a threshold of 60, as explained before. The resulting image highlights all skin pixels in the frame and is henceforth referred to as the "skin image". Figure 5.3 illustrates the skin image.

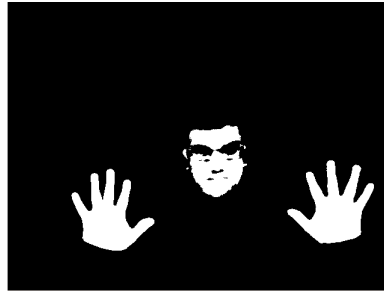


FIGURE 5.3: The skin image.

Gaussian blur is applied to the skin image to reduce sources of noise and obtain a clearer outline of the hands. Figure 5.4 illustrates the image after Gaussian smooth has been applied.

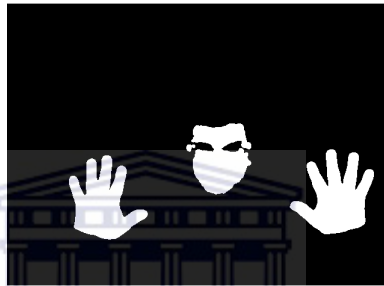


FIGURE 5.4: The skin image after Gaussian smooth has been applied.

5.1.2 The Locating Procedure

Frame differencing is used to separate the background from the foreground. This technique provides information on the location of the moving objects – in this case the hands. The resulting image highlights all areas in the frame that have moved relative to the previous frame and is henceforth called the “motion image”. A logical AND operation is performed between the motion image and skin image resulting in an image that only contains those pixels in the frame that are, both, skin and have moved i.e. moving skin pixels. This image is depicted in Figure 5.5 and is henceforth referred to as the “moving skin image”. This procedure is effective in filtering out the majority of sources of noise in the background. The face is also eliminated since its movements are below the motion threshold.

Upon acquiring the moving skin image an approach is used to locate the hands. The moving skin image is scanned from left to right until the first skin pixel is encountered. This is regarded as the side of the hand and its coordinates are used as a reference point for the next step. Figure 5.6 illustrates the point representing the side of the hand which is displayed as a green dot in the figure. Since the side of the hand has been located,



FIGURE 5.5: The moving skin image.

the entire image does not have to be scanned again in order to locate the top, bottom and opposite side of the hand.

Using the reference point, the horizontal bound of a search area is determined by adding the width of the face to the x coordinate of the reference point, since the hand is proportionate to the face [19]. Figure 5.7 illustrates the new point representing the horizontal bound of the search area on the right side of the hand. Using the y coordinates of the two points, the vertical bounds of the search area are determined by adding one facial height above and below the two points. The vertical extent of the search area is illustrated as the yellow lines in Figure 5.8.



FIGURE 5.6: The side of the hand located.



FIGURE 5.7: The right side of the hand located.

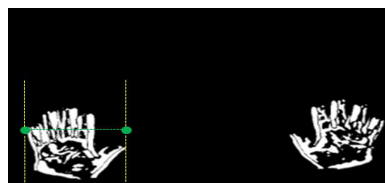


FIGURE 5.8: The extent of the search area to locate the top and bottom of the hand.

The resulting search area is scanned to locate the hand precisely. A scan is carried out in the search area from right to left to locate the exact coordinates the right-most pixel

of the hand. A search is also carried out upwards from the bottom, and downwards from the top of the search area to locate the top-most and bottom-most bounds of the hand. This procedure results in a rectangle that isolates the hand precisely. This method is used to isolate the hand regardless of the shape of the hand. Figure 5.9 illustrates various hand shapes that are correctly isolated with a dynamic box drawn around it.

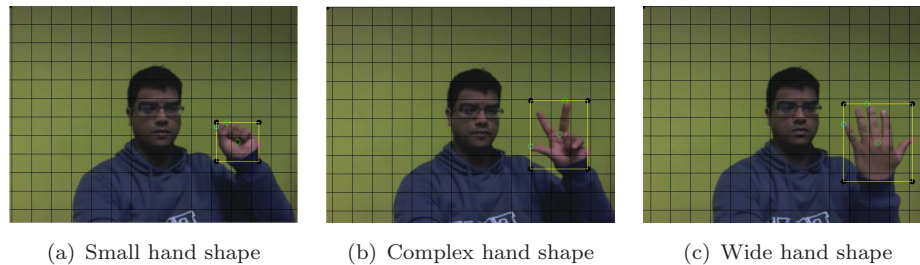


FIGURE 5.9: Various hand shapes that are correctly isolated by a dynamic box.

This procedure is repeated by scanning the moving skin image from right to left until the first skin pixel of the second hand is encountered. The same procedure ensues to draw a dynamic box around the second hand.

5.1.3 The Correction Procedure

For small motions of the hand between frames, a correction procedure is applied to avoid the computational cost of re-locating the hand in every frame. For example, the case where the finger tips have moved marginally outside the dynamic box is portrayed in Figure 5.10 (a). The metric that is used to determine whether correction should be applied is a skin count within the current dynamic box. If the skin count is above 50% of the original amount, it is determined that the majority of the hand is still within the dynamic box and a only a correction needs to be applied.

To achieve correction, a region growing approach is applied in which a search area is computed by increasing the bounds of the dynamic box by 10% and a search carried out for new bounds of the hand. Figure 5.10 (b) depicts the new search area represented as the red box. This ensures that the hands are correctly and accurately located for small movements without incurring the computational cost of an image-wide search.

This procedure is applied to each hand separately, as necessary.

5.1.4 Hand Shape Recognition

Connected component analysis (CCA) is applied within the dynamic box of each hand image to obtain a hand contour. In each case, the hand is regarded as the largest

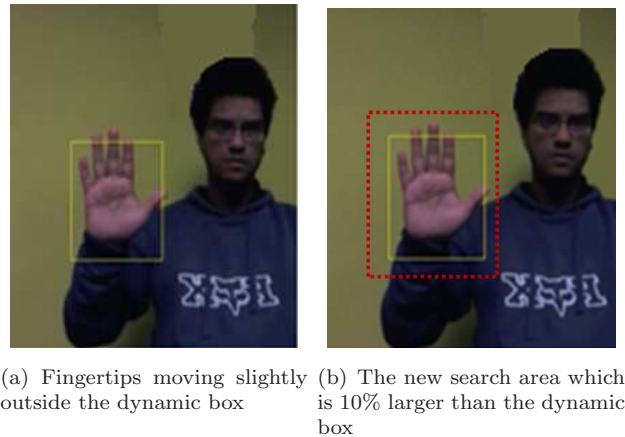


FIGURE 5.10: The correction procedure.

contour. A geometrical computation algorithm is used to draw an oriented minimum bounding box around each hand contour. Figure 5.11 illustrates the hand contour with a minimum bounding box drawn around it. Each minimum bounding box is rotated to an upright position by aligning its principal axis with the vertical axis to overcome misalignment invariance [44].

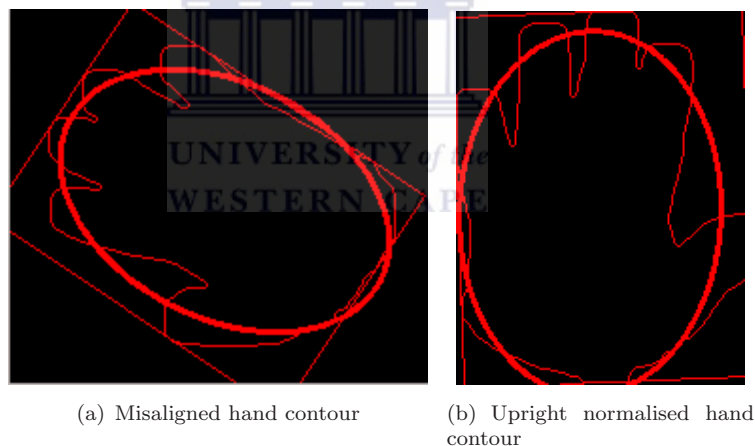


FIGURE 5.11: Normalising the hand contours with a minimum bound box.

The resulting hand contour is isolated and scaled to a size of 60×60 pixels and used as input to a Support Vector Machine for hand shape classification. Figure 5.12 illustrates the scaled binary image of the hand contour used by the SVM to predict the hand shape.

The training, optimisation and testing of the SVM for hand shape recognition is discussed in Chapter 6.



FIGURE 5.12: A scaled binarised hand contour image.

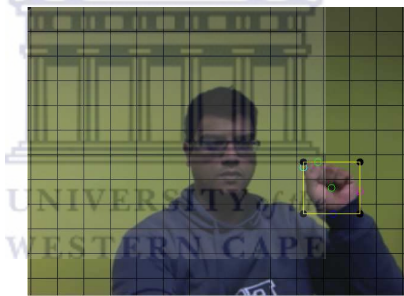


FIGURE 5.13: A dynamic grid centred on and proportional to the face used to represent the location of the hands.

5.1.5 Hand Motion Extraction

Across the entire image sequence, the locations of the hands in each frame are accumulated as a motion vector describing the motion of the hands across the image sequence. Based on Naidoo's [49] work, in order to represent the location of the hands in the image discretely, a grid centred on, and proportional to the face is drawn. The grid cells are a fourth of the signers facial size, as depicted in Figure 5.13. This results in a total of 98 grid cells. For each frame in the sequence, the amount of skin in each individual cell is computed and used with the following criteria, based on Naidoo's work, to assign a skin score to the cell:

1. if the skin pixels within the cell are less than 40%, the cell receives a value of 0.
2. if the skin pixels are greater than 40%, the cell receives a value of 1.

5.2 Feature Vector

This section discusses the structure of the feature vectors for the HMM and HM-SVMs.

The HMM-based system is divided into two categories: HMMs using hand motion as a feature vector and HMMs using the hand motion and hand shape as a feature vector. To achieve the feature vector in the first case, the output of the hand motion recognition is accumulated. To achieve the feature vector in the latter case, the output of the hand recognition result is combined with the result of the motion. The training and testing procedure of the hand shape recognition component is discussed in Chapter 6.

Table 5.1 illustrates a snippet of the motion feature vector. Each unique row of the vector is considered a state in the HMM for training a model $\Phi(A, B, \pi)$ where A is the state probability distribution and B is the observation symbol which is computed using the feature vector. For example, the unique motion sequence in Frame 1, which was also observed in Frame 2, is labelled as state 0. The feature vector consists of 98 features corresponding to the 98 grid cells. Thus, an ergodic HMM would produce 2^{98} possible states in which every state is connected to every other state. The human body has physical constraints and only a few number of these states will ever occur [49]. Also, a moderate signing speed is expected which dictates that the hands can only move to specific states from any given state e.g. they cannot skip from the bottom of the frame to the top of the frame directly. Therefore, an ergodic HMM is not used. Rather, new states will be introduced to the model as they appear in training. The resulting model will only consist of the appropriate states with a state probability distribution matrix determining the transitions between states.

Given an unseen image sequence, this feature vector is produced and used as input to the trained models of each SASL gesture. The model that produces the highest probability is chosen as the correct interpretation of the SASL gesture. The testing of the HMM will be discussed in Chapter 6.

The feature vector for the hand motion and hand shape combination is produced by adding the label of the hand shape predicted by the SVM in each frame to the beginning of the hand motion sequence of that frame. Each unique shape and motion sequence is considered a unique state in this case. A separate HMM is trained on this feature vector in the same manner as for the motion-only HMM. The procedure for predicting a test feature vector using the hand shape and motion HMM also remains the same. The testing of this HMM is also discussed in Chapter 6.

The HM-SVM is also trained on the feature vector that includes hand motion and hand shape. Each unique sequence is similarly taken as a unique state in the HMM.

Chapter 6

Experimental Results and Analysis

This chapter describes the experiments carried out in order to answer the research questions posed in Chapter 1 and assess the accuracy of components of the proposed gesture recognition system and the proposed gesture recognition system as a whole.

Section 6.1 describes the experimental setup used in all the experiments. Section 6.2 describes the testing of the proposed hand tracking component. This is the basis of all subsequent components of the proposed gesture recognition component and the experiment aims to demonstrate that this component is accurate. Section 6.3 describes the experiments carried out in order to optimise the hand shape feature vector and SVM used to recognise hand shapes, and subsequently assess the accuracy of this component. This component is the basis for accurate gesture recognition using a combination of hand shape and hand motion.

Finally, 6.4 describes the two experiments carried out in order to answer the two research questions posed in Chapter 1. The first experiment aims to answer the question of whether a combination of hand shape and hand motion would be more accurate than using hand motion only. The second experiment aims to answer the question of how HMMs and HM-SVMs compare in terms of accuracy when applied to SASL gesture recognition using hand shape and motion as gesture descriptors.

The chapter is then concluded.

6.1 Experimental Setup

The experiments carried out in this research were performed on an AMD FX 8120, 3.11 GHz CPU and 8 GB RAM desktop PC running the Kubuntu Linux 12.04 x64 operating system. A Logitech HD Pro Webcam C910 running at a resolution of 640×480 was used to capture the video frames. A illustration of the experimental setup is depicted in Figure 6.1.



FIGURE 6.1: Experimental setup.

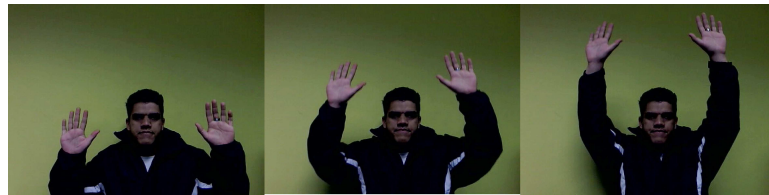
6.2 Testing of the Hand Tracking Component

A vital component of the system is hand tracking. The latter steps of feature extraction such as hand motion extraction and hand shape recognition are all based on acquiring the hands correctly. It is important to ensure that the hands are correctly located and tracked. Thus, this component was tested first. This experiment intends to determine the accuracy of the hand tracking strategy within the scope of the research set out in Chapter 1. While a demonstration of an accurate final gesture recognition result in a subsequent section would have implied accurate hand tracking as well, this experiment is included for completeness.

6.2.1 Hand Tracking Experimental Procedure

For this experiment, ten subjects of different skin tone were instructed to perform a specific hand motion sequence. Figure 6.2 (a) and 6.2 (b) describe the hand motion

sequence visually. It can be observed that the sequence is well representative of hand motions in the signing space. These sequences were captured resulting in ten videos.



(a) Subject moving his hands vertically.



(b) Subject moving his hands horizontally and outside the frame.

FIGURE 6.2: The hand motion sequence performed by test subjects in the hand tracking experiment.

The training method discussed in Chapter 5 was used to track the hands in each of the ten videos. Similar to the hand tracking experimentation carried out in [44], each frame of each motion sequence was manually inspected to determine if the hands were correctly tracked to determine the accuracy of the hand tracking strategy. The standard for a successfully tracked frame was defined as the hand(s) in the frame being encased by dynamic boxes, with all the fingers and the entire region beyond the wrist fully encased inside the boxes. If this was not the case for either or both hands, the frame was marked as incorrectly tracked.

6.2.2 Results and Analysis

The success rate of the hand tracking procedure is illustrated in Table 6.1. As observed, the total number of frames in the video of each subject was different. This value is included in each case.

Referring to Table 6.1, every subject obtains a virtually perfect tracking accuracy of over 98.5%. The tracking procedure achieves a virtually perfect overall average success rate of 99.0%. This result is attributed to the strategic assumptions made in Chapter 1 in order to simplify the problem of hand tracking and focus specifically on gesture recognition in the research. In any case, this result clearly demonstrates that the tracking strategy employed is accurate and highly robust to variations in the skin colour and body dimensions of test subjects. As such, the first objective set out in Chapter 1 has been successfully achieved.

Subject	Total Frames	Correctly Tracked	Percentage Correct (%)
1	623	619	99.3
2	612	609	99.5
3	586	581	99.1
4	526	519	98.6
5	541	537	99.2
6	633	628	99.2
7	591	586	99.1
8	607	602	99.1
9	578	571	98.7
10	618	613	99.1
Total	5915	5865	99.0

TABLE 6.1: Hand tracking accuracy results.

6.3 Training and Testing of the Hand Shape Recognition Component

This section describes the training and optimisation of the SVM used to recognise hand shapes and the subsequent experiment carried out to assess the accuracy of this component.



6.3.1 Training Set

To gain a clear understanding of the testing procedure in this section, the training data and testing is explained. The next section explains the data set of 35 SASL gestures from the Fulton School for the Deaf SASL Dictionary [32] that were selected and used to train and test the proposed gesture recognition system. At this stage, it is important to note that, the dictionary was used to determine which gestures to be used and 35 gestures that make use of 10 unique hand shapes were selected. Figure 6.3 depicts the 10 hand shapes. A dataset was constructed of hand images of five subjects, with each subject differing in hand size. Figure 6.4 depicts the five subjects used to train the hand shape recognition component. Each subject was instructed to perform the 10 different hand shapes sequentially from hand shape 1 to hand shape 10. Each subject was asked to hold each hand shape for at least 3 seconds. A training set was produced by selecting 10 images of each hand shape from each subject resulting in a total of 500 examples – 50 examples for each of 10 hand shapes.



FIGURE 6.3: The 10 unique hand shapes in the 35 SASL gestures (Top row, left to right: shapes 1 to 5) (Bottom row, left to right: shapes 6 to 10).



FIGURE 6.4: The five subjects used to train the hand shape recognition component.

6.3.2 Training and Optimisation of the Radial Basis Function Kernel Parameters

It was decided in Section 4.1.1 that the RBF kernel is the most suitable kernel for hand shape recognition and thus used in the proposed system.

The optimisation of the RBF kernel involves selecting optimal values for the two parameters C and γ in the context of the classification problem. This can be manually carried out in a brute-force approach of iterating through a large range of C and γ values, evaluating the accuracy of the model resulting from the use of each parameter pair and selecting the pair that achieved the highest accuracy. The grid-search function of LibSVM [13] carries out this procedure. It iterates through a large range of C and γ pairs and for each pair, the cross validation accuracy on the training set is computed. The pair that achieves the highest v -cross validation accuracy is selected as the optimum pair. v -Cross validation divides the data set into training sets and testing sets whereby $v - 1$ samples are used to train the model and the remaining sample is used to test the system and obtain an accuracy. A total of v different combinations of training and testing sets are produced and the average accuracy across all v sets is the cross validation accuracy.

In addition to optimising the C and γ values, it was expected that the resolution of the hand contour image used as a feature vector would affect the accuracy of the system. Therefore, four different resolutions were evaluated to determine the most suitable one.

The resolutions investigated were 20×30 , which was the only resolution proposed by Li [44], as well as three additional resolutions of 40×40 , 50×50 , and 60×60 pixels.

Each hand shape image in the hand shape training set was pre-processed using the procedure mentioned in Section 5.1.4 in order to obtain a hand contour image. The image was scaled to 4 resolutions of 20×30 , 40×40 , 50×50 , and 60×60 . This resulted in training sets of 500 hand shape images for each resolution. Thereafter, the grid search function of LibSVM was used to determine the optimal C and γ values for the training set of each resolution. The results are summarised in Table 6.2. The table summarises the optimal C and γ value in each case and the corresponding cross validation accuracy obtained from the optimal parameter pair.

Resolution	C	γ	Accuracy (%)
20×30	8	0.000122	92.68
40×40	2	0.007812	92.68
50×50	2	0.078125	92.68
60×60	32	0.000003	95.12

TABLE 6.2: C and γ optimisation results and accuracy for various hand contour resolutions.

Table 6.2 illustrates that at a resolution of 60×60 , the system predicts hand shapes correctly at a high accuracy of 95.12%, whereas the remaining resolutions all predict at a lower accuracy of 92.68%. Therefore, a resolution of 60×60 is selected for use. This serves as an optimisation of Li's hand shape recognition system. The final hand shape recognition SVM was trained on the images of the hand shape training set at a resolution of 60×60 using the optimal C and γ parameters provided in Table 6.2. The accuracy of the resulting model was tested on unseen data, described in the next subsection.

6.3.3 Hand Shape Recognition Experimental Procedure

A framework such as depicted in Figure 5.2 would be used here in order to determine the hand shape recognition accuracy of the proposed approach, 10 subjects were instructed to perform the 10 hand shapes. These subjects were different to those used in the hand shape training set. For each hand shape performed by each subject, five random frames of the hand shape were obtained. This resulted in a total of 500 frames: five frames of 10 hand shapes performed by 10 subjects, 50 frames per hand shape.

Each image in the testing set was pre-processed as explained before and a contour at a resolution of 60×60 pixels was obtained. The resulting images were used as input to the final hand shape recognition SVM. The prediction of the SVM was marked as a

correct output if the label matched that of the input hand shape. If not, it was marked as an incorrect output.

6.3.4 Results and Analysis

Table 6.3 illustrates the overall accuracy results of the hand recognition component. The component achieves a high average accuracy of 91% across all hand shapes and subjects. The lowest accuracy obtained was 90% for hand shape 10 and the highest accuracy achieved was 94% for hand shape 3. It is clear that the system is highly accurate and consistent across the 10 hand shapes.

Hand Shape	Correct Output	Accuracy(%)
1	45	90
2	45	90
3	47	94
4	45	90
5	45	90
6	46	92
7	46	92
8	45	90
9	47	94
10	45	90
Total	456	91

TABLE 6.3: Hand shape recognition accuracy results.

Table 6.4 summarises the accuracy per subject per hand shape as a percentage of the 5 input images in each case i.e. 5 images for hand shape 1 by Subject 1, 5 for hand shape 2 by Subject 2 etc. It can be observed from the table that in all cases, the recognition rate was either 100% (5 out of 5 correct) or 80% (4 out of 5 correct). No case registers a recognition rate lower than 80%. Focusing on the average accuracy per subject (the last row of the table), it is observed that Subjects 3, 5, and 10 achieving the highest average accuracy of 94%. It is extremely encouraging to note that the lowest accuracy was 88% which is by no means low. This was achieved by Subjects 4 and 9.

Considering that the subjects were are of diverse skin tone, hand dimensions, body dimensions and gender, these results once again clearly demonstrate that the system is highly consistent and robust to variations in subjects. This is a very encouraging result and demonstrates a successful, accurate and robust hand shape recognition component. As such, the second objective set out in Chapter 1 has been successfully achieved.

Hand Shape	Subject									
	1	2	3	4	5	6	7	8	9	10
1	80	100	100	80	100	80	80	100	80	100
2	100	80	100	80	80	100	100	80	100	80
3	100	100	80	100	100	80	100	100	80	100
4	100	80	100	80	80	100	100	80	100	80
5	80	100	100	80	100	80	80	100	80	100
6	100	80	80	100	100	100	100	80	80	100
7	100	80	100	100	80	100	100	80	100	80
8	80	100	100	80	100	80	80	100	80	100
9	100	80	80	100	100	100	100	80	100	100
10	80	100	100	80	100	80	80	100	80	100
Average	92	90	94	88	94	90	92	90	88	94

TABLE 6.4: Hand shape recognition accuracy results per subject.

6.4 Testing of the Proposed SASL Gesture Recognition System

This section describes the experiment carried out in order to answer the two research questions posed in Chapter 1. The first research question is: “How should a combination of hand shape and hand motion be used to recognise more SASL phrases with a higher accuracy than hand motion alone?” This question will be answered by training two HMMs, one using the hand motion as a feature vector and the other using both hand motion and hand shape as a feature vector, and comparing the gesture recognition accuracy of the two approaches. This experiment is henceforth called the “Feature Vector Comparison” experiment. Once the optimal gesture descriptor is determined in this experiment, the next experiment uses this gesture descriptor to answer the second research question.

The second research question is: “How do HMMs and HM-SVMs compare in terms of accuracy when applied to SASL gesture recognition using hand shape and motion as gesture descriptors?” To answer this question, the HMM which performed at a higher accuracy in the previous experiment will be compared to an HM-SVM trained on the same feature vector and the accuracy of the two machine learning approaches will be compared. This experiment is henceforth called the “HMM versus HM-SVM Comparison” experiment.

The method of training the HMMs and HM-SVM was described in the previous chapter. The data set used to train the HMMs and HM-SVM and subsequently carry out this experimentation is described in Section 6.4.1. Sections 6.4.2 and 6.4.3 then explain the experimental procedure, results and analysis of results of each experiment carried out.

6.4.1 Data Set

Currently, there is no standard SASL gesture database available for use. It was initially desired that the data sets of Naidoo [49] or Rajah [58] be used in experimentation but these sets were not available. As such, a direct comparison is not possible. However, the vocabulary size and accuracy will be compared in the conclusion of the chapter. A SASL data set was manually collected.

The Fulton School for the Deaf SASL Dictionary [32] was used to select a larger number of signs to Naidoo and Rajah. As explained in Chapter 1, Naidoo and Rajah’s systems recognised 20 and 23 signs respectively. Thirty five signs were selected as the vocabulary to be tested by the proposed system. This constitutes a 52% increase in vocabulary size compared to Rajah’s system and a 75% increase compared to Naidoo’s system. These signs were chosen such that a wide variety of motions and locations in the signing space were represented. Table 6.5 summarises the signs selected.

A data set was created by collecting videos of 15 subjects performing the 35 gestures. The subjects were chosen such that a wide variety of skin tones and body dimensions were well represented. Figure 6.5 depicts the subjects used in this experiment. It can be seen that the subjects are diverse in skin tone. All but four of these subjects were different to those used in all previous experiments. Each subject was instructed to perform the 35 gestures thrice, resulting in a total of 1575 videos. This data set will henceforth be referred to as the “gesture data set”.



FIGURE 6.5: The 15 subjects for gesture recognition (Top row, left to right: Subjects 1 – 5) (Middle row, left to right: Subjects 6 – 10) (Bottom row, left to right: Subjects 11 – 15).

Number	Gesture	Description
1	Stand	Left hand flat + right hand two fingers
2	Lie down	Both hands flat + right on left
3	Hop	Left hand flat + right hand one finger
4	Puppy	Both hands bent
5	Umbrella	Both hands fist + right on left
6	Breast	Both hands palms facing chest
7	House	Both hands bent facing each other
8	Table lamp	Left hand 1 finger + right hand flat + right on left
9	Gate	Both hands flat opposite in front of chest
10	Bye	Right hand palm open and waving
11	Colour	Both hands open move in circle
12	Window	Both hands 1 finger + display window
13	Rain	Both hands open + wiggle fingers
14	Christmas tree	Both hands open above head move to the side
15	Tea	Left hand 3 fingers + right hand opposite side
16	That	Right hand 1 finger pointing to the right
17	Bright	Both hands open + move above head + move to side
18	Microwave	Right hand 1 finger moving in circle
19	Kitchen	Both hands 1 finger up and down
20	Take off clothes	Both hands fist moving up above chest
21	Up	Right hand 1 finger moving up
22	Newspaper	Both hands fist + moving way from chest
23	Strong	Both hands fist + strong pose
24	Tractor	Both hands fist + mimic steering wheel
25	Hat	Both hands fist moving against head
26	Motorbike	Both hands + mimic motorbike
27	Finish	Both hands opposite + reverse hands moving away
28	Dolphin	Right backhand wave
29	How many	Both hands fist moving away to opposite hand open
30	Cut	Right hand 2 fingers + scissor
31	Cow	Both hands + pink thumb + against head
32	Crab	Both hands 2 fingers + open and close
33	Tadpole	Right hand fist + 1 finger right hand
34	Bovril	Right hand open + right hand 2 fingers
35	Hair	Hold 3 fingers close to hair

TABLE 6.5: Description of the selected gestures.

6.4.2 Feature Vector Comparison Experiment

This experiment aimed to determine whether a combination of hand shape and hand motion can be used to achieve a higher accuracy than using hand motion only using a large data set. The following subsections describe the experimental procedure, results and analysis of results for this experiment in order to answer this question.

6.4.2.1 Feature Vector Comparison Experimental Procedure

Two HMMs were trained, one with the feature vector only including hand motion information and the other with the feature vector including both hand motion and hand shape. The training data consisted of the videos of 7 of the 15 subjects – Subjects 9 to 15 – from the gesture data set. This resulted in 21 videos per gesture and a total of 735 videos, used for training each HMM. Thereafter, a testing data set was constructed which comprised of the data of the remaining 8 subjects – Subjects 1 to 8 – resulting in a total 840 videos, 24 videos per gesture, for testing each HMM.

In testing, each video in the test set was used as input to each HMM. Each HMM produced an output label corresponding to the predicted gesture. If the predicted label corresponded to the correct gesture, this was recorded as a correct classification, otherwise it was marked as an incorrect classification.

Before an analysis of the results could be carried out, the success rate of using random guessing was considered in place of the classifier under the conditions of this experiment as a base comparison. For each video of each sign, a classification into one of 35 classes is carried out. For each video, the random guessing success rate is then:

$$\frac{1}{35} \times 100 \approx \mathbf{3\%} \quad (6.1)$$

Therefore, for all 840 gesture videos across all signs and subjects, the success rate for each video using a random guess is approximately 3%. Computing the average success rate across all the videos of all signs also results in an average success rate of approximately 3% for the use of a random guess in place of using a classifier. Therefore, achieving an accuracy that is higher than this success rate is considered to be better than random guessing.

6.4.2.2 Results and Analysis

Table 6.6 depicts the average number of correctly classified videos, as a percentage of the 24 videos per gesture, for both HMMs. The number of correctly recognised videos in each case is provided in Table A.1 in Appendix A. Analysing Table 6.6, it is observed that the accuracies for the HMM with motion range from 50.0% to 75.0%. The average overall accuracy for this approach is 55.1%. The accuracies for the HMM with motion and hand shape range from 62.5% to 87.5%, with an overall average accuracy of 70.7%. The standard deviations for the shape and motion HMM and the motion-only HMM are 6.0% and 6.5% respectively.

It is noted, first and foremost, that both approaches achieve acceptable recognition rates of above 50%. It is important to compare this result to the random guessing base case. An accuracy of 50% by the classifiers is about 16 times higher than random guessing – an increase of 47%. Therefore, this result can by no means be considered low but rather a very encouraging accuracy on the part of both classifiers. The standard deviations are small which indicates that the differences in accuracy amongst gestures is small, implying recognition approaches that are consistent across variations in gestures, with no outliers observed. The second point to note is that the HMM that uses both hand motion and hand shape clearly performs much better than the HMM that uses only hand motion. The HMM with two parameters achieves an increase in average accuracy of 15% as compared to the HMM with one parameter. This is a very large increase in accuracy.

Figure 6.6 graphically depicts the accuracies in Table 6.6 for the two methods. Analysing the graph reveals that the increase registered by including the hand shape in the feature vector is clearly beneficial to every gesture, as opposed to some gestures and not others. Every gesture of the HMM that used hand shape and hand motion registers a higher accuracy than their motion-only counterparts. Also, it may be mentioned that, while the standard deviations of the two approaches are very similar, the HMM with hand motion and hand shape has a slightly lower value indicating slightly more consistent recognition across variations in gestures.

It is already apparent that the use of both hand shape and hand motion helps sustain a high accuracy on a large vocabulary, whereas the accuracy clearly suffers on the large vocabulary if only hand motion is used. In order to confirm this, a statistical test was used to determine whether the difference in accuracy between the HMM with motion and the HMM with motion and hand shape is significant for each gesture. To test for significant difference, a non-parametric Signed Rank Test was used since the accuracy values are bounded by 100. The Signed Rank Test showed that there is a significant

Gesture	HMM Motion (%)	HMM Motion and Hand Shape (%)
1	75.00	83.33
2	62.50	75.00
3	54.16	66.66
4	54.16	66.66
5	58.33	70.83
6	50.00	70.83
7	58.33	75.00
8	50.00	62.50
9	54.16	75.00
10	50.00	66.66
11	62.50	75.00
12	54.16	70.83
13	54.16	70.83
14	50.00	62.50
15	62.50	70.83
16	50.00	70.83
17	75.00	87.50
18	50.00	66.66
19	50.00	70.83
20	58.33	83.33
21	54.16	70.83
22	50.00	66.66
23	50.00	62.50
24	54.16	66.66
25	50.00	70.83
26	54.16	66.66
27	50.00	62.50
28	50.00	70.83
29	54.16	70.83
30	54.16	75.00
31	50.00	62.45
32	54.16	70.83
33	50.00	66.66
34	58.33	70.83
35	62.50	79.16
Average	55.11	70.70

TABLE 6.6: Average accuracy of the HMM with motion and the HMM with motion and hand shape per gesture.

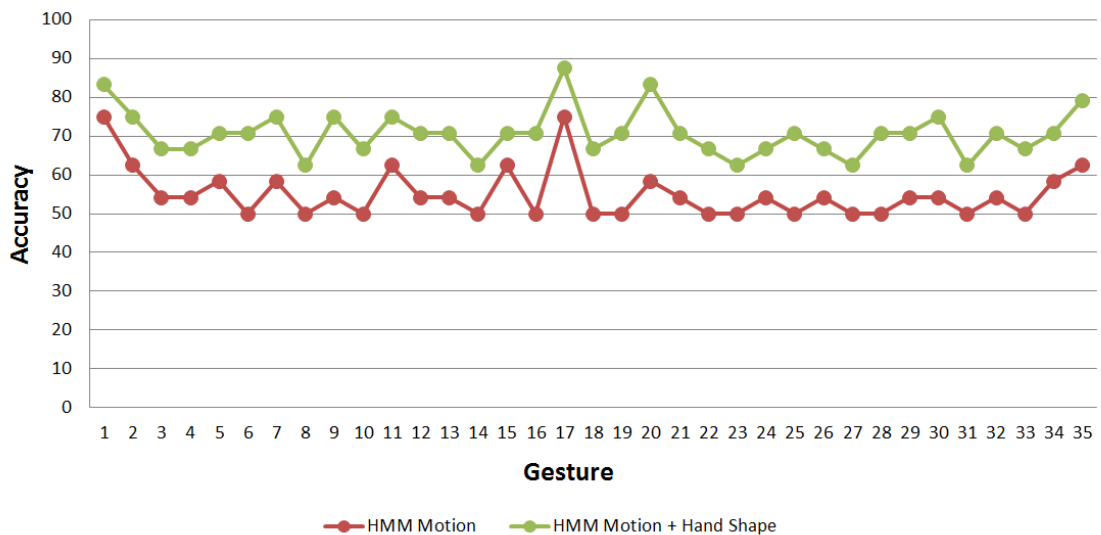


FIGURE 6.6: Graph of gesture recognition accuracy of HMM with only motion versus HMM with motion and hand shape.

increase in gesture recognition using two parameters for every gesture, with the p -value being $p < 0.0001$ for all 35 gestures.

Therefore, it can be firmly concluded that, in response to the first research question, the use of a combination of hand motion and hand shape is highly effective in recognising a larger set of SASL gestures, as compared to using only hand motion. For comparison, it should be considered that Naidoo's approach achieved an average accuracy of 69% for a vocabulary of 20 SASL gestures and Rajah's approach achieved an average accuracy of 68% for a set of 23 SASL gestures. The proposed system recognises a much larger set of SASL gestures than both of these systems – 75% and 52% more signs – and is able to do so at a higher accuracy than both systems – an average of 70% for 35 SASL gestures – in spite of the increase in vocabulary size. This is indicative of a highly successful hand tracking and feature extraction process. It is also concluded at this stage that objectives 3, 4 and 5 mentioned in Chapter 1 have been met.

As an additional note, an analysis of the average accuracy per test subject is provided and analysed. Table 6.7 illustrates the average accuracy results per subject for the two methods. The accuracy per subject of the HMM with only hand motion ranged from 53.3% to 57.1%, with an average accuracy of 55.1%. It is clear that the range in accuracies across subjects is very small. The accuracy per subject of the HMM with hand motion and hand shape ranged from 68.5% to 73.3%, with an average accuracy of 70.7%. In this case as well, the range in accuracy across subjects is very small. There are no outliers in either case. In all cases, the average accuracy of the HMM with both hand motion and hand shape is greater than that of the HMM with hand motion only. Considering that there a wide diversity in the skin tone and body dimensions of subjects,

this result is very encouraging and indicates a feature extraction process that is robust to variations in test subjects.

Subject	HMM Motion (%)	HMM Motion and Hand Shape (%)
1	54.28	70.47
2	57.14	73.32
3	57.14	73.32
4	54.28	69.52
5	53.33	70.47
6	55.23	68.57
7	54.28	69.52
8	55.23	70.47
Average	55.11	70.70

TABLE 6.7: Average accuracy of the HMM with motion and the HMM with motion and hand shape per subject.

6.4.3 HMM versus HM-SVM Comparison Experiment

This experiment aims to answer the second research question: “How do HMMs and HM-SVMs compare in terms of accuracy when applied to SASL gesture recognition using hand shape and motion as gesture descriptors?” The following subsections describe the experimental procedure, results and analysis of the results of this experiment in order to answer this question.

6.4.3.1 HMM versus HM-SVM Comparison Experimental Procedure

This experiment made use of the HMM trained in the previous section on the feature vector that includes hand shape and hand motion. Henceforth, this HMM is referred to simply as “the HMM”. A HM-SVM was trained on the same data and feature vector as this HMM and a comparison in accuracy of the two approaches was carried out.

Thus, the HM-SVM was trained on the videos of 7 of the 15 subjects used to train the HMM and tested on the videos of the remaining 8 subjects. Each video was used as input to the HM-SVM and the output label corresponding to the predicted gesture was compared to the actual label of the video. If the predicted label corresponded to the correct gesture, this was recorded as a correct classification, otherwise it was marked as an incorrect classification.

6.4.3.2 Results and Analysis

Table 6.8 illustrates the average number of correctly classified videos, as a percentage of the 24 videos per gesture, for the HMM and the HM-SVM. The number of correctly recognised videos in each case is provided in Table A.2 in Appendix A.

Analysing Table 6.8, it is observed that the accuracy of the HM-SVM ranges from 66.6% to 91.6% across gestures, with an overall average accuracy of 75.0%. The standard deviation for the HM-SVM is 5.3%. The accuracies for the HMM range from 62.5% to 87.5%, with an overall average accuracy of 70.7%. The standard deviation for the HMM is 6.0%.

It is noted that both approaches achieve very accurate results. The standard deviations in accuracy for both approaches are small which indicates, once again, that both approaches are consistent across variations in gestures, with no outliers observed. This is very encouraging given that a larger vocabulary has been used. It is also noted that the HM-SVM registers an increase in average accuracy of 4.5% compared to the HMM. This is in line with the results of other researchers presented in Chapter 2.

Figure 6.7 graphically depicts the accuracies in Table 6.8 for the two methods. Analysing the graph reveals that, with exception of only one gesture, the increase registered by using the HM-SVM was also clearly beneficial to every gesture, as opposed to some gestures and not others. The exception is for gesture 6 – the word “Breast” – which is highlighted in bold font in the table. In this case, the HM-SVM achieved the same accuracy as the HMM. In every other case, using the HM-SVM helps achieve an improved classification result. Once again, while the standard deviations of the two approaches are very similar, the HM-SVM has a slightly lower value indicating slightly more consistent recognition across variations in gestures.

In order to determine whether the increase in accuracy per gesture of the HM-SVM over the HMM was significant, the non-parametric Signed Rank Test was applied to the results per gesture. The results indicate that the p -value obtained is $p < 0.0001$ for every gesture, except gesture 6 which had a p -value of 1. Therefore, a significant increase is achieved for every gesture when using the HM-SVM over the HMM, except for gesture 6 for which the accuracy is the same in both cases. It may be concluded that the HM-SVM generally performs better than the HMM, and in a very small number of cases, it is at least as good as the HMM.

Therefore, it is firmly concluded that, in response to the second research question, HM-SVMs are a more suitable and accurate classification technique than HMMs in the context of SASL gesture recognition using hand shape and hand motion as gesture

Gesture	HMM (%)	HM-SVM (%)
1	83.33	87.50
2	75.00	79.16
3	66.67	75.00
4	66.67	70.83
5	70.83	75.00
6	70.83	70.83
7	75.00	79.16
8	62.50	66.66
9	75.00	79.16
10	66.67	75.00
11	75.00	79.16
12	70.83	75.00
13	70.83	75.00
14	62.50	66.66
15	70.83	75.00
16	66.67	70.83
17	87.50	91.67
18	66.67	70.83
19	70.83	75.00
20	75.00	79.16
21	70.83	75.00
22	66.67	70.83
23	62.50	70.83
24	66.67	75.00
25	70.83	75.00
26	66.67	70.83
27	62.50	66.66
28	70.83	75.00
29	70.83	75.00
30	75.00	79.16
31	62.50	70.83
32	70.83	75.00
33	66.67	70.83
34	70.83	75.00
35	79.17	83.33
Average	70.70	75.00

TABLE 6.8: Average accuracy between the HMM with motion and hand shape and the HM-SVM per gesture.

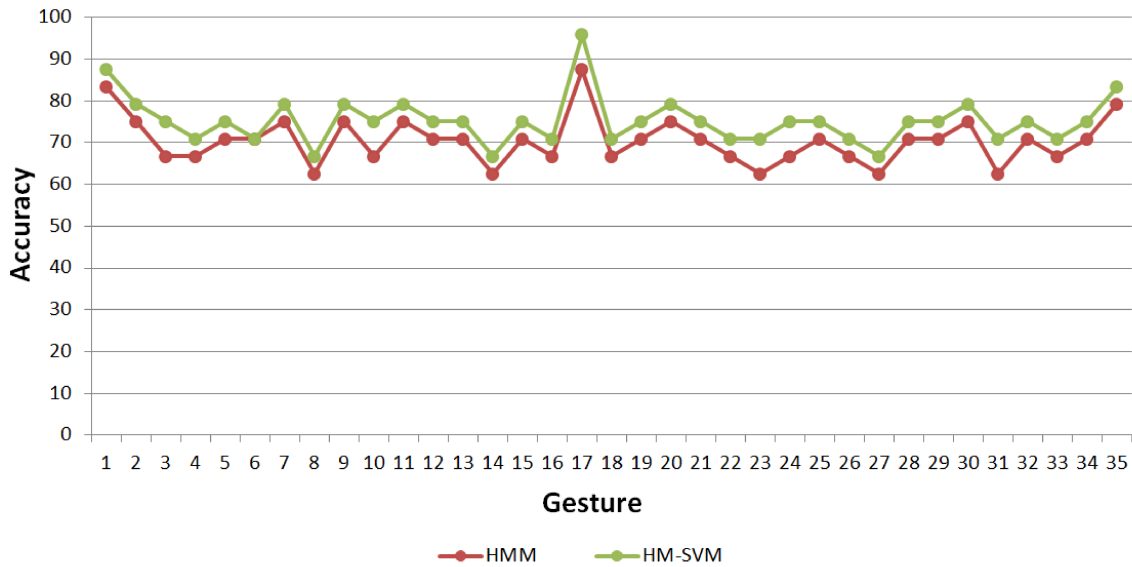


FIGURE 6.7: Graph of gesture recognition accuracy of HMM vs HM-SVM.

descriptors. It is also concluded that the final two objectives 6 and 7 mentioned in Chapter 1 have been met.

Once again, as an additional note, an analysis of the average accuracy per test subject is provided and analysed. Table 6.9 illustrates the average accuracy results per subject for the two methods. The accuracy per subject of the HMM ranged from 68.5% to 73.3%, with an average accuracy of 70.7%. It is clearly observed that the range in accuracies across subjects is very small. The accuracy per subject of the HM-SVM ranged from 72.3% to 78.1%, with an average accuracy of 75.0%. For this method as well, the range in accuracy across subjects is very small. No outliers are observed in either case. In all cases, the average accuracy of the HM-SVM is greater than that of the HMM. Again, taking the wide diversity in test subjects into account, this is a very encouraging result and indicates a highly successful feature extraction process that is robust to variations in test subjects.

Subject	HMM (%)	HM-SVM (%)
1	70.47	78.10
2	73.32	77.14
3	73.32	77.14
4	69.52	72.38
5	70.47	75.23
6	68.57	75.23
7	69.52	72.38
8	70.47	72.38
Average	70.70	75.00

TABLE 6.9: Overall average accuracy per subject.

6.5 Conclusion

This chapter first described the experiments carried out in order to assess the accuracy of the hand tracking and hand shape recognition components of the proposed gesture recognition system. Thereafter the two experiments carried out to answer the two research questions posed in Chapter 1 were described.

The findings demonstrated that the hand tracking component of the system achieved a near-perfect tracking accuracy of 99.0%. The experiment was carried out on 10 subjects with varied skin tones and body dimensions and it was thus shown that the strategy is virtually invariant to variations in test subjects.

A contribution was made to the field of hand shape recognition, although this was not the focus of this research. Four different hand contour resolution sizes were compared to determine the size that achieves the highest cross-validation accuracy using the SVM. It was found that a resolution of 60×60 was the optimal resolution, achieving an accuracy of 95.12%. This is a higher accuracy as compared to the resolution of 20×30 previously proposed by Li, which achieved an accuracy of 92.68%. Based on this finding, the SVM was trained. An experiment was then carried out in order to assess the accuracy of the hand shape recognition model on unseen data from 10 new subjects. It was found that the system can recognise hand shapes at a high accuracy of 91% and is robust to variations in test subjects.

Thereafter, two experiments were carried out to answer the two research questions posed in Chapter 1. The first experiment compared the SASL gesture recognition accuracy of two HMMs, one trained on hand motion only, and the other trained on a combination of hand motion and hand shape. It was demonstrated that using a combination of hand motion and hand shape results in a higher accuracy for every SASL gesture tested, with an average increase of 15%. The response to the first research question was stated as “the use of a combination of hand motion and hand shape is highly effective in recognising a larger set of SASL gestures, as compared to using only hand motion”.

The second experiment compared the SASL gesture accuracy of an HMM and HM-SVM, both using a combination of hand motion and hand shape as gesture descriptors. It was demonstrated that the HM-SVM generally performs better than the HMM, and in a very small number of cases, it is at least as good as the HMM. On average, the HM-SVM achieved a 5% increase over the HMM. Therefore the HM-SVM is preferred to the HMM. The response to the second question was therefore stated as “HM-SVMs are a more suitable and accurate classification technique than HMMs in the context of SASL gesture recognition using hand shape and hand motion as gesture descriptors”.

Finally, in both cases, it was demonstrated that the system is robust to variations in test subjects.



Chapter 7

Conclusion

This research has made several significant contributions to the field of South African Sign Language (SASL) gesture recognition.

The most important of these contributions is a fully automatic gesture recognition framework that is able to recognise a larger gesture vocabulary than previous systems such as those of Naidoo and Rajah. An improved feature vector containing two sign language parameters and an accurate extraction process were proposed to achieve a high accuracy. The previous systems only used one parameter as a feature vector. The gesture recognition accuracy obtained was higher than that of previous systems, in spite of the larger vocabulary.

It is crucial to note that the proposed gesture recognition system does not require any specialised equipment such as DataGloves, 3D cameras or coloured markers. It was shown that the majority of research in the field of sign language gesture recognition either make use of specialised equipment to achieve a high accuracy and large vocabulary, or use a simple web camera configuration but achieve a low accuracy with a small vocabulary. The proposed system does not make use of any specialised equipment but recognises 35 SASL gestures at a high accuracy.

Another important and novel contribution was the use of HM-SVMs to recognise SASL gestures. This technique has not been implemented in the context of SASL gesture recognition before and results indicate that there is an improvement over the conventional HMMs. The research clearly demonstrated that HM-SVMs are better than HMMs in terms of classification accuracy in most cases, but always at least as good as HMMs. This indicates that HM-SVMs are a better choice. It is hoped that this finding benefits other researchers in the field.

An additional contribution was made to the field of hand shape recognition for SASL. The hand shape recognition component was optimised by using a larger hand contour resolution and achieving a higher accuracy than that of Li. It was shown that a hand contour resolution of 60×60 pixels achieved a higher accuracy than the resolution of 20×30 pixels originally proposed by Li. This is also hoped to benefit research in the field of hand shape recognition.

7.1 Directions for Future Work

The following subsections provide directions for future work.

7.1.1 Optimising HM-SVMs

Conducting further investigation into the work on HM-SVMs could allow for more accurate results. HM-SVMs use SVMs and it is possible to achieve a higher accuracy by optimising each SVM. Possible optimisations include investigating the use of various kernels and optimising their corresponding parameters.

7.1.2 Parallel HMMs

Making use of multiple parallel HMMs could increase the accuracy of the framework and the processing speed per gesture. Each hand could have its own HMM and set of rules. As explained in Chapter 2, some researchers categorize the two hands, one as being a “dominant hand” and the other as a “non-dominant hand”. Using separate HMMs for each hand could therefore affect the accuracy result.

7.1.3 Including Additional Sign Language Parameters

This research has shown that two sign language parameters can facilitate the recognition of a larger vocabulary at a high accuracy than using one parameter. In future, an investigation into including further sign language parameters such as facial expressions and hand orientation can be carried out. The feature extraction process could be extended to extract these features as well.

7.2 Concluding Remarks

This research has served as a very remarkable experience to the researcher. It is hoped that this research will benefit others in the field towards the eventual realisation of a full automatic South African Sign Language machine translation system.



Appendix A

Additional Test Results



Gesture	HMM Motion (24)	HMM Motion and Hand Shape (24)
1	18	20
2	15	18
3	13	16
4	13	16
5	14	17
6	12	17
7	14	18
8	12	15
9	13	18
10	12	16
11	15	18
12	13	17
13	13	17
14	12	15
15	15	17
16	12	17
17	18	21
18	12	16
19	12	17
20	14	20
21	13	17
22	12	16
23	12	15
24	13	16
25	12	17
26	13	16
27	12	15
28	12	17
29	13	17
30	13	18
31	12	15
32	13	17
33	12	16
34	14	17
35	15	19
Average	13	17

TABLE A.1: Number of correctly recognised videos by the HMM using only motion and the HMM using motion and hand shape, per gesture.

Gesture	HMM (24)	HM-SVM (24)
1	20	21
2	18	19
3	16	18
4	16	17
5	17	18
6	17	17
7	18	19
8	15	16
9	18	19
10	16	18
11	18	19
12	17	18
13	17	18
14	15	16
15	17	18
16	17	17
17	21	22
18	16	17
19	17	18
20	20	19
21	17	18
22	16	17
23	15	17
24	16	18
25	17	18
26	16	17
27	15	16
28	17	18
29	17	18
30	18	19
31	15	17
32	17	18
33	16	17
34	17	18
35	19	20
Average	17	18

TABLE A.2: Number of correctly recognised videos by the HMM and HM-SVM, both using hand motion and hand shape, per gesture.

Bibliography

- [1] I. Achmed, “Upper body pose recognition and estimation towards the translation of South African Sign Language,” Master’s thesis, University of the Western Cape, Computer Science, 2010.
- [2] U. Ahlvers, R. Rajagopalan, and U. Zölzer, “Model-free face detection and head tracking with morphological hole mapping,” in *Proceedings of the European Signal Processing Conference*, 2005.
- [3] Y. Altun, I. Tsochantaridis, and T. Hofmann, “Hidden markov support vector machines,” Brown University, USA, 2003.
- [4] L. Alvarez, P.-L. Lions, and J.-M. Morel, “Image selective smoothing and edge detection by nonlinear diffusion. ii,” *SIAM J. Numer. Anal.*, vol. 29, no. 3, pp. 845–866, Jun. 1992.
- [5] A. Atsalakis, N. Papamarkos, and I. Andreadis, “On estimation of the number of image principal colors and color reduction through self-organized neural networks.”
- [6] Y. Bengio, “Markovian models for sequential data,” Universit de Montral, 1996.
- [7] G. Bradski and A. Kaehler, *Learning OpenCV*. O’Reilly Media, 2008.
- [8] J. Brand and J. Mason, “A comparative assessment of three approaches to pixel level human skin-detection,” in *Proceedings of the 15th International Conference on Pattern Recognition*, 2000, pp. 1056–1059.
- [9] D. Brown, “Faster upper body pose recognition and estimation using compute unified device architecture,” Master’s thesis, University of the Western Cape, Computer Science, 2013.
- [10] R. Brunelli, *Template Matching Techniques in Computer Vision: Theory and Practice*. Wiley; 1 Edition, 2009.
- [11] M. Castrill, O. Dniz, D. Hernandez, and J. Lorenzo, “A comparison of face and facial feature detectors based on the viola and jones general object detection framework,” *Mach. Vision Appl.*, vol. 22, no. 3, pp. 481–494, May 2011.

- [12] D. Chai and K. Ngan, "Face segmentation using skin-color map in videophone applications," *IEEE Trans. Cir. and Sys. for Video Technol.*, vol. 9, no. 4, pp. 551–564, 1999.
- [13] C. Chang and C. Lin, "Libsvm: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 27:1–27:27, May 2011.
- [14] S. Cheung and C. Kamath, "Robust techniques for background subtraction in urban traffic video," Center for Applied Scientific Computing Lawrence Livermore National Laboratory, 2007.
- [15] R. Clark, G. Schweikert, C. Toomajian, S. Ossowski, G. Zeller, P. Shinn, N. Warthmann, T. Hu, G. Fu, D. Hinds, H. Chen, K. Frazer, D. Huson, B. Schlkopf, M. Nordborg, G. Rtsch, J. Ecker, and D. Weigel, "Common sequence polymorphisms shaping genetic diversity in arabidopsis thaliana," *Science*, vol. 317, no. 5836, pp. 338–342, 2007.
- [16] M. Collins, "Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms," in *Proceedings of the ACL-02 conference on Empirical methods in natural language processing - Volume 10*, ser. EMNLP '02. Stroudsburg, PA, USA: Association for Computational Linguistics, 2002, pp. 1–8.
- [17] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, "Online passive-aggressive algorithms," *J. Mach. Learn. Res.*, vol. 7, pp. 551–585, Dec. 2006.
- [18] N. Cristianini, "Support vector and kernel machines," *Tutorial at ICML*, 2001.
- [19] L. da Vinci and J. Hawkins, *A Treatise on Painting*. J. Taylor, 1802.
- [20] F. Dadgostar and A. Sarrafzadeh, "A fast real-time skin detector for video sequences," in *Image Analysis and Recognition*, 2005, pp. 804–811.
- [21] F. Dadgostar and A. Sarrafzadeh, "An adaptive real-time skin detector based on hue thresholding: A comparison on two motion tracking methods," *Pattern Recognition Letters*, vol. 27, no. 12, pp. 1342–1352, 2006.
- [22] K. Duan, S. S. Keerthi, W. Chu, S. K. Shevade, and A. N. Poo, "Multi-category classification by soft-max combination of binary classifiers," in *Proceedings of the 4th international conference on Multiple classifier systems*, ser. MCS'03. Berlin, Heidelberg: Springer-Verlag, 2003, pp. 125–134.
- [23] T. Finley and T. Joachims, "Training structural svms when exact inference is intractable," in *Proceedings of the 25th international conference on Machine learning*, ser. ICML '08. New York, NY, USA: ACM, 2008, pp. 304–311.

- [24] M. Fleck, D. Forsyth, and C. Bregler, "Finding naked people," in *Proceedings of the 4th European Conference on Computer Vision*, 1996, pp. 593–602.
- [25] G. Forney, "The viterbi algorithm," *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, 1973.
- [26] H. Freeman and R. Shapira, "Determining the minimum-area encasing rectangle for an arbitrary closed curve," *Commun. ACM*, vol. 18, no. 7, pp. 409–413, Jul. 1975.
- [27] M. Gales and S. Young, "The application of hidden markov models in speech recognition," *Found. Trends Signal Process.*, vol. 1, no. 3, pp. 195–304, Jan. 2007.
- [28] M. Ghaziasgar, "The use of mobile phones as service-delivery devices in a sign language machine translation system," Master's thesis, University of the Western Cape, Computer Science, 2010.
- [29] T. Hofmann, I. Tsochantaridis, and Y. Altun, "Learning over structured output spaces via joint kernel functions," in *Proceedings of the Sixth Kernel Workshop*, 2002.
- [30] T. Hofmann, B. Schölkopf, and A. J. Smola, "Kernel methods in machine learning," *The Annals of Statistics*, vol. 36, no. 3, pp. 1171–1220, 2008.
- [31] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*. Prentice Hall; 3 Edition, 2006.
- [32] S. Howard, *Finger Talk: South African Sign Language dictionary*. South Africa: Mondri, 2008.
- [33] C. Hsu, C. Chang, and C. Lin, "A practical guide to support vector classification," National Taiwan University, Tech. Rep., 2003.
- [34] C. Hsu and C. Lin, "A comparison of methods for multiclass support vector machines," *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 415–425, 2002.
- [35] R. Hsu, M. Abdel-Mottaleb, and A. Jain, "Face detection in color images," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002, pp. 696–706.
- [36] X. Huang, Y. Ariki, and M. Jack, *Hidden Markov Models for Speech Recognition*. New York, NY, USA: Columbia University Press, 1990.
- [37] R. Hughey and A. Krogh, "Hidden markov models for sequence analysis: extension and analysis of the basic method," 1996.

- [38] D. Ilea and P. Whelan, "Adaptive pre-filtering techniques for colour image analysis," in *Machine Vision and Image Processing Conference, 2007. IMVIP 2007. International*, 2007, pp. 150–157.
- [39] T. Joachims, T. Finley, and C. Yu, "Cutting-plane training of structural svms," *Mach. Learn.*, vol. 77, no. 1, pp. 27–59, Oct. 2009.
- [40] P. Kakumanu, S. Makrogiannis, and N. Bourbakis, "A survey of skin-color modeling and detection methods," *Pattern Recognition*, vol. 40, no. 3, pp. 1106–1122, Aug. 2007.
- [41] S. S. Keerthi and C. Lin, "Asymptotic behaviors of support vector machines with gaussian kernel."
- [42] W. Kelly, A. Donnellan, and D. Molloy, "Screening for objectionable images: a review of skin detection techniques," in *Proceedings of the 2008 International Machine Vision and Image Processing Conference*, 2008, pp. 151–158.
- [43] W. Kong and S. Ranganath, "Sign language phoneme transcription with rule-based hand trajectory segmentation," *J. Signal Process. Syst.*, vol. 59, no. 2, pp. 211–222, May 2010.
- [44] P. Li, "Hand shape estimation for south african sign language," Master's thesis, University of the Western Cape, Computer Science, 2012.
- [45] H. Lin and C. Lin, "A study on sigmoid kernels for svm and the training of non-psd kernels by smo-type methods," Tech. Rep., 2003.
- [46] A. Maruch, "Talking with the hearing-impaired," January 2010, [Online] Available at <http://www.deafsa.co.za/index-2.html>.
- [47] T. Matsuo, Y. Shirai, and N. Shimada, "Automatic generation of hmm topology for sign language recognition," in *Pattern Recognition, 2008. ICPR 2008. 19th International Conference*, 2008, pp. 1–4.
- [48] D. Mohr and G. Zachmann, "Fast: Fast adaptive silhouette area based template matching," in *Proceedings of the British Machine Vision Conference*. BMVA Press, 2010, pp. 39.1–39.12.
- [49] N. Naidoo, "South African Sign Language recognition using feature vectors and hidden markov models," Master's thesis, University of the Western Cape, Computer Science, 2009.
- [50] A. Niissalo, "Hidden markov support vector machines," 2008, [Online] Available at <http://www.cs.helsinki.fi/group/smart/teaching/58308109/niissaloPrint.pdf>.

- [51] W. Noble, "What is a support vector machine?" *Nature Biotechnology*, vol. 24, no. 12, pp. 1565–1567, 2006.
- [52] M. Pantic, M. Valstar, R. Rademaker, and L. Maat, "Web-based database for facial expression analysis," in *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference*, 2005.
- [53] P. Peer and F. Solina, "An automatic human face detection method," in *Proceedings of the Computer Vision Winter Workshop*, 1999, pp. 122–130.
- [54] J. C. Platt, N. Cristianini, and J. Shawe-taylor, "Large margin dags for multiclass classification," in *Advances in Neural Information Processing Systems 12*, 2000, pp. 547–553.
- [55] R. J. Povinelli and X. Feng, "Temporal pattern identification of time," in *Artificial Neural Networks in Engineering*, 1998, pp. 691–696.
- [56] R. J. Povinelli and X. Feng, "A new temporal pattern identification method for characterization and prediction of complex time series events," *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 2, pp. 339–352, 2003.
- [57] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," in *Proceedings of the IEEE*, 1989, pp. 257–286.
- [58] 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, Computer Science, 2006.
- [59] L. Ren, G. Shakhnarovich, J. Hodgins, H. Pfister, and P. Viola, "Learning silhouette features for control of human motion," in *ACM SIGGRAPH 2004 Sketches*, 2004, p. 129.
- [60] G. Rigoll, A. Kosmala, and S. Eickeler, "High performance real-time gesture recognition using hidden markov models," in *In Proc. Gesture Workshop*. Springer, 1998, pp. 69–80.
- [61] J. Rousu, C. Saunders, S. Szedmak, and J. Shawe-Taylor, "Kernel-based learning of hierarchical multilabel classification models," *J. Mach. Learn. Res.*, vol. 7, pp. 1601–1626, Dec. 2006.
- [62] J. Rus, "Rgb color solid cube," [Online] Available at http://commons.wikimedia.org/wiki/File%3ARGB.color_solid_cube.png.
- [63] I. Sandjaja and N. Marcos, "Sign language number recognition," in *Proceedings of the 2009 Fifth International Joint Conference on INC, IMS and IDC*, ser. NCM '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 1503–1508.

- [64] H. Schneiderman and T. Kanade, "A statistical method for 3d object detection applied to faces and cars," in *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, vol. 1, 2000, pp. 746–751 vol.1.
- [65] O. Schreer and S. Ngongang, "Real-time gesture recognition in advanced video-communication services," in *Image Analysis and Processing, 2007. ICIAP 2007. 14th International Conference on*, 2007, p. 253258.
- [66] scikit-learn developers, "Support vector machines," 2013, [Online] Available at <http://scikit-learn.org/stable/modules/svm.html>.
- [67] F. Sha and L. K. Saul, "Large margin hidden markov models for automatic speech recognition," in *Advances in Neural Information Processing Systems 19*. MIT Press, 2007, pp. 1249–1256.
- [68] A. Shamaie and A. Sutherland, "Hand tracking in bimanual movements," *Image Vision Comput.*, vol. 23, no. 13, pp. 1131–1149, Nov. 2005.
- [69] N. Shimada, K. Kimura, and Y. Shirai, "Real-time 3d hand posture estimation based on 2d appearance retrieval using monocular camera," in *Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems*, 2001, pp. 23–30.
- [70] M. Shin, K. Chang, and L. Tsap, "Does colorspace transformation make any difference on skin detection?" in *Proceedings of the 6th IEEE Workshop on Applications of Computer Vision*, 2002, pp. 275–279.
- [71] L. Sigaland and S. Sclaroff, "Estimation and prediction of evolving color distributions for skin segmentation under varying illumination," Boston University, Tech. Rep., 1999.
- [72] P. Simon, *Too Big to Ignore: The Business Case for Big Data*. Wiley; 1 Edition, March 2013.
- [73] W. Skarbek and A. Koschan, "Colour image segmentation: A survey," University of Berlin, Tech. Rep., 1994.
- [74] E. Stergiopoulou and N. Papamarkos, "Hand gesture recognition using a neural network shape fitting technique," *Eng. Appl. Artif. Intell.*, vol. 22, no. 8, pp. 1141–1158, Dec. 2009.
- [75] W. Stokoe., "Sign language structure: An outline of the visual communication systems of the american deaf," in *Deaf Studies and Deaf Education*, 2005, pp. 3–37.
- [76] C. Sutton and A. McCallum, "An introduction to conditional random fields," 2010.

- [77] C. Sutton, A. McCallum, and K. Rohanimanesh, "Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data," *J. Mach. Learn. Res.*, vol. 8, pp. 693–723, May 2007.
- [78] B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin, "Learning structured prediction models: a large margin approach," in *Proceedings of the 22nd international conference on Machine learning*, ser. ICML '05. New York, NY, USA: ACM, 2005, pp. 896–903.
- [79] B. Taskar, C. Guestrin, and D. Koller, "Max-margin markov networks." MIT Press, 2003.
- [80] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*. Academic Press-Elsevier, 2003.
- [81] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun, "Support vector machine learning for interdependent and structured output spaces," in *Proceedings of the twenty-first international conference on Machine learning*, ser. ICML '04. New York, NY, USA: ACM, 2004, p. 104.
- [82] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun, "Large margin methods for structured and interdependent output variables," *J. Mach. Learn. Res.*, vol. 6, pp. 1453–1484, Dec. 2005.
- [83] N. Tsumura, R. Usuba, T. Nakaguchi, M. Shiraishi, N. Ojima, N. Okiyama, K. Takase, K. Hori, S. Okaguchi, and Y. Miyake, "Real-time image-based control of skin melanin texture," in *ACM SIGGRAPH 2005 Sketches*, 2005.
- [84] A. Tzotsos, "A support vector machine approach for object based image analysis," in *Proceedings of the 1st International Conference on Object-based Image Analysis*, 2006, pp. 3099–3104.
- [85] M. F. Valstar and M. Pantic, "Combined support vector machines and hidden markov models for modeling facial action temporal dynamics, humancomputer interaction," 2007.
- [86] V. N. Vapnik, *The nature of statistical learning theory*. New York, NY, USA: Springer-Verlag New York, Inc., 1995.
- [87] V. Vezhnevets, V. Sazonov, and A. Andreeva, "A survey on pixel-based skin color detection techniques," in *Proceedings of the Graphicon*, 2003, pp. 85–92.
- [88] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference*, vol. 1, 2001, pp. 511–518.

- [89] S. V. N. Vishwanathan and A. Smola, “Fast kernels for string and tree matching,” 2004.
- [90] C. Vogler and D. Metaxas, “Parallel hidden markov models for american sign language recognition,” in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference*, vol. 1, 1999, pp. 116–122 vol.1.
- [91] L. Wang, Y. Hang, S. Luo, X. Luo, and X. Jiang, “Deblurring gaussian-blur images: A preprocessing for rail head surface defect detection,” in *Service Operations, Logistics, and Informatics (SOLI), 2011 IEEE International Conference on*, 2011, pp. 451–456.
- [92] J. Whitehill, “Automatic real-time facial expression recognition for signed language translation,” Master’s thesis, University of the Western Cape, Computer Science, 2006.
- [93] H. Y. Sato, M. Saito and Koike, “Real-time input of 3d pose and gestures of a user’s hand and its applications for hci,” in *Virtual Reality, 2001. Proceedings. IEEE*, 2001, pp. 79–86.
- [94] Z. Yang, Y. Li, W. Chen, and Y. Zheng, “Dynamic hand gesture recognition using hidden markov models,” in *Computer Science Education (ICCSE), 2012 7th International Conference on*, 2012, pp. 360–365.
- [95] A. Youssif, A. Aboutabl, and H. Ali, “Arabic sign language (arsl) recognition system using hmm,” in *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 1, no. 11, 2011.
- [96] B. D. Zarit, B. J. Super, and F. K. H. Quek, “Comparison of five color models in skin pixel classification,” in *Proceedings of the International Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems*, 1999, pp. 58–66.
- [97] L. Zhang, Y. Chen, G. Fang, X. Chen, and G. Wen, “A vision-based sign language recognition system using tied-mixture density hmm,” in *Proceedings of the 6th international conference in Multimodal interfaces*, ser. ICMI ’04. New York, NY, USA: ACM, 2004, pp. 198–204.
- [98] W. Zhong-Hua and Q. Hui, “Chinese chunk recognition using hmsvm method,” in *Artificial Intelligence and Computational Intelligence (AICI), 2010 International Conference*, vol. 1, 2010, pp. 3–7.
- [99] H. Zhou and T. Huang, “Okapi-chamfer matching for articulate object recognition,” in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference*, vol. 2, 2005, pp. 1026–1033 Vol. 2.