UNIVERSITY OF THE WESTERN CAPE

A Comparison of Machine Learning Techniques for Hand Shape Recognition

by

Roland Foster

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 2015

Declaration

I, Roland Foster, declare that this thesis “A Comparison of Machine Learning Techniques for Hand Shape Recognition” 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: ....

"Our deepest fear is not that we are inadequate. Our deepest fear is that we are powerful beyond measure. It is our light, not our darkness that most frightens us. We ask ourselves, 'Who am I to be brilliant, gorgeous, talented, fabulous?' Actually, who are you not to be? You are a child of God. Your playing small does not serve the world. There is nothing enlightened about shrinking so that other people won't feel insecure around you. We are all meant to shine, as children do. We were born to make manifest the glory of God that is within us. It's not just in some of us; it's in everyone. And as we let our own light shine, we unconsciously give other people permission to do the same. As we are liberated from our own fear, our presence automatically liberates others."

Marianne Williamson

Abstract

There are five fundamental parameters that characterize any sign language gesture. They are hand shape, orientation, motion and location, and facial expressions. The SASL group at the University of the Western Cape has created systems to recognize each of these parameters in an input video stream. Most of these systems make use of the Support Vector Machine technique for the classification of data due to its high accuracy. It is, however, unknown how other machine learning techniques compare to Support Vector Machines in the recognition of each of these parameters.

This research lays the foundation for the process of determining optimum machine learning techniques for each parameter by comparing Support Vector Machines to Artificial Neural Networks and Random Forests in the context of South African Sign Language hand shape recognition. Li, a previous researcher at the SASL group, created a state-of-the-art hand shape recognition system that uses Support Vector Machines to classify hand shapes.

This research re-implements Li's feature extraction procedure but investigates the use of Artificial Neural Networks and Random Forests in the place of Support Vector Machines as a comparison. The machine learning techniques are optimized and trained to recognize ten SASL hand shapes and compared in terms of classification accuracy, training time, optimization time and classification time.

Keywords

Hand Shape Recognition, Face Detection, Skin Detection, Background Subtraction, Morphological Operations, Haar Features, Support Vector Machine, Artificial Neural Networks, Random Forests, Optimization.

Acknowledgements

I would first like to thank our Heavenly Father for making all of this possible by granting me the wisdom, strength and focus I needed to study at the University of the Western Cape and to complete this research. Thank you for guiding me throughout my decisions in my life, especially the decision to study further to pursue my MSc degree.
A big thank goes to my parents for all of their support, both emotionally and financially, throughout my life. I am very fortunate in the fact that I could put my focus solely into my studies and you guys would see to all my needs. Thank you for your words of wisdom and motivation and for always showing an interest in my studies.
A special thank you goes to my supervisor Mehrdad Ghaziasgar for your guidance, support, wisdom and patience. You truly showed me that it doesn't matter what the size of the problem is, if you break it down into simpler subsets, anything can be achieved. I thank you for all of the help you have given me during this research, it has been highly appreciated. To my co-supervisor James Connan, thank for all of your support and guidance from afar.
To my friends and extended family, I'd like to say thank you for all your kind words of support and motivation. Thank you for always being there in times of need. To my lab mates Kenzo, Nathan, Diego, Dane, Ibraheem, Warren and Imran, I thank you guys for making our lab a fun place of learning. I would also like to thank you guys for your advice and assistance.
I would like to thank the University of the Western Cape for the opportunity to complete my studies here. For the past 6 years of my life, I've found the lecturers very helpful. I extend my deepest gratitude to all of my lecturers throughout my years of study.
Lastly, but most importantly, to Shannon, thank you for always being there for me and for your support, patience, understanding and love. It really means the world to me and so do you.

Publications

- **Title:** A Comparison of Machine Learning Techniques for Hand Shape Recognition.

  Authors: Roland Foster, Mehrdad Ghaziasgar, James Connan, Reginald Dodds.

  Published in the SATNAC Conference in Port Elizabeth, 2014.

- **Title:** A Comparison of Machine Learning Techniques for Hand Shape Recognition.

  Authors: Roland Foster, Mehrdad Ghaziasgar, James Connan.

  Published in the SATNAC Conference in Stellenbosch, 2013.

Contents

List of Figures

List of Tables

Abbreviations

ANN	**A**rtificial **N**eural **N**etworks
ASL	**A**merican **S**ign **L**anguage
BART	**B**ayesian **A**dditive **R**egression **T**rees
CAMShift	**C**ontinously **A**daptive **M**ean **S**hift
CART	**C**lassification **a**nd **R**egression **T**rees
CCA	**C**onnected **C**omponents **A**nalysis
CPU	**C**entral **P**rocessing **U**nit
DAG	**D**irected **A**cyclic **G**raph
DNBC	**D**ynamic **N**aive **B**ayesian **C**lassifiers
DSP	**D**igital **S**ignal **P**rocessor
FPS	**F**rames **P**er **S**econd
GB	**G**igabyte
GHz	**G**iga **h**ertz
GMM	**G**aussian **M**ixture **M**odels
HMM	**H**idden **M**arkov **M**odels
H-S	**H**ue **S**aturation
HSV	**H**ue **S**aturation **V**alue
JSL	**J**apanese **S**ign **L**anguage
***k*-NNs**	*k*-**N**earest **N**eighnbours
KFPS	**K**orean **F**ingerspelling **P**ractice **S**ystem
KSL	**K**orean **S**ign Language
LibSVM	**L**ibrary of **S**upport **V**ector **M**achines
LR	**L**ogistic **R**egression
MCU	**M**icro-**C**ontroller **U**nit
ML	**M**aximum **L**ikelihood
MLP	**M**ulti-**L**ayer **P**erceptron

MLRF	**M**ulti-**L**ayered **R**andom **F**orest
NB	**N**aive **B**ayes
PC	**P**ersonal **C**omputer
RAM	**R**andom **A**ccess **M**emory
RBF	**R**adial **B**asis **F**unction
RF	**R**andom **F**orests
RGB	**R**ed **G**reen **B**lue
SASL	**S**outh **A**frican **S**ign **L**anguage
SIFT	**S**cale **I**nvariant **F**eature **T**ransform
SVM	**S**upport **V**ector **M**achines
SVM-RBF	**S**upport **V**ector **M**achine - **R**adial **B**asis **F**unction
SVM-POLY	**S**upport **V**ector **M**achine - **P**olynomial **K**ernel

Chapter 1

Introduction

1.1 Background and Motivation

Communication is a key part of the everyday life of a human being. We, as human beings, communicate in order to interact with one another, to express our feelings and to share our experiences and knowledge with one another. More importantly, we communicate in order to access various public services and seek help when required. The ability to communicate is a necessity, as we use it to share ideas with colleagues in the workplace, to learn with and from our classmates at school, to interact with doctors and nurses in hospitals, and interact with our families in our homes. Numerous other similar contexts of use also exist.

About 600,000 Deaf people in South Africa use South African Sign Language as their first and only language [39]. At this point, it is important to point out two common misconceptions that exist in popular belief. The first misconception is that there is a single sign language used by all Deaf people worldwide. The second misconception, that appears to lead on from the first misconception, is that this (single) sign language is a signed-gestural equivalent of a specific spoken language, perhaps English. This in turn leads to the belief that the Deaf all understand a spoken language, perhaps English, and can, at the very least, read and write.

Stokoe—a prominent sign language linguist—showed, first of all, that each country appears to have a sign language that is unique to that country [59]. A few examples are Chinese Sign Language, Australian Sign Language, German Sign Language and South African Sign Language (SASL). Similar to spoken languages, these sign languages may have some overlap, but are completely distinct and unique, similar to how English and German, for instance, are distinct and unique. Stokoe also showed that sign languages are fully-fledged natural languages that are completely independent of spoken languages [59].

The implication of these facts is that a language barrier exists between hearing people and Deaf people, whose first and only language is a sign language. This gives rise to a distinction between two groups of deaf people: the *deaf* with a small *d*, which are deaf people educated and able to communicate in spoken languages in some form; and the *Deaf* with a capital *D*, which are deaf people whose first and only language is a sign language, and are mostly or completely illiterate in spoken languages [31].

The inability to communicate in spoken languages means that the South African Deaf have limited access to public services such as education and health-care [5]. The majority of the South African Deaf are completely illiterate in spoken languages [58]. This means that, similar to hearing people who are illiterate in spoken languages, employment opportunities for such people are scarce. As such, it is found that Deafness in South Africa is characterized by poverty and unemployment. Unlike hearing people who are illiterate, however, the South African deaf have been marginalized and have very little to no easy access to many essential basic services in a language that they can understand [32]. Given a choice between a hearing person that is illiterate in spoken language, but can speak and understand spoken language, and a Deaf person that can't read, write, speak or understand spoken language, it is clear that employers may prefer the former.

Sign language interpreters can be employed to translate between spoken and sign language in certain contexts. The cost of SASL interpreters, however, is limiting to a largely poor Deaf community [22]. SASL interpreters are also very scarce and cannot service a large population, regardless of the cost [22, 64]. In many instances, the presence of an interpreter is inappropriate such as in medical or psychological consultations. While SASL interpreters can and have assisted in alleviating communication problems, a technological translation system that can complement and supplement their services is desired.

The South African Sign Language (SASL) group at the University of the Western Cape is currently developing a machine translation system for the automatic translation between South African Sign Language and English [21]. The main goal of the system is to be able to translate a recorded SASL video into English audio and vice versa. This allows for communication between Deaf users and English speaking users. The project aims to use commodity hardware such as simple web cameras to ensure low cost and simplicity of the final system. The eventual aim of the project is to realize a mobile-based translation system that can be used in any context or place.

Such a system is expected to have a significant impact on the lives of Deaf people in a variety of contexts. It can be used, for instance, in an academic environment to enable Deaf students to study at universities. A Deaf university student can use such a system to make use of spoken lectures. The system can automatically translate lectures in English to SASL and translate SASL questions or comments into English for the Deaf student. The previously mentioned health-care context is another instance in which the eventual SASL system can significantly impact the life of the Deaf.

One significant aspect of the process involved in the SASL machine translation system is the ability to recognize SASL gestures from a video stream captured by a commodity web camera. There are five fundamental parameters that characterize any sign language gesture [59]. These are: hand location, hand motion, hand orientation, hand shape and facial expressions. The recognition of SASL gestures necessarily involves the recognition of each of these parameters in a video stream. This has been one major focus of the research at the SASL group thus far.

The SASL group has produced systems to recognize the hand shape [36], hand location [2, 12], hand motion [3, 20, 45] and facial expressions [44, 69] of a signer in a video stream. The hand shape recognition system of the SASL project, proposed by Li [36], extracts features pertaining to the hand shape from a web camera, making very few assumptions about the input images and providing freedom to the user, as required by the SASL project. It has also been shown to be robust to variations in users such as skin colour, body dimensions and gender.

Li's system, as well as the majority of the other SASL parameter recognition systems mentioned, make use of Support Vector Machines (SVMs) to achieve accurate recognition. SVMs have proven to be very accurate in all of these cases. Li achieved an accuracy of 83.3% in recognizing 10 key SASL hand shapes. However, it is unknown how well other machine learning techniques compare to SVMs in the context of sign language parameter recognition. It is known that specific machine learning techniques may be better suited to specific classification problems than others. While SVMs may be accurate, they may not be optimal in this context.

This research proposes to compare SVMs to two other promising machine learning techniques in classifying the features derived from Li's hand shape feature extraction methodology. While a variety of machine learning techniques exist, two machine learning techniques—Random Forests (RFs) and Artificial Neural Networks (ANNs)—have shown promise in a variety of classification contexts [1, 33, 34, 46, 47, 62].

This research aims to explore the use of SVMs, RFs and ANNs, and compare these machine learning techniques to determine which is best suited for use in the specific context of SASL hand shape recognition. The comparison between the machine learning techniques focuses on four key comparative factors: time-to-optimize, time-to-train, computational speed and accuracy.

Each machine learning technique has one or more parameters that can be optimized to achieve an optimal classification model. The time taken to achieve this optimization differs from technique to technique. Once the optimal parameters are determined, the classification model is trained using the parameters. The time taken to obtain this model also varies between techniques. Once optimization and training is carried out, the final model is characterized by a specific classification accuracy and classification speed, given a new arbitrary image that needs to be classified, called an "unseen image". Both of these factors, once again, differ from technique to technique. All of these factors will be considered in this research.

Ultimately, the comparison of machine learning techniques has to be carried out for all of the systems of the SASL project that recognize each of the SASL parameters. This research lays the foundation for this process by starting with the hand shape recognition system. It can then be extended, in future, to find the best machine learning techniques for the recognition of the other four parameters which recognize SASL gestures.

1.2 Research Question

The following research question can be specified based on the previous section: "How do Support Vector Machines, Artificial Neural Networks and Random Forests compare in the context of SASL hand shape recognition?".

The main research question can be broken down into the following research sub-questions:

1. How do the techniques compare in terms of the time taken for optimization and training?
2. How do the techniques compare in terms of the final classification accuracy on unseen images once they have been optimized and trained?
3. How do the techniques compare in terms of the time taken to achieve a classification result on a single input once they have been optimized and trained?

1.3 Research Objectives

The following objectives will be met in this research in order to arrive at answers to the research question and sub-questions specified in the previous section:

1. Implement Li's complete hand shape feature extraction procedure. The process of locating, tracking and extracting features from the hand will be implemented using Li's methodology.
2. Train and optimize a SVM, RF and ANN, and time these procedures. Each of the machine learning techniques will be trained on a set of hand shapes. The optimal parameters will be found for each machine learning technique and these procedures will be timed and compared.
3. Determine the classification accuracy and classification speed on the testing data. Once optimized and trained models are obtained for each technique, they will be compared in terms of how accurately they classify hand shapes and the computational speed at which they perform the classification.

1.4 Premises

The following assumptions are made in this research:

- It is assumed that the user of the system sits or stands in view of a web camera, without any extra hardware such as sensory gloves or special markers attached to his/her hands or body. The skin colour of the user and the background of the environment in which the user is sitting or standing are also assumed to be arbitrary. Doing so creates a system which provides the most natural experience to the signer, which is a requirement of the SASL project.
- It is assumed that only one signer is present in view of the web camera at any time when performing the SASL hand shapes. This assumption is justified because the user of the eventual system can easily isolate himself/herself into a quiet area when using the system. This is also typical of spoken conversations in which busy and loud environments are avoided.
- It is assumed that the signer will hold up their hand with an open palm until the tracking component of the system is initialized. This is done in order to be able to easily and automatically locate the user's hand initially. From there on, the signer can move his/her hand freely and perform the various SASL hand shapes.

1.5 Thesis Outline

The remainder of the thesis is arranged as follows:

Chapter 2: *Related Work*: This chapter reviews existing literature in order to build a base of understanding of machine learning in the context of existing sign language hand shape, gesture and general recognition systems. It demonstrates that SVMs, RFs and ANNs have been used extensively to achieve high-accuracy classification. It also demonstrates the need to compare machine learning techniques in a specific classification context by showing that specific machine learning techniques may perform well in specific contexts, but poorly in other contexts. It also explains Li's feature extraction procedure and demonstrates that the system is best suited to this research, demonstrating other systems as making stringent assumptions about the input data or using complex and expensive hardware.

Chapter 3: *Techniques for Hand Shape Recognition*: This chapter is split into two main sections. The first section details key image processing techniques required to implement the hand shape feature extraction procedure, as a pre-cursor to classification by the machine learning techniques, and as a base of understanding for the chapters that follow. The second section gives a theoretical discussion of the classification mechanism of the three machine learning techniques—Support Vector Machines, Artificial Neural Networks and Random Forests—which are compared in the context of SASL hand shape recognition in this research, is provided in this chapter as a base of understanding.

Chapter 4: *Design and Implementation of the Hand Shape Recognition System*: This chapter details the implementation of the hand shape recognition system. It discusses and illustrates the feature extraction procedure implemented in order to achieve Objective 1 set out in this chapter. It further explains the process involved in training the three machine learning techniques and using the trained models to obtain a classification label for a given unseen image, as a basis for the experiments detailed in the chapter that follows.

Chapter 5: *Experimental Results and Analysis*: This chapter describes the experiments carried out to optimize each of the machine learning techniques, and subsequently produce optimal trained classification models of each technique, thereby achieving Objective 2 set out in this chapter. It further details the experiments carried out to assess the computational accuracy and speed of each of the machine learning techniques and compare the results of the three techniques, in order to meet Objective 3 set out in this chapter. An analysis of these results culminates in an answer to the main research question and the sub-research questions posed in this chapter.

Chapter 6: *Conclusion*: A summary of the findings of this research and the conclusions drawn is presented, and the thesis is concluded, in this chapter. A discussion of possible directions for future work are also provided.

Chapter 2

Related Work

This chapter looks at related studies in the contexts of recognition and machine learning. The purpose of this chapter is to demonstrate from the literature that:

1. Support Vector Machines (SVMs), Random Forests (RFs) and Artificial Neural Networks (ANNs) can be (and have been) used to achieve high-accuracy recognition in a variety of contexts, hence their selection in this research.
2. The accuracy of different classification techniques varies, and depends greatly on the features used. It is therefore important to compare various classification techniques with a given set of features and select an optimum classifier, which is the basis of this research.
3. Li's feature extraction procedure is robust, flexible and is the most suitable procedure for this research. It is low-cost, provides freedom to the user and is robust to variations in skin colour and body dimensions. Other solutions either use specialized, expensive and/or cumbersome hardware or make stringent assumptions about the nature of the input which puts limitations on the use of such systems.

The chapter is organized into four sections: hand shape recognition using machine learning techniques, gesture recognition using machine learning techniques, comparisons of machine learning techniques and a summary and discussion of these systems.

Hand shape recognition involves an extraction of features pertaining to the shape of the hand from an input source and a subsequent classification of those features into pre-defined category classes using a classification technique. The first section discusses studies that have used various feature extraction and classification techniques to achieve hand shape recognition. Where possible, studies that have used SVMs, ANNs and RFs will be focused on in order to demonstrate that these techniques can be used to achieve high-accuracy classification. Also, Li's feature extraction procedure is discussed and shown to be a robust and suitable feature extraction method as a precursor to the comparison of machine learning techniques carried out.

Gesture recognition is similar to hand shape recognition in that features are extracted and classification is carried out, but it involves the extraction of features that pertain to an entire gesture, such as hand location, orientation etc. Therefore, the second section provides a discussion into gesture recognition systems, detailing the feature extraction and classification techniques used in this regard. Once again, a focus on SVMs, ANNs and RFs aims to demonstrate that these techniques are suitable candidates in the comparison carried out in this research.

The third section discusses selected studies which have compared the use of various machine learning techniques for the purpose of classification in computer vision as well as general classification problems. The section demonstrates the fact that, depending on the specific context and the features used, a specific classifier may perform better than all others, hence the need to carry out a comparison as is the case in this research.

A summary and conclusions section concludes the chapter.

2.1 Hand Shape Recognition Using Machine Learning Techniques

Hand shape recognition systems in the literature can broadly be sub-divided into two categories: hardware-based systems and vision-based systems.

Hardware-based systems are systems that make use of special hardware such as a colour-coded clothing, Data Gloves and depth sensing, stereo or 3D cameras for the extraction of hand shape features. Using such equipment, both, increases the accuracy of the features extracted, and simplifies the extraction procedure. Using such hardware, however, places constraints on the system and the user, and can greatly increase the cost of the system and reduce the freedom of the user of the system.

Vision-based systems are systems that use only an inexpensive web camera to capture input in the form of a video stream. The systems then either rely heavily on a variety of image processing techniques to reduce noise in, and extract clear features from, the input images, or they make stringent assumptions on the nature of the input in order to simplify the process. Making use of a vision-based setup with few stringent assumptions can be very challenging and requires a robust set of algorithms to achieve high-accuracy recognition. However, it ensures a low cost and natural feel to the system.

The following subsections describe the hardware-based and vision-based hand shape recognition systems, respectively, in the literature.

2.1.1 Hardware-Based Systems

Tabata and Kuroda [60] proposed a system to recognize hand shapes for finger spelling in Japanese Sign Language (JSL) which they call "Stringlove". The system uses a custom-made glove fitted with sensors to capture features of the fingers. The glove consists of 24 inductcoders and nine contact sensors which jointly help determine several parameters such as: the joint flexion/extension of the fingers, adduction/abduction angles of fingers, thumb and wrist rotations and the contact position between the fingertips of the fingers.

The system encodes finger parameters and shapes into sign notation code using a Digital Signal Processor (DSP) embedded in the glove. The notation codes acquired are then used to recognize hand shapes on the basis of the distinctive features of each finger-spelling hand posture. A matching procedure is carried out between the notation code of the hand shape obtained from the glove and notation codes of various hand shapes stored in a hand shape database. The closest matching hand shape is determined as the correct recognition result. The system could recognize six JSL finger spelling hand shapes, namely: "A", "U", "TE", "FU", "RO" and "WA".

JSL Hand Shape	Accuracy (%)
A	66
U	100
TE	94
FU	100
RO	84
WA	100

TABLE 2.1: The mean recognition accuracy for each JSL hand shape by Tabata and Kuroda.

A preliminarily experiment was carried out using two subjects to determine the recognition accuracy of the system. Each subject performed each of the six JSL hand shapes three times. For each time, the system would attempt to recognize the hand shape multiple times over a period of time, although exactly how many times and how long are not clear from the literature. The proposed method showed the notation codes of a hand shape from the measured data. Table 2.1 illustrates the average accuracy of the system for each JSL finger spelling hand shape across both subjects and all classification attempts.

The results demonstrate that the use of custom hardware can provide very high recognition accuracies, even as high as 100% for many of the hand shapes, with the exception of the JSL hand shape for ‘A’ which achieves a 66% accuracy.

In a subsequent study, the same researchers worked to improve the Stringlove prototype data glove to use only six sensors, as opposed to the 24 sensors used previously [61]. The newer system follows the same finger categorization method as mentioned previously. The system was also trained to recognize a much larger number of JSL hand shapes–28 hand shapes as shown in Figure 2.1. In testing the system, the newer prototype achieved an 82% accuracy for the 28 JSL hand shapes.

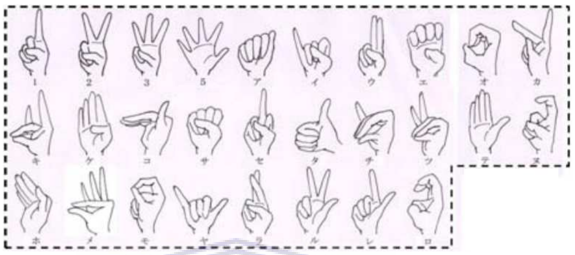


FIGURE 2.1: The 28 JSL finger spelling hand shapes recognized by Tabata *et al.* [61].

Kuznetsova *et al.* proposed a system for real-time recognition of American Sign Language (ASL) using a depth camera [34]. The system's feature extraction procedure involves depth processing on a depth image of an isolated hand, as shown in Figure 2.2. The input hand image is captured and a depth threshold is used to segment the hand in the image, that is, the hand is assumed to consist of all pixels that are closer than a specific distance/threshold to the camera. This depth processing forms a depth image which is converted into a point cloud by means of inverse perspective transformation.

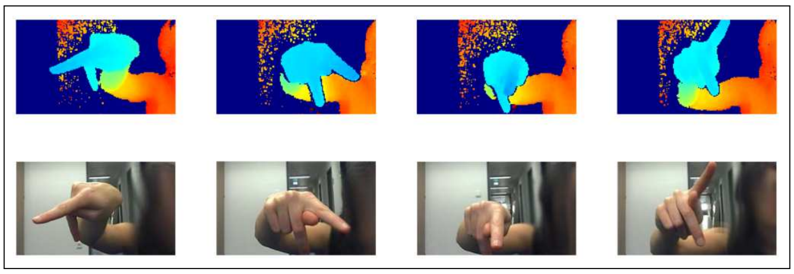


FIGURE 2.2: The isolated hand depth images used for classification [34].

A series of concatenated histograms of the resulting image form the feature vector needed to train a Random Forest-variant called a Multi-Layered Random Forest (MLRF). Data clustering of the feature vectors is first performed before training the MLRF. Once data clustering is complete, the first level of the random forest is trained on the aggregated feature vectors. A cluster label is assigned to each incoming vector of the forest. After the first level training, for each of the clusters, a separate random forest is trained on the full feature vectors to distinguish between similar signs.

For testing, each sample passes through the first-level forest to determine the cluster label of the sample. The sample is then passed to the corresponding forest on the second level to determine its class label.

A public dataset of 24 ASL finger spelling signs consisting of 65000 images from 5 subjects was used to evaluate the accuracy of the MLRF. The data of four of the five subjects was used to train the MLRF and the system was tested using the data of the fifth subject. The system was shown to yield a recognition accuracy of 97.4% across all finger spelling signs. Another test was carried out in which half of the data was used in training and the other half in testing. The system accuracy deteriorated to 84.7%.

Another system for finger spelling recognition of ASL signs was proposed by Otiniano-Rodríguez *et al.* [53]. The proposed system uses a Microsoft Kinect sensor to collect RGB-D information from images.

The system is comprised of four stages. In stage one, a depth map is used to segment the hand area from the background. The Kinect provides both depth and colour data which are used to extract the exact hand shape. Stage two entails the extraction of features from the depth map and intensity images using Gradient descriptors and the Scale Invariant Feature Transform (SIFT) descriptors respectively. During stage three, the Bag-of-Visual-Words model is applied to obtain semantic information about the RGB-D images.

For the use of the Bag-of-Visual-Words model in the research, an image is considered to be the document and the “words” are the visual entities found in the image. A Support Vector Machine (SVM) was used for classification of the ASL signs.

The ASL Finger Spelling Dataset [51] was used to train and test the system. Three types of experiments were performed in order to test the classification accuracy of the SVM. The first experiment used only RGB images with colour data for testing and training, and the SVM accuracy achieved was 62.70%. The second experiment made use of depth images for testing and training and the SVM achieved an accuracy 85.18%. The third experiment made use of the RGB-D images for testing and training and the SVM achieved the highest accuracy of 91.26%.

It can be observed that the above systems all use specialized hardware for hand shape recognition. It is noted in each case that the accuracies obtained are very high, but each proposed setup is complex, costly and cumbersome.

2.1.2 Vision-Based Systems

Li, a former student of the SASL group at the University of the Western Cape, developed a state-of-the-art system to recognize ten SASL hand shapes in real-time [36]. The system takes in live video frames of a signer's upper or entire body from a consumer web camera and continuously recognizes SASL hand shapes performed by the signer in real-time.

The system detects the face of the signer in the initial video frame using Haar-like features. Once the face has been detected, the position of the nose is determined by isolating the centre of the facial frame. The skin colour distribution of the detected nose is computed and used to highlight the skin pixels of the signer in every frame of the video sequence thereafter. In order to achieve skin highlighting, histogram back projection is applied using the skin colour distribution determined from the nose region. Gaussian Mixture Models (GMM) are used to achieve background subtraction to separate the background and foreground of the image. Doing this ensures that only the moving parts, in this case the signer's hands, are present in the image.

The hand is located in the resulting image using Hierarchical Chamfer matching only once on the initial frame. This is used to initialize the CAMShift tracking algorithm, which continuously tracks the located hand. Rotations of the hand are normalized by aligning the hand region to the vertical axis. Connected Components Analysis (CCA) is used to highlight the contour of the hand region in every frame and the contour image is resized to a resolution of 20×30 pixels. The resulting image is used as a feature vector for the hand shape recognition process.

A SVM was used to classify SASL hand shapes. The SVM was trained to recognize ten SASL hand shapes. The system was demonstrated as being very accurate, achieving an accuracy of 83.3% across all hand shapes on even complex backgrounds. It was also demonstrated to be highly robust to variations in test subjects such as skin colour and hand dimensions. Figure 2.3 depicts Li's system in action.

Li's feature extraction procedure is seen as very suitable for the purposes of this research as it only uses an inexpensive web camera, but is still highly accurate and robust to complex backgrounds and variations in users. It also provides freedom to the user and makes no assumptions about the position of the hand in the frame.

Nyugen *et al.* proposed a system for the recognition of ten American Sign language (ASL) hand shapes with the use of a consumer web camera to capture input [46].

Extraction of features by the system first involves hand detection using a static skin colour filter proposed by [18]. The result of applying the skin colour filter is shown in

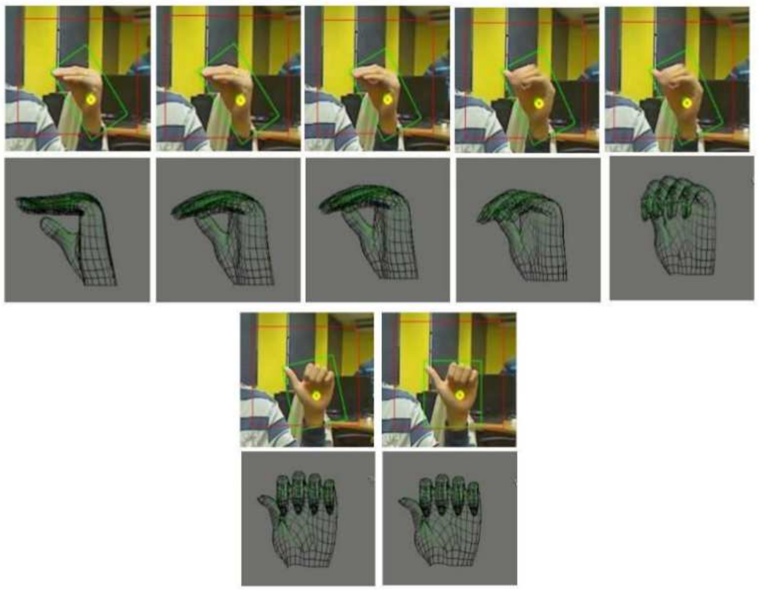


FIGURE 2.3: An example of the Li's hand shape estimation system [36].

Figure 2.4. A median filter is then used to reduce noise in the image. The largest object in the input frame is assumed to be the hand and all smaller objects are the removed from the frame as seen in Figure 2.5.

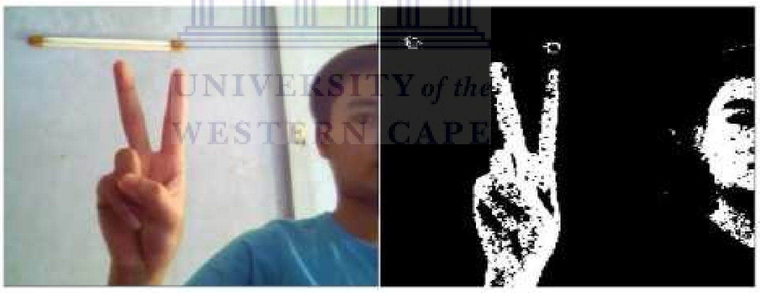


FIGURE 2.4: The skin colour filter used by Nyugen *et al.* [46].

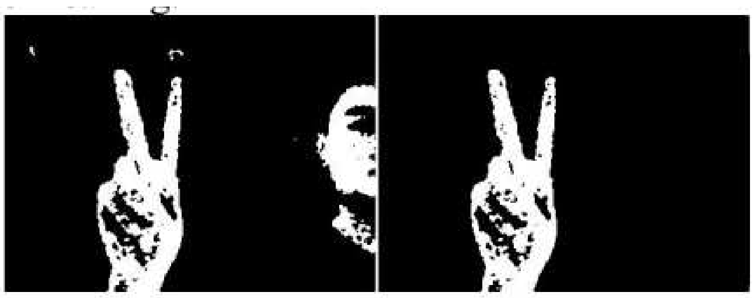


FIGURE 2.5: The result of selecting the largest object in Nyugen *et al.*'s system [46].

A flood fill operation is used to fill the noisy hand contour in the input frame. A wrist detection algorithm is used to find the position of the wrist in order separate the hand from the arm as shown in Figure 2.6. The feature vector used is composed of three main features: the change of the horizontal/vertical object pixels, the shape of the boundary of the hand and the scalar description of the hand.

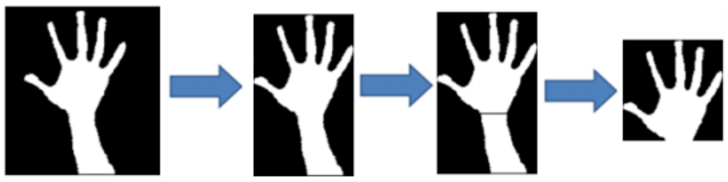


FIGURE 2.6: Segmentation of the hand from the arm [46].

Artificial Neural Networks (ANNs) were selected as the machine learning technique for hand shape classification. A Multi-Layer Perceptron (MLP) network was the type of ANN used. It consists of three layers: an input layer, which in this case had 48 neurons corresponding to the size of the feature vector, a hidden layer which consists of neurons, the quantity of which was decided using a process of trial-and-error, and an output layer with ten neurons corresponding to the ten ASL hand shapes to be recognized.

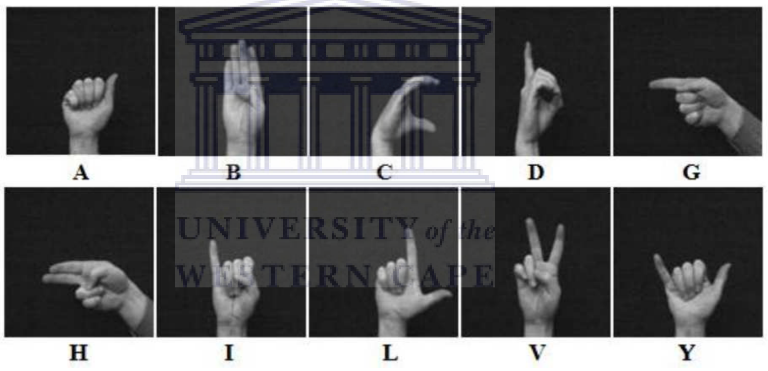


FIGURE 2.7: The ten ASL hand shapes to be recognized by Nyugen *et al.* [46].

The data used to train the ANN was collected from an American Sign Language hand posture dataset [38]. Figure 2.7 depicts some of the images of the dataset. A custom dataset collected using a Logitech 9000 web camera on a simple background with stable lighting conditions was used to test the system. The videos were taken only of the hand of the user for easier segmentation. Five people were used to collect this video data, with each person performing each of the ten ASL hand shapes once.

The training data consisted of a total of 450 samples across all hand shapes, and the testing data consisted of a total of 445 samples across all hand shapes. Four different vector sizes were used in testing the system. Feature vector 1 had a vector size of 24 elements and achieved an accuracy of 97.1%. Feature vector 2 achieved an accuracy of 97.3% with a vector size of 32 elements. Feature vectors 3 and 4 both achieved an accuracy of 98.0% with vector sizes of 40 and 48 elements, respectively.

While the accuracies achieved are very high, stringent assumptions are made about the nature of the input data in order to simplify the feature extraction procedure and achieve these accuracies. Specifically, it is assumed that the input frames consist mostly or only of a single vertically aligned hand on a simple background. This severely limits the freedom of the user in interacting with the system.

Kulkarni and Lokhande [33] also created a system for the automatic translation of 26 ASL finger spelling hand shapes. The system uses three image processing techniques for feature extraction and an ANN for recognition of hand shapes. An overview of the proposed system is shown in Figure 2.8 which is taken from their work. Figure 2.9 depicts sample images of the 26 ASL hand shapes recognized by the system. As seen in the figure, the system uses images containing only a single hand on a simple background in a vertical position, similar to Nyugen *et al.*.

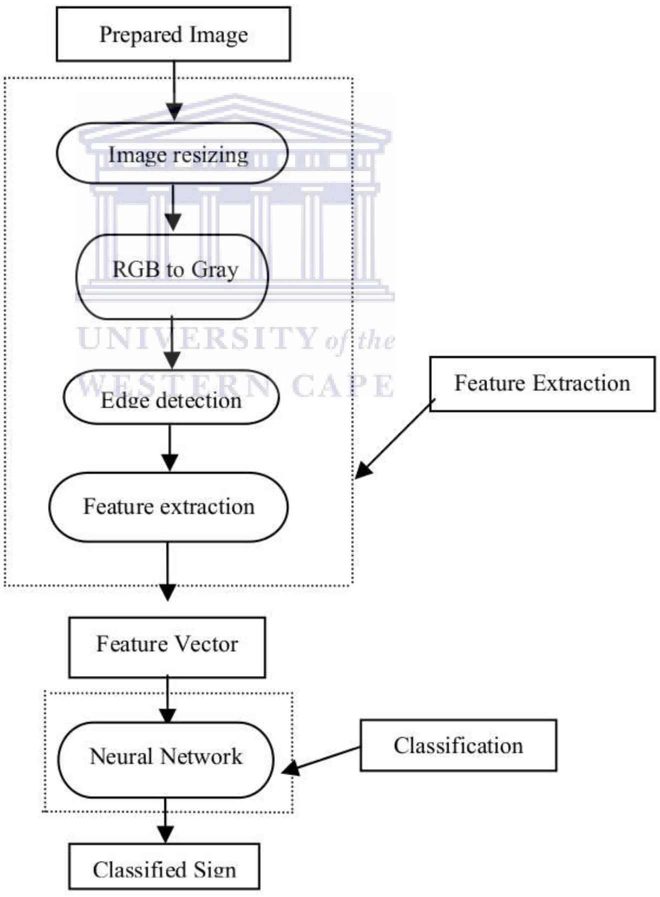


FIGURE 2.8: An overview of Kulkarni and Lokhande's image processing procedure [33].

In extracting features, the input image is first resized to a resolution of 80×64 pixels and this image, which is in the default Red-Green-Blue (RGB) colour space, is converted to grayscale. Canny edge detection [13] is used to highlight the edges in the image which

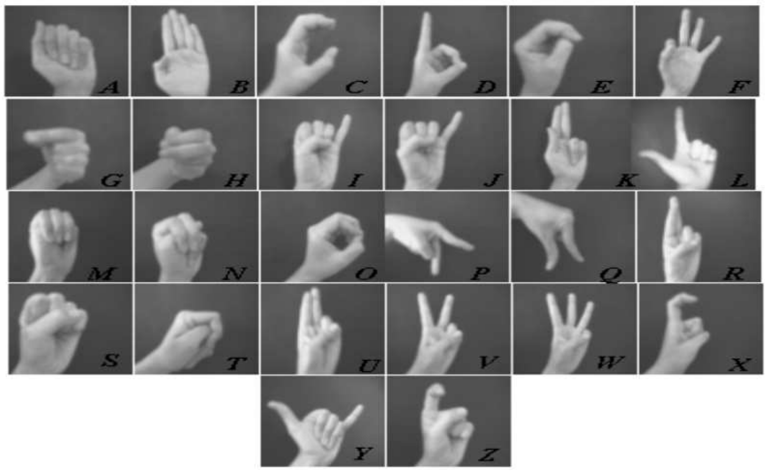


FIGURE 2.9: Sample ASL finger spelling images recognized by Kulkarni and Lokhande's system [33].

collectively form the contour of the hand in the grayscale image. These edges form the features used to train the ANN.

A Multi-Layer Perceptron (MLP) Neural Network is used to classify input images as one of the 26 ASL hand shapes. Testing was carried out by making use of both training and testing data to test the accuracy of the system. The dataset consisted of eight volunteers performing each of the 26 ASL letters. Hence, there were 8 samples per ASL letter, with 5 of the 8 samples used to train the ANN and the remaining 3 samples used for testing. For 12 of the 26 ASL letters, all 3 samples were correctly recognized. For an additional 12 letters, 2 out of 3 samples were correctly recognized. For the remaining 3 letters, only 1 of the 3 samples were correctly recognized.

Once again, while a simple hardware setup is used, specific stringent assumptions are made about the input data in order to simplify the feature extraction procedure. While the accuracies achieved are high, the assumptions limit the freedom of the user which contravenes the requirements of the SASL project. It is also noted from both of the previous studies that the use ANNs can yield very promising results given a robust set of hand shape features.

2.2 Gesture Recognition Using Machine Learning Techniques

Similar to hand shape recognition systems, gesture recognition systems in the literature can also broadly be sub-divided into the two distinct categories previously described: hardware-based systems and vision-based systems. The following subsections describe the hardware-based and vision-based gesture recognition systems, respectively, in the literature.

2.2.1 Hardware-Based Systems

Kadous compared two machine learning techniques in the recognition of Australian Sign Language gestures [29]. Australian Sign Language is also known as Auslan by the Australian deaf community. There are four thousand well defined signs in this language. Instrumented gloves are used by the system in order to track the user's hand and extract features for recognition. The justification given for the use of data gloves is that they have been used extensively for direct manipulation in virtual environments and can therefore also be used in gesture and sign language recognition [29]. The instrumented glove chosen for this task was the Nintendo PowerGlove.

The PowerGlove was originally designed for use with the Nintendo gaming system. It is a gaming accessory which provides a set of three attributes in order for feature extraction to take place. The first set of attributes is the x, y and z positions of the glove relative to a point of synchronization. The second set of attributes is the degree of rotation of the wrist given in 30 degree increments. Finally, the degree to which each of the first four fingers is bent on a scale of 1 to 4 is also provided for each finger.

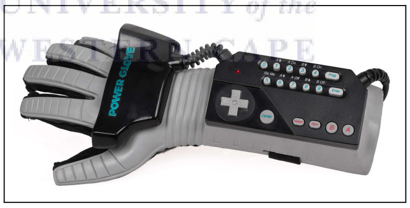


FIGURE 2.10: The Nintendo PowerGlove: The instrumented glove chosen for feature extraction by Kadous [29].

Two machine learning techniques were used for gesture recognition and compared in this regard. They were instance-based learning and decision tree building. Instance-based learning stores all training instances in "attribute space". Given an unseen instance, it finds the nearest instance in the attribute space and classifies the test instance according to this nearest neighbour. Decision tree building builds a hierarchy of decisions based on attribute values. The attribute values of an unseen instance can then be used to retrace a series of decisions to a specific class.

A set of 95 Auslan signs performed by 3 signers were selected to train and test the system. These signs were one-handed signs collected using a PowerGlove worn on the right hand of test subjects. The glove was attached to an SGI Iris 4D workstation for processing. Each of the signers contributed between 8 and 20 samples for each of the

95 signs. To avoid the effects of fatigue on the results of individual signs, the order of the signs was randomly changed between signers. A total of 6650 signs were collected and these were used to compare the recognition accuracy of the two machine learning techniques.

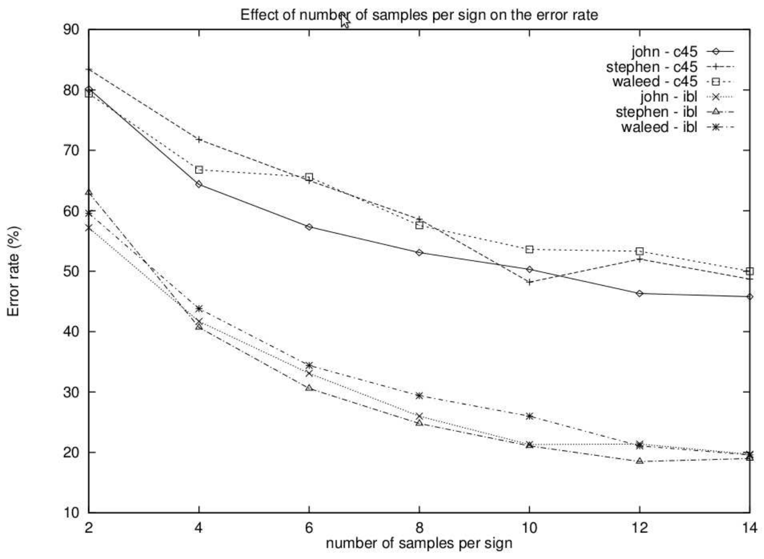
Five-fold cross validation was used to ensure a good measure of recognition accuracy. n-fold cross validation involves dividing the collected data into n sets and in each set, $n-1$ parts of the data are used to train the classifier and the remaining part is used for testing. The average accuracy over all n parts provides a good measure of the recognition accuracy of the classifier. The Instance-based learning technique achieved an accuracy of 80% and the C4.5 implementation of a decision tree builder achieved a significantly lower accuracy of 55%.

Kadous analysed the behaviour of each of the machine learning techniques. Figure 2.11a depicts the error rate of the system with respect to the number of samples per sign used to train each classifier. Figure 2.11b summarizes the error rate with respect to the size of the lexicon recognized. He analysed the effect of the number of samples per sign on the error rate, and it was expected that a larger number of samples would reduce the rate of error. Figure 2.11a confirms the expectation that an increase in the number of samples per sign lowers the error rate, thus improving recognition accuracy. As regards the influence of the lexicon size on the error rate, it was expected that a larger lexicon would make it harder for each the classifiers to discern between signs, hence, a higher error rate. Figure 2.11b once again confirms this expectation.

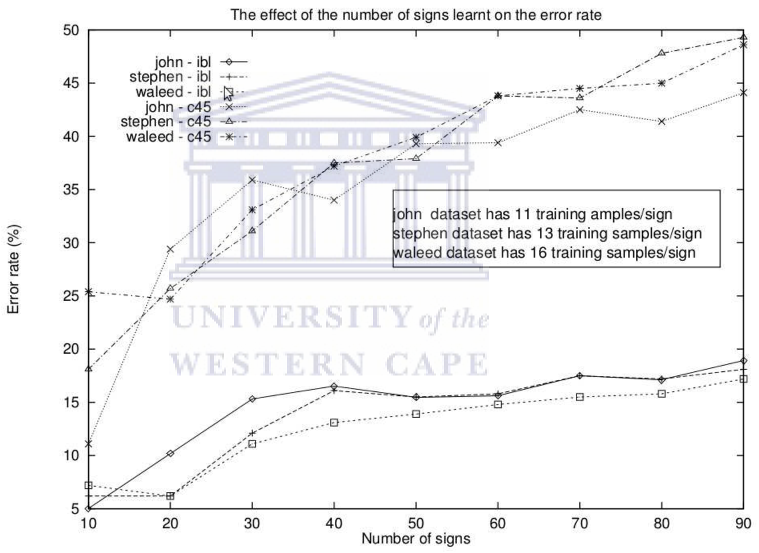
Lee *et al.* proposed a Korean Fingerspelling Practice System (KFPS) which uses a data glove [35]. The KFPS is comprised of three modules: letter, word and short sentence recognition of Korean Sign Language (KSL), as well as a gesture-based game. There are a total of twenty-four hand shapes for KSL letters which are composed of fourteen consonants and ten vowels. The twenty-four KSL letters are shown in Figure 2.12. Combinations of these letters form words and short sentences.

The custom-made data glove used by the system to capture gestures consists of 10 sensors: 5 flex-sensors, 3 pressure-sensors and 2 tilt-sensors. These sensors measure the gesture postures of the palm and fingers for the extraction of features. The glove is shown in Figure 2.13.

As can be seen from Figure 2.13, there is a flex sensor placed on each of the five fingers, a pressure sensor between the fingers and tilt sensors are located on the back and the palm of the hand. A Micro-controller Unit (MCU) is used to capture the data from each sensor. The data is then transmitted via the Slave Bluetooth to the Master Bluetooth device, both depicted in Figure 2.13, which is connected to a computer for processing and



(a)



(b)

FIGURE 2.11: The effects of the lexicon size and the number of samples of each sign on the accuracy of Kadous' system [29].

recognition. The twenty-four letters of KSL are subdivided into four groups according to their tilt orientation. To obtain the finger posture measurement, the flex and pressure sensors of the glove are used. The values gathered from these sensors are stored in a database.

The classification of the gestures is performed using a k-means algorithm. Five subjects of mixed gender were used to test the KFPS system. Each subject had to perform a series of tasks which involved performing various KSL letters, words and sentences. Each task was carried out three times by each of the five subjects. The KFPS system achieved a gesture recognition rate of 80.27%.

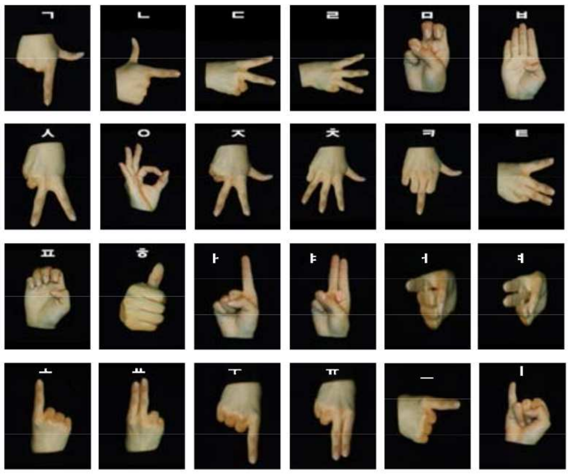


FIGURE 2.12: Samples from the dataset consisting of 24 Korean Sign Language letters [35].

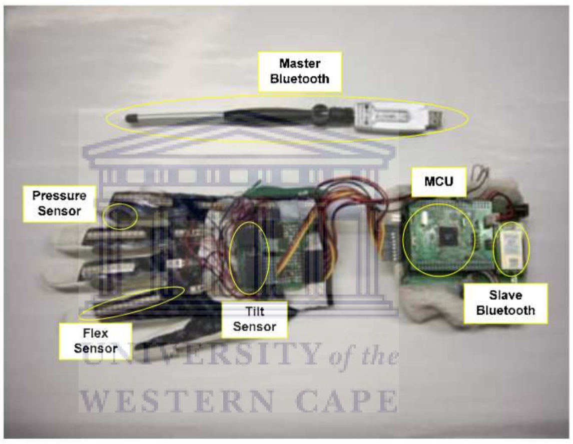


FIGURE 2.13: The proposed data glove of Lee *et al.* and its components [35].

The previous studies once again demonstrate that the use of specialized hardware can yield very high accuracy recognition at the expense of simplicity, cost and user freedom. It is also noted that different machine learning techniques can yield very different accuracies and it is, hence, crucial to compare techniques to determine an optimal classifier.

2.2.2 Vision-Based Systems

Avilés *et al.* presented a study to assess the performance of Dynamic Naive Bayesian Classifiers (DNBCs) and Hidden Markov Models (HMMs) [6] for gesture recognition. An adaptive skin detection scheme was used to handle users of different ethnicities. The adaptive skin detection scheme first finds the user's face using the Viola-Jones Face detection algorithm [66]. Once the face has been detected, the dimensions of the facial frame and known average proportions of the body are used to estimate the positions of the user's torso and right hand as shown in Figure 2.14.

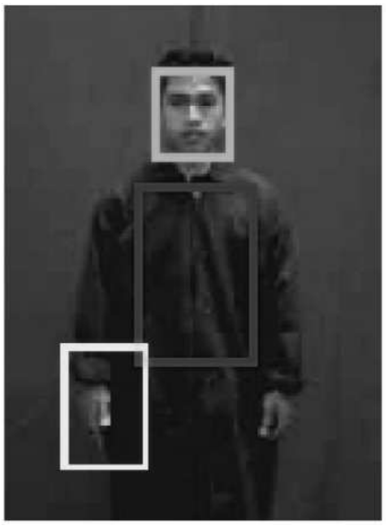


FIGURE 2.14: Estimation of the torso and hand positions of the person by Avilés *et al.* [6].

A Bayes classifier is used according to [28] to label pixels in the colour image as skin or non-skin pixels. A small skin-colour search window is applied to the hand detected in order to track it. Tracking of the hand is achieved using the CAMShift tracking algorithm [10]. This algorithm accurately tracks hand motion over a sequence of image frames under their experimental conditions. A manually collected dataset composed of 10 gestures, shown in Figure 2.15, performed by 10 men and 5 women using the right arm was collected and used to train and test the system. Each gesture was also performed at varying distances to the camera and at varying rotations for an experiment mentioned below. Each of the participants supplied a different number of gesture samples but no less than 50 samples of each gesture. In total, the dataset contains 7308 gesture samples.

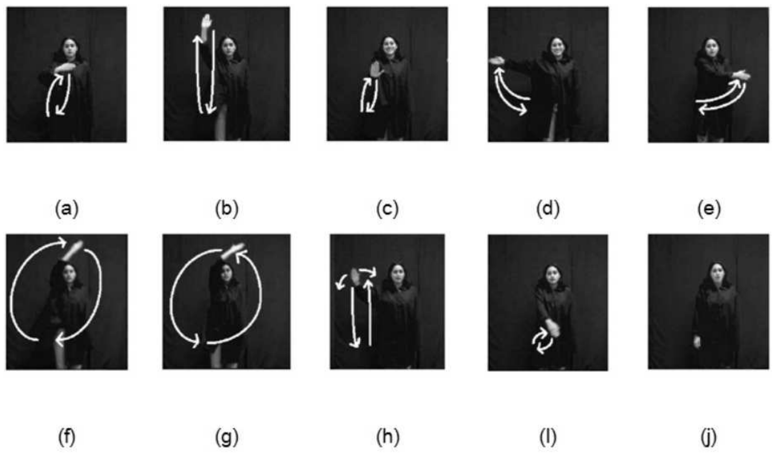


FIGURE 2.15: The 10 gestures recognized by Avilés *et al.*'s system [6].

The hardware used for the experiments included an IBM Intel Pentium 1.6 GHz computer with 512Mb RAM, a Sony EVI-D30 camera and a WinTV frame grabber. Two experiments were conducted in order to compare the classification and learning performances of the DNBCs and HMMs. In both experiments, two sets of features were used to train and test the classifiers to compare their effectiveness as feature representation methods. The first set of features included only information about the motion of the right hand ("motion data") while the second set included both right-hand motion information and information about the posture of the left hand ("posture and motion data").

For each machine learning technique, 15 trained classifiers were created corresponding to each of the 15 subjects. In other words, each classifier was trained to recognize the gestures of one of the 15 subjects, yielding 15 HMM classifiers and 15 DNBC classifiers. The classifiers constructed for each person were used to classify gestures from the other 14 subjects. This method provides a very good indication of the ability of each technique to generalise to other signers. Two samples per gesture were randomly selected from the images of each subject for this test, yielding a total of 48 samples per gesture. In this test, DNBCs achieved an average recognition rate of 73.85% with posture and motion data and 52.80% with motion data. HMMs achieved an average recognition rate of 74.80% with posture and motion data and 51.60% with motion data. It is clear that both techniques provide comparable results in this case, but HMMs perform slightly better with features that include both motion and posture, while DNBCs perform slightly better with motion data only.

The second experiment focused on assessing the robustness of the classifiers to variations in rotation of the gestures and the distance of the subjects to the camera. To assess the robustness to the distance from the camera, 15 samples of each gesture, as performed at 2m and 4m, were randomly chosen, resulting in a test set of 30 samples per gesture. In this case, DNBCs outperform HMMs for both posture and motion data features and motion data features. To assess the robustness to rotations, once again 30 samples of each type of gesture performed at an angle of $\pm 45^\circ$ were randomly selected. In this case, HMMs outperformed DNBCs for posture and motion data features with an average difference of 4.61%, while the use of motion data features yields poor results for both classification techniques.

It is clear that a specific machine learning technique can yield a high accuracy with a specific set of features, but a low accuracy with a slightly different set of features. As such, it is crucial to compare a variety of machine learning techniques with a specific set of features to select an optimum technique, as is the objective of this research.

2.3 Comparisons of Machine Learning Techniques

This section discusses studies which compare machine learning techniques in the context of various classification problems.

Trigueiros *et al.* compared *k*-Nearest Neighbours (*k*-NNs) classifiers, Naive Bayes (NB) classifiers, Artificial Neural Networks (ANNs) and Support Vector Machines (SVMs) in the recognition of 10 generic hand gestures [62]. The study used the Microsoft Kinect camera and Rapid Miner for the development of the experiments performed on the machine learning techniques. Two datasets consisting of different hand features were created. The first dataset comprised of the following features: the angle of the hand, the mean and variance in the grey values, the area and perimeter and the number of convexity defects of the segmented hand. The second dataset comprised of the following features: the angle of the hand, the mean and variance in the grey values, an orientation histogram and the radial signature of the segmented hand. However, the number of samples and test subjects of each dataset is unclear from the literature.

An application using the Kinect camera collected the grey image values and the depth image values and stored these into a database. A 10-fold cross-validation technique was used to determine the recognition accuracy of the machine learning techniques on both datasets. The Rapid Miner application was used to analyse the results and compare the performance of the four machine learning techniques. An Intel i7 with a 2.8GHz processor and 4GB RAM was used with the RapidMiner 5.2 application to carry out the experiments.

	Classifier	*k*-NNs	Naive Bayes	ANNs	SVMs
Dataset 1	Accuracy(%)	92.45	25.87	96.99	91.66
	Time(s)	8	1	2793	190
Dataset 2	Accuracy(%)	88.52	66.50	85.18	80.02
	Time(s)	1	1	32	68

TABLE 2.2: Recognition accuracy and training time using datasets 1 and 2 by Trigueiros *et al.* [62].

Table 2.2 shows the recognition accuracies and the training time of each classifier for the two tested datasets. For dataset 1, ANNs achieve the highest recognition accuracy of 96.99%. This accuracy, however, comes at the expense of training time which is orders of magnitude larger than the other classifiers. *k*-NNs and SVMs also achieve very high accuracies of 95.45% and 91.66% respectively. The lowest recognition accuracy of 25.87% was achieved by the Naive Bayes classifier, which also takes the least time to train. Overall, *k*-NNs provide the best combination of accuracy and time in this case.

The results for dataset 2 in Table 2.2 show that k-NNs once again achieve the highest accuracy of 88.52% and smallest training time, with ANNs and SVMs closely following with accuracies of 85.18% and 80.02%, respectively. The training times of the ANN and SVM were significantly reduced on this dataset, attributed to the smaller dimensionality of the features used. Once again, the Naive Bayes classifier achieves the lowest recognition accuracy of 66.50%.

Nitze *et al.* compared four machine learning techniques in the classification and recognition of agricultural crop types [47]. Random Forests (RFs), Support Vector Machines (SVMs), Artificial Neural Networks (ANNs) and Maximum Likelihood (ML) were the four machine learning techniques chosen for classification. A multi-temporal set of RapidEye images were used for classification. These images cover the optical electromagnetic spectrum in five bands: blue, green, red, red-edge and near-infrared, and have a ground sampling distance of five meters.

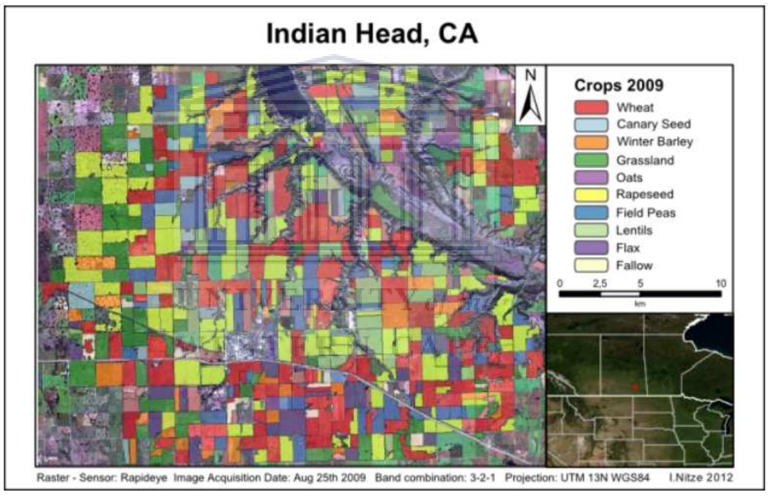

Indian Head, CA
N
Crops 2009
Wheat
Canary Seed
Winter Barley
Grassland
Oats
Rapeseed
Field Peas
Lentils
Flax
Fallow
0 2.5 5 10
km
Raster - Sensor: Rapideye Image Acquisition Date: Aug 25th 2009 Band combination: 3-2-1 Projection: UTM 13N WGS84 I.Nitze 2012

FIGURE 2.16: Overview of the study area of Nitze *et al.* showing the field boundaries with crop types [47].

The study area, shown in Figure 2.16, is located in Indian Head in Canada and spans an area of 20×25 km. A total of 512 agricultural fields of known crop and cultivation types grew in the study area during the summer months of 2009. Ten distinct crop types, summarized in Table 2.3, were selected for classification after excluding the very small and semantically similar classes.

The following implementations of the classifiers previously mentioned were compared: Naive Bayes in the form of Maximum Likelihood (ML); Random Forests (RF); Artificial Neural Networks (ANN) in the form of a Multi-Layer Perceptron; and the LibSVM implementation of the radial-basis-function kernel (SVM-RBF) and the polynomial kernel (SVM-POLY). For each run, the training and testing datasets were randomly selected. Data was split as follows: 80% was used as testing data and 20% as training data.

Crop type	# of fields
Wheat	161
Rapeseed	136
Grassland	79
Field Peas	52
Barley	40
Lentils	38
Flax	37
Oats	30
Fallow	19
Canary Seed	18

Table 2.3: Cultivated crops recognized by Nitze *et al.* and the number of fields in the study area with each type of crop.

Classifier	Recognition Accuracy (%)	Training Time (s)	Classification Time (s)
ANN	87.1	15.145	0.003
ML	78.9	0.005	0.017
RF	87.4	6.205	0.083
SVM-POLY	87.8	0.296	0.020
SVM-RBF	88.1	0.292	0.039

Table 2.4: The classification accuracy, training time and classification time of each machine learning technique by Nitze *et al.*

Table 2.4 summarizes the results of the experiment. Although all the methods achieved a generally high recognition accuracy, the highest accuracy for the classification of crop types was 88.1% with the SVM-RBF and the second highest was the SVM-POLY implementation which achieved 87.8%. In this case, ML had the lowest, but not low, recognition accuracy.

In terms of training time, ANNs took the longest to train, followed by RFs. Both SVM implementations and ML took less than a second to train. On the other hand, the ANN had the fastest classification time of 3 milliseconds. All other methods were at least one order of magnitude slower, with the slowest method being RFs. The classification time across all methods was, however, generally fast.

Nimeh *et al.* compared the recognition accuracy of several machine learning methods including Logistic Regression (LR), Classification and Regression Trees(CART), Bayesian Additive Regression Trees (BART), Support Vector Machines (SVM), Random Forests (RF), and Artificial Neural Networks (ANNs) to detect phishing emails [1]. The dataset comprised of 1718 legitimate emails and 1171 raw phishing emails. A total of 43 features were extracted from emails and used for the training and testing of the afore-mentioned machine learning techniques. A detailed account of the features used can be found in [1].

A 10-fold cross-validation technique was used to determine the classification accuracy of the machine learning techniques. Table 2.5 summarizes the mean error rate achieved by each of the machine learning techniques. It should be noted that a lower error rate is desirable as it indicates a higher success rate, and a higher accuracy.

Classifier	Error Rate (%)
RF	7.72
CART	8.13
LR	8.85
BART	9.69
SVM	9.90
ANN	10.73

TABLE 2.5: Phishing email error rate (lower is better) using various classifiers by Nimeh *et al.*

Although all the classifiers had a generally comparable detection accuracy, in this application, RFs performed slightly better than all other classifiers. ANNs and SVMs performed the worst, albeit only by a small margin.

All of the above studies clearly demonstrate that each machine learning technique may yield a very high accuracy in a specific context and given a specific set of features, but perform poorly in a different context or with a different set of features. It is therefore crucial to compare a variety of machine learning techniques for a specific classification problem and set of features to determine the optimal technique in that context. It is also demonstrated that ANNs and RFs can potentially yield excellent classification results. ANNs, however, may be costly (in terms of time) to train, depending on the specific ANN configuration used.

2.4 Summary and Conclusion

This chapter presented a comprehensive literature survey in the fields of sign language recognition, gesture recognition and general recognition. Where possible, the pre-processing and feature extraction procedure, the hardware and the machine learning technique used in each case was described. Several important conclusions can be drawn from the studies detailed in the chapter.

The first conclusion to be drawn is that the real-time hand shape recognition system created by Li [36] is highly accurate and robust, and makes very few assumptions about the scene and the type of hardware used. The input frame contains the entire signer which allows for the system to be extended for the recognition of two handed sign language gestures. It also does away with cumbersome and expensive hardware requirements by making use of an inexpensive web camera.

Several other studies make use of various types of complex hardware such as data gloves, Kinect cameras and other 3D depth sensing camera configurations for the extraction of features, which in most cases result in high accuracy classification. Unfortunately, such configurations are usually seen as cumbersome and expensive and are not suitable for the purposes of the SASL project which aims to use inexpensive and simple commodity hardware.

A number of studies do make use of only a web camera for the feature extraction procedure. However, many of these make stringent assumptions about the type of input data, in many cases isolating the input image to only the manually segmented hand region of the signer. This limits the freedom of the user and is not suitable for the eventual goal of the SASL project which is to capture whole-body gestures, including the hand shapes, to infer the meaning of the gestures spoken. Other studies make less stringent assumptions but still do not provide the freedom and robustness of Li's feature extraction procedure. As such, Li's method is seen as the most suitable solution for the feature extraction procedure of this project.

The chapter also clearly demonstrated the potential of Artificial Neural Networks (ANNs) and Random Forests (RFs) as accurate alternatives to Support Vector Machines (SVMs). Both techniques were shown to achieve high accuracies when applied to various classification problems such as hand shape recognition and gesture recognition.

ANNs were demonstrated to be excellent classifiers, achieving accuracies of over 90% in all but one of the studies discussed [47] where it achieved an accuracy of 87% which is still clearly a very high accuracy. Random Forests were also demonstrated to be excellent classifiers, achieving high accuracies of 97.4% in [34] and 87.4% in [47] and a low error rate of 7.72% in [1]. Thus, their selection for a comparison with the SVMs in the context of hand shape recognition using Li's feature extraction procedure is justified.

It was also made clear that various classifiers may be better or worse-suited to classification using specific features. As such, it is crucial to carry out a comparison in order to determine the optimum classifier in each context separately, as is the case in this research.

Finally, it was demonstrated in studies [33] and [1] that the selection and optimization of the hidden layers of ANNs is a process of trial and error. This approach is used in this research to train a ANN in a subsequent chapter.

The next chapter discusses the image processing techniques required for the extraction of features and the machine learning techniques used for hand shape recognition.

Chapter 3

Techniques for Hand Shape Recognition

This chapter consists of two sections Image Processing Techniques for Hand Shape Recognition 3.1 and Machine Learning Techniques 3.2. Section 3.1 provides a theoretical background on the key image processing techniques which are integral to the extraction of features necessary for hand shape recognition. Section 3.2 gives background knowledge on the three machine learning techniques and discusses how they are used for classification.

3.1 Image Processing Techniques for Hand Shape Recognition

Image Processing is the analysis and/or manipulation of images or video frames in digital format to extract useful information from them. In the case of this research, images are processed to extract features for hand shape recognition.

The following image processing techniques discussed are: Canny edge detection, face detection, adaptive skin detection, background subtraction using Gaussian Mixture Models, hierarchical Chamfer matching, connected components analysis and CAMShift tracking. Each of these techniques is discussed in a separate subsection below.

3.1.1 Canny Edge Detection

Edge detection is the process of finding the edges within an image. An edge is defined as a point in an image with a discontinuity in brightness, or, in simple terms, a sharp change in brightness [4]. Edge detection simplifies an image representation to that of only its structural appearance-based information. Canny developed the Canny Edge detection technique [13] in 1986 and it is one of the most popular and robust edge detection techniques [56]. The Canny algorithm strives to meet the following three criteria:

1. A low error rate: The detection of edges should be as accurate as possible. The edges found in an image should not be falsely overlooked because omission of these edges could affect a system's performance.
2. Good localization: The distance between detected edge pixels and the actual edge pixels must be minimized.
3. Minimal response: Multiple responses to an edge should be avoided by limiting detection to only a single response per edge.

The Canny edge detection algorithm involves four steps [37]. These are: smoothing the image using a Gaussian filter; computation of the gradients in the image to highlight potential edges; applying non-maximum suppression to achieve thin edges; and double thresholding to suppress edge streaks. These steps are explained in the subsections below .

3.1.1.1 Smoothing the Image Using a Gaussian Filter

The initial step of Canny edge detection involves mitigating any excess noise in an image. Images generally contain some amount of noise. These sources of noise can easily, but mistakenly, be detected as edges–sharp changes in brightness–within the image.

As such, the image is smoothed by means of a Gaussian filter [67]. This involves convolving a Gaussian kernel K with the image I. Below is an example of a Gaussian kernel of size 5×5 using a standard deviation of $\sigma = 1.4$ which can be used, but larger kernels can be used as well.

$$K = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}$$

3.1.1.2 Computation of the Image Gradients

Once the image has been smoothed and excess noise has been filtered out from it, the next step is to determine the intensity gradients of the image. These gradients are computed because they give an indication of the strength of edges in the image. At each pixel in the smoothed image, the gradients are determined using the Sobel operator as follows.

The gradients are approximated using a pair of 3×3 convolution masks, S_x and S_y. S_x highlights the edges in the x-direction while S_y highlights the edges in the y-direction. These convolution masks are given as:

$$S_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \tag{3.1a}$$

$$S_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} \tag{3.1b}$$

Convolving the two masks with the original image results in two gradient images G_x and G_y. The Equation 3.2 below is then computed at each pixel (i, j) to find the gradient strength at that pixel using the law of Pythagoras:

$$|G(i,j)| = \sqrt{G_x(i,j)^2 + G_y(i,j)^2} \tag{3.2}$$

A simpler measure can also be used to approximate the gradient at pixel (i, j) in the form of the Manhattan distance measure given by:

$$|G(i,j)| = |G_x(i,j)| + |G_y(i,j)| \tag{3.3}$$

The direction of the edge θ is also computed at each pixel (i, j). The exact direction of the edge is determined using the following equation:

$$\theta(i,j) = \arctan\left(\frac{|G_x(i,j)|}{|G_y(i,j)|}\right) \tag{3.4}$$

The calculated direction θ of the edge is then rounded off to the nearest 45° angle representing the directions of the horizontal and vertical neighbours and those of the two diagonal neighbours. As such, it is rounded of to one of four possible angles: 0°, 45°, 90° or 135°.

3.1.1.3 Applying a Non-maximum Suppression

Once the direction and magnitude of the edges have been determined, a non-maximum suppression is applied to thin out edges by discarding non-maximum pixels in each edge. This results in accurate thin edges in the image, as required.

This is achieved by examining the gradient values of the neighbours on either side of each pixel (i, j) in the direction perpendicular to the direction of the pixel. If the gradient value of the pixel is greater than that of both neighbours, it is marked as being an edge pixel. If it is not, it is discarded i.e. set to 0.

For example, if the gradient direction for a pixel $\theta(i, j)$ is 0, meaning that it is North-South aligned, it is compared to the two neighbours on either side in the East-West direction. If its gradient value is greater than that of these neighbours, it is marked as an edge pixel. If not, it is set to 0.

3.1.1.4 Double Thresholding

After the non-maximum suppression has been applied, a double threshold is used to eliminate false edges which can cause features such as edge streaks. The double threshold consists of an upper and lower threshold. The steps below are followed to complete the edge detection process:

1. The upper threshold is applied to identify all ‘strong’ edges. A pixel is considered a ‘strong’ or confirmed edge pixel if the gradient value of that edge exceeds the upper threshold.

2. The lower threshold is applied to identify all ‘weak’ edges. A pixel is considered a ‘weak’ or rejected edge pixel if the pixel gradient is below the lower threshold. Such edges are discarded.

3. All pixels that have a gradient value between the upper and lower threshold are considered as edge pixels if they are connected to a strong edge pixel in a 3×3 neighbourhood area.

4. If pixels with a gradient value between the upper and lower threshold are not connected to a strong edge pixel in a 3×3 neighbourhood, but are connected to at least one other pixel that has a gradient value between the upper and lower threshold in the same neighbourhood area, the previous step is repeated with a 5×5 neighbourhood. If no strong edges are found in this expanded area, the edge is discarded.

Canny recommended a double threshold ratio (upper:lower) of between (2:1) and (3:1) [13]. Figure 3.1 provides an example of an image to which the Canny edge detection algorithm has been applied.

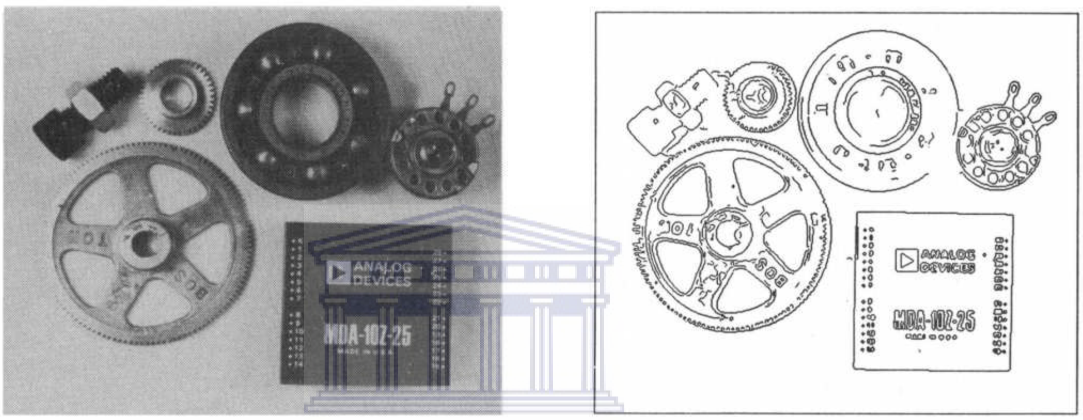


FIGURE 3.1: Canny edge detection: (a) The original image. (b) Application of the Canny edge detection algorithm [13].

3.1.2 Face Detection

The Viola-Jones [65] framework is a very popular framework for object detection. The framework has been applied to face detection and it has proven to be highly accurate and computationally efficient [66, 68, 70].

The Viola-Jones object detection framework classifies objects in images using simple fundamental features called Haar-like wavelets. It additionally uses a novel data structure called an Intergral Image to significantly speed up the detection of these features. Finally, a modified Adaboost classifier is used to arrange a series of weak classifiers trained to detect various Haar-like features into a rejection cascade. This setup results in a strong and highly efficient object detector.

The following subsections describe each of these steps, namely: the nature and computation of haar-like features; the use of an integral image to speed up the computation of haar-like features; the use of Adaboost to select appropriate features for face detection; and the use of a final rejection cascade as a face detector.

3.1.2.1 Haar-Like Wavelet Feature Detection

The object detection approach of the Viola-Jones algorithm makes use of features that are based on the principle of Haar wavelets called Haar-like wavelet features. Haar-like wavelets consist of a set of alternating rectangles of the same size and shape that are either "light" or "dark", and are either vertically or horizontally adjacent. Figure 3.2 illustrates two-rectangle, three-rectangle and four-rectangle features.

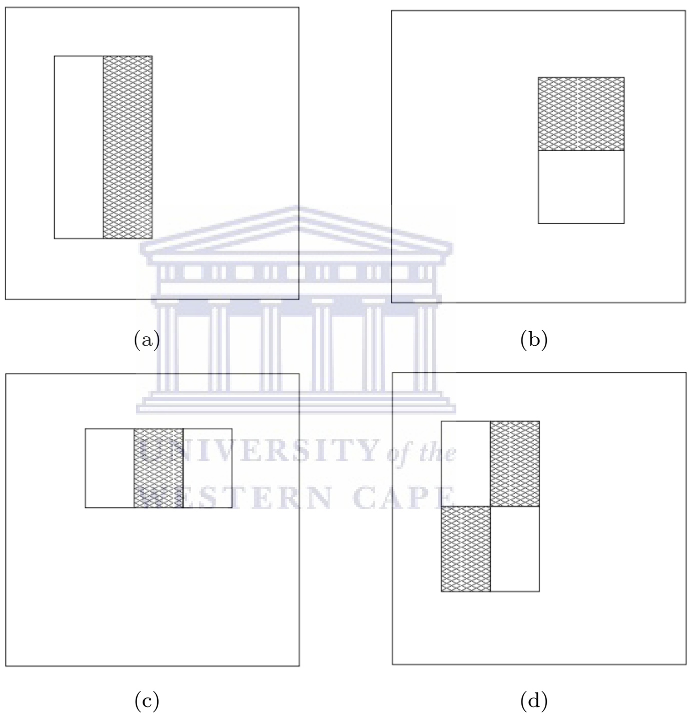


FIGURE 3.2: Three types of Haar-like wavelet features used by the Viola-Jones face detector [66].

Each type of feature is passed over a target image at various scales and positions. At each scale and position, the sum of the pixels corresponding to the dark region are subtracted from the sum of the pixels corresponding to the light region. If the result of this computation exceeds a threshold value, this specific feature is determined to be present at this location and scale.

Two-rectangle features are calculated by computing the sum of all the pixels in the dark region and subtracting these from the sum of all pixels in the light region and applying an acceptance threshold to the result. Three-rectangle features are computed by applying an acceptance threshold to the difference between the combined sum of the pixels in the two light rectangles and the dark rectangle. Four-rectangle features are calculated by applying an acceptance threshold to the difference between the combined sum of the pixels in the diagonal pairs of rectangles.

3.1.2.2 The Use of An Integral Image to Compute Haar-Like Features

Computing the values of various features at every scale and position in an image is a very computationally expensive operation. Viola and Jones proposed an intermediate representation of an image called an Integral Image which enables the rapid computation of the sums of various features at any scale and position in the image.

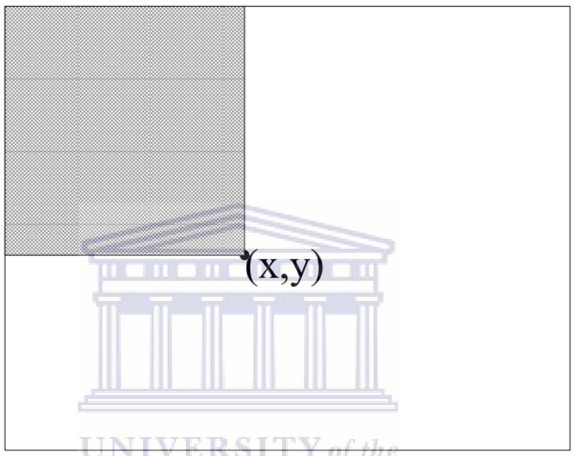


FIGURE 3.3: Computation of the Integral Image: The value of the Integral Image at (x, y) is the sum of all pixels to the top-left of the pixel, in the shaded region [66].

Given an image I, the integral image representation G at any position (x, y) is the sum of the pixels to the top-left of (x, y), as shown in Figure 3.3, given by:

$$G(x, y) = \sum_{i \leq x, j \leq y} I(i, j) \tag{3.5}$$

An alternative definition of the Integral Image is given in terms of the cumulative row sum $S(x, y)$ at (x, y) as the following pair of recurrence relations which can be used to compute the image in a single pass:

$$G(x, y) = G(x - 1, y) + S(x, y) \tag{3.6a}$$

$$S(x, y) = S(x, y - 1) + I(x, y) \tag{3.6b}$$

where

$$S(x, -1) = 0 \tag{3.6c}$$

and

$$G(-1, y) = 0 \tag{3.6d}$$

Using the Integral Image, it is possible to compute any Haar-like feature using only a few lookups in the image by easily computing the sum of any rectangle in the original image, as required. Referring to Figure 3.4, it is possible to compute the sum of the pixels inside the rectangle labeled D by subtracting the Integral Image value at point 4 from the sum of the Integral Image values at points 2 and 3, and adding back the Integral Image value at point 1 to counteract the excess caused by the intersection of rectangles $(A + B)$ represented by point 2 and rectangles $(A + C)$ represented by point 3.

The ability to compute the sum of pixels in any rectangle implies the ability to compute any Haar-like feature at any scale or location.

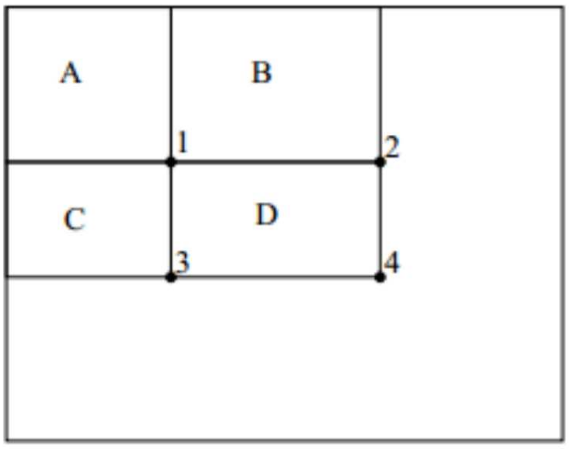


FIGURE 3.4: An example of the computation of the integral image [66].

3.1.2.3 The Use of AdaBoost to Select Haar-Like Features

AdaBoost is a learning algorithm which improves the classification performance of weak classifiers. A modified version of the algorithm is used by the Viola-Jones face detection system to choose an optimal subset of the potentially large number of features and train a classifier based on these features [65].

Even though each feature can be computed at a high speed, the computation of the set of features can be very slow since there are a large number of rectangular features associated with each image sub-window. Only those features are selected which best distinguish between positive and negative examples, thus limiting the number of features that are required to achieve a strong classifier.

3.1.2.4 A Rejection Cascade of Weak Feature Classifiers

A rejection cascade of classifiers is constructed in such a manner as to achieve a high accuracy while significantly lowering the computational cost for negative examples. The principle behind this idea is that simpler, and thus faster, boosted classifiers can be created to reject most of the negative sub-windows while still being able to detect almost all of the positive instances.

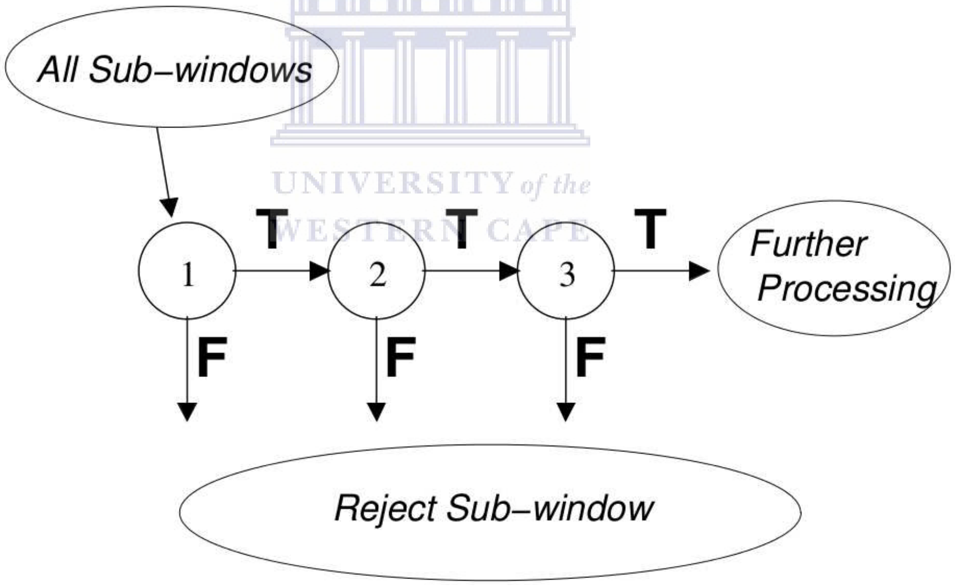


FIGURE 3.5: The typical structure of a rejection cascade [66].

The rejection cascade has the structure of a degenerate decision tree and it is depicted in Figure 3.5. With reference to Figure 3.5, when the first classifier obtains a positive result, it triggers the evaluation of the second classifier, and a positive result from the second classifier triggers the third classifier. As long as every classifier returns a positive result, this process continues on to the final classifier, after which a face is determined to have been detected in the sub-window in question.

On the other hand, if the result is negative at any classifier, the sub-window is immediately rejected. This significantly reduces the computational overhead of the algorithm for sub-windows in which no face exists.

3.1.2.5 Evaluation of the Face Detection System

The Viola-Jones face detection system was evaluated on the MIT+CMU frontal face dataset [55]. Some examples of the dataset with face detection performed on them are shown in Figure 3.6. The evaluation aimed to measure the speed as well as the accuracy of the technique. The system was shown to achieve a real-time detection speed of 15 frames per second (fps) on images with a resolution of 384×288 pixels when operating on a 700 MHz Intel Pentium III computer. The system achieved an accuracy of 93.9% with only 167 false detections.

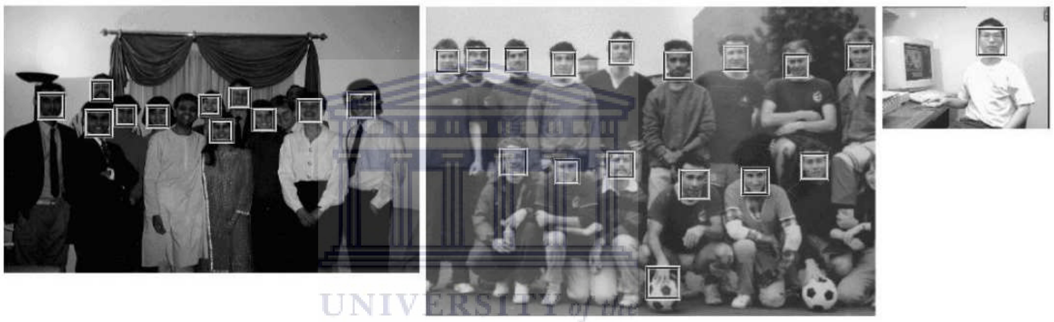


FIGURE 3.6: Example of the testing data from the MIT+CMU dataset [55].

3.1.3 Adaptive Skin Detection

Skin detection is an image processing technique which segments skin pixels from non-skin pixels. It eliminates all non-skin pixels in an image and highlights only the skin pixels in the image. Applications of skin detection include human-computer interaction, human detection, hand tracking, face detection and face recognition [17, 27, 36]. Skin detection in this research assists in initializing and maintaining the hand tracking algorithm to track the hands of the user. The adaptive skin detection algorithm used was initially proposed by Achmed [2] and used in the feature extraction procedure of Li [36].

The procedure works as follows: the face is detected; the skin colour distribution of the user is extracted from the face; it is back projected onto the original image to obtain a skin probability distribution; finally, the skin probability distribution is thresholded to obtain a binary skin map of the original image. Each step of this procedure is explained in further detail in the following subsections.

3.1.3.1 Face Detection

The Viola-Jones face detection algorithm is used to determine the position of the face. A 10×10 pixel area at the centre of the detected facial frame is extracted and used as a representative skin colour distribution in the form of a histogram. Achmed showed that this region represents the skin colour very well as it is usually void of non-skin obstructions such as shadows, hair, eyes and spectacles [2]. The 10×10 pixel area of the nose is converted from the default Red, Green and Blue (RGB) colour space to the Hue, Saturation and Value (HSV) colour space. A histogram of the Hue and Saturation channels of the region is computed and taken as the representative skin colour distribution of the user.

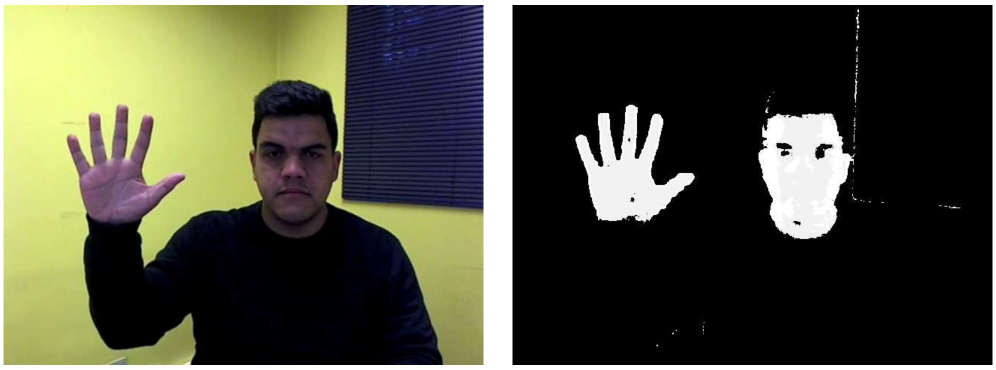
3.1.3.2 Histogram Back Projection and Thresholding

The skin colour histogram is back-projected onto the original input frame resulting in a skin probability distribution of the input frame. The back-projection is achieved as follows. Given C represents the colour of a pixel in the image, and F is the probability that the pixel is skin, $P(C|F)$ is the probability of drawing that colour when the pixel is actually skin. Then $P(F|C)$ is the probability that the pixel is skin given its colour. This yields the following equation:

$$P(F|C) = \frac{P(F)}{P(C)}P(C|F) \tag{3.7}$$

The resulting back-projected skin probability image is converted into a binary image in which skin pixels have a value of 255 (white) and non-skin pixels have a value of 0 (black). This is achieved by thresholding the image using a threshold value of 60. This static threshold value was determined as being optimum by Brown [12] An example of a back-projected image is illustrated in Figure 3.7.

As seen in the figure, this technique effectively segments skin pixels from non-skin pixels. There are, however, factors such as background noise or colours in the background which are similar to that of skin colour which can cause noise in the image. To this effect, background subtraction in the form of Gaussian Mixture Models, described in the next section, are used to mitigate such sources of noise.



(a) Original image (b) Skin-detected image

FIGURE 3.7: a) Original image and b) Skin-detected image.

3.1.4 Background Subtraction Using Gaussian Mixture Models

Background subtraction is the segmentation of objects/regions in an image or a sequence of video frames that are of interest to an application, referred to as the foreground, from those that are not of interest, referred to as the background [57]. In the current case, the foreground consists of the hand of the user, while all other objects in the frame constitute the background.

Gaussian Mixture Models (GMMs) are a probabilistic method that can be used for effective background subtraction. They can be used to highlight moving pixels in a frame with a history indicator over a set number of frames such that the brightness of a pixel indicates the recency of its motion, and regions with no motion over a number of frames appear as completely black.

Given an image sequence I, the history of a pixel at (i, j) at a specific time t can be represented as follows:

$$\{I_1, \ldots, I_t\} = \{I(i, j, x) : 1 \leq x \leq t\} \tag{3.8}$$

Each pixel can be modeled as a mixture of k Gaussian distributions. Letting $W_{x,t}$ represent the weight estimate of the x-th Gaussian, the probability of a pixel possessing the value I_t at time t can be expressed using the equation below:

$$P(I_t) = \sum_{x=1}^{k} W_{x,t} \times \eta(I_t, \mu_{x,t}, \Sigma_{x,t}) \tag{3.9}$$

where $\eta(I_t, \mu_{x,t}, \Sigma_{x,t})$ is the normal distribution of the x-th Gaussian component with a mean of $\mu_{x,t}$ and expressed as:

$$\eta(I_t, \mu_{x,t}, \Sigma_{x,t}) = \frac{1}{(2\pi)^{\frac{n}{2}} \mid \Sigma_{x,t} \mid^{\frac{1}{2}}} e^{\frac{-1}{2}(I_t - \mu_{x,t})^T \Sigma_{x,t}^{-1}(I_t - \mu_{x,t})} \tag{3.10}$$

where $\Sigma_{k,t} = \sigma_{k,t}^2 \mathbf{I}$ is the covariance of the k-th Gaussian component given $\mathbf{I}$ is the identity matrix.

A fitness value $\frac{W_{x,t}}{\sigma_{x,t}}$ is used as a reference when ordering the number of distributions k and the first M distributions are used for modeling the background scene, where the estimate of M is given by:

$$M = \text{argmin}_m (\sum_x^m W_{x,t} > T_h) \tag{3.11}$$

where T_h is the threshold that represents the minimum portion of the background model.

Given an updated background, foreground detection is then achieved by labeling all pixels which are determined to be more than a standard deviation of 2.5 away from any of the M distributions as foreground pixels. If there is a match between the test value and the x-th Gaussian component $W_{x,t}$, it is updated as shown below:

$$W_{x,t} = W_{x,t-1} \tag{3.12a}$$
$$\mu_{x,t} = (1-\rho)\mu_{x,t-1} + \rho I_t \tag{3.12b}$$
$$\sigma_{x,t}^2 = (1-\rho)\sigma_{x,t-1}^2 + \rho(I_t - \mu_{x,t})^T(I_t - \mu_{x,t}) \tag{3.12c}$$
$$\rho = \alpha\eta(I_t \mid \mu_k, \Sigma_k) \tag{3.12d}$$

where $\frac{1}{\alpha}$ is defined as the time constant which determines change. If there is no match between the Gaussian component and the test value, then it is updated as follows:

$$W_{x,t} = (1-\alpha)W_{x,t-1} \tag{3.13a}$$
$$\mu_{x,t} = \mu_{x,t-1} \tag{3.13b}$$
$$\sigma_{x,t}^2 = \sigma_{x,t-1}^2 \tag{3.13c}$$

If the test value does not match any of the Gaussian components, a new Gaussian component with a high variance, low weight parameter, and the test value as its mean replaces the Gaussian component with the lowest probability. An example of GMMs applied to highlight the moving foreground of an image is illustrated in Figure 3.8.

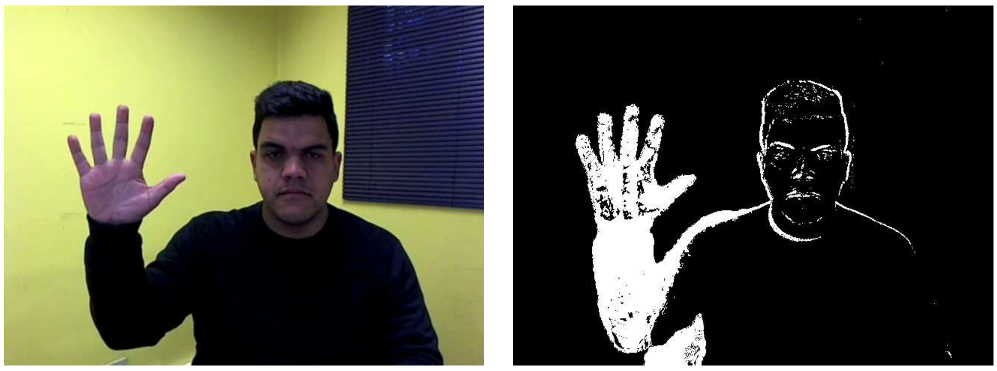


FIGURE 3.8: The application of Gaussian Mixture Models (GMMs) to achieve background subtraction: a) Original image and b) Background-subtracted image.

3.1.5 Hand Detection Using Hierarchical Chamfer Matching

The hierarchical chamfer matching technique is explained in this section [8]. It is a matching algorithm used to detect a template object in an image. In the case of this research, it is used to detect the location and size of the signer's hand and initialize the hand tracking algorithm. A template silhouette of the hand is used to find a match in the input image.

Chamfer matching involves three stages: computation of an edge image on the image in which the search is carried out; computation of a Chamfer distance transform on the image in which the search is carried out; and edge matching of the template edge image with the search image distance transform. Subsections 3.1.5.1 and 3.1.5.2 describe the computation of the Chamfer distance transform and the edge matching process, respectively.

A hierarchical approach can be used to significantly speed up the edge matching process. This is described in Subsection 3.1.5.3.

3.1.5.1 Computation of the Chamfer Distance Transform

A distance transform is an algorithm which converts an edge image into a distance image. Each non-edge pixel of the hand template silhouette image is given an intensity value ranging from 0 to 255. The intensity value is a measurement of the distance of the pixel to the closest edge pixel.

Various distance masks can be used to effectively calculate the distance image. Li showed that a 3×3 mask with a $(3, 4)$ distance transform produced excellent matching results.

The process of computing a distance transform involves two passes which are made over an image by propagating the computed distance values across the image like a wave. First a "forward" pass from left to right and from top to bottom is carried out, followed by a "backward" pass from right to left and from bottom to top. For an image V of size $W \times H$ pixels, a computation of the forward pass is given by:

$$V_{i,j} = \text{minimum}(V_{i-1}, V_{j-1} + 4, V_{i-1,j} + 3, V_{i-1,j+1} + 4, V_{j-1} + 3, V_{i,j}) \tag{3.14a}$$

$$\forall \quad i = \{2, \ldots, H\} \text{ and } j = \{2, \ldots, W\} \tag{3.14b}$$

The backward pass is given by:

$$V_{i,j} = \text{minimum}(V_{i,j}, V_{i,j+1} + 3, V_{i+1,j-1} + 4, V_{i+1,j} + 3, V_{i+1,j+1} + 4) \tag{3.15a}$$

$$\forall \quad i = \{H - 1, \ldots, 1\} \text{ and } j = \{W - 1, \ldots, 1\} \tag{3.15b}$$

The computation of the distance transform from an edge image provides a basis for template-based shape matching, even in conditions where the foreground image is unclear/noisy.

3.1.5.2 Chamfer Distance for Template Matching

Chamfer distance matching is the process of determining the position in the search image distance transform of greatest similarity to the template silhouette image. In the case of this research the template is a hand silhouette image which is created by combining skin and motion cues.

Template matching is achieved by passing the template silhouette over the search image distance transform column-wise and row-wise. At each position of the template over the search image, the sum of all distances corresponding to pixels in the search image that overlap with edges in the template is computed.

The summed value is known as the distance measure and the region with the smallest sum value is considered the closest matching position.

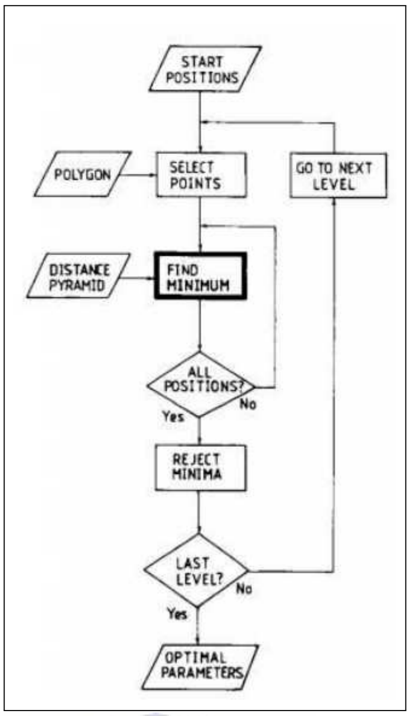


FIGURE 3.9: Flowchart of the Hierarchical Chamfer Matching Algorithm

3.1.5.3 Hierarchical Template Matching

Chamfer matching does a good job of detecting a target object if the size of the object in the search image is exactly the same as that of the target image. If the target object changes size in the search image, such as if the hand moves closer to or further away from the camera, or as is observed with variations in users, matching needs to be done at several different scales. Scanning the image at various scales can be very computationally expensive.

Hierarchical Chamfer matching offers a solution to this problem. It provides a coarse-to-fine resolution search using a pyramid of images at various resolutions to boost the chamfer matching process. This pyramid of images, also known as a resolution hierarchy, consists of multiple duplicates of the original search image at various resolutions.

Figure 3.9 depicts a flowchart of the Hierarchical Chamfer matching algorithm. Chamfer distance matching is initially executed on the lowest resolution image to obtain an approximation for the general region of the target object in the search image. The process is repeated on a higher-resolution image down the next level of the hierarchy, limiting the search in the new image only to the area determined in the previous level. This process is repeated until the search is performed on the original image to locate the target object.

The main advantage of using this approach is the reduction in computational cost, as the number of scans is strategically reduced.

3.1.6 Connected Component Analysis

Connected Component Analysis (CCA) is an algorithm for the detection and extraction of the contours of objects in an image [19]. The technique, also known as Connected Components labeling, passes over an image at a pixel-by-pixel level to search for all connected pixel regions. It can be performed on binary images, as well as grayscale images. Regions are said to be connected when adjacent pixels share the same set of intensity values V. In the case of a binary image, $V = \{255\}$. The 4-connectivity and 8-connectivity labeling operators are shown in Figure 3.10.

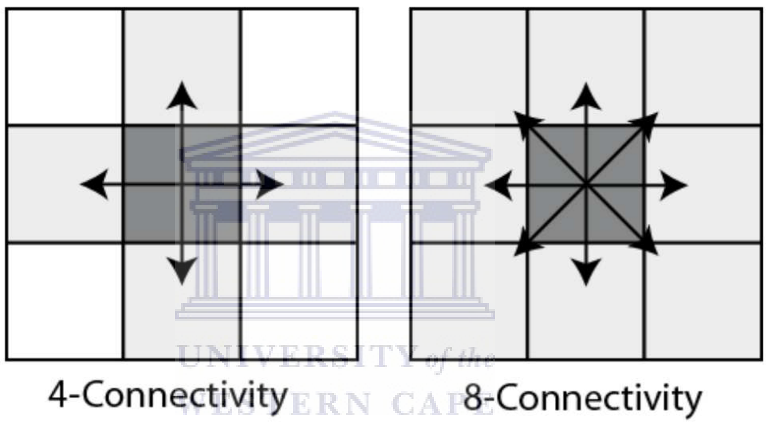


FIGURE 3.10: An example of the 8-connectivity labeling operator [16]

In order to compute the connected components of a binary image using the 8-connectivity operator, each pixel p that has an intensity value $V = 255$ is scanned and labeled. If $V = 255$ for the current p, the 8 neighbours of p which have been encountered before in the scan are examined and p is labeled using the following criteria:

1. If all the neighbours of p are of the intensity value 0, then assign a label q.
2. If all of the neighbours of p possess the value 255, then assign a label p.
3. If more than one of the neighbours have the intensity value of 255, assign the label of one of the neighbours to p and keep track of the equivalences.

Once the scan has been completed, a secondary pass is carried out to replace each label resulting from the first pass with its equivalent class label. Pixels labeled p are considered as foreground. Connected foreground blobs are then labeled as separate foreground objects, each with a unique index.

3.1.7 CAMShift

CAMShift stands for Continuously Adaptive Mean Shift and it is a modified version of the Mean-shift algorithm [10]. It is a colour-based tracking technique that is accurate, yet simple and computationally efficient. The algorithm is capable of tracking colour objects in real-time and does so efficiently.

CAMShift is a robust non-parametric technique which climbs the density gradients in order to find the mode/peak of the probability distribution. In this case the object to be found is the skin colour of the user's hand which is to be tracked in the video sequence.

The algorithm performs well in noisy environments since it has the ability to find and track the mode of a dynamically changing probability distribution. It is also very effective in overcoming transient occlusions such as when the hand passes over the face. This is attributed to the fact that the search window usually first absorbs the occlusion but then reverts to the dominant distribution when the occlusion passes.

A probability distribution is used to represent the pixel data of a video sequence. Each pixel $I(u, v)$ at location (u, v) in a frame is assigned a probability value $P(u, v)$ which represents the likelihood that the pixel belongs to the target.

The object to be tracked has to explicitly specified once to initialize the algorithm. Li used hierarchical Chamfer matching to find the hand and initialize the algorithm. A 1D histogram is computed and used as the model of the desired object to be tracked. The Hue channel of the HSV colour space is used in this computation. Depending on the range of the hue in the histogram, the probability value $P(u, v)$ is assigned a value in the range $[0, 1]$.

The probability distribution for the algorithm is computed within a search window, rather than on the entire image, to improve performance. Ideally, the search window is small enough to realize greater computational efficiency, but large enough to capture the motion of the object in any direction.

The histogram is used as a lookup-table. After determining the probability distribution $P(u, v)$, the maximum of the distribution is located. The location of the maximum represents the focal point of the target object in the actual frame. To calculate the maximum probability within the search window, statistical moments of the zeroth- and first-order are used.

Letting the probability distribution be σ, a statistical moment of order p and q can generally be formulated as [10]:

$$m_{pq} = \sum_{(u,v)\in\sigma} P(u,v)\cdot u^p \cdot v^q \tag{3.16}$$

The zeroth moment m_{00} is therefore given by:

$$m_{00} = \sum_{(u,v)\in\sigma} P(u,v) \tag{3.17}$$

This corresponds to the integral over the distribution . Similarly, the moments of first order are given by:

$$m_{10} = \sum_{(u,v)\in\sigma} P(u,v)\cdot u \tag{3.18a}$$

$$m_{01} = \sum_{(u,v)\in\sigma} P(u,v)\cdot v \tag{3.18b}$$

The position of the centre of the target object $L = (L_x, L_y)$ is then calculated as:

$$L_x = \frac{m_{10}}{m_{00}} \tag{3.19a}$$

$$L_y = \frac{m_{01}}{m_{00}} \tag{3.19b}$$

Once the location of the target object has been found and the location of the search window has been updated, the new size (w_s, h_s) of the search window is determined for the next frame. To achieve this, the moments of the zeroth-order and the maximum value of the distribution P_{max} are used as follows:

$$w_s = s\cdot\sqrt{\frac{m_{00}}{P_{max}}} \tag{3.20a}$$

$$h_s = 1.2\cdot w_s \tag{3.20b}$$

3.2 Machine Learning Techniques

This section discusses the three machine learning techniques which are compared in the context of SASL hand shape recognition. The three machine learning techniques which are used are Support Vector Machines (SVMs), Artificial Neural Networks (ANNs) and Random Forests (RFs). This section provides a base of understanding on each of these machine learning techniques, focusing on the mechanism of classification of data used in each case. The section is organized into four subsections. Subsection 3.2.1 discusses Support Vector Machines, Subsection 3.2.2 discusses Artificial Neural Networks and Subsection 3.2.3 discusses Random Forests.

3.2.1 Support Vector Machines

A Support Vector Machine (SVM) is a supervised machine learning technique which is comprised of a group of statistical learning models. Vapnik [14] initially introduced the SVM as a binary classification technique and it was later adapted for multi-class classification problems. SVMs have been used extensively within the SASL research group to recognize various SASL parameters including: hand shape in [36], hand location in [2], hand motion in [3], hand orientation in [36] and facial expressions in [44, 69]. These parameters are necessary for the recognition of SASL and each of the systems mentioned achieve very encouraging accuracies. In this regard, SVMs have been shown to be accurate, robust and easy to use.

This discussion on SVMs is divided into three parts: Subsection 3.2.1.1 discusses the underlying principle behind classification by SVMs; Subsection 3.2.1.2 mentions a variety of kernels that can be used with SVMs; and Subsection 3.2.1.3 describes prominent techniques used to achieve multi-class classification using SVMs.

3.2.1.1 Support Vector Machine Classification

Consider a Cartesian plane with a set of points shown in Figure 3.11. The figure depicts a two-class classification problem. The red and blue points on the graph belong to two separate classes. The general goal of classification is to find a hyperplane that separates the two classes of points. Thus, the classification problem entails drawing a hyperplane between the points of the two classes. It can be seen in the figure that many different separating hyperplanes can be placed between the two classes to separate them. The key idea of SVM classification is to find a hyperplane that ensures the biggest gap or "maximum margin" between the classes. This hyperplane is referred to as the "optimal hyperplane" and is depicted in Figure 3.12.

A decision rule is formulated below to conform to the decision boundary which determines where a class lies in feature space. Let the set of N points in Figure 3.12 be $X = \{x_i | i = 1, \dots, N\}$. The points in X can be assigned to one of two classes, a positive

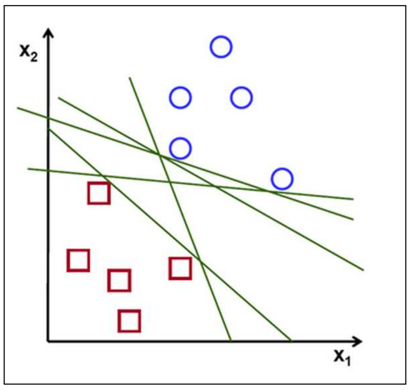


FIGURE 3.11: A two-class classification problem and the various hyperplanes that can be used to separate the two classes [48].

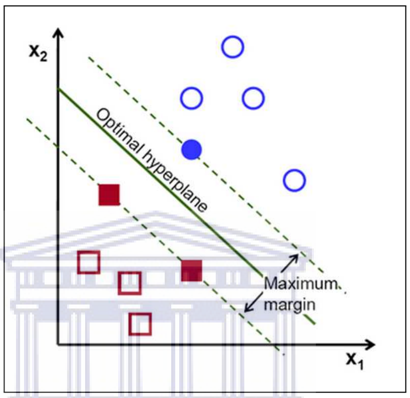


FIGURE 3.12: The optimal hyperplane for separating two classes [48].

class C^+ and a negative class C^-, respectively corresponding to the blue and the red classes in the figure. In the simplest terms, given an arbitrary point x that forms a vector $\bar{u}$ from the origin and lies anywhere in the cartesian plane, the problem of assigning a class to x amounts to determining on which side of the hyperplane the point lies. Let $\bar{w}$ be a vector perpendicular to the hyperplane, then for some constant C that depends on the specific optimal hyperplane [14]:

$$\bar{w} \cdot \bar{u} \geq C \quad \text{if} \quad x \in C^+ \tag{3.21a}$$

$$\bar{w} \cdot \bar{u} < C \quad \text{if} \quad x \in C^- \tag{3.21b}$$

Restructuring Equation 3.21 and introducing b such that $b = -C$ for convenience yields:

$$\bar{w} \cdot \bar{u} + b \geq 0 \quad \text{if} \quad x \in C^{+} \tag{3.22a}$$

and

$$\bar{w} \cdot \bar{u} + b < 0 \quad \text{if} \quad x \in C^{-} \tag{3.22b}$$

Assuming $\bar{x}_{+}$ and $\bar{x}_{-}$ to be arbitrary positive and negative samples, respectively, in X, a simple rescaling of $\bar{w}$ yields:

$$\bar{w} \cdot \bar{x}_{+} + b \geq 1 \tag{3.23a}$$

and

$$\bar{w} \cdot \bar{x}_{-} + b \leq -1 \tag{3.23b}$$

A variable y_i is introduced to simplify Equations 3.23a and 3.23b and arrive at a single generic equation for the positive and negative classes. The variable defines a set of labels $\{y_i | i = 1, \dots, N, y_i \in \{1, -1\}\}$ assigned to each point x_i, such that the point $x_i \in C^{+}$ if $y_i = 1$ and $x_i \in C^{-}$ if $y_i = -1$ for any $i \in \{1, \dots, N\}$. Multiplying y_i by either Equation 3.23a or Equation 3.23b yields exactly the same outcome as follows:

$$y_i(\bar{x}_i \cdot \bar{w} + b) \geq 1 \quad \forall \; x_i \tag{3.24}$$

For the specific points x_i that lie on the boundary of the margin on either side, referred to as support vectors, Equation 3.24 is expressed as:

$$y_i(\bar{x}_i \cdot \bar{w} + b) = 1 \tag{3.25}$$

The goal is to separate the positive and negative samples by the widest hyperplane possible. This requires for a formulation of the size of the margin D which can be expressed by determining a vector formed by taking the difference between a support vector of the positive class $\bar{x}_{+}$ and a support vector of the negative class $\bar{x}_{-}$ and projecting it onto a unit vector perpendicular to the separating hyperplane. This can be formulated as:

$$\begin{aligned} D &= (\bar{x}_+ - \bar{x}_-) \cdot \frac{\bar{w}}{||w||} \\ &= \frac{\bar{w} \cdot \bar{x}_+ - \bar{w} \cdot \bar{x}_-}{||w||} \end{aligned} \tag{3.26}$$

Substituting for $\bar{w} \cdot \bar{x}_+$ and $\bar{w} \cdot \bar{x}_-$ in Equation 3.26 using Equation 3.25 results in:

$$\begin{aligned} D &= \frac{1 - b + 1 + b}{||w||} \\ &= \frac{2}{||w||} \end{aligned} \tag{3.27}$$

As such, a maximization of the margin amounts to maximizing Equation 3.27 which amounts to minimizing the inverse of that equation subject to Equation 3.25 as follows:

$$\begin{aligned} \max \frac{2}{||w||} &\Longrightarrow \min \frac{||w||}{2} \\ &\Longrightarrow \min ||w|| \\ &\Longrightarrow \min \frac{1}{2}||w||^2 \end{aligned} \tag{3.28a}$$

subject to:

$$y_i(\bar{x}_i \cdot \bar{w} + b) = 1 \tag{3.28b}$$

Lagrange multipliers are used to maximize Equation 3.28a. Let L be the boundary to be maximized by subtracting Equation 3.28a from a summation of all the constraints found in Equation 3.25 as shown below:

$$L = \frac{1}{2}||\bar{w}||^2 - \sum_i^N \alpha_i[y_i(\bar{w} \cdot \bar{x}_i + b) - 1] \tag{3.29}$$

Differentiating L with respect to $\bar{w}$ yields the minimization expression:

$$\frac{\partial L}{\partial \bar{w}} = \bar{w} - \sum_{i}^{N} \alpha_i y_i \bar{x}_i = 0$$
$$\bar{w} = \sum_{i}^{N} \alpha_i y_i \bar{x}_i \tag{3.30}$$

This indicates that $\bar{w}$ is the linear sum of all of the samples in X along with their corresponding class labels. Differentiating L with respect to b yields the minimization expression:

$$\frac{\partial L}{\partial b} = -\sum_{i}^{N} \alpha_i y_i = 0$$
$$\sum_{i}^{N} \alpha_i y_i = 0 \tag{3.31}$$

The expression for $\bar{w}$ in Equation 3.30 can now be substituted back into equation 3.29 to obtain the following:

$$L = \frac{1}{2}\left(\sum_{i}^{N} \alpha_i y_i \bar{x}_i\right) \cdot \left(\sum_{j}^{N} \alpha_j y_j \bar{x}_j\right) - \left(\sum_{i}^{N} \alpha_i y_i \bar{x}_i\right) \cdot \left(\sum_{j}^{N} \alpha_j y_j \bar{x}_j\right) - b\sum_{i}^{N} \alpha_i y_i + \sum_{i}^{N} \alpha_i \tag{3.32}$$

Substituting the expression in Equation 3.31 into Equation 3.32 allows for the Lagrangian to be rewritten as follows:

$$L = \sum_{i}^{N} \alpha_i - \frac{1}{2}\sum_{i}^{N}\sum_{j}^{N} \alpha_i \alpha_j y_i y_j \bar{x}_i \cdot \bar{x}_j \tag{3.33}$$

The formulation for the discriminant of the optimal hyperplane is therefore:

$$f(\bar{x}) = \sum_{i \in V} \alpha_i y_i \bar{x}_i \cdot \bar{x} + b \tag{3.34}$$

where V is a set containing the indices of the support vectors in X and:

$$x \in C^{+} \quad \text{if} \quad f(x) \geq 0, \tag{3.35a}$$

$$x \in C^{-} \quad \text{if} \quad f(x) < 0 \tag{3.35b}$$

3.2.1.2 Kernel Functions

In cases where the data of the two classes are not linearly separable, the so-called "kernel trick" [14] can be used to successfully map data from the current space onto a higher-dimensional space in which the data is linearly separable. In this case, a kernel is used to achieve this mapping. There are many different kernel functions that can be used. Four common kernel functions which are based on Mercer's theorem [25] are as follows:

1. Linear Kernel: $K(\bar{x}, \bar{x}') = (\bar{x})^T \cdot (\bar{x}')$
2. Polynomial Kernel: $K(\bar{x}, \bar{x}') = (\gamma(\bar{x})^T \cdot (\bar{x}') + b)^d$
3. Radial Basis Function (RBF) Kernel: $K(\bar{x}, \bar{x}') = \exp(-\gamma(||\bar{x} - \bar{x}'||_2^2))$
4. Sigmoid Kernel: $K(\bar{x}, \bar{x}') = \tanh(\gamma(\bar{x})^T \cdot \bar{x}' + b)$

where γ, b and d are kernel parameters. The choice of kernel can affect the classification accuracy of a SVM. The choice of a specific kernel depends on the specific classification problem. However, several studies have concluded that the RBF kernel is the best-suited kernel to most classification problems [3, 36, 44, 47, 53]. As such, the RBF kernel is selected for use in this research. A comparison of other kernels may yield interesting results, but is outside the scope of this research and is left for future work.

3.2.1.3 Multi-class SVM Techniques

Support Vector Machines are, by definition, designed to handle binary classification problems. Binary classifiers are limited to solving only two-class problems. A comparative study in [26] describes three techniques that have been proposed to modify SVMs to handle multi-class classification. Most of these multi-class classification techniques involve the combination of several binary classifiers along with a strategic decision to choose a single class. The following subsections describe three of these techniques.

One-Against-All

Given an M-class problem, the problem is to separate the data points belonging to each class i from the data points of the remaining classes, where $i \in \{1, 2, \ldots M\}$.

In this approach, the data points of all the classes besides class i are combined to form a single class and a binary classifier with a label representation for the class i and a different label representation for the combination of the remaining classes is trained. This procedure is repeated for each of the classes $i \in \{1, 2, \ldots M\}$ and results in a total M binary classifiers.

Given an unknown pattern that requires classification into one of the M classes, the pattern is presented to each of the M classifiers. The class of the pattern is then taken to be the class that receives the maximum number of votes across all classifiers.

This technique is seen as inefficient because of the lengthy training and testing times as a result of possibly large datasets of points in each combination pair of classes.

One-Against-One

This technique is similar to the previous technique in that a series of binary classifiers are created. However, in this case, each classifier is trained to distinguish between two specific classes u and v, where $u \neq v$, for every distinct pair (u, v) using the data points associated with those classes.

The samples of class u are used as positive examples and those of class v, as negative examples. Each classifier is able to distinguish between these specific classes. This results in a total of $\frac{M(M-1)}{2}$ binary classifiers.

As with the previous technique, an unknown test pattern is presented to all the classifiers. The class that receives the largest number of votes across all the classifiers is resolved to be the class of the input pattern.

Directed Acyclic Graph Support Vector Machine

The Directed Acyclic Graph Support Vector Machine was first proposed by Platt *et al.* [50]. The training phase of the Directed Acyclic Graph (DAG) Support Vector Machine (SVM) is carried out according to the one-against-one technique and results in a total of $\frac{M(M-1)}{2}$ binary SVMs.

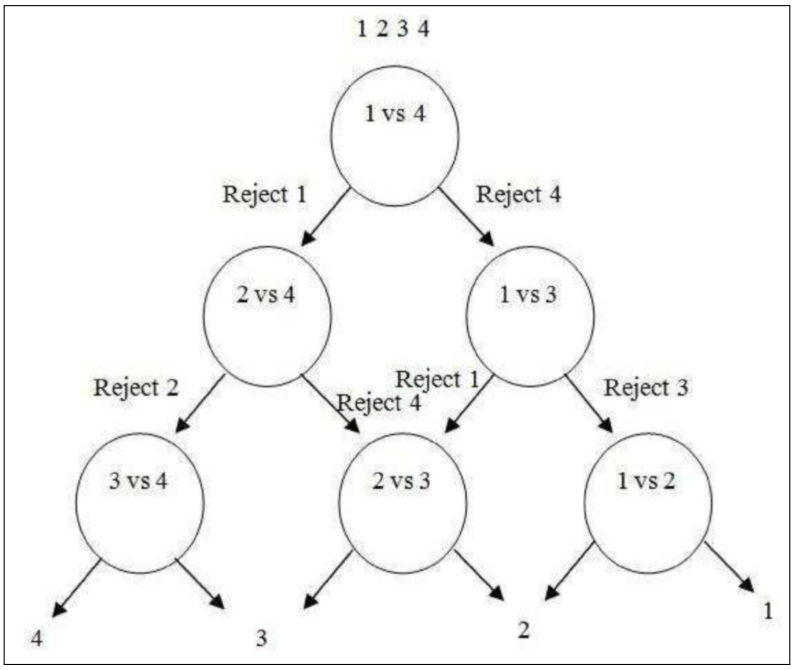


FIGURE 3.13: A Directed Acyclic Graph (DAG) of a 4-class problem.

Thereafter, a rooted binary directed acyclic graph consisting of $\frac{M(M-1)}{2}$ internal nodes and M leaves is used in the testing phase to classify an unknown test pattern. Each node in the graph is a binary SVM of the classes u and v.

Figure 3.13 illustrates a 4-class problem with the class $i \in \{1, 2, 3, 4\}$. Beginning at the root node, classes 1 and 4 are compared. If class 1 is determined to be the correct class, it is also important to note that class 4 was rejected and it is then resolved that classifiers involving class 4 will no longer be invoked.

This concept is propagated down into all the remaining nodes and at each stage one class is rejected, and the other, accepted. At the end of the process after $M - 1$ steps and at the bottom of the graph, only a single class remains which is taken to be the predicted class.

This technique combines the efficiency in training of the one-against-one technique and provides a better classification efficiency than the one-against-all technique.

3.2.2 Artificial Neural Networks

An Artificial Neural Network (ANN) is a group of interconnecting artificial neurons which mimic the biological neurons of the human brain. The ANN concept was first introduced by McCulloch and Pitts [40] who created a computational model of the concept in 1943. Rosenblatt then went on to create the first Perceptron in 1958 [54].

In 1969 Minsky and Papert [42, Pages 105–110] introduced an advanced version of the Perceptron called the Multi-Layer Perceptron (MLP).

The following subsections describe: the basic Perceptron in Subsection 3.2.2.1; various activation functions that are used in ANNs in Subsection 3.2.2.2; and the Multi-Layer Perceptron which is used in this research in Subsection 3.2.2.3.

3.2.2.1 The Perceptron

The Neural Network model is based on an over-simplified mathematical model of the biological neuron called a Perceptron. The basic computational unit of a Perceptron is a neuron [41, Pages 7–10]. A Perceptron consists of one or more neurons arranged in a specific pattern, hence the name artificial neural network, i.e. a network of neurons. The neurons can be organized into several layers, as explained in a subsequent section.

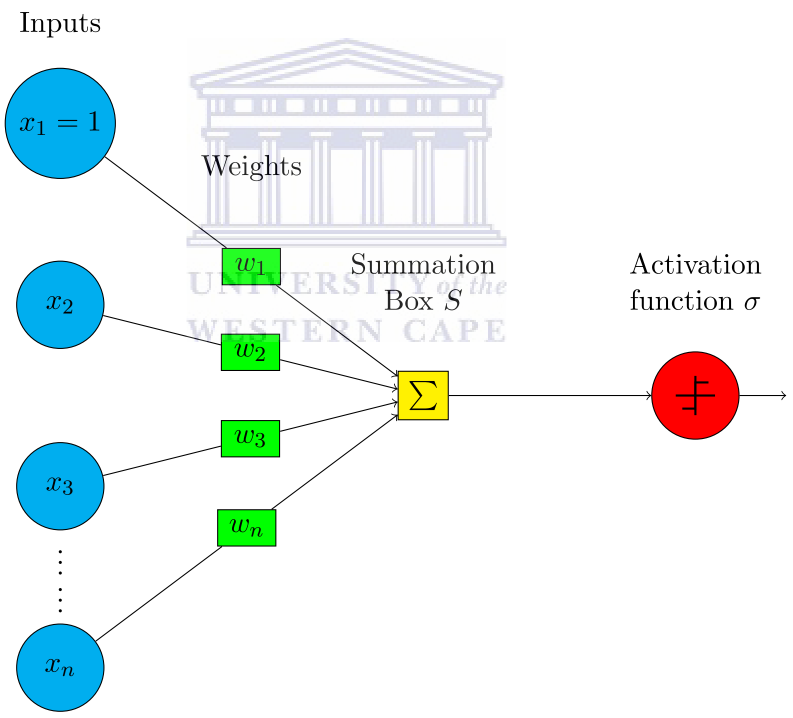


FIGURE 3.14: An example of a Perceptron.

A basic Perceptron structure consisting of a single neuron is illustrated in Figure 3.14. This Perceptron can be described as having a set of n input nodes $\{x_1, \ldots, x_n\}$, each of which link to a summation box S. A weighted sum S of the input nodes is calculated at the summation box, where a set of weights $\{w_1, \ldots, w_n\}$, each corresponding to each input node, are used to calculate this sum. The weighted sum is then used as input to a function σ, called an activation function, the output of which is taken as the output value of the neuron. In this case, the output of this neuron is also the output of the Perceptron.

The weighted sum S is given by:

$$S = \sum_{i=1}^{n} w_i x_i \tag{3.36}$$

There are a variety of activation functions that can be used. The classical Perceptron uses a basic step activation function σ which produces a binary output as follows [41, Pages 11–13]:

$$\sigma(S) = \begin{cases} 1 & if S \geq 0 \\ 0 & if S < 0 \end{cases} \tag{3.37}$$

3.2.2.2 Activation Functions

The choice in activation function is crucial to achieving a high-accuracy ANN. It depends on the specific classification problem at hand. The following are examples of activation functions σ besides the step activation function described previously:

- Linear Activation Function: The value of this activation function grows proportional to the value of the input. It is given by:

$$\sigma(S) = S \tag{3.38}$$

- Logistic/Sigmoid Activation Function: This activation function limits the value of the output to the specific range $[0, 1]$. As such, the output can be thought of as a probability measure. It is expressed as:

$$\sigma(S) = \frac{1}{1 + e^{-S}} \tag{3.39}$$

There is no known method of selecting one or other activation function. Selection of an appropriate function is a matter or trial and error. However, the Sigmoid activation function has been used extensively to solve a variety of classification problems with a high accuracy [30, 49, 52]. In this respect, it shows excellent promise for the hand shape classification problem in this research. As such, it is selected as the activation function for the ANN in this research.

3.2.2.3 Multilayer Perceptron

Practically speaking, the Perceptron is only capable of solving simple problems that are linearly separable. An example of a non-linear function that the Perceptron is unable to solve is the XOR function [43].

Adding extra layers to the Perceptron structure results in a Multilayer Perceptron (MLP) structure [7] which is able to solve non-linearly separable problems. A typical MLP consists of three layers, although a larger number of layers can also be used. It has an input and output layer like the basic single-layer Perceptron, but also has a hidden layer as shown in Figure 3.15. Note that the symbol $x_i^{(L)}$ in the figure represents the value of the i-th node in the L-th layer and $w_{i,j}^{(L)}$ represents the weight corresponding to the connection that flows from the i-th node in the L-th layer to the j-th node in the $(L+1)$-th layer.

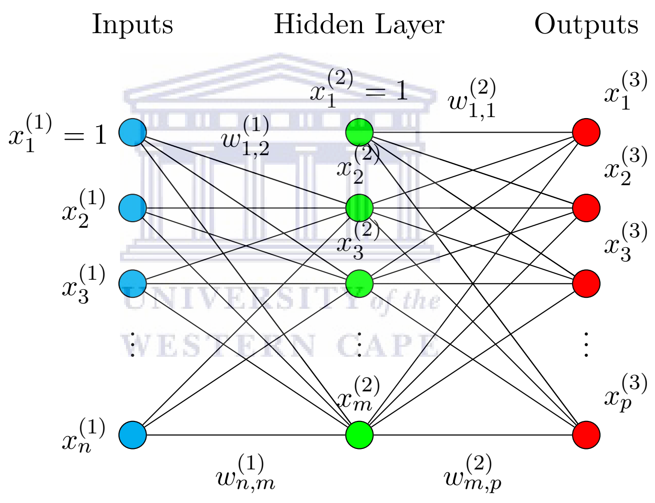


FIGURE 3.15: A Multilayer Perceptron example.

The MLP in the figure consists of n input nodes $\{x_i^{(1)} | i = 1, \ldots, n\}$, m hidden nodes $\{x_i^{(2)} | i = 1, \ldots, m\}$ and p output nodes $\{x_i^{(3)} | i = 1, \ldots, p\}$. It is important to note that some nodes and connector arrows have been omitted from the figure due to space constraints, but every node in a layer is connected to every node in the next layer, with the exception of the first node in the input and hidden layer which have a fixed value as follows:

$$x_1^{(1)} = 1 \tag{3.40a}$$

$$x_1^{(2)} = 1 \tag{3.40b}$$

The value of each node in the input layer is taken as the input value to that node. The value of each node $x_i^{(L)}$ in every other layer L is computed by means of applying the activation function σ to the weighted sum $S_i^{(L)}$ of all inputs to that node given by:

$$x_i^{(L)} = \sigma(S_i^{(L)}), \quad L \in \{2, 3\} \tag{3.41}$$

where the weighted sum $S_i^{(L)}$ is given by:

$$S_i^{(L)} = \sum_{j=1}^{C} x_j^{(L-1)} w_{j,i}^{(L-1)} \tag{3.42}$$

where C is the number of nodes in layer $L-1$. The activation function σ considered is the sigmoid function given by:

$$\sigma(S) = \frac{1}{1+e^{-S}} \tag{3.43}$$

Assuming that all the weights of the MLP are known i.e. a trained MLP model is available, propagating the input values of an unseen input sample through the MLP and receiving a prediction value at the output nodes requires a set of simple computations using the above equations. The problem, however, is to determine the weights given a training set—a set of input samples labeled with known output values. A technique called Backpropagation is the training method used to achieve this outcome [24].

Backpropagation is the application of gradient descent to ANNs. Given an input vector $\bar{u} = \{u_1, \ldots, u_n\}$, the ANN can be generally thought of as a function F that takes in a vector of weights $\bar{w}$ and the input vector $\bar{u}$ to produce a computed output vector $\bar{z}$ which consists of the outputs of the MLP $\{x_1^{(3)}, \ldots, x_p^{(3)}\}$ as follows:

$$\bar{z} = F(\bar{u}, \bar{w}) \tag{3.44}$$

Note that F may be a good or bad approximator of the actual function G that produces the vector of actual or required outputs $\bar{d} = \{d_1, \ldots, d_p\}$ given the same input vector $\bar{u}$ as follows:

$$\bar{d} = G(\bar{u}) \tag{3.45}$$

The problem is to determine the weight vector $\bar{w}$ such that the actual and computed outputs are very close, according some metric. In this case, the mean squared error (MSE) is used to provide the error P as follows:

$$P = \text{MSE} = \frac{1}{p}\sum_{i=1}^{p}(d_i - x_i^{(3)})^2 \tag{3.46}$$

Minimizing the difference between $\bar{d}$ and $\bar{z}$ amounts to minimizing P. Applying gradient descent to this function allows for the formulation of a generic update rule for any weight $w_{i,j}^{(L)}$, starting with some random value for that weight:

$$w_{i,j}^{(L)} \leftarrow w_{i,j}^{(L)} - \alpha\frac{\partial P}{\partial w_{i,j}^{(L)}} \tag{3.47}$$

where α is the learning rate, the optimum value of which is determined by trial and error. Consider a weight $w_{i,j}^{(2)}$ that connects the hidden layer to the output layer. A computation of the weight update rule requires the partial derivative of the function P with respect to this weight. The chain rule can be used to derive this partial derivative as follows:

$$\begin{aligned}\frac{\partial P}{\partial w_{i,j}^{(2)}} &= \frac{\partial P}{\partial x_j^{(3)}}\frac{\partial x_j^{(3)}}{\partial S_j^{(3)}}\frac{\partial S_j^{(3)}}{\partial w_{i,j}^{(2)}} \\ &= \frac{\partial P}{\partial x_j^{(3)}}\frac{\partial x_j^{(3)}}{\partial S_j^{(3)}}x_i^{(2)}\end{aligned} \tag{3.48}$$

Noting from Equation 3.41 that $x_j^{(3)} = \sigma(S_j^{(3)})$, the unknown partial derivative in Equation 3.48 is in fact the derivative of the activation function $\sigma(S)$ given by:

$$\sigma'(S) = \frac{d}{dS}(\sigma(S)) = \sigma(S)(1-\sigma(S)) \tag{3.49}$$

As such, Equation 3.48 becomes:

$$\begin{aligned}\frac{\partial P}{\partial w_{i,j}^{(2)}} &= \frac{x_j^{(3)} - d_j}{x_j^{(3)}(1-x_j^{(3)})}x_j^{(3)}(1-x_j^{(3)})x_i^{(2)} \\ &= (x_j^{(3)} - d_j)x_i^{(2)}\end{aligned} \tag{3.50}$$

The weight update rule for $w_{i,j}^{(2)}$ is then:

$$w_{i,j}^{(2)} \leftarrow w_{i,j}^{(2)} - \alpha(x_j^{(3)} - d_j)x_i^{(2)} \tag{3.51}$$

The same approach can be used to obtain a weight update expression for any weight $w_{i,j}^{(1)}$ that connects the input layer to the hidden layer. The partial derivative of the function P with respect to this weight can also be obtained using the chain rule, expanded and simplified as in the previous equations as follows:

$$\frac{\partial P}{\partial w_{i,j}^{(1)}} = \frac{\partial P}{\partial x_j^{(2)}} \frac{\partial x_j^{(2)}}{\partial S_j^{(2)}} \frac{\partial S_j^{(2)}}{\partial w_{i,j}^{(1)}} \tag{3.52a}$$

$$= (x_l^{(3)} - d_l)w_{j,l}^{(2)}u_j^{(2)}(1 - u_j^{(2)})u_i^{(1)} \tag{3.52b}$$

Using this expression, the update rule for $w_{i,j}^{(1)}$ is expressed as:

$$w_{i,j}^{(1)} \leftarrow w_{i,j}^{(1)} - \alpha \sum_{l=0}^{p} \left[(x_l^{(3)} - d_l)w_{j,l}^{(2)} \right] x_j^{(2)}(1 - x_j^{(2)})x_i^{(1)} \tag{3.53}$$

In practice, the values of all weights are initially chosen randomly and the backpropagation procedure is repeated either until the error in the predicted result of the ANN is smaller than a threshold, or until a maximum number of iterations is reached.

3.2.3 Random Forests

Random Forests (RFs) are an ensemble of decision trees which collectively form a forest [11]. Each group of decision trees votes for a class of some sample data and the class which receives the most votes from all groups of trees is said to be the predicted result. The underlying strategy behind this method is a technique called bagging which is the process of constructing a group of classifiers on different random subsets of an overall training dataset.

In the case of RFs, the classifiers used are decision trees. Decision trees, on their own, are very simple and poor-accuracy predictors and yield a high prediction variance [11]. Combining multiple decision trees, however, increases the prediction accuracy. In this respect, RFs are powerful classification techniques since the underlying classification principle used is simple, but a high prediction accuracy can be obtained.

A formal definition of RFs is given as follows [11]: a Random Forest R is a collection of B individual decision tree classifiers given by:

$$R = \{T_b(X, \Theta_b), b = 1, \ldots, B\} \tag{3.54a}$$

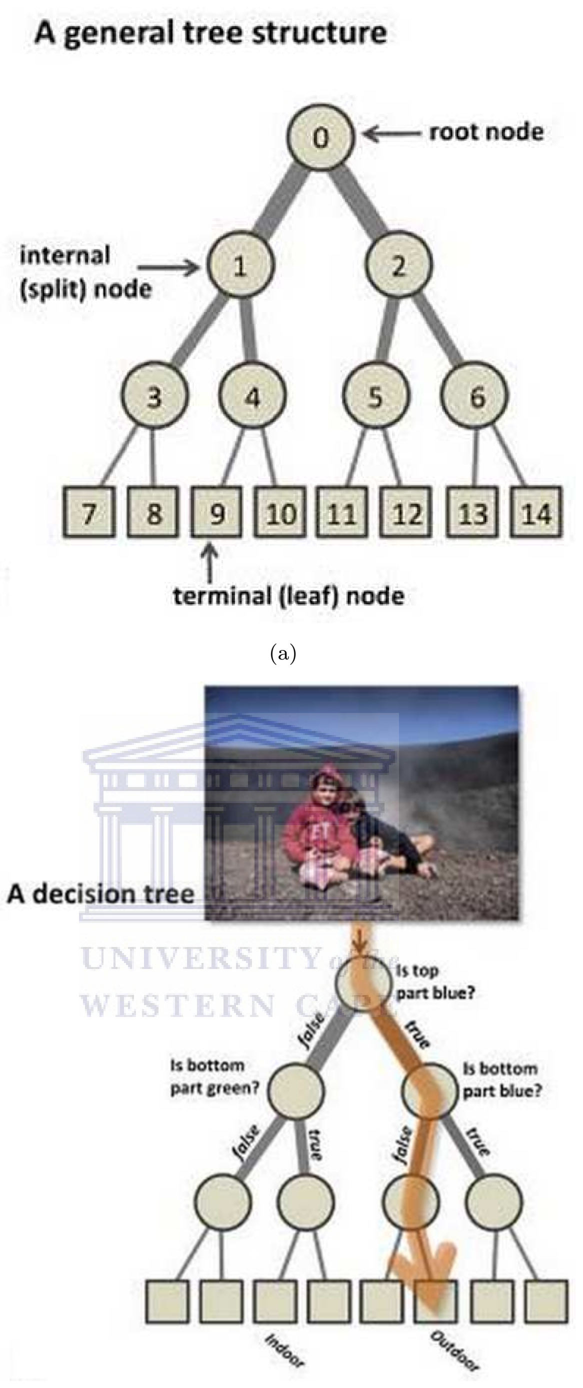
where Θ_k is a set of independently distributed samples used to construct a unique decision tree, T_b refers to the b-th tree in the forest, and each decision tree votes for the most popular class given input X. The prediction of the forest is the class that achieves a plurality across all decision trees. It is important to note that the prediction of decision trees is unweighted, meaning that the prediction of no individual decision tree is treated as any better than that of any other. This discussion on RFs is divided into two parts: Subsection 3.2.3.1 describes the underlying principle behind classification using a decision tree and Subsection 3.2.3.2 provides the algorithm used to construct a random forest given a set of training data.

3.2.3.1 The Decision Tree

It is important to understand what decision trees are before RFs can be discussed. A decision tree is a very simple classifier that consists of a set of classification questions about a given input organized into a hierarchy [15, 63]. When classifying a given input, the input is passed down the hierarchy of the decision tree, which amounts to asking a pre-defined set of classification questions about the known characteristics of the input in order to predict the nature of some unknown characteristic of the input. Each successive question depends on the answer of all previous questions asked in the hierarchy.

Figure 3.16a is a visualization of an abstracted decision tree and Figure 3.16b is an example of a decision tree that attempts to predict whether or not a given input image was taken outdoors or indoors.

Referring to Figure 3.16a, a decision tree consists of a set of nodes and edges which are organized into a hierarchy [15]. The tree structure is comprised of internal nodes, also called split nodes, and terminal nodes, also called leaf nodes. The internal nodes are represented by circles and terminal nodes appear as squares in Figure 3.16a. A split function is used at each internal node to determine the next node to which a given input should be redirected. The terminal node of a decision tree provides a predicted class output.



(b)

FIGURE 3.16: (a) The structure of a typical decision tree and (b) An example decision tree to predict whether an input image was taken indoors or outdoors[15].

Referring to the path highlighted in orange in Figure 3.16b, a photograph is received as input and undergoes a series of checks to determine an eventual class—indoor or outdoor. The first node performs a check to determine whether the top half of the input image is blue. This could possibly indicate that the sky is present in the top background. If the result of this check is true, the right sub-tree of the node is activated. The check at the next internal node deals with whether or not the bottom half of the photograph is (also) blue. If the result of this check is false, the decision tree produces the predicted result “outdoor scene”.

On their own, decision trees are very simple and poor classifiers [11]. However, grouped into a forest of a large number of decision trees, all querying and voting on various aspects of an input sample, they develop a strong classification characteristic.

3.2.3.2 Random Forest Algorithm

Consider a dataset consisting of N labeled points $\{(x_i, y_i)|i = 1, \ldots, N\}$ in which x_i refers to a single training example and y_i is the label corresponding to that example. Each x_i consists of p features that can be used in classification. A total of B decision trees are created using this training set.

For each tree, a random but uniform dataset of n samples with replacement from the original dataset of N samples is extracted using bootstrapping. Bootstrapping is the process of selecting unique random subset datasets from a main dataset and using these unique datasets to train different classifiers. p_s variables are selected from the total set of p variables available as candidates for splitting. At each node in the tree, p_s variables are selected at random from p and the best split positions in the tree on these variables are selected. Those nodes are split into two child nodes. This is repeated until the depth of the tree D_b is equal to a threshold $D_{\min}$. This entire procedure is repeated for each of the B trees, resulting in a RF $\{T_b|b = 1, \ldots, B\}$.

With all these symbols defined, this procedure can be summarized in algorithmic form as follows [23]:

Algorithm 1 Random Forest algorithm

```
1: for $b = 1$ to $B$ do
2:     Draw a bootstrap sample $Z^*$ of size $n$ from the training data to create tree $T_b$
3:     while Number of nodes in the current tree $D_b < D_{\min}$ do
4:         Select $p_s$ variables at random from the $p$ variables available
5:         Pick the best variable/split-point among the $p_s$ variables
6:         Split the node into two daughter nodes
7:     end while
8: end for
9: Output the ensemble of trees $\{T_b|b = 1, \ldots, B\}$
```

While the value of p_s can be optimized, in practice, using a value of $p_s = \log_2 p + 1$ has been shown to produce sufficiently accurate results [11]. Once the RF has been trained, classification takes place by determining the class with the largest number of votes from all the decision trees in the forest. Letting $\hat{C}_b(x)$ be the chosen class of the b-th tree in the RF on input x, the chosen class of the RF $\hat{C}_{\mathrm{rf}}(x)$ is given by:

$$\hat{C}_{\mathrm{rf}}(x) = \left\{\hat{C}_b(x)\right\}_1^B \tag{3.55a}$$

3.2.4 Summary

Section 3.1 discussed fundamental image processing techniques that are implemented in the proposed system to extract hand shape features from an image sequence, as a basis for subsequent chapters.

The image processing techniques discussed in this section of the chapter were edge detection, face detection, skin detection, background subtraction, hierarchical Chamfer matching, Connected Components Analysis and CAMShift tracking. These techniques are referred to in a subsequent chapter that details the feature extraction procedure.

Section 3.2 discussed the three machine learning techniques which are compared in the context of SASL hand shape recognition. The three techniques discussed were Support Vector Machines, Artificial Neural Networks and Random Forests. In each case, the underlying principle behind the classification strategy of each technique was discussed in detail.

The next chapter describes the proposed system implementation used to carry out the comparison in SASL hand shape accuracy between the three techniques.

Chapter 4

Design and Implementation of the Hand Shape Recognition System

This chapter discusses the design and implementation of the hand shape recognition system. Figure 4.1 illustrates the process flow, as well as the various image processing techniques and machine learning techniques used within the hand shape recognition system. The operation of the system can be broken down into two main components, namely, the feature extraction and the classification.

In the feature extraction component, image processing techniques are used to locate and track the hand of the signer, and extract the features relating to the shape of the hand. The classification component has two phases: a training phase and a testing phase. In the training phase, a set of labeled training images are used to produce a classification model of each machine learning technique. In the testing phase, the classification model is used to classify a previously unseen image into one of the pre-defined hand shape classes.

This chapter is organized into three sections. The first section—Section 4.1—discusses the feature extraction component used to extract hand shape features from an input video stream in detail, in order to successfully achieve Objective 1 set out in Chapter 1. Section 4.2 details the classification component which involves the training and use of the three machine learning techniques—SVMs, RFs and ANNs—towards recognizing SASL hand shapes, which is used in the next chapter to achieve Objectives 2 and 3 set out in Chapter 1. The chapter is then concluded.

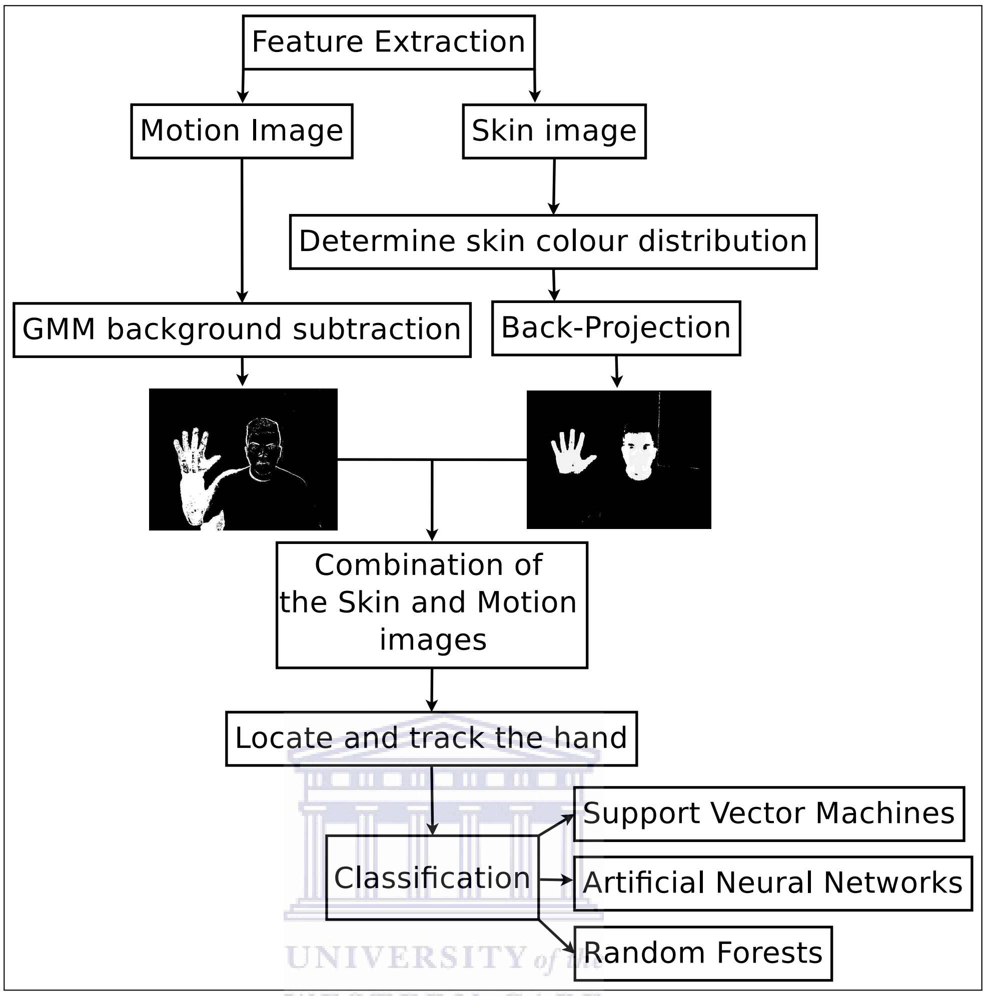

Feature Extraction
Motion Image
Skin image
Determine skin colour distribution
GMM background subtraction
Back-Projection
Combination of the Skin and Motion images
Locate and track the hand
Classification
Support Vector Machines
Artificial Neural Networks
Random Forests

FIGURE 4.1: An overview of the hand shape recognition system.

4.1 Feature Extraction

This section explains the procedure carried out to locate and track the hand of the signer and subsequently extract features related to the hand shape of the signer. Referring to Figure 4.1, the feature extraction component involves concurrently computing a skin image, described in Subsection 4.1.1 below, and a motion image, detailed in Subsection 4.1.2 below, and subsequently combining these images to obtain a combined image, explained in Subsection 4.1.3. This procedure happens on every frame of the input video stream.

Hierarchical Chamfer matching is applied to the combined image only once in the initial stage of the procedure to locate the signer's hand, described in Subsection 4.1.4. This location is then used to initialize the CAMShift tracking algorithm which continuously tracks the signer's hand thereafter. This is described in Subsection 4.1.5. Finally, the contours of the hand are normalized and extracted from the isolated hand region. This is described in Subsection 4.1.6 below.

4.1.1 Skin Image

An adaptive skin detection algorithm created by [2] is implemented to locate the skin pixels of the signer. The algorithm is able to detect skin pixels within an image. It is also robust to various skin tones and illumination conditions.

In order to determine the skin colour distribution of the signer, the face of the signer is detected. This is achieved using the Viola-Jones face detection algorithm [66]. The detected face of the signer is depicted in Figure 4.2a.

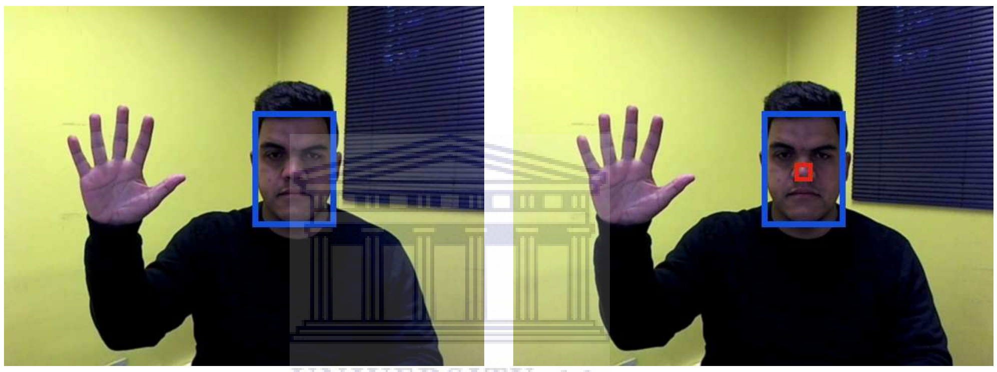
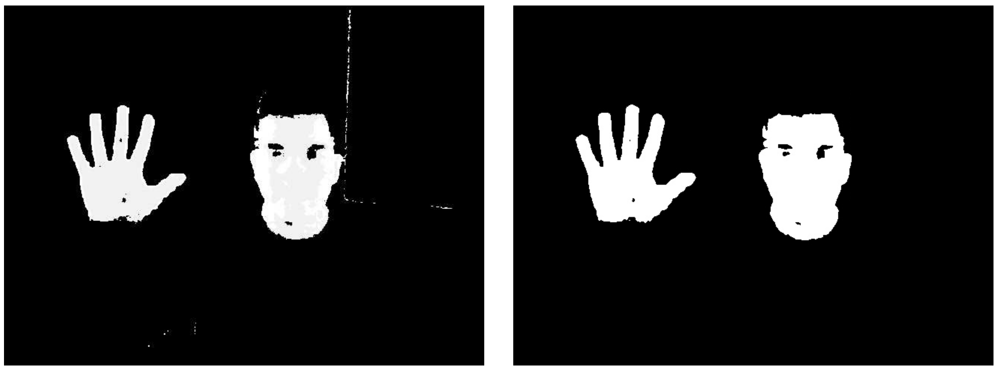

(a) Face detection applied to the image (b) The center of the facial frame locates the tip of the nose

FIGURE 4.2: Locating the signer's nose.

Once the face has been detected, the next step is for the position of nose of the signer to be detected. The tip of the nose is located by extracting a 10×10 pixel region of the center of the detected face of the signer, as shown in Figure 4.2b. This specific region is used as it is a region that is void of any non-skin obstructions such as hair, discolouration, etc. It provides a robust indication of the skin colour distribution of the signer. This region is converted from the RGB (Red-Green-Blue) colour space to the HSV (Hue-Saturation-Value) colour space.

A 1-dimensional H-S histogram is computed from hue component of the HSV colour representation of the extracted nose region. The hue component is used because it has been shown to be robust to illumination variations. The computed histogram is then backprojected onto the original frame of the signer, which produces a skin probability image i.e. brighter pixels have a high probability of being skin. Brown found that applying a threshold value of 60 to the resulting probability image provided an optimal skin binarization result [12]. The resulting binary image highlights only skin pixels in the image. Skin pixels are represented with an intensity value of 255 and non-skin pixels are represented by 0, as shown in Figure 4.3a.



(a) The skin image (b) Application of Gaussian blur to reduce noise

FIGURE 4.3: The skin image: a) before and b) after applying Gaussian blur to smoothen the image and reduce noise.

A Gaussian blur operation is applied to smooth the image in order to reduce sources of noise in the image, the result of which can be seen in Figure 4.3b. The contours of skin regions are smoothened and small regions of noise, such as the contours of the blinds in the top-right region of the image, are eliminated.

4.1.2 Motion Image

The skin image is not sufficient to locate the hand on its own, as there could be other large skin-coloured objects, especially the face, in the frame that should constitute the background but would be highlighted. As such, background subtraction is used to segment the moving foreground from the stationary background of the image and eliminate such sources of noise.

This is achieved using the Gaussian Mixture Models (GMMs) background subtraction technique described in Section 3.1 of Chapter 3. GMMs are applied to the original input image, resulting in an image—the motion image—in which only the moving pixels are highlighted. The motion image is depicted in Figure 4.4.

4.1.3 Combination of the Skin and Motion Images

The skin image provides an accurate mapping of the skin pixels in an input image, but may highlight non-skin areas that are skin-coloured. The motion image effectively highlights all moving parts of an image, but these parts include objects besides the hand of the signer. Combining the skin image shown in Figure 4.3b and the motion image shown in Figure 4.4 effectively eliminates most sources of noise and objects that are not

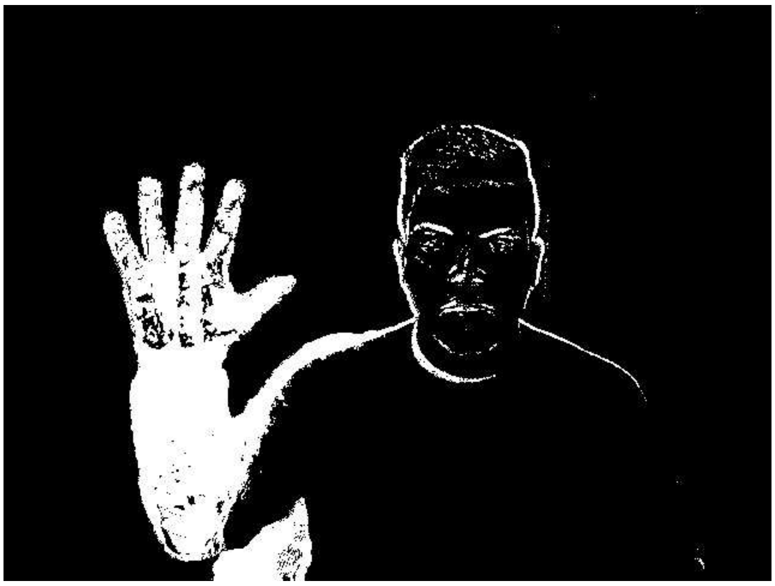


FIGURE 4.4: The result of applying Gaussian Mixture Models to highlight moving pixels—the motion image.

of interest in either image. This image contains only the moving skin pixels present in the input frame. Figure 4.5 depicts the moving skin pixel image produced by combining Figures 4.3b and 4.4.

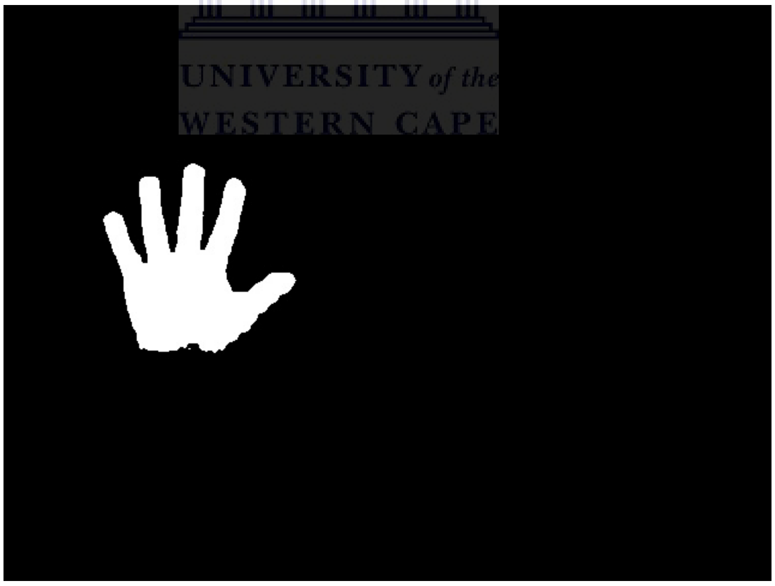


FIGURE 4.5: The skin image and motion image combined to form a moving skin image.

4.1.4 Locating the Hand

Hierarchical Chamfer Matching is used to locate the signer's hand only once during the initial stage of the entire procedure. This is used to initialize the CAMShift tracking algorithm tracking window. The system assumes that, only initially, the signer holds up an open palm until tracking is initialized. A pre-computed template silhouette of the open palm is used to locate the signer's open palm in the moving skin image.

The matching is performed at various scales to find the best match. The most probable match location is determined. Searching for the hand silhouette in the moving skin image virtually eliminates the chance of determining an incorrect matching location.

4.1.5 Using CAMShift for Hand Tracking

The location of the search window of the CAMShift algorithm is set to the location of the hand. CAMShift then continuously tracks the signer's hand as it moves, and it does so in real time. Figure 4.6 shows the CAMShift tracking algorithm in action, tracking the signer's hand.

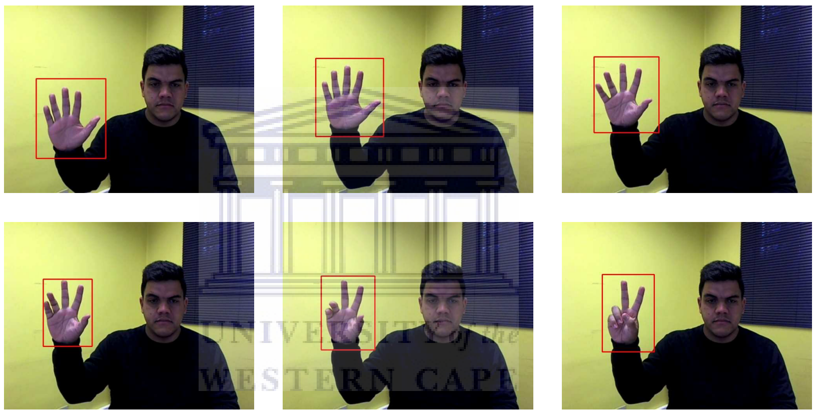


FIGURE 4.6: The hand of the signer tracked by the CAMShift tracking window.

4.1.6 Feature Extraction and Normalization

The hand of the signer is now consistently tracked across each frame of the video stream using CAMShift. Connected Components Analysis (CCA) is used to compute the contours of the objects in the region in the CAMShift tracking window, which mostly consists of the signer's hand. It is assumed that the largest connected component in the region is the hand contour. As such, all connected components that are not connected to or part of the largest connected component are eliminated. This results in an image of the isolated hand.

In some situations the hand could be tilted in-plane slightly which can cause incorrect classification of a hand shape. One solution is to train the machine learning techniques on a large number of hand rotation and orientations. A much more cost-effective and efficient solution is to normalize the hand by aligning it to one of the principal axes. The latter method is used.

A minimum bounding box is drawn around the hand as depicted in Figure 4.7a. Assuming a small rotation of the hand in-plane ($< 45°$), the principal axis of the box is rotated and aligned with the vertical image axis, as shown in Figure 4.7b. After rotation, the hand contour is scaled down to a resolution of 30×40 pixels, depicted in Figure 4.7c.

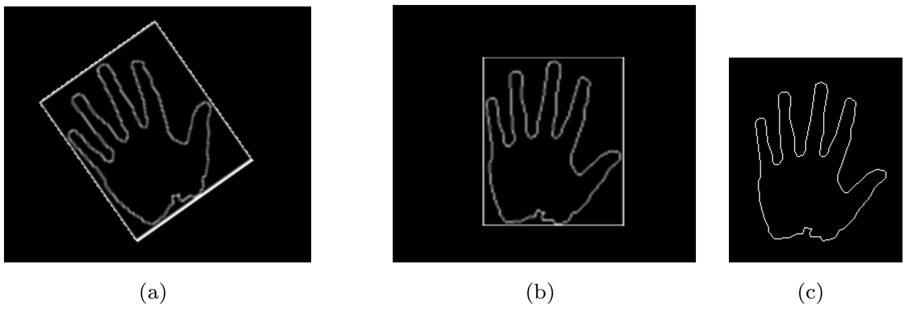

(a) (b) (c)

FIGURE 4.7: The process of extraction and normalization of the hand contour: (a) The original tilted hand contour with a minimum bounding box drawn around it (b) The minimum bounding box aligned with the vertical axis (c) The normalized and resized hand contour.

This normalized binary contour image is the feature representation of the hand shape. Assuming that the value of a pixel at position (i, j) in the normalized binary contour image I is given by $I(i, j)$, a final feature vector V is computed by concatenating the pixels of the image, row-by-row and column-by-column into a single linear feature vector as follows:

$$V = \{R_1, \ldots, R_{40}\} \tag{4.1}$$

where

$$R_j = \{I(1, j), \ldots, I(30, j)\}$$

where R_j represents the j-th row in the image I.

4.2 Classification

With the extracted hand contour feature vector available, it is now possible to train and test the three machine learning techniques. The ten SASL hand shapes, shown in Figure 4.8, are used to train and test the machine learning techniques. Overall, the classification procedure for any of the machine learning techniques used takes place in two phases.

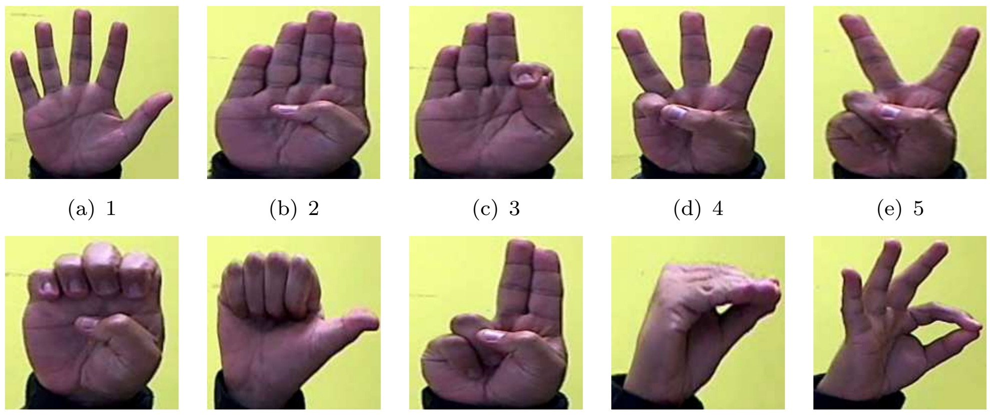

(a) 1 (b) 2 (c) 3 (d) 4 (e) 5
(f) 6 (g) 7 (h) 8 (i) 9 (j) 10

Figure 4.8: The ten SASL hand shapes.

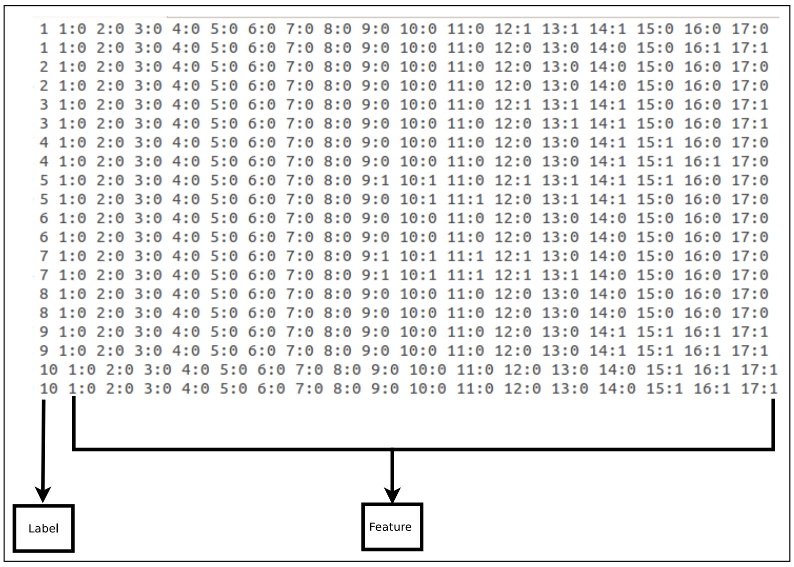
The first phase called the "training phase" involves optimizing and training the machine learning technique using a set of labeled images, referred to as the "training set", to obtain a classification model. Optimization involves determining the optimal values of specific parameters, specific to each machine learning technique, that achieve the optimum classification accuracy on the training set. Training then involves using the optimal parameters to produce a final optimal classification model. The experiments carried out to optimize and train the SVM, ANN and RF are detailed in the next chapter.

The second phase called the "testing phase" involves classifying an unknown or unseen image into one of the classes using the trained and optimized models on a continuous basis. Given a set of labeled images different to those of the training set called a "testing set", it is possible to determine the classification accuracy of the machine learning technique. The next chapter also discusses the experiments carried to determine and compare the accuracy of each machine learning technique using a testing set.

This section describes the process involved in training each machine learning technique in the training phase given a training set, and classifying a set of unseen images in the testing phase using a testing set. The following subsections describe these procedures for Support Vector Machines in Subsection 4.2.1, Artificial Neural Networks in Subsection 4.2.2 and Random Forests in Subsection 4.2.3.

4.2.1 Classification Using the Support Vector Machine

The very popular LibSVM [25] implementation of Support Vector Machines is used in this research. The library provides a set of tools to create Support Vector Machine classification models and classify a given input using an existing or pre-created model.



```
1 1:0 2:0 3:0 4:0 5:0 6:0 7:0 8:0 9:0 10:0 11:0 12:1 13:1 14:1 15:0 16:0 17:0
1 1:0 2:0 3:0 4:0 5:0 6:0 7:0 8:0 9:0 10:0 11:0 12:0 13:0 14:0 15:0 16:1 17:1
2 1:0 2:0 3:0 4:0 5:0 6:0 7:0 8:0 9:0 10:0 11:0 12:0 13:0 14:0 15:0 16:0 17:0
2 1:0 2:0 3:0 4:0 5:0 6:0 7:0 8:0 9:0 10:0 11:0 12:0 13:0 14:0 15:0 16:0 17:0
3 1:0 2:0 3:0 4:0 5:0 6:0 7:0 8:0 9:0 10:0 11:0 12:1 13:1 14:1 15:0 16:0 17:1
3 1:0 2:0 3:0 4:0 5:0 6:0 7:0 8:0 9:0 10:0 11:0 12:0 13:1 14:1 15:0 16:0 17:1
4 1:0 2:0 3:0 4:0 5:0 6:0 7:0 8:0 9:0 10:0 11:0 12:0 13:0 14:1 15:1 16:0 17:0
4 1:0 2:0 3:0 4:0 5:0 6:0 7:0 8:0 9:0 10:0 11:0 12:0 13:0 14:1 15:1 16:1 17:0
5 1:0 2:0 3:0 4:0 5:0 6:0 7:0 8:0 9:1 10:1 11:0 12:1 13:1 14:1 15:1 16:0 17:0
5 1:0 2:0 3:0 4:0 5:0 6:0 7:0 8:0 9:0 10:1 11:1 12:0 13:1 14:1 15:0 16:0 17:0
6 1:0 2:0 3:0 4:0 5:0 6:0 7:0 8:0 9:0 10:0 11:0 12:0 13:0 14:0 15:0 16:0 17:0
6 1:0 2:0 3:0 4:0 5:0 6:0 7:0 8:0 9:0 10:0 11:0 12:0 13:0 14:0 15:0 16:0 17:0
7 1:0 2:0 3:0 4:0 5:0 6:0 7:0 8:0 9:1 10:1 11:1 12:1 13:0 14:0 15:0 16:0 17:0
7 1:0 2:0 3:0 4:0 5:0 6:0 7:0 8:0 9:1 10:1 11:1 12:1 13:1 14:0 15:0 16:0 17:0
8 1:0 2:0 3:0 4:0 5:0 6:0 7:0 8:0 9:0 10:0 11:0 12:0 13:0 14:0 15:0 16:0 17:0
8 1:0 2:0 3:0 4:0 5:0 6:0 7:0 8:0 9:0 10:0 11:0 12:0 13:0 14:0 15:0 16:0 17:0
9 1:0 2:0 3:0 4:0 5:0 6:0 7:0 8:0 9:0 10:0 11:0 12:0 13:0 14:1 15:1 16:1 17:1
9 1:0 2:0 3:0 4:0 5:0 6:0 7:0 8:0 9:0 10:0 11:0 12:0 13:0 14:1 15:1 16:1 17:1
10 1:0 2:0 3:0 4:0 5:0 6:0 7:0 8:0 9:0 10:0 11:0 12:0 13:0 14:0 15:1 16:1 17:1
10 1:0 2:0 3:0 4:0 5:0 6:0 7:0 8:0 9:0 10:0 11:0 12:0 13:0 14:0 15:1 16:1 17:1
```

FIGURE 4.9: Illustration of the data file format used by LibSVM.

Given the feature vectors of a set of training images in the training phase produced as previously explained, a data file is produced in a specific format. The format of the data file consists of label and feature pairs as shown in Figure 4.9. Each row of data represents the label and feature vector of a specific hand shape image. Each row in the figure has a class label, and the rest of the data consists of "index:feature" pairs of binary values of the normalized hand shape contour image of that row.

As previously noted, a feature vector consisting of 30 × 40 pixels for the hand shape contour image implies a total of 1200 pixels and, therefore, 1200 features per row. Due to space constraints, the figure omits features 18 to 1200 which are implied to exist in the data file. Also note that each feature value is a binary value in which a '1' indicates a contour pixel and a '0' indicates a non-contour pixel. The result of training on this data file is a SVM classification model.

In the testing phase, given the feature vectors of a set of testing images whose labels are not known, a data file in the same format mentioned previously is produced, except that the label of each row is set to a default value of −1, since the true class is not known. The output of the SVM prediction, given the trained model, is a file consisting of predicted labels of each row in the original data file.

4.2.2 Classification Using the Artificial Neural Network

The OpenCV implementation of ANNs was used. A Feed Forward 3-Layer Perceptron ANN was used in the recognition of the ten SASL hand shapes. The network consists of an input layer, one hidden layer and an output layer. The input layer consists of 1200

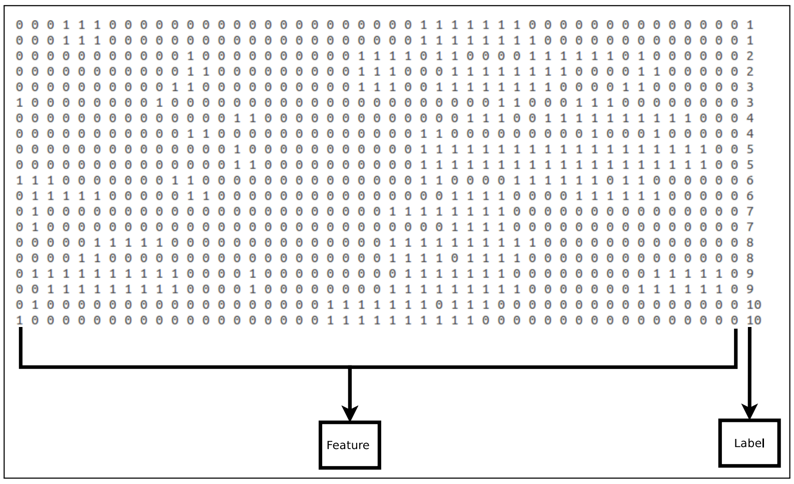

```
0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 1 1 0 0 0 0 1 1 1 1 1 1 0 1 0 0 0 0 0 0 2
0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 1 1 0 0 0 0 0 2
0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 1 1 1 1 1 1 1 1 0 0 0 0 1 1 0 0 0 0 0 0 3
1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0 3
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0 4
0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 4
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 5
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 5
1 1 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 1 1 1 1 1 0 1 1 0 0 0 0 0 0 6
0 1 1 1 1 1 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 6
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7
0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 8
0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 8
0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 9
0 0 1 1 1 1 1 1 1 1 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 1 1 1 1 1 1 0 9
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 10
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 10
```

FIGURE 4.10: The data file of the Artificial Neural Network

neurons corresponding to each of the 1200 features in the hand shape feature vector. The output layer consists of 10 neurons, each corresponding to one of the 10 hand shape classes to be recognized. While the number of input and output neurons is known, the number of hidden neurons m that can yield an optimal classification accuracy is not known. The optimization of the number of neurons in the hidden layer is carried out in the next chapter.

In the training phase, similar to the case of SVMs, given the feature vectors of a set of training images, a data file is produced. The format in this case is quite similar to that of SVMs, but consists of only the values of the set of binary digits of the 1200 features of the hand shape separated by spaces, followed by a corresponding class label, illustrated in Figure 4.10. Each row corresponds to a single hand shape image in the training set. Given this data file, backpropagation is used to determine the optimal weights for the ANN used to produce a classification model.

In the testing phase, given the feature vectors of a set of testing images, a data file in the same format as the training phase is produced, but the labels of all the images are, similar to SVMs, set to a default value of -1. Given the trained classification model and the data file, a output file is produced containing the predicted labels of the images represented in each row of the file.

Since the sigmoid activation function is used at each neuron, including the output neurons, the eventual output at each output neuron is a value in the range [0,1]. Given the inputs of a single image to be classified, the class represented by the output neuron with the highest value is taken as the correctly predicted class.

4.2.3 Classification Using the Random Forest

Similar to ANNs, the OpenCV implementation of RFs was used. As such, the format of the data file to be constructed for either training or testing sets was very similar to the one described in the previous subsection, with the exception that the features in the file are comma-delimited, and the last value in every row represents the label of the image represented by the features in that row. Once again, every row represents a single hand shape image.

In the training phase, the training data file is used to produce a forest of decision trees using the optimal parameters. As explained in the previous chapter, RFs have three possible parameters that can be optimized to achieve optimal classification accuracy on a training set. These are: the total number of trees in the forest B, the depth of the individual decision trees in the forest D_{min} and the number of split variables per node p_s. The next chapter describes the experiment carried out to optimize the parameters of the RF used.

In the testing phase, the feature vectors of a set of testing images are placed into the same format as in the training phase, with the exception that the labels of all the images are set to the default place-holder -1 value, indicating that the class of these images is unknown. The output of the forest is a file in which each row contains a label that the majority of decision trees in the forest determined as being the class of the image in that row in the input data file.

4.3 Summary

This chapter detailed the implementation of the feature extraction and classification components of the system. It described, in detail, the image processing procedure used to locate and track the hand, and subsequently extract and represent the features of the hand shape. As such, it is concluded at this stage that Objective 1 set out in Chapter 1 has succesfully been achieved. The chapter also described the process used to train each machine learning technique, as well as use the trained classification models of each technique to produce a predicted class for a previously unseen hand image.

The next chapter describes the experiments carried out to optimize the parameters of each of the machine learning techniques and subsequently train them, as well as the experiments carried out to assess and compare the accuracy and computational speed of each technique in order to meet Objectives 2 and 3 set out in Chapter 1, and answer the research questions.

Chapter 5

Experimental Results and Analysis

This chapter discusses the experimentation carried out to optimize and train the three machine learning techniques, and evaluate the classification accuracy and speed of the three techniques. These experiments ultimately lead to an answer to the main research question and all three research sub-questions set out in Chapter 1. The results culminate in the selection of the best technique out of the three techniques compared in the context of SASL hand shape recognition.

All experiments were carried out on a Lenovo ThinkPad Edge PC with a 2.2GHz i3 CPU and 4 GB of RAM. For ease of reference in this chapter, a reference to a specific hand shape will be denoted as "HS x", where x is the hand shape number. For example, Hand shape 10 will be denoted HS 10.

The discussion in the chapter is sub-divided into the following sections. Section 5.1 explains the dataset collected and used in the optimization, training and testing of the machine learning techniques.

Section 5.2 details the experiments carried out to optimize each machine learning technique in order to meet Objective 2 set out in Chapter 1. A detailed analysis of the results culminates in a response to Research Sub-question 1 set out in Chapter 1.

Section 5.3 discusses the experiments carried out to determine the classification accuracy and time of each technique in order to meet the final objective, Objective 3 set out in Chapter 1.

Section 5.4 then provides a summary of the comparisons in previous sections in order to obtain a response to the two remaining Research Sub-questions 2 and 3 set out in

Chapter 1, as well as motivate for the selection of the best technique, out of the three techniques compared, for SASL hand shape recognition.

5.1 Training and Testing Datasets

For the collection of SASL hand shape images, twelve subjects of varying skin colour shown in Figure 5.1 were asked to perform each of the ten SASL hand shapes. It is clear from the figure that the subjects chosen were of varied skin tones in order to demonstrate the robustness of the system to variations in skin tone.

A video of no less than 30 seconds was recorded per subject for each of the ten SASL hand shapes. This resulted in ten videos collected from each of the twelve subjects, one for each SASL hand shape. Equivalently, this resulted in twelve videos per SASL hand shape, and 120 videos in total. These videos were recorded at 24 frames per second and, in total, the entire dataset consisted of over 80000 images.

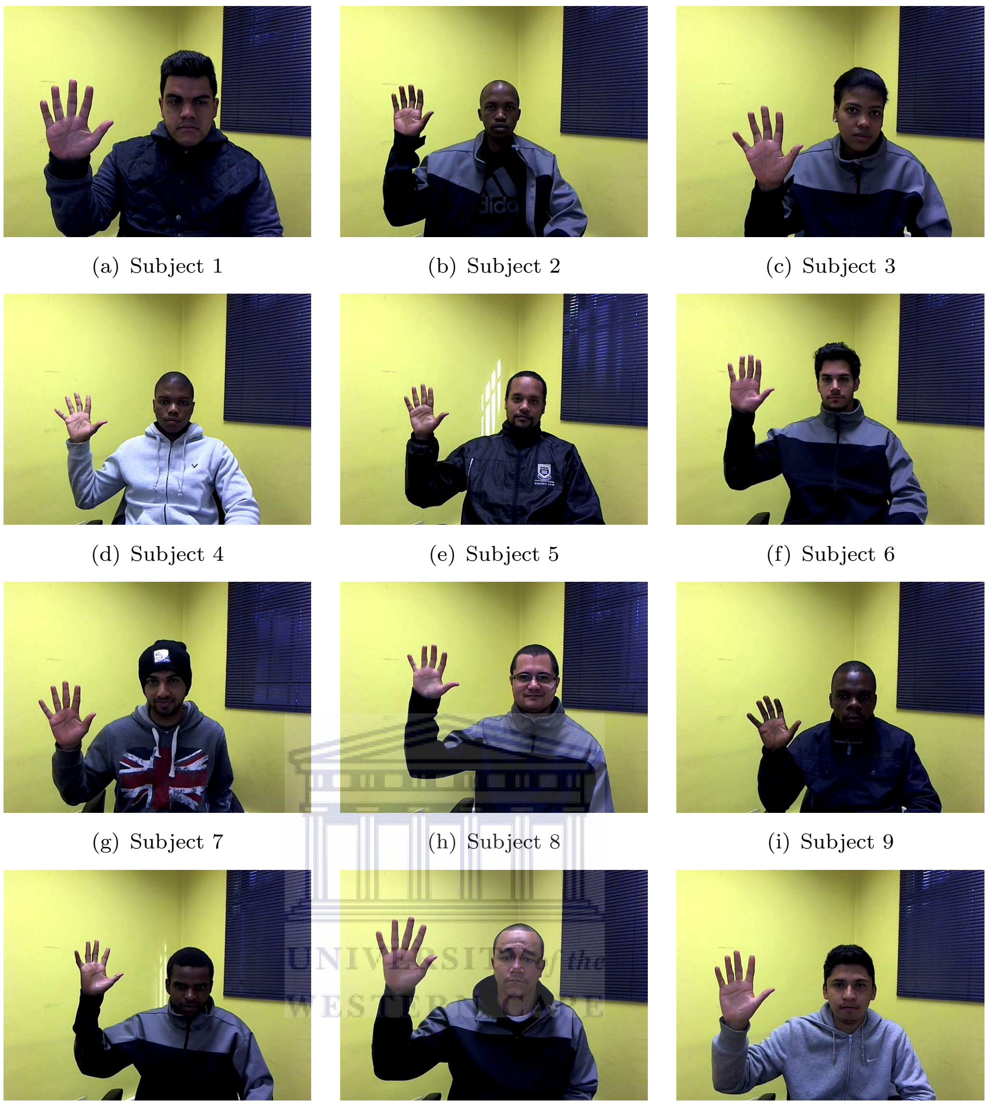
The dataset was divided into two equal parts. The images of Subjects 1 to 6 were used as a training set for the optimization experiments in Section 5.2. A total of 50 images of each hand shape for each subject were randomly chosen and combined to form a training set. The resulting set consisted of a total of 3000 images; 300 images per hand shape and 500 images per subject. This set is henceforth referred to as the "training set" and is used in the optimization experiments and training procedures detailed in Section 5.2.

The images of Subjects 7 to 12 were used as a testing set for the classification experiments in Section 5.3. Like the training set, a total of 50 images of each hand shape for each subject were randomly chosen and combined to form a testing set. The resulting set also consisted of a total of 3000 images; 300 images per hand shape and 500 images per subject. This set is henceforth referred to as the "testing set".

The training and testing sets were so devised as to include subjects of a variety of skin tones in both sets, as observed in Figure 5.1.

5.2 Optimization Experimentation

This section describes the experiments carried out to determine the optimal parameters of each machine learning technique, and subsequently produce an optimal classification model for each technique, in accordance with Objective 2 set out in Chapter 1. A comparison in the results in order to answer Research Sub-question 1 is also provided.


(a) Subject 1 (b) Subject 2 (c) Subject 3

(d) Subject 4 (e) Subject 5 (f) Subject 6

(g) Subject 7 (h) Subject 8 (i) Subject 9

(j) Subject 10 (k) Subject 11 (l) Subject 12

FIGURE 5.1: The 12 subjects used for training and testing.

Section 5.2.1 below first explains the *k*-fold cross-validation technique that is used as an accuracy measure when optimizing each technique. Since the optimization procedure is completely different for each technique, the discussion of these procedures is provided separately in Sections 5.2.2, 5.2.3 and 5.2.4 for the SVM, ANN and RF, respectively, with Section 5.2.5 finally comparing these results and answering Research Sub-question 1.

5.2.1 *k*-fold Cross Validation

For optimization and training, the *k*-fold cross-validation technique was used. The technique is a method of validating the accuracy of a machine learning technique to aid with parameter selection. It involves first splitting up the training set into k equally-sized subsets. Thereafter, the machine learning technique is trained on $k-1$ subsets and tested on the remaining set in a revolving fashion, to make a total of k trials or "turns".

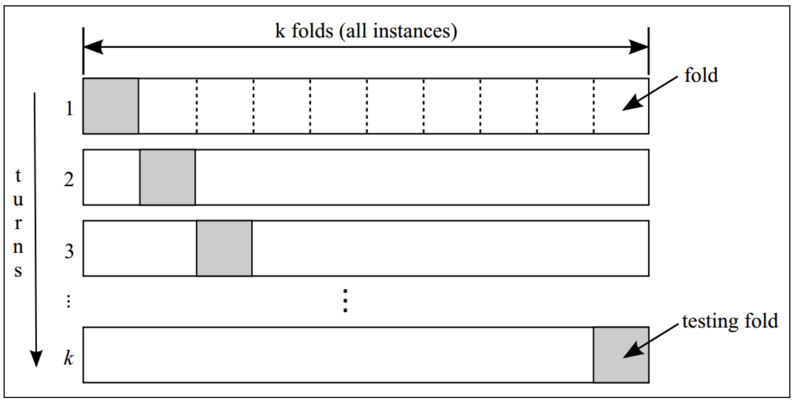


FIGURE 5.2: A visual representation of k-fold cross validation [9]

This process is illustrated in Figure 5.2. It is seen that in turn 1, the first subset or "fold" which is shaded is used as a testing set, and folds 2 to k are used as a combined training set. In turn 2, the second fold is used as a testing set, and a combination of folds 1 and 2 to k are used a training set. Repeating this procedure, represented by a downward projection through the figure, results in k unique turns. Finally, the average accuracy across all k folds is computed. This accuracy is referred to as the k-fold cross validation accuracy or cross validation accuracy for short. It is a very good measure of the overall classification accuracy of the model and its ability to generalize to a variety of unseen images and subjects [9].

In this research, the size of k was chosen as 6. This was chosen in order to be able to divide the training set such that each fold contains all the images of 1 of the 6 subjects in the set. This ensures that each cross-validation turn makes use of a different set of subjects for training and testing, giving a strong indication of the ability of the system to generalize to a variety of test subjects. As such, in each cross-validation turn, the testing fold consists of the images of 1 of the training subjects, and the combined training fold consists of the images of the remaining subjects.

5.2.2 Optimization of the Support Vector Machine

As mentioned in Subsection 3.2.1 in Chapter 3 , the Radial Basis Function (RBF) kernel is used with the SVM in this research. This kernel has two parameters—C and γ—that can be optimized to yield an optimum SVM classification model. The optimization of these parameters is carried out by a process of trial-and-error involving investigating the classification accuracy of various (C, γ) pairs as follows.

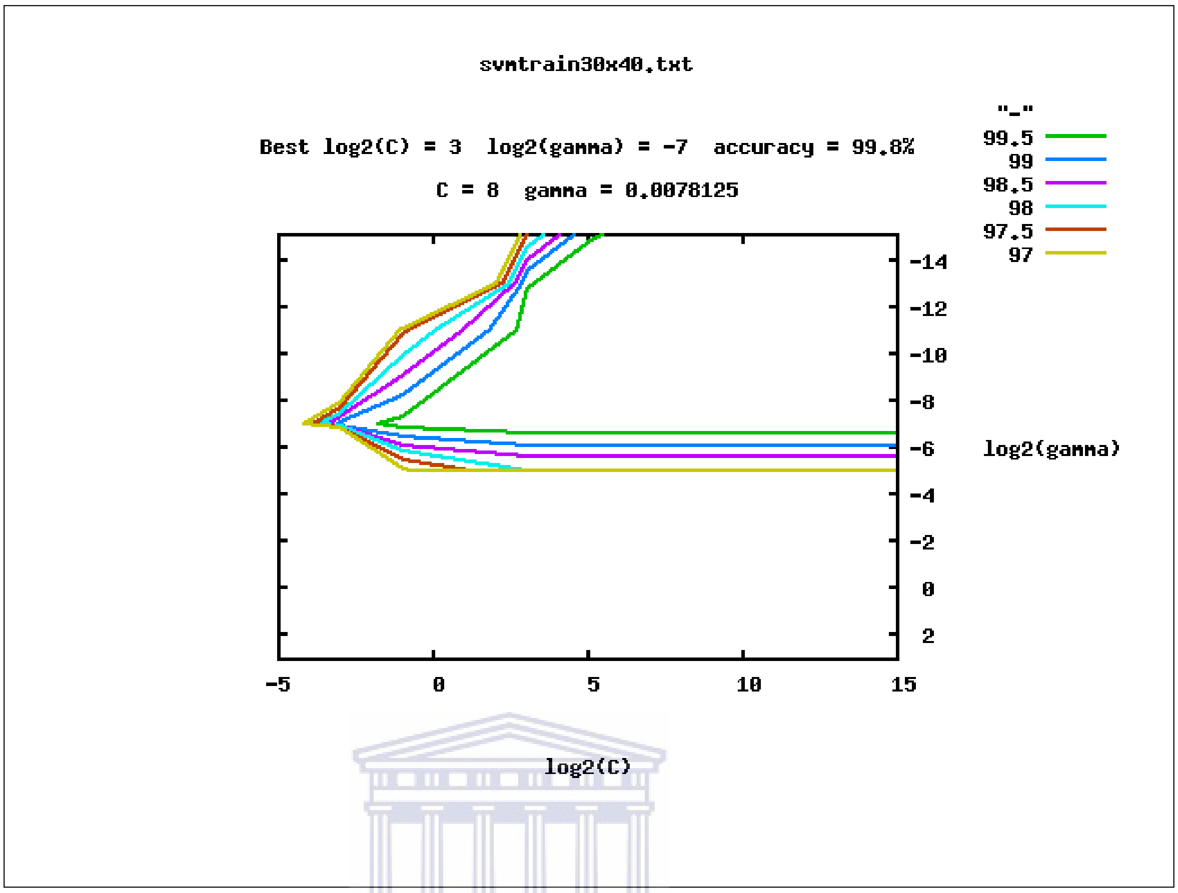


FIGURE 5.3: Optimization of the SVM: A graph depicting the grid-search optimization results.

In each trial, a (C, γ) pair is selected. 6-fold cross-validation is then used to determine the classification accuracy of this specific (C, γ) pair. The pair that produces the highest cross-validation accuracy is selected as the optimum parameter setting. LibSVM provides a grid-search tool that carries out this procedure and returns the (C, γ) pair that yields the highest cross-validation accuracy, and this tool was used in this research.

The graphical output of the grid-search is illustrated in Figure 5.3. It can be seen that the pair $(C = 8, \gamma = 0.0078125)$ produced the optimum cross-validation accuracy of 99.8%. The entire grid-search optimization procedure was timed, and took 109 seconds to complete. The optimal parameter values were then used to train the SVM on all of the samples of the training set, and this was timed, and took 21 seconds to complete.

5.2.3 Optimization of the Artificial Neural Network

As mentioned in the previous chapter, a 3-Layer Perceptron consisting of an input layer with 1200 neurons and an output layer with 10 neurons was used. The number of neurons in the hidden layer m is unknown and is optimized.

As demonstrated in Chapter 2, a process of trial and error can be used to determine the optimal number of hidden neurons [1, 33]. Typically, the use of between 5 and 100 hidden neurons is investigated [23]. To be consistent with the optimization of the SVM, the same 6-fold cross-validation technique previously explained was used to determine the optimal number of hidden neurons m. The number of hidden neurons m was varied from 2 to 50. For each m, the 6-fold cross-validation accuracy was determined and recorded.

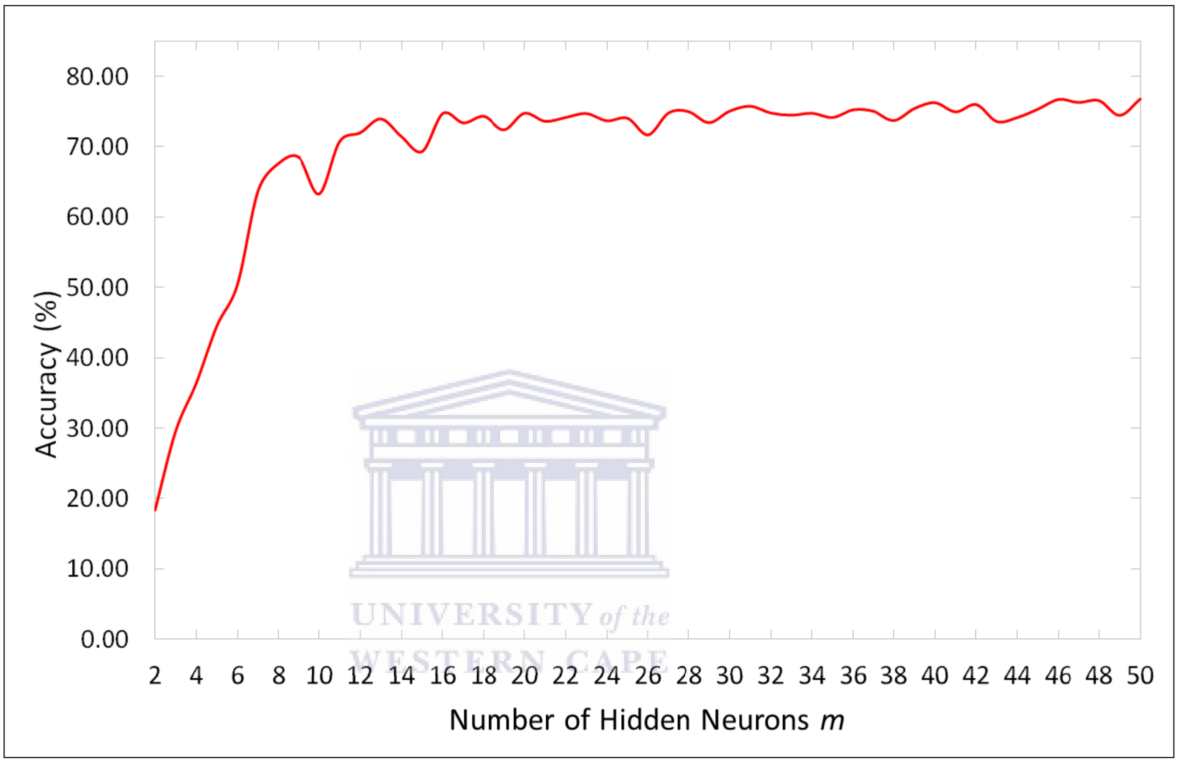


FIGURE 5.4: Optimization of the ANN: A graph of the cross-validation accuracy for each number of hidden neurons m used.

Table A.1 in Appendix A contains the complete optimization results, and Figure 5.4 summarizes these results graphically. Referring to Figure 5.4, it can be seen that the cross-validation accuracy of the graph initially increases sharply with each added neuron, but eventually converges. In the region of the graph between 2 and 11 neurons, the accuracy increases very rapidly from 18.33% to 70.73%. Thereafter, the accuracy stabilizes and converges to the range between 72% and 76%.

The optimal number of neurons chosen is 16, since this is the best trade-off between cross-validation accuracy and processing speed. A larger number of hidden neurons, in this case, may yield a slight increase in cross-validation accuracy of about 1% or 2%, but greatly increases the training and processing time due to a significantly larger number of computations required in the latter case.

The total time taken to carry out the optimization procedure for all $m \in \{2, \ldots, 50\}$ was 3589 seconds. The time taken to train the final model using the optimal number of hidden neurons was 39 seconds.

5.2.4 Optimization of the Random Forest

As explained in the previous chapter, RFs have three possible parameters that can be optimized to achieve optimal classification accuracy on a training set. These are: the total number of trees in the forest B, the depth of the individual decision trees in the forest $D_{\min}$ and the number of split variables per node p_s. As mentioned in Chapter 3.2, it is common practice to set $p_s = \log_2 p + 1$. Given the number of features $p = 1200$, the value of p_s is computed as $p_s = 11$. This produces sufficiently accurate results and significantly limits the parameter optimization search problem to two variables. This substantially simplifies and speeds up the otherwise expensive optimization process. This approach is used.

For consistency, the same 6-fold cross-validation accuracy used to optimize the previous two machine learning techniques was used as a measure of classification optimality of each $(B, D_{\min})$ combination. The value of $D_{\min}$ was increased in steps of 2 from 2 upwards. For each $D_{\min}$ value, values of B from 5 upwards in steps of 5 were used and the cross-validation accuracy of each combination was computed.

Table A.2 in Appendix A contains the complete optimization results, and Figure 5.5 summarizes these results graphically. Each curve in the graph represents the cross-validation accuracy of a specific depth value $D_{\min}$. Two trends are noted from Figure 5.5.

First, by observing each individual curve in the graph, it is noted that as the number of trees B increases, the cross-validation accuracy initially also increases rapidly, but the accuracy eventually converges, at which point increasing the number of trees does not generally appear to yield any significant increase in accuracy. This trend is consistent for all of the curves i.e. for all the different depth values $D_{\min}$. It also appears that for higher values of $D_{\min}$, the accuracy appears to converge at approximately $B = 40$ or $B = 45$.

Second, by comparing the accuracy level of the curves, it is noted that increasing the depth also initially yields a rapid increase in cross-validation accuracy, embodied by the upward shift in the curves, but the accuracy once again converges starting at a depth value of $D_{\min} = 8$, and no significant increase in accuracy is realized by increasing the depth after this point, represented by the general overlap between the curves of $D_{\min} > 8$.

A tree depth $D_{\min} = 12$ is observed to yield a slightly higher cross-validation accuracy of 70.03% compared to other depth values. Considering the curve of this depth value, it is seen that no change in the cross-validation accuracy is observed from 50 trees onwards. Therefore, the number of trees is taken as $B = 50$. The optimal parameter pair is then $(B = 50, D_{\min} = 12)$

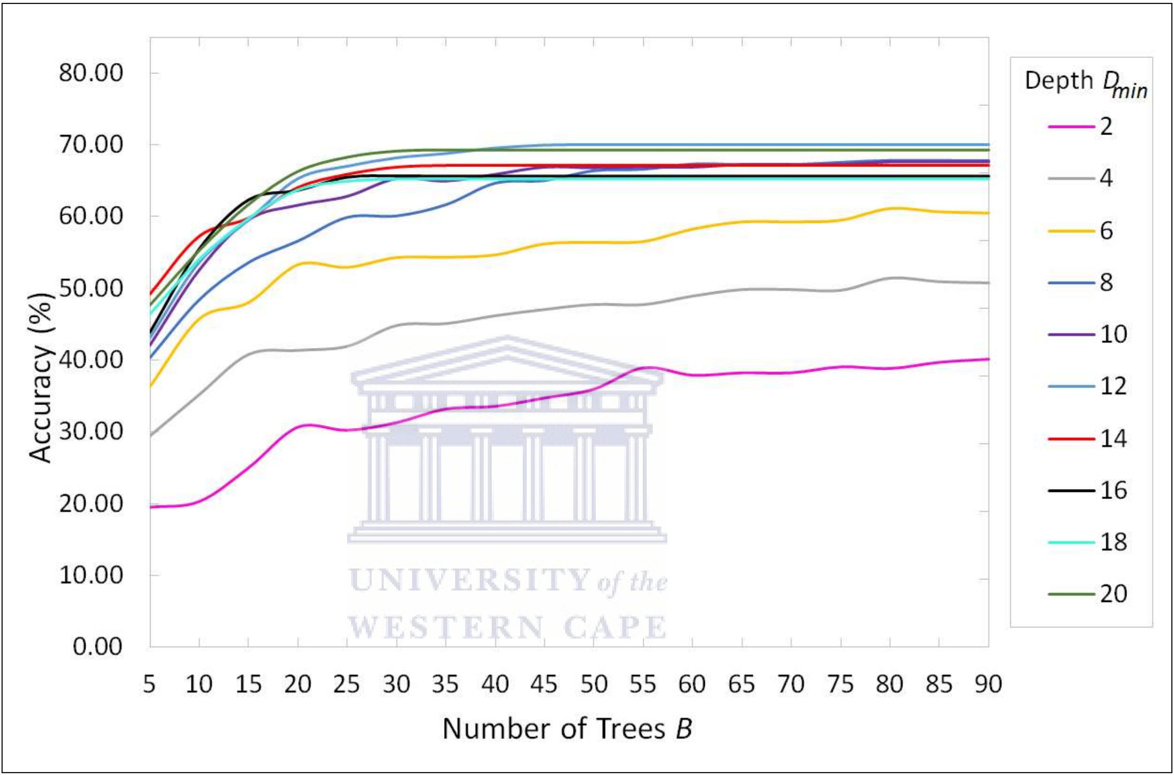


FIGURE 5.5: Optimization of the RF: The cross-validation accuracy for each increment in the number of trees B for various depths $D_{\min}$.

The total time taken to carry out the entire optimization procedure was 14916 seconds. The time taken to train the final RF model using the optimal parameter pair $(B = 50, D_{\min} = 12)$ was 101 seconds.

5.2.5 Comparison in Optimization and Training Procedures

It is stated that, at this stage, Objective 2 set out in Chapter 1 has successfully been achieved.

It is very clear, at this point, that the SVM is by far the quickest technique to optimize and train. The optimization time of this technique is, respectively, 1 and 2 orders of magnitude smaller than those of the ANN and RF. The training time is also considerably smaller than that of both techniques, being about half that of the ANN and about 5 times smaller than that of the RF.

As such, it is very clear that, in this respect, the SVM is superior to both other techniques.

The ANN is also clearly faster to optimize and train than the RF. The ANN considered in this case only requires one parameter to be optimized—the number of neurons in the hidden layer. In comparison, the RF requires a minimum of two parameters to be optimized—the number of trees in the forest and the minimum tree depth.

The increase in optimization complexity for the RF is very apparent in the timings of the two procedures. The time required to optimize the ANN is about 4 times smaller than the time required to optimize the RF. The time required to train the ANN is about 2.5 times less than that of the RF.

It is interesting to note that, despite the severe difference in optimization and training times, the eventual cross-validation accuracies of the two techniques were comparable.

As such, in response to Research Sub-question 1 which asks "How do the techniques compare in terms of the time taken for optimization and training?", it is stated that the SVM takes considerably less time to optimize and train than the ANN and the RF, and the ANN takes considerably less time to optimize and train than the RF.

5.3 Classification Experimentation

This section describes the experiments performed to determine the recognition accuracy and computational speed of each machine learning technique on the testing set using the optimized and trained models in accordance with Objective 3 set out in Chapter 1. A detailed analysis and discussion of the results is carried out, and the discussion for all three techniques is combined and compared in this case, since the testing procedures were all the same. The comparison of the results between the three machine learning techniques in order to answer Research Sub-questions 2 and 3 is carried out in a subsequent section.

Section 5.3.1 describes the experimental procedure used to obtain the classification accuracy and time results. Section 5.3.2 analyses, compares and discusses the overall classification accuracy and time of the techniques. Sections 5.3.3 and 5.3.4 then analyse, compare and discuss the classification accuracy of each technique in terms of variations across hand shapes and test subjects, respectively, to obtain an indication of the robustness of each technique to such variations.

For ease of reference in this section, Figure 4.8, provided in the previous chapter which depicts the 10 hand shapes recognized, is provided here again in Figure 5.6.

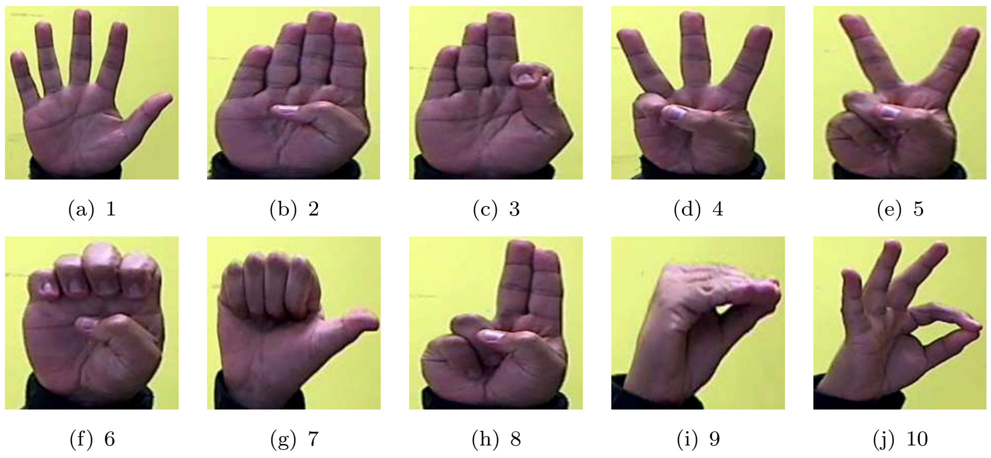


FIGURE 5.6: The ten SASL hand shapes.

5.3.1 Experimental Procedure

The feature vector of each image of the testing set was used as input into the trained classification model of each machine learning technique. As mentioned in the previous chapter, given an input feature vector, each of the models predicts a label between 1 and 10, corresponding to the hand shape class that most accurately matches the input according to that classification model.

As such, for each image in the testing set, the result of the predicted hand shape label was compared with the actual hand shape label of the image to produce a dichotomous outcome as follows: if the two labels matched, the result was recorded as correctly recognized; if the two labels did not match, the result was recorded as incorrectly recognized. These results were then used to determine an overall per-sign and per-subject accuracy for each technique.

In addition to recording the outcome of recognition for each image, the time taken by the classification model to produce an output was also determined. It was found, experimentally, that the time taken to classify a single image was negligibly small since the processing speed of the ANN and RF on a single image was exceptionally high.

Therefore, rather than timing the classification procedure on each image, the procedure was timed on all 3000 images. However, in order to obtain a measure of the standard deviation in the total time, the process of classifying all 3000 images was repeated 3000 times for each technique. For each iteration, the total time taken to classify all 3000 images was recorded. This was used to compute the average total time and the standard deviation in this value.

Before an analysis of the results is carried out, it is very important to take note of the success rate of random guessing in place of each of the classifiers as a base comparison.

For each image of each hand shape, a classification into one of 10 hand shape classes is under taken. A random guess for each image is successful $\frac{1}{10}$ of the time, which is an accuracy of 10%. Taking the average success rate across all the images then also results in an average overall success rate of 10% when using random guessing in place of the classifiers.

An accuracy any higher than this success rate, henceforth referred to as the "guess accuracy", is considered to be superior to random guessing.

5.3.2 Results and Analysis – Overview and Comparison

It is stated at this point that the final Objective 3 set out in Chapter 1 has been successfully achieved.

A comprehensive set of results per subject and hand shape for the SVM, ANN and RF are provided in Tables B.1, B.2 and B.3 in Appendix B. Also, confusion matrices of the hand shape recognition outcome of each classifier are provided in Tables B.4, B.5 and B.6 of the same appendix. For convenience, this and all subsequent sections draw and provide relevant summarized excerpts of the results as required by the discussion and analysis.

Overall, the SVM achieved a classification accuracy of 84.3%, correctly classifying 2529 of the 3000 images in the testing set. It is also very important to note that, in spite of the fact that a dataset that was completely different to Li's dataset was used in this research (since Li's dataset was not available), the hand shape recognition accuracy obtained is very close to Li's hand shape recognition accuracy of 83.3%. This result leads to the following conclusions:

1. It confirms that Li's feature extraction procedure was correctly re-implemented in this research.
2. It clearly demonstrates the robustness of Li's feature extraction procedure.
3. In reverse, it also confirms that the accuracy arrived at by Li was an accurate result.

The ANN, overall, obtained a classification accuracy of 85.9% on the testing set. This is a marginally larger average accuracy than the SVM—1.6% higher. Out of the 3000 images, 2578 were correctly classified.

The Random Forest achieved the lowest, but very comparable, accuracy of 81.3%. It correctly recognized 2440 of the 3000 SASL hand shape testing images.

It is noted that the average accuracies of all three techniques are very comparable, but the ANN achieves a marginally larger accuracy than the SVM and RF. Also, comparing all of these accuracies to the guess accuracy of 10% leads to the realization that all three classifiers perform exceptionally well.

Before analysing the classification time, it should be noted that the classification time and the classification speed are inversely related. As such, a low classification time directly implies a high or fast classification speed. Conversely, a large classification time directly implies a low or slow classification speed. Therefore, this discussion may use these terms interchangeably to reference the classification time which is the subject of analysis.

In terms of classification time, it is observed that the ANN and RF achieve very comparable and exceptionally small classification times indicating very high processing speeds, whereas the SVM is significantly slower than the previous two techniques.

The classification process for the RF was observed to be exceptionally fast, and the fastest of the three techniques. The time to classify all 3000 images in the testing set using this technique was 0.033 seconds with a standard deviation of 0.005 seconds. The classification process for the ANN was also observed to be exceptionally fast. On average, the time to classify all 3000 images in the testing set was only 0.061 seconds with a standard deviation of 0.003 seconds. The SVM was significantly slower than the ANN and RF and took an average of 20.974 seconds with a standard deviation of 0.071 seconds to classify all 3000 images in the testing set.

The average classification speed of the SVM may, at first glance, be considered slow when compared to the speeds achieved by the ANN and RF. However, it should be noted that a time of 20.974 seconds for 3000 images equates to approximately 0.007 seconds per image. Comparing this to the minimum real-time processing speed of 15 frames per second which equates to 0.067 seconds per frame reveals that the time of 0.007 seconds per frame achieved by the SVM is actually approximately 10 times faster than real-time. Therefore, it cannot be said that the SVM is slow. Rather, it should be concluded that the ANN and RF are exceptionally fast.

Overall, although both the RF and ANN achieve exceptionally fast and comparable classification speeds, the RF achieves a faster classification time, which, when compared to the classification time of the ANN, is approximately 2 times faster. In this respect, the RF is the most suitable classifier.

5.3.3 Results and Analysis – Accuracy Per Hand Shape

Table 5.1 summarizes the average accuracy per hand shape class across all test subjects for each machine learning technique and these results are depicted graphically in Figure 5.7.

The table indicates the percentage of images that were correctly classified out of a total of 300 images per hand shape for each machine learning technique. In each row of the table, the accuracy of the highest performing technique for the hand shape corresponding to that row has also been highlighted.

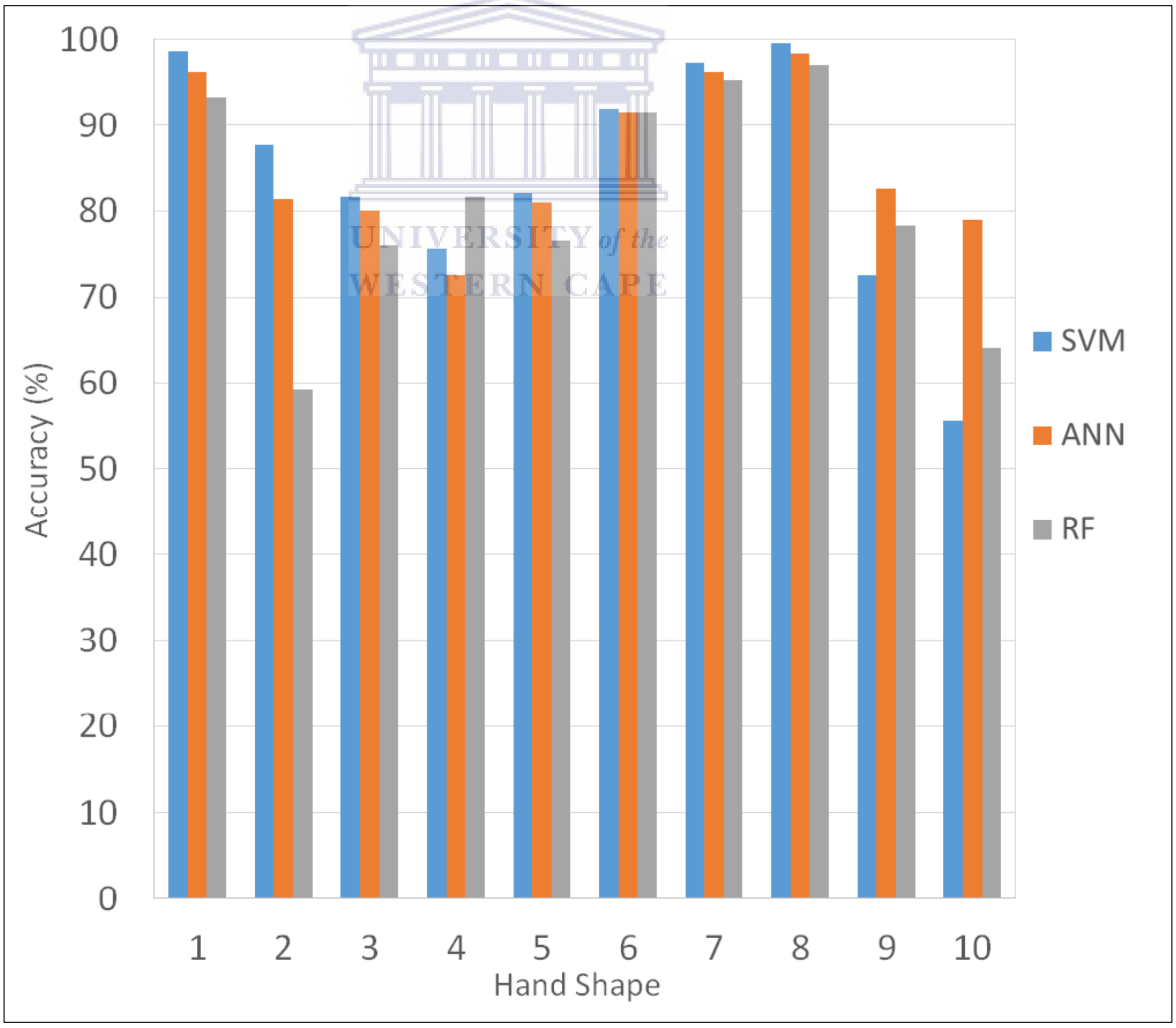


FIGURE 5.7: Average accuracy per hand shape class across all test subjects.

Hand Shape	**Accuracy (%)**		
	SVM	**ANN**	**RF**
1	**98.6**	96.3	93.3
2	**87.6**	81.3	59.3
3	**81.6**	80.0	76.0
4	75.6	72.6	**81.6**
5	**82.0**	81.0	76.6
6	**92.0**	91.6	91.6
7	**97.3**	96.3	95.3
8	**99.6**	98.3	97.0
9	72.6	**82.6**	78.3
10	55.6	**79.0**	64.0
Overall	84.30	**85.93**	81.33

TABLE 5.1: Classification accuracy of the each machine learning technique per hand shape.

Referring to Table 5.1 and Figure 5.7, it is seen that HS 8 consistently achieves the highest accuracy for every technique. The hand shape achieves near-perfect accuracies of 99.6%, 98.3% and 97.0% for the SVM, ANN and RF, respectively.

This consistently high accuracy can be attributed to the unique visual appearance of the hand shape, depicted in Figure 5.6. It is not easily confused with other hand shapes. Only HS 2 may be considered to have some similarity with the hand shape. Both HS 8 and HS 2 have the index and middle fingers raised, but with a major difference in the position of the ring and index fingers which are lowered in the former and raised in the latter. As such, even these two shapes are not very similar. Therefore, it is not surprising that HS 8 obtains the highest accuracy.

The hand shape and value of the lowest accuracy, however, is very different for each machine learning technique. The lowest accuracy obtained by the SVM is for HS 10 with an accuracy of 55.6%. The RF obtains a marginally higher minimum accuracy of 59.3% attributed to HS 2. The ANN obtains the highest minimum accuracy of 72.67% corresponding to HS 4. As such, the ANN has a much smaller range in accuracy across hand shapes and it may be deduced that the ANN is more robust to a large number of hand shape classes.

However, the SVM and RF are also robust to hand shapes. It should be considered that the minimum accuracies of the SVM and the RF are outliers amongst the accuracies of the other 9 hand shapes. Aside from HS 10 of the SVM, it is observed that all other hand shapes achieve very high accuracies of higher than 70%, with 4 hand shapes above 90%, 3 hand shapes above 80% and 2 hand shapes above 70%. A similar observation is made for the RF whereby, apart from HS 2, the RF generally yields high accuracies of higher than 60%, with 4 hand shapes above 90%, 1 hand shape above 80%, 3 hand shapes above 70% and 1 hand shape above 60%.

Furthermore, comparing the accuracies in Table 5.1 on a row-by-row basis and considering the highest value per row highlighted in bold font reveals that the SVM achieves slightly higher accuracies in the majority of individual hand shape classes, with the ANN achieving higher accuracies for HS 9 and HS 10 and the RF achieving a higher accuracy for HS 4.

Also, regarding the lowest accuracies of 55.6% by HS 10 for the SVM and 59.3% by HS 2 for the RF, while these accuracies may, at first glance, be considered low accuracies and will be analysed further in this section, it is important to note that they are still several times larger than the guess accuracy of 10% and should be considered, at the very least, satisfactory or good, if not very good accuracies.

As such, it is clear that the SVM and RF, while less consistent than the ANN in accuracy across hand shapes, still clearly perform exceptionally well.

In conclusion, although the SVM achieves slightly higher accuracies for most individual hands shapes, the ANN is the preferable classifier since the individual hand shape accuracies are only very slightly lower than those of the SVM, but it achieves a greater consistency in high accuracy recognition across all hand shapes than the SVM and RF and a considerably lower range in hand shape accuracy, demonstrating greater consistency and robustness than both other classifiers.

Predicted Class	**Image Count**
1	130
2	0
3	0
4	0
5	0
6	0
7	0
8	3
9	0
10	167
Total	300

Table 5.2: Confusion summary of the SVM for Hand shape 10.

An analysis of the results was carried out to obtain an indication of the cause of the relatively lower accuracies obtained by HS 10 for the SVM and HS 2 for the RF.

Table 5.2 is an excerpt of the confusion matrix of the SVM provided in Appendix B and summarizes the confusion result of only HS 10.

The matrix summarizes the number of images of HS 10 that were classified either correctly or incorrectly as each indicated label by the SVM. For example, the first row of the table after the title row indicates that 130 of the 300 images of HS 10 were misclassified as HS 1. The second last row of the table indicates that 167 of the 300 images of HS 10 were correctly classified as HS 10.

Analysing the table reveals that, with the exception of a very small number of images that were incorrectly classified as HS 8, HS 10 was consistently confused with HS 1, in a total of 130 images—43.4% of the images of HS 10. The fact that the hand shape is consistently confused with one other hand shape indicates, first, that the model is not generally inaccurate since, if this was the case, the 167 incorrect classification cases would have been scattered more evenly across several or all other hand shape classes.

Second, it indicates that the cause of confusion can not be attributed to random variations in the dataset, since the number of incorrect classifications is both high and consistent. Errors attributed to random variations in the data would have, once again, been scattered more evenly across several or all other hand shape classes.

As such, the cause of this confusion is primarily attributed to the underlying classification model, the manner in which the classes are separated in the model, and a strong similarity between the feature vectors of the two classes in feature space.

Comparing the two hand shapes HS 1 and HS 10 in Figure 5.6 reveals that the two hand shapes are visually similar when viewed in a two-dimensional perspective and when considering only the outer contours of the shapes. The two hand shapes have similar positions and poses for the thumb, pinky, ring and middle fingers. It is only the position of the index finger that is different in the two hand shapes. Slight variations in the hand contours of images can easily confuse the classification model in this respect.

Interestingly, this confusion trend is similarly observed for the ANN and RF. Table 5.3 provides the confusion results for HS 10 for all three machine learning techniques.

In the ideal case, the table should contain a total image count of 300 in the row corresponding to the predicted class of 10 for every technique, since this would indicate that all 300 images of HS 10 for each technique were correctly predicted to be HS 10. While a large number of images are predicted as such for all three techniques, it is seen that a number of images are mostly incorrectly predicted as HS 1 for all three techniques. It is very evident that HS 10 is confused with HS 1 in the majority of cases, although this is more pronounced for the SVM than for the ANN or RF.

This observation further strengthens the belief that the hand shapes are intrinsically similar in feature space. In any case, as noted in [23], an analysis of the classification

Predicted Class	**Image Count**		
	SVM	**ANN**	**RF**
1	130	23	76
2	0	2	0
3	0	0	7
4	0	6	0
5	0	0	0
6	0	3	8
7	0	0	3
8	3	6	5
9	0	23	9
10	167	237	192
Total	300	300	300

Table 5.3: Confusion summary of Hand shape 10 for all three machine learning techniques.

result by a classifier can yield an indication of the cause of the classification decision, but it is difficult to determine the exact cause of the decision. What is important is that, overall, the classifier should achieve a high recognition accuracy, which is observed in this case.

Predicted Class	**Image Count**
1	0
2	178
3	108
4	0
5	0
6	9
7	0
8	5
9	0
10	0
Total	300

Table 5.4: Confusion summary of the Random Forest for Hand shape 2.

A similar analysis is carried out for the RF for HS 2 which achieves the lowest accuracy for this technique. Table 5.4 is an excerpt of the confusion matrix of the RF provided in Appendix B and summarizes the confusion result of only HS 2.

Once again, the table should ideally have all 300 images classified correctly as HS 2, but it is observed that 178 images are classified correctly. In a large number of cases—108 cases, 36% of the cases—the hand shape is consistently misclassified as HS 3, and in a very small number of random cases, as HS 6 and HS 8.

As seen in Figure 5.6, HS 2 and HS 3 also appear to be visually similar in a two-dimensional perspective and when considering only the outer contours of the shapes.

Both shapes have the pinky, ring and middle fingers in exactly the same configuration. Although the thumb is differently placed in the two hand shapes, the hand contours of the two shapes are not affected by this difference. In fact, the two hand shapes only significantly differ in the configuration of the index finger, which is raised in HS 2 and lowered in HS 3.

Similar to HS 10, it is interesting to note that the confusion trend of HS 2 is also manifested in the results of the ANN and SVM. Table 5.5 provides the confusion results for HS 2 for all three machine learning techniques.

Predicted Class	**Image Count**		
	SVM	**ANN**	**RF**
1	0	0	0
2	263	244	178
3	35	49	108
4	0	0	0
5	0	0	0
6	2	7	9
7	0	0	0
8	0	0	5
9	0	0	0
10	0	0	0
Total	300	300	300

TABLE 5.5: Confusion summary of Hand shape 2 for all three machine learning techniques.

The table should ideally contain a total image count of 300 in the row corresponding to the predicted class of HS 2 for every technique, since this would indicate that all 300 images of HS 2 for each technique were correctly predicted to be HS 2.

While a large number of images are predicted as such for all three techniques, it is seen that the majority of incorrectly predicted images of HS 2 are misclassified as HS 3 for all three techniques. It is very evident that HS 2 is confused with HS 3 in the majority of cases, although this is more pronounced for the RF than for the SVM or ANN.

This further strengthens the belief that the incorrect predictions for this hand shape are primarily attributed to the intrinsic similarity between the two hand shapes.

5.3.4 Results and Analysis – Accuracy Per Subject

An analysis of the accuracy per subject was carried out to determine the level of robustness of each technique to variations in test subjects. Table 5.6 summarizes the average recognition accuracy of each machine learning technique per test subject, expressed as a percentage of 500 images of each test subject for each technique. In each row of the table, the accuracy of the highest performing technique for the test subject corresponding to that row has also been highlighted. Figure 5.8 depicts these results graphically.

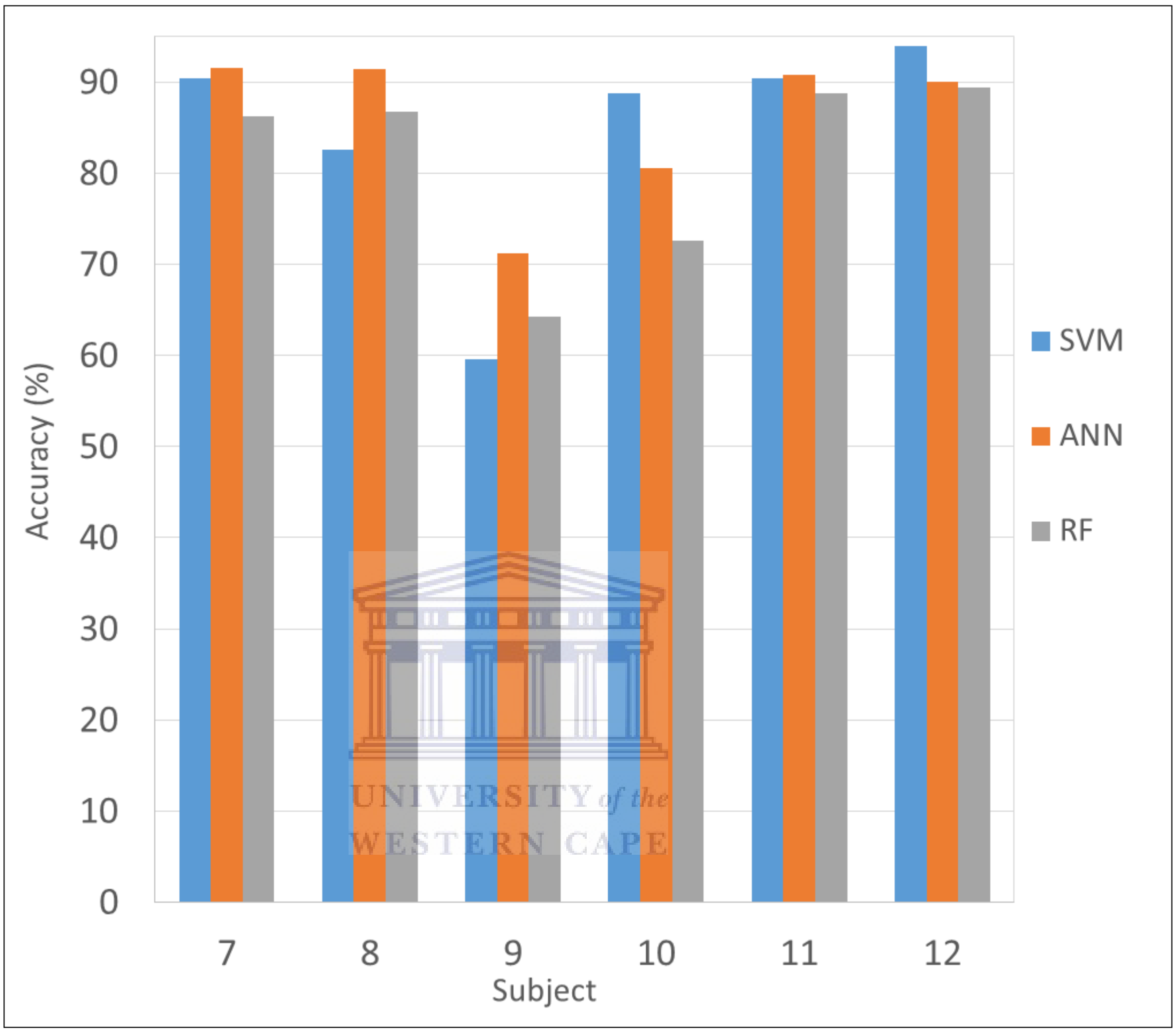


FIGURE 5.8: Average recognition accuracy of each machine learning technique per test subject.

Subject	**Accuracy (%)**		
	SVM	**ANN**	**RF**
7	90.4	**91.6**	86.2
8	82.6	**91.4**	86.8
9	59.6	**71.2**	64.2
10	**88.8**	80.6	72.6
11	90.4	**90.8**	88.8
12	**94.0**	90.0	89.4
Overall	84.30	**85.93**	81.33

TABLE 5.6: Classification accuracy of each machine learning technique per test subject.

Referring to the table and graph, it is observed, first, that Subject 9 consistently achieves the lowest accuracy for all three techniques and is clearly an outlier amongst the subjects for all three techniques. This is analysed in greater detail shortly. With the exception of this subject, it is clear that the majority of other accuracies are consistent, comparable and very high, mostly no less than 80%, but many on or around the 90% mark.

Of the cases, 39% of the cases in the table are above 90% accuracy and a total of 78% of the cases are above 80% accuracy. Comparing these results to the guess accuracy of 10% allows for an appreciation of the robust and exceptionally high accuracies obtained across all test subjects and techniques.

Subject 12 obtains the highest recognition accuracy of 94.0% for the SVM, which means that 94% of 500 images of that subject were correctly classified by the SVM. The RF also appears to have the highest accuracy for Subject 12, but it is noted that this accuracy of 89.4% is very close to the accuracies of other subjects such as Subject 11 for the same technique. For the ANN, the subject that achieves the highest accuracy is Subject 7, the accuracy of which is again only marginally higher than that of other subjects for the same technique.

Even the lowest accuracy of 59.6% for Subject 9 by the SVM translates to very close to 300 of 500 images correctly classified, which is a very encouraging classification performance. It is also encouraging to observe that no subject caused the system to completely fail. Considering 59.6% is very close to 60% and can be considered as such, it can be said that no subject achieved an accuracy below 60%.

This is indicative of a very effective and appropriate feature extraction procedure that yields classification results that are highly robust to variations in test subjects.

Referring to Subject 9, the SVM obtained the lowest accuracy for this subject, closely followed by the RF with an accuracy of 64.2%. While still the lowest value amongst other subjects for the ANN, the accuracy for the same subject for the ANN was considerably higher at 71.2%.

Thus, similar to the accuracy in hand shapes, this implies a considerably smaller range in accuracy across test subjects for the ANN, compared to the SVM and RF. Furthermore, in addition to achieving the highest overall accuracy, it also achieves better accuracies for more individual subjects in this case.

It is clearly concluded that, although all of the classifiers achieve high accuracies across all subjects and are very robust to variations in test subjects, the ANN outperforms the SVM and RF by achieving a higher overall accuracy, better individual accuracies, and a smaller range in accuracies. Therefore, it can be considered more robust to variations in test subjects than the RF and SVM.

Hand Shape	**Accuracy (%)**
1	100
2	100
3	**2**
4	**34**
5	**2**
6	98
7	86
8	100
9	**10**
10	64
Overall	59.6

TABLE 5.7: Classification accuracy of the Support Vector Machine per hand shape for Subject 9.

An analysis was carried out to determine possible causes for the accuracy achieved by Subject 9. Table 5.7 summarizes the recognition accuracies obtained by Subject 9 for each hand shape class as percentages of the total of 50 images of this subject for each hand shape. The table demonstrates that the subject achieves a very low accuracy in only 4 of the 10 hand shapes that have been highlighted in bold font: HS 3, 4, 5 and 9.

For the other hand shapes, however, the subject achieves exceptionally high accuracies, 3 of which are as high as 100%. Therefore, it is clear that the relatively lower average accuracy for this subject is only attributed to a few specific hand shapes, rather than a general intolerance to this subject. The fact that the subject achieves 100% accuracy for HS 1, HS 2 and HS 8 makes it clear that the system is robust to the subject.

In order to obtain an indication of whether these low accuracies are attributed to the classification model, to the intrinsic similarity between the poorly performing hand shapes, or to the data of the specific subject and the manner in which the subject performed these hand shapes in the dataset videos, it is necessary to determine whether other subjects also generally performed poorly for these hand shapes.

Hand Shape	**Accuracy (%)**			
	Subject 9	**Subject 7**	**Subject 10**	**Subject 12**
3	2	100	88	100
4	34	86	80	100
5	2	100	98	100
9	10	100	100	100

TABLE 5.8: Classification accuracy of the Support Vector Machine per hand shape for Subject 9.

Table 5.8 summarizes the accuracies, as percentages of 50 images of each hand shape for each subject, obtained by Subjects 7, 10 and 12, with Subject 9 included for comparison, for HS 3, HS 4, HS 5 and HS 9. The accuracies obtained by Subjects 7, 10 and 12 for these hand shapes are perfect examples of the exceptionally high accuracies, many as high as 100%, that the classification model achieves for other subjects for these hand shapes.

Noting that the accuracies in the table are percentages of 50 images, an accuracy of 100% indicates that every one of the 50 images of the specific hand shape for the specific subject were correctly recognized, which is a very impressive classification performance.

Therefore, this clearly demonstrates that the low accuracies observed for the same hand shapes by Subject 9 can not be attributed to the classification model or to a possible intrinsic similarity between these hand shapes, since if either of these cases were true, the same low-accuracy trend would have been manifested for other subjects as well.

As such, the low accuracy observed for these hand shapes by only Subject 9 can only be attributed to the manner in which these specific hand shapes, and not other hand shapes, were performed by the subject. This statement is further confirmed by the fact that the same subject also achieved very low accuracies for exactly the same hand shapes for the ANN and RF, as demonstrated in Figure 5.9. Figure 5.9 graphically summarizes the recognition accuracy obtained for each hand shape by each machine learning technique for Subject 9.

The figure demonstrates that hand shapes HS 3, 4, 5 and 9 are the lowest performing hand shapes for Subject 9 for all three techniques. It should also be noted that the Subject achieves very high accuracies for all other hand shapes. This further confirms that the low accuracies observed for these specific hand shapes for this specific subject are attributed to the actual data of this subject for only these specific hand shapes, and not to the classification model.

One example of an incorrectly performed hand shape by Subject 9 is provided in Figures 5.10a and 5.10b. Figure 5.10a depicts an example of the subject performing HS 3 in a manner that looks very similar to HS 10 depicted in Figure 5.10b.

5.4 Summary of Comparisons and Selection of the Optimal Technique

This section discusses and compares the recognition accuracies of the three machine learning techniques.

Before comparing the results of the different machine learning techniques, it is necessary to determine the priority of each of the four comparative factors referenced in Research Sub-Questions 1, 2 and 3.

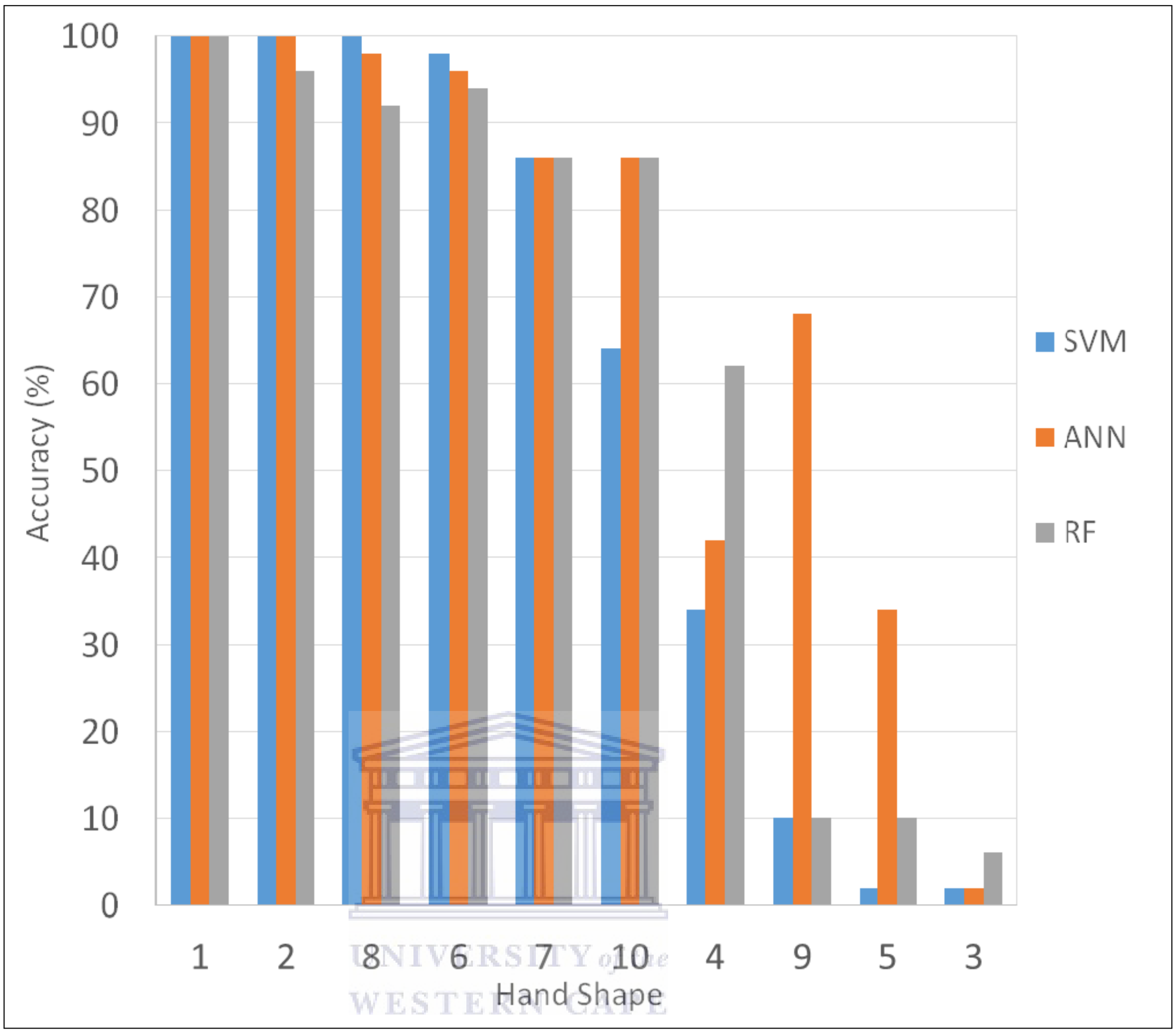


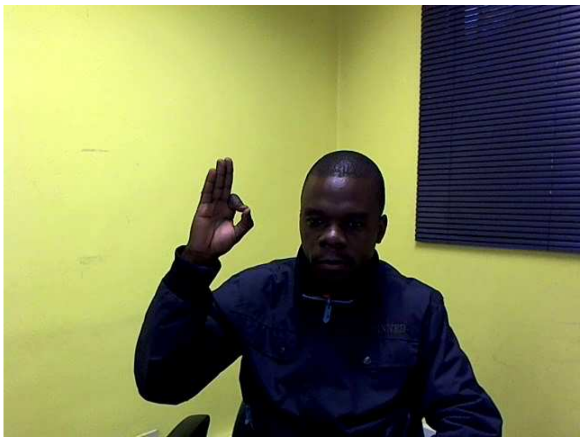
FIGURE 5.9: The recognition accuracy obtained for each hand shape by each machine learning technique for Subject 9.

For the eventual SASL machine translation system, the first and most important factor to consider is the classification accuracy. The classifier must be reliable and, therefore, must achieve high accuracies. Without a high classification accuracy, a classifier may not be considered a classifier at all, depending on the accuracy that it achieves. Therefore, classification accuracy is the most important factor to consider when comparing the techniques.

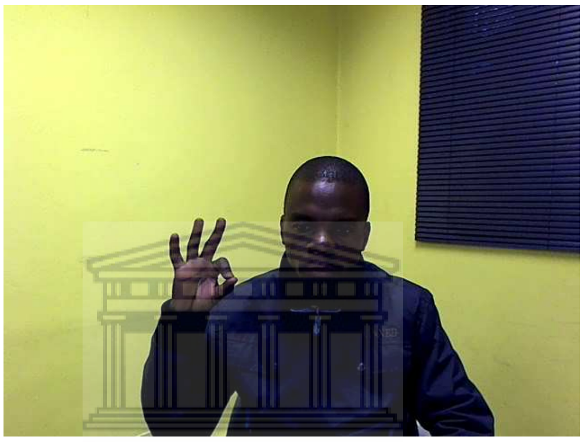
All other factors equal, the technique with a higher classification accuracy will be considered the better technique.

Accuracy includes the overall average classification accuracy of a classifier and the robustness of the classifier to variations in test subjects and various hand shape classes.

The second-most important factor to consider is the classification speed. A classifier must perform its predictions in the shortest time possible while it runs in a real-time



(a) Hand Shape 3



(b) Hand Shape 10

FIGURE 5.10: Subject 9 performing (a) Hand Shape 3 in an incorrect manner that is very similar to Hand shape 10 and (b) Hand Shape 10 performed correctly.

system.

Training and optimization time are important factors and are equally important, but less important than the previous two factors. This is because they are only ever performed once before the system is deployed and can be thought of as a once-off cost. A high resulting accuracy and a fast processing speed can justify a lengthy once-off optimization and training procedure.

Table 5.9 summarizes all the experimental results of all three machine learning techniques. The rows of the table have been divided into three distinct groups. The first group contains factors pertaining to classification accuracy. The second group contains the classification time factor. The third group combines the optimization and training time that are of equal importance and take place together.

It should first be noted that the classification times are presented correct to 3 decimal places, whereas the optimization and training times are presented correct to the nearest

Factor	**SVM**	**ANN**	**RF**
Overall Accuracy (%)	84.3	**85.93**	81.33
Robust to Subjects	High	**Best**	High
Robust to Hand shapes	High	**Best**	High
Classification Time (s)	20.974	0.061	**0.033**
Optimization Time (s)	**109**	3589	14916
Training Time (s)	**21**	39	101

Table 5.9: Summary of results and analysis for all three machine learning techniques.

second. This is because the training and optimization procedures were once-off procedures and were timed only once. The measured time is presented. On the other hand, the classification times were measured over multiple images and iterations, and can be presented at a higher precision.

Referring to the table, it is observed that the ANN performs the best in terms of accuracy: it achieves the highest overall accuracy, although the accuracy of the SVM and RF are comparable to this accuracy. In terms of robustness, however, the ANN is substantially more robust to variations in test subjects and hand shape classes than the SVM and RF, with a substantially smaller range in accuracy, as explained in a previous section. Therefore, it can generally be said that the ANN has the best classification accuracy.

The RF performs the best in terms of classification time/speed: it achieves the shortest classification time of 0.033 seconds for classifying a total of 3000 images. The SVM speed is orders of magnitude slower than this speed, but the ANN speed is very comparable to this speed. It has a comparable classification time of 0.061 seconds for all 3000 images.

The SVM performs the best in terms of optimization and training time: the optimization and training times of the SVM are the shortest at 109 seconds and 21 seconds, respectively. This optimization time is one order of magnitude smaller (and hence faster) than that of the ANN which, in turn, is one order of magnitude smaller than that of the RF.

To conclude, it should first be stated that all the classifiers perform very well in the context of SASL hand shape recognition. However, given the ANN achieves the best accuracy, is the most robust to variations in subjects and hand shapes, and has a classification time that is very comparable to that of the RF, it is concluded that this is the best classifier out of the three techniques for SASL hand shape recognition.

Finally, in response to Research Sub-question 2 which asks "How do the techniques compare in terms of the final classification accuracy on unseen images once they have been optimized and trained?", it is stated that all three techniques are very accurate and comparable in terms of accuracy, but the ANN can be considered more accurate than the SVM and RF, and the SVM can be considered more accurate than the RF.

In response to Research Sub-question 2 which asks "How do the techniques compare in terms of the time taken to achieve a classification result on a single input once they have been optimized and trained?", it is stated that the ANN and RF both take an exceptionally small amount of time in this respect, and are both many orders of magnitude faster than the SVM but the RF takes considerably less time to classify a single image than the ANN.

5.5 The Hand Shape Recognition System results and Nitze et al's experimental results.

Table 5.10 compares the accuracy, training and classification times achieved by the hand shape recognition system on the ten hand shapes and the results achieved by Nitze et al from Table 2.4. The results of Nitze et al's work is denoted by "(Nitze)" and the results of the hand shape recognition system are denoted by "(HSR)" in Table 5.10. From Table 2.4 only the results of the relevant machine learning techniques such as ANN, SVM and RF data were put in the Table 5.10. Both the experiments make use of ten unique classes to be classified. The hand shape recognition system does preprocessing before the recognition can take place and Nitze's system uses a multi-temporal set of RapidEyes images for classification. As shown in Table 5.10 the SVM implementation of Nitze achieves the highest accuracy of 88.1%. The SVM-RBF implementation of Nitze achieves the fastest training time at 0.292 seconds and the ANN of Nitze achieves the fastest classification time of 0.003 seconds.

Machine Learning Technique	**Recognition Accuracy (%)**	**Training Time (s)**	**Classification Time (s)**
ANN (Nitze)	87.1	15.145	0.003
ANN (HSR)	85.9	39	0.061
RF (Nitze)	87.4	6.205	0.083
RF(HSR)	81.33	101	0.033
SVM-RBF (Nitze)	88.1	0.292	0.039
SVM (HSR)	84.3	21	20.974

Table 5.10: Timing and accuracy results of the hand shape recognition (HSR) and Nitze et al (Nitze).

5.6 Summary and Conclusion

This chapter discussed the experimentation carried out to optimize and train the three machine learning techniques and the experimentation carried out to evaluate the classification accuracy and speed of the three techniques. These experiments ultimately yielded clear answers to the three research sub-questions set out in Chapter 1.

At this stage, it is possible to provide an answer to the main research question which was posed as follows: "How do Support Vector Machines, Artificial Neural Networks and Random Forests compare in the context of SASL hand shape recognition?".

In response to this question, it is stated that the SVM is considerably quicker to optimize and train that the ANN and RF, the ANN is more accurate and consistent than the SVM and RF, given a trained and optimized classification model, and the RF is considerably faster when it comes to classifying a single input image, given a trained and optimized classification model, than the ANN and SVM.

A detailed analysis and discussion of the results culminated in the selection of the ANN machine learning technique as the best technique for SASL hand shape recognition, as compared to the SVM and RF. The motivation for this choice was the fact that it achieves a higher and more consistent accuracy than both other techniques and has a classification time that is comparable to that of the RF. While the optimization and training time was considerably higher than that of the SVM, this is a once-off cost that can definitely be worth it, given the final classifier is more accurate and much faster.

It was also found that the classification accuracy result obtained by Li using a completely different dataset to the one used in this research was very close to the classification accuracy achieved by the SVM in this research. This demonstrates the robustness and accuracy of the framework, and demonstrates that the feature extraction procedure was re-implemented correctly in this research.

The next chapter concludes the thesis.

Chapter 6

Conclusion

This research aimed to compare the use of Support Vector Machines (SVMs), used extensively in the SASL research group [2, 12, 44, 45, 69], with other promising machine learning techniques, in this case Artificial Neural Networks (ANNs) and Random Forests (RFs) in the context of SASL hand shape recognition. Four factors were considered in this comparison, namely: classification accuracy which is the most important factor, classification speed which is the second-most important factor, and the time required to optimize and train the technique, which are both very important, but not as important as the two previous factors.

In response to the first research sub-question posed as "How do the techniques compare in terms of the time taken for optimization and training?", it was concluded that the SVM takes considerably less time to optimize and train than the ANN and the RF, and the ANN takes considerably less time to optimize and train than the RF.

In response to the second research sub-question posed which asked "How do the techniques compare in terms of the final classification accuracy on unseen images once they have been optimized and trained?", it was concluded that while all three techniques are very accurate and comparable in terms of classification accuracy, all three achieving exceptionally high overall accuracies of over 80%, the ANN can be considered more accurate than the SVM and RF, and the SVM can be considered more accurate than the RF.

In response to the third and final research sub-question posed as "How do the techniques compare in terms of the time taken to achieve a classification result on a single input once they have been optimized and trained?", it was concluded that the ANN and RF both take an exceptionally small amount of time to classify a single image, and are both many orders of magnitude faster than the SVM, but the RF takes considerably less time to classify a single image than the ANN.

Therefore, and finally, in response to the main research question which was phrased as "How do Support Vector Machines, Artificial Neural Networks and Random Forests compare in the context of SASL hand shape recognition?", it was concluded that the ANN is more accurate and consistent than the SVM and RF, the SVM is considerably quicker to optimize and train than the ANN and RF given a trained and optimized classification model, and the RF is considerably faster than the ANN and SVM when it comes to classifying a single input image given a trained and optimized classification model.

Overall, it was concluded that the ANN is the most suitable classifier, given it is the most accurate and consistent classifier, and has an exceptionally high classification speed that is comparable to that of the RF, and both of these factors justify the optimization and training time which is more than the SVM but less than the RF.

These finding have made a significant contribution to the field of hand shape recognition, and to the research of the SASL project. They have clearly demonstrated that the basis of this research—carrying out a comparison of machine learning techniques in the context of a specific classification problem—is crucial. This is because, while an arbitrary machine learning technique such as SVMs can serve as a good classifier, as was used originally by Li [36], this research has shown that it may not be, and in the context of SASL hand shape recognition, is not the optimal choice.

This research has also significantly contributed to the SASL group, first, by producing an improved SASL hand shape classifier. More importantly, it has produced a methodology that can be used in future to determine optimal machine learning techniques for other classification problems such as facial expression, hand location, hand orientation and hand motion recognition.

6.1 Future Work

The ANN technique has proven to be the better technique amongst the three chosen machine learning techniques in the context of SASL hand shape recognition. In future, the ANN-based system can, therefore, be incorporated into the SASL gesture recognition system to achieve an improved accuracy, and exceptional computational speed.

This research provides a basis and methodology for comparing machine learning techniques in a specific context. In future, this approach can be used to determine optimal classifiers for each of the SASL systems that recognize various SASL parameters.

Finally, while this research has determined that ANNs are better than SVMs and RFs, the investigation may be extended to other machine learning techniques such as Naive Bayes classifiers and Hidden Markov Models, which may prove to be better than ANNs for SASL hand shape recognition.

6.2 Concluding Remarks

The researcher has found the research and experiments conducted throughout this course to have been an excellent growth experience. It is hoped that this research can serve as a basis and methodology for the selection of optimal machine learning techniques for other sign language parameters by the SASL group, and for classification problems in general.

Appendix A

Additional Optimization Results

Number of Hidden Neurons m	**Accuracy (%)**	**Number of Hidden Neurons** m	**Accuracy (%)**
2	18.33	27	74.80
3	29.53	28	74.96
4	36.36	29	73.40
5	44.50	30	75.03
6	50.40	31	75.73
7	63.73	32	74.76
8	67.63	33	74.46
9	68.43	34	74.73
10	63.26	35	74.13
11	70.73	36	75.23
12	71.96	37	75.00
13	73.93	38	73.70
14	71.40	39	75.40
15	69.30	40	76.23
16	74.70	41	74.93
17	73.37	42	75.97
18	74.30	43	73.57
19	72.37	44	74.13
20	74.73	45	75.33
21	73.60	46	76.70
22	74.13	47	76.26
23	74.70	48	76.53
24	73.66	49	74.43
25	74.03	50	76.77
26	71.67		

(cont. right)

Table A.1: The hidden neurons and their corresponding cross-validation accuracies

No. of Trees	**Depth $D_{\mathbf{min}}$**									
B	2	4	6	8	10	12	14	16	18	20
5	19.50	29.43	36.30	40.33	42.07	43.10	49.20	43.87	46.40	45.47
10	20.30	35.20	45.73	48.40	52.57	53.63	57.27	55.50	54.03	53.53
15	25.00	40.80	48.03	53.60	59.57	59.53	59.80	62.33	59.73	60.33
20	30.67	41.36	53.30	56.60	61.60	65.30	64.07	63.67	63.73	65.20
25	30.23	41.93	52.93	59.90	62.83	67.03	65.90	65.50	64.93	67.30
30	31.30	44.83	54.30	60.10	65.30	68.20	66.90	65.67	65.23	68.20
35	33.20	45.10	54.33	61.67	64.97	68.80	67.13	65.63	65.23	68.37
40	33.57	46.20	54.67	64.67	65.90	69.57	67.13	65.63	65.23	68.37
45	34.73	47.07	56.20	65.03	66.90	69.97	67.13	65.63	65.23	68.37
50	35.97	47.77	56.40	66.40	66.80	70.03	67.13	65.63	65.20	68.37
55	38.93	47.73	56.53	66.60	66.93	70.03	67.13	65.63	65.23	68.37
60	37.90	48.93	58.27	67.33	66.90	70.03	67.13	65.63	65.23	68.37
65	38.23	49.83	59.27	67.23	67.27	70.03	67.13	65.63	65.23	68.37
70	38.23	49.83	59.27	67.23	67.27	70.03	67.13	65.63	65.23	68.37
75	39.07	49.73	59.50	67.57	67.27	70.03	67.13	65.63	65.23	68.37
80	38.83	51.43	61.13	67.83	67.63	70.03	67.13	65.63	65.23	68.37
85	39.70	50.93	60.67	67.83	67.63	70.03	67.13	65.63	65.23	68.37
90	40.13	50.77	60.50	67.80	67.63	70.03	67.13	65.63	65.23	68.37

Table A.2: The cross-validation accuracies for Random Forests

Appendix B

Additional Test Results

Subject	**Hand Shape**									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
7	50	46	50	43	50	50	49	50	50	14
8	50	49	50	50	49	50	50	50	13	2
9	50	50	1	17	1	49	43	50	5	32
10	46	48	44	40	49	49	50	49	50	19
11	50	48	50	27	47	30	50	50	50	50
12	50	22	50	50	50	48	50	50	50	50
Total	296	263	245	227	246	276	292	299	218	167

Table B.1: Classification accuracy per subject of the Support Vector Machine.

Subject	**Hand Shape**									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
7	50	45	50	41	50	49	48	48	50	27
8	50	49	50	49	48	50	50	49	14	48
9	50	50	1	21	17	48	43	49	34	43
10	44	50	39	24	31	49	48	49	50	19
11	50	44	50	33	47	30	50	50	50	50
22	45	6	50	50	50	49	50	50	50	50
Total	289	244	240	218	243	275	289	295	248	237

Table B.2: Classification accuracy per subject of the Artificial Neural Network.

Subject	**Hand Shape**									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
7	49	20	50	48	48	49	50	47	50	20
8	44	47	50	50	47	50	50	50	30	16
9	50	48	3	31	5	47	43	46	5	43
10	38	17	25	41	35	50	43	48	50	16
11	49	44	50	25	45	31	50	50	50	50
12	50	2	50	50	50	48	50	50	50	47
Total	280	178	228	245	230	275	286	291	235	192

Table B.3: Classification accuracy per subject of the Random Forest.

Actual	**Predicted**									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	**296**	0	0	3	0	0	0	0	1	0
2	0	**263**	35	0	0	2	0	0	0	0
3	0	4	**245**	0	0	0	6	2	0	43
4	66	0	0	**227**	7	0	0	0	0	0
5	46	1	0	3	**246**	0	0	4	0	0
6	0	24	0	0	0	**276**	0	0	0	0
7	1	0	2	0	2	0	**292**	1	2	0
8	0	0	1	0	0	0	0	**299**	0	0
9	0	0	0	41	0	0	35	6	**218**	0
10	130	0	0	0	0	0	0	3	0	**167**

Table B.4: Confusion matrix for the recognition accuracy of the Support Vector Machine.

Actual	**Predicted**									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	**289**	0	0	1	0	0	3	1	1	5
2	0	**244**	49	0	0	7	0	0	0	0
3	0	4	**240**	1	0	0	1	5	1	48
4	13	4	0	**218**	53	0	9	0	1	2
5	19	0	0	17	**243**	0	1	10	0	10
6	0	25	0	0	0	**275**	0	0	0	0
7	5	0	2	1	2	0	**289**	0	0	1
8	0	1	0	1	2	0	0	**295**	1	0
9	0	0	0	10	0	0	36	6	**248**	0
10	23	2	0	6	0	3	0	6	23	**237**

Table B.5: Confusion matrix for the recognition accuracy of the Artificial Neural Network.

Actual	**Predicted**									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	**280**	1	2	0	4	0	6	1	4	2
2	0	**178**	108	0	0	9	0	5	0	0
3	7	15	**228**	0	0	0	0	1	0	49
4	31	0	0	**245**	9	0	8	0	0	7
5	6	2	0	1	**230**	0	1	55	0	5
6	0	24	1	0	0	**275**	0	0	0	0
7	1	1	2	0	1	0	**286**	2	2	5
8	0	1	2	2	0	4	0	**291**	0	0
9	0	14	0	12	0	14	23	2	**235**	0
10	76	0	7	0	0	8	3	5	9	**192**

TABLE B.6: Confusion matrix for the recognition accuracy of the Random Forest.

Bibliography

[1] S. Abu-Nimeh, D. Nappa, X. Wang, and S. Nair, "A comparison of machine learning techniques for phishing detection," in *Proceedings of the Anti-phishing Working Groups 2nd Annual eCrime Researchers Summit.* ACM, 2007, pp. 60–69.

[2] 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.

[3] I. Achmed, I. M. Venter, and P. Eisert, "A framework for independent hand tracking in unconstrained environments," in *Proceedings of the Southern Africa Telecommunication Networks and Applications Conference 2012*, George, South Africa, 2012.

[4] M. Ali and D. Clausi, "Using the Canny edge detector for feature extraction and enhancement of remote sensing images," in *Geoscience and Remote Sensing Symposium, 2001. IGARSS'01*, vol. 5, 2001, pp. 2298–2300.

[5] K. Asmal and W. James, "Education and democracy in South Africa today," *Daedalus*, pp. 185–204, 2001.

[6] H. Avilés-Arriaga, L. Sucar-Succar, C. Mendoza-Durán, and L. Pineda-Cortés, "A comparison of dynamic naive Bayesian classifiers and Hidden Markov Models for gesture recognition," *Journal of Applied Research and Technology*, vol. 9, no. 1, pp. 81–100, 2011.

[7] G. Bebis and M. Georgiopoulos, "Feed-forward neural networks," *Potentials, IEEE*, vol. 13, no. 4, pp. 27–31, 1994.

[8] G. Borgefors, "An improved version of the chamfer matching algorithm," in *Proceedings of the 7th International Conference on Pattern Recognition*, vol. 2, Montreal, Canada, 1984, pp. 1175–1177.

[9] T. Borovicka, M. Jirina Jr, P. Kordik, and M. Jirina, "Selecting representative data sets," *Advances in Data Mining Knowledge Discovery and Applications. Intech*, 2012.

[10] G. Bradski, "Real time face and object tracking as a component of perceptual user interface," in *Proceedings of the Fourth IEEE Workshop on Applications of Computer Vision*, 1998, pp. 214–219.

[11] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[12] D. Brown, "Upper body pose recognition and estimation towards the translation of South African Sign Language," Master's thesis, University of the Western Cape, Computer Science, 2013.

[13] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 6, pp. 679–698, 1986.

[14] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[15] A. Criminisi and J. Shotton, *Decision forests for computer vision and medical image analysis.* Springer, 2013.

[16] M. Dewar, "Characterization and evaluation of aged 20Cr32Ni1Nb stainless steels," Master's thesis, Department of Chemical and Materials Engineering, University of Alberta, Canada, 2013.

[17] A. Elgammal, C. Muang, and D. Hu, *Skin detection—A short tutorial.* Springer, 2009.

[18] M. M. Fleck, D. A. Forsyth, and C. Bregler, "Finding naked people," in *Computer VisionECCV'96.* Springer, 1996, pp. 593–602.

[19] H. Freeman, "Computer processing of line-drawing images," *ACM Computing Surveys*, vol. 6, no. 1, pp. 57–97, 1974.

[20] I. Frieslaar, "Robust south african sign language gesture recognition using hand motion and shape," Master's thesis, University of the Western Cape, Computer Science, 2014.

[21] 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.

[22] M. Glaser and W. D. Tucker, "Telecommunications bridging between deaf and hearing users in South Africa," in *Proceedings of the Conference and Workshop on Assistive Technologies for People with Vision and Hearing Impairments*, Granada, Spain, 2004.

[23] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning.* Springer, 2009.

[24] R. Hecht-Nielsen, “Theory of the backpropagation neural network,” in *International Joint Conference on Neural Networks, IJCNNA’89*, 1989, pp. 593–605.

[25] C. Hsu, C. Chang, and C. Lin, “A practical guide to support vector classification,” National Taiwan University, Tech. Rep., 2003.

[26] 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.

[27] 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.

[28] M. Jones and J.M.Rehg, “Statistical color models with application to skin detection,” Cambridge Research Laboratory, Tech. Rep., 1996.

[29] M. Kadous, “Machine recognition of Auslan signs using Powergloves:towards large-lexicon recognition of sign language,” Master’s thesis, University of New South Wales, Computer Science and Engineering, 1995.

[30] B. Karlik and A. V. Olgac, “Performance analysis of various activation functions in generalized MLP architectures of neural networks,” *International Journal of Artificial Intelligence and Expert Systems*, vol. 1, no. 4, pp. 111–122, 2011.

[31] E. Keating and G. Mirus, “American sign language in virtual space: Interactions between deaf users of computer-mediated video communication and the impact of technology on language practices,” *Language in Society*, vol. 32, no. 5, pp. 693–714, 2003.

[32] N. B. Kiyaga and D. F. Moores, “Deafness in sub-Saharan Africa,” *American Annals of the Deaf*, vol. 148, no. 1, pp. 18–24, 2003.

[33] V. S. Kulkarni and S. Lokhande, “Appearance based recognition of American Sign Language using gesture segmentation,” *International Journal on Computer Science and Engineering*, vol. 2, no. 03, pp. 560–565, 2010.

[34] A. Kuznetsova, L. Leal-Taixé, and B. Rosenhahns, “Real-time sign language recognition using a depth camera,” in *IEEE International Conference on Computer Vision Workshops*, 2013, pp. 83–90.

[35] Y. Lee, S. Min, H. Yang, and K. Jung, “Motion sensitive glove-based Korean finger-spelling tutor,” in *Proceedings of the 2007 International Conference on Convergence Information Technology*, ser. ICCIT ’07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 1674–1677.

[36] P. Li, “Hand shape estimation for South African Sign Language,” Master’s thesis, University of the Western Cape, Computer Science, 2010.

[37] R. Maini and H. Aggarwal, “Study and comparison of various image edge detection techniques,” *International Journal of Image Processing (IJIP)*, vol. 3, no. 1, pp. 1–11, 2009.

[38] S. Marcel, “Hand posture and gesture datasets,” [Online] Available at http://www.idiap.ch/resource/gestures.

[39] A. Maruch, “Talking with the hearing-impaired,” January 2010, [Online] Available at http://www.deafsa.co.za/index-2.html.

[40] W. S. McCulloch and W. Pitts, “A logical calculus of the ideas immanent in nervous activity,” *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[41] K. Mehrotra, C. K. Mohan, and S. Ranka, *Elements of Artificial Neural Networks.* MIT Press, 1997.

[42] M. Minsky and S. Papert, *Perceptrons.* MIT Press, 1969.

[43] M. Minsky and S. Papert, *Perceptrons: An introduction to computational geometry, Expanded Editon.* MIT Press, 1988.

[44] D. Mushfieldt, “Robust facial expression recognition in the presence of rotation and partial occlusion,” Master’s thesis, University of the Western Cape, Computer Science, 2014.

[45] W. Nel, “An integrated sign language recognition system,” Master’s thesis, University of the Western Cape, Computer Science, 2014.

[46] T.-N. Nguyen, H.-H. Huynh, and J. Meunier, “Static hand gesture recognition using artificial neural network,” *Journal of Image and Graphics*, 2013.

[47] I. Nitze, U. Schulthess, and H. Asche, “Comparison of machine learning algorithms random forest, artificial neural network and support vector machine to maximum likelihood for supervised crop type classification,” in *Proceedings of the 4th Geobia, Rio de Janeiro—Brazil*, 2012.

[48] OpenCV documentation, “Introduction to support vector machines,” 2014. [Online]. Available: http://docs.opencv.org/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html

[49] V. Patil and S. Shimpi, “Handwritten English character recognition using neural network,” *Elixir Computi. Sci. Eng.*, vol. 41, pp. 5587–5591, 2011.

[50] J. C. Platt, N. Cristianini, and J. Shawe-Taylor, "Large margin dags for multiclass classification." in *Advances in Neural Information Processing Systems 12*, 1999, pp. 547–553.

[51] N. Pugeault, "The ASL finger spelling dataset," [Online] Available at http://personal.ee.surrey.ac.uk/Personal/N.Pugeault/index.php?section=FingerSpellingDataset.

[52] A. Réda and B. Aoued, "Artificial neural network-based face recognition," in *First International Symposium on Control, Communications and Signal Processing, 2004.*, 2004, pp. 439–442.

[53] K. O. Rodríguez and G. C. Chávez, "Finger spelling recognition from RGB-D information using kernel descriptor," in *Proceedings of the 2013 XXVI Conference on Graphics, Patterns and Images*, ser. SIBGRAPI '13. Washington, DC, USA: IEEE Computer Society, 2013, pp. 1–7.

[54] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.

[55] H. Rowley, "Frontal face images," [Online] Available at http://vasc.ri.cmuedu/ idb/ html/ face/frontal_images/.

[56] M. Sharifi, M. Fathy, and M. Tayefeh Mahmoudi, "A classified and comparative study of edge detection algorithms," in *Proceedings of the International Conference on Information Technology: Coding and Computing, 2002*, 2002, pp. 117–120.

[57] C. Stauffer and W. Grimson, "Adaptive background mixture models for real-time tracking," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 246–252, Jun. 1999.

[58] C. Stobbart and E. Alant, "Home-based literacy experiences of severely to profoundly deaf preschoolers and their hearing parents," *Journal of Developmental and Physical Disabilities*, vol. 20, no. 2, pp. 139–153, 2008.

[59] W. C. Stokoe, "Sign language structure: An outline of the visual communication systems of the american deaf. studies in linguistics: Occasional papers," Buffalo: Dept. of Anthropology and Linguistics, University of Buffalo, Tech. Rep. 8, 1960.

[60] Y. Tabata and T. Kuroda, "Finger spelling recognition using distinctive features of hand shape," in *International Confernece on Disability, Virtual Reality and Associated Technologies with Art Abilitation*, no. 7, 2008, pp. 287–292.

[61] Y. Tabata, T. Kuroda, and K. Okamoto, "Development of a glove-type input device with the minimum number of sensors for Japanese finger spelling," in *Proceedings of*

International Conference on Disability, Virtual Reality and Associated Technologies, no. 9, 2012, pp. 305–310.

[62] P. Trigueiros, F. Ribeiro, and L. P. Reis, "A comparison of machine learning algorithms applied to hand gesture recognition," in *Proceedings of the 7th Iberian Conference on Information Systems and Technologies (CISTI)*. IEEE, 2012, pp. 1–6.

[63] P. E. Utgoff, "Incremental induction of decision trees," *Machine learning*, vol. 4, no. 2, pp. 161–186, 1989.

[64] D. van Wyk, "Virtual human modelling and animation for sign language visualisation," Master's thesis, University of the Western Cape, Computer Science, 2008.

[65] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the IEEE Computer Society International Conference on Computer Vision and Pattern Recognition*, 2001, pp. 511–518.

[66] P. Viola and M. Jones, "Robust real-time object detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004.

[67] L. Wang, Y. Hang, S. Luo, X. Luo, and X. Jiang, "Deblurring Gaussian-blur images: A preprocessing for rail head surface defect detection," in *Proceedings of IEEE International Conference on Service Operations, Logistics, and Informatics (SOLI)*, 2011, pp. 451–456.

[68] Y. Wang, H. Ai, B. Wu, and C. Huang, "Real time facial expression recognition with adaboost," in *Proceedings of the 17th International Conference on Pattern Recognition (ICPR)*, vol. 3. IEEE, 2004, pp. 926–929.

[69] J. Whitehill, "Automatic real-time facial expression recognition for signed language translation," Master's thesis, University of the Western Cape, Computer Science, 2006.

[70] J. Whitehill and C. W. Omlin, "Haar features for FACS AU recognition," in *7th International Conference on Automatic Face and Gesture Recognition, 2006*. IEEE, 2006, pp. 97–101.