UNIVERSITY OF THE WESTERN CAPE

Hand Shape Estimation for South African Sign Language

by

Pei Li

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

in the
Faculty of Science
Department of Computer Science

March 2012

Supervisor: James Connan
Co-supervisor: Mehrdad Ghaziasgar

Declaration of Authorship

I declare that *Hand Shape Estimation for South African Sign Language* is my own work, that it has not been submitted for any degree or examination in any other university, and that all the sources I have used or quoted have been indicated and acknowledged by complete references.

Signed: _____

Date: _____

UNIVERSITY OF THE WESTERN CAPE

Abstract

Faculty of Science
Department of Computer Science

Master of Science

by Pei Li

Hand shape recognition is a pivotal part of any system that attempts to implement Sign Language recognition. This thesis presents a novel system which recognises hand shapes from a single camera view in 2D. By mapping the recognised hand shape from 2D to 3D, it is possible to obtain 3D co-ordinates for each of the joints within the hand using the kinematics embedded in a 3D hand avatar and smooth the transformation in 3D space between any given hand shapes. The novelty in this system is that it does not require a hand pose to be recognised at every frame, but rather that hand shapes be detected at a given step size. This architecture allows for a more efficient system with better accuracy than other related systems. Moreover, a real-time hand tracking strategy was developed that works efficiently for any skin tone and a complex background.

Acknowledgements

My sincerest gratitude and appreciation is given to my supervisor James Connan. His continuous inspiration, guidance and encouragement has instilled in me the motivation to succeed. My co-supervisor Mehrdad Ghaziasgar, his patience, perseverance and the valuable time he spent guiding me in my research is invaluable to me. Thank you for your assistance and guidance.

During my research, Imran Achmed has shared his experience with me. His sound advice and constructive comments, from the beginning of my research and until the end, was of great benefit. An enormous thanks goes to him for his generous assistance.

I am indebted to my parents, who taught me the value of hard work and who tried their best to help me in my studies in whichever way they could, even though they never had the opportunity to study towards a tertiary education themselves. This enormous support has pushed me to excel. I could not have reached this goal without them.

Last but not least, a special thank you to my wife, Liu Yang. Her love, patience, support and understanding throughout my studies, has given me the momentum to see through each year. She is my source of inspiration.

Contents

List of Figures

List of Tables

Abbreviations

2D	- Two Dimensional
3D	- Three Dimensional
CamShift	- Continuously Adaptive Mean Shift
CPU	- Computer Processing Unit
DF	- Distance Transform
DOF	- Degree Of Freedom
EM	- Expectation Maximisation
FPS	- Frames Per Second
GB	- GigaByte
GMM	- Gaussian Mixture Model
GHz	- GigaHertz
H-Anim	- Humanoid Animation Working Group
HSV	- Hue, Saturation, Value
HCI	- Human Computer Interaction
LibSVM	- Library of Support Vector Machines
MB	- MegaByte
MHz	- MegaHertz
MSEPF	- Mean-Shift Embedded Particle Filter
NN	- Neural Network
PC	- Personal Computer
PDI	- Probability Colour Density Image
RAM	- Random Access Memory
RBF	- Radial Basis Function
ROI	- Regions of Interest
SASL	- South African Sign Language
SIR	- Sequential Importance Re-sampling
SVM	- Support Vector Machine
UWC	- University of the Western Cape
YCbCr	- Luminance and Chrominance

Chapter 1

Introduction

1.1 Background and Motivation

The ability to communicate is a necessary skill used in places such as schools, workplaces, shopping malls, hospitals and homes. An individual's ability to study, work, shop, see a doctor and chat with family or friends depends solely on that individual's ability to communicate.

Many deaf people use a sign language as their primary language. Sign languages use a different modality of communication than spoken languages. Sign language signs are characterized by 6 parameters: hand motion, orientation, shape and location, and non-manual gestures such as facial expressions, and body movements. Just as spoken languages are primarily communicated using voice, sign languages are primarily communicated using the hands.

As a result, many deaf people are completely unable to communicate with hearing people. The barrier to such communication is the difference in, both, the language and the modality of communication. Therefore, many deaf people are faced with many social problems. Those that primarily use sign language to communicate are unable to communicate with hearing people even in their own area, home town, and country.

In South Africa, there are 600,000 deaf people that use South African Sign Language (SASL) [30] as their primary language. In comparison with hearing people, deaf people in South Africa receive very few opportunities to study and work. 75% of the South African deaf are functionally illiterate and 70% are unemployed [30].

Many institutes and research groups, many of which are based at universities, have attempted to use modern techniques to help deaf people breach this barrier. Such attempts

[19, 52, 77, 110] take the form of systems that convert voice to sign language and/or sign language to voice. The "Integration of Signed and Verbal Communication: Sign Language Recognition, Animation and Translation" group, more conveniently called the SASL group [25], founded at the University of the Western Cape, is one such group. This group has made many advances in its goal of developing a system to convert sign language to voice and voice to sign language using commodity hardware. A main objective of the group is to make this system as natural as possible by tending away from cumbersome devices such as data suits, data gloves and colour-coded clothes.

Many research projects have been conducted within the group. Whitehill [113] developed a robust facial expression recognition system. Naidoo [79] developed a whole-gesture recognition system to interpret entire sign language gestures to English. This system lacked the ability to recognise and include the effects of different hand shapes in the same gesture. Segers [89] developed a hand shape recognition system. However, this system is only able to operate on a static background and with only the hand in the frame. Using the 3D humanoid avatar developed by van Wyk [109], Achmed [2] developed a system that can estimate the movements of the arms in 3D space. However, no system currently exists in the group to estimate the shape and motion of the hands.

The aim of this research is to develop a system that can track, recognise and estimate hand shapes from a video sequence on a non-static background. It has been mentioned that the hand parameters form 4 of the 5 parameters that wholly characterize a sign language sign. Therefore, this research forms one of the most pivotal parts of the SASL project.

1.2 Research Problem

Sign language signs contain varied hand shapes. During the course of speech, the hand shape may transition from one hand shape to another several times. The human hand is an articulated object with many degrees of freedom. Therefore, the hand is highly deformable [55, 66, 99]. Also, hand motions and deformations are often characterized by self-occlusions. These factors imply that it is difficult to extract hand information pertaining to sign language.

Data gloves and data suits can be used as reliable sources of such hand information [61, 116]. However, these are very costly and cumbersome and reduce the natural feel of the system. This goes against the objective of the SASL group. The use of computer vision, which uses image processing and machine learning algorithms, can make the extraction of such hand information from a video feed possible. Such a solution constitutes

a comfortable non-contact user interface using commodity hardware – a commodity webcam – as required by this research.

In a bid to simplify the extraction of information from a video feed, many studies have added the requirement of colour-coded markers to be worn by users of the system – also known as contact solutions – and/or have used multi-view cameras. Such constraints defy the objective of the SASL group. Therefore, the problem exists to be able to extract hand information from a video sequence without contact solutions and using a monocular view.

The problem of extracting hand information in a video sequence can be broken down into three sub-problems:

1. Detecting and tracking the hand in each frame.
2. Recognizing the hand shape in the frame. This can be a complicated process since there are a multitude of different hand shapes formed by the hand that need to be recognised.
3. Estimating intermediate hand shapes in the transition between two recognised hand shapes.

Considering South Africans are of many different skin tones, the SASL project requires that this system is able to work for people of any skin tone. It also requires that this system be able to work in different environments (simple or complex backgrounds).

1.3 Research Question

Referring to the challenges mentioned in the previous section, the following research question can be used to guide the research process: “Can South African Sign Language hand shapes be recognised accurately and estimated for a person of any skin tone from a monocular video feed on an arbitrary background?”.

1.4 Research Objectives

The objective of this research is to develop a system that can accurately:

1. Detect and track the hand in each frame.

2. Recognise the hand shape of the detected hand.

3. Estimate intermediate hand shapes in the transition between two recognised hand shapes.

4. The system should also be robust and accommodate a wide range of skin tones and users.

1.5 Thesis Outline

The remainder of the thesis is outlined as follows:

Chapter 2: *Related Work*: This chapter reviews the current techniques for hand tracking, hand shape recognition and estimation. Based on these techniques, techniques suitable for this research are proposed.

Chapter 3: *Hand Tracking*: This chapter describes the hand tracking strategy used in this research. The procedures involved in this process are also discussed. These are: adaptive skin detection, Gaussian mixture model background subtraction, hand detection and hand tracking.

Chapter 4: *Hand shape Recognition*: This chapter discussed the hand shape recognition aspect of the system proposed by this research. An overview of the proposed recognition strategy is presented. A detailed discussion of the pre-processing procedure used to extract and normalize features from images in the video sequence is provided. In selecting a suitable machine learning classification strategy to classify hand shapes using the normalized features, a detailed comparative study between support vector machine and neural network is also presented. The machine learning training and testing procedure is explained. It also details specific modifications made to the recognition framework to overcome recognition errors and enhance accuracy.

Chapter 5: *Hand shape Estimation*: This chapter provides an overview of the hand shape estimation method proposed by this research. In order to describe clearly the three dimensions hand model used, a review of "Van Wyk's work" is also provided [109]. Details of each step of the estimation process are provided.

Chapter 6: *Experimental Results and Analysis*: This chapter describes the tests carried out to assess the performance of the 3 sub-systems of this research: hand tracking, hand shape recognition and hand shape estimation.

Chapter 7: *Conclusion and Future Work*: This chapter concludes the thesis. It explains the main contributions that this research has made to the field of study. Directions for future research are also provided.

Chapter 2

Related Work

The aim of this research is to demonstrate the extraction of sign language hand shapes from a monocular video sequence. The analysis of hand shape variations in a video sequence requires three major tasks: hand tracking, hand shape recognition and hand shape estimation. Hand tracking is the process of detecting the location of the hand in each frame of the video. Hand shape recognition is the process of determining the shape of the hand using the detected hand location. Accordingly, hand shape estimation is the approximation of hand shapes that are formed in transitions between two recognised hand shapes.

This chapter presents a detailed survey on hand tracking, hand shape recognition and estimation. The review has been separated in the following sections: Sections 2.1, 2.2 and 2.3 present related work in the fields of tracking, recognition and estimation of articulated objects, respectively; Section 2.4 concludes the chapter. In each case, an overview of the method is presented as a base along with studies that have implemented the method.

There are two categories of methods used to track articulated objects: model-based methods [23, 101] and model-free methods [54, 81, 91, 97]. Model-based methods, also known as shape-based methods, are methods which use shape features to detect the target in each frame and include algorithms like chamfer distance matching and distance functions. Model-free methods are methods which use the colour and intensity features of the target to track it. These include algorithms such as mean-shift, particle filters and optical flow. Section 2.1 explains why model-free methods are more suitable for tracking articulated objects than model-based methods and excludes the latter from this research. Sections 2.1.1–2.1.4 explain related work in the field of object tracking using four methods: mean-shift, particle filters, methods that combine mean-shift and particle filters, and optical flow.

2.1 Articulated Object Tracking

Model-based methods have been applied to the field of articulated object tracking [66, 99, 115]. It has, however, been found that such implementations are generally unsuitable for this task due to a number of reasons. Primarily, model-based methods are only able to track a limited set of shapes. In this case, it is not possible to track objects of unpredictable shape. The alternative is to use a large training set that contains all possible variations of a target object. It is, however, very difficult to obtain such a training set and the time cost involved in constructing a converged model with all these shapes is substantial and infeasible. Additionally, the resulting model will be slow and less than real-time [66, 99, 115]. Therefore, this research excludes model-based methods as the basis for tracking.

Model-free tracking methods capture hand motions effectively. These include tracking algorithms such as mean-shift, particle filters and optical flow. In principle, these algorithms track any deformable or articulated object regardless of the shape and therefore do not require shape-based models. The only factor that affects the performance of these algorithms is the colour and intensity of the object. Therefore, the more distinct and uniform the colour of the object, the higher the performance of the algorithm. By combining multiple features such as colour, edge and motion, model-free tracking algorithms can achieve an accurate and efficient result, even in complex environments.

These methods have successfully been applied to the task of hand tracking [54, 91, 97]. From Section 2.1.1–Section 2.1.4, we discuss and analyse studies that make use of three model-free tracking methods to track articulated objects, namely: mean-shift, particle filters and optical flow. In Section 2.1.5, a conclusion is drawn on a tracking algorithm that is suitable for tracking sign language hand shapes.

2.1.1 Mean-Shift

Mean-shift is an effective colour-based tracking technique which is not computationally expensive. It is a non-parametric clustering procedure. In comparison to the k-means clustering approach, there are no embedded assumptions on the shape of the distribution or the number of modes or clusters [16]. It is applied as a basic component for many advanced trackers and HCI applications such as a perceptual user interface [17].

The mean-shift algorithm runs as follows [24]:

1. The first time the algorithm is run, the target object is manually selected and colour cue information about this region is extracted. A search window is positioned around the target object.

2. The colour cue probability distribution of the search window is computed. In other words, a probability is computed for each pixel in the search window that indicates the probability that it matches the computed colour cue.

3. The peak of the probability distribution, also known as the centre of mass, is computed. This peak represents the region with the highest number of high probability pixels, that is, the target object.

4. The search window is repositioned at the centre of mass, that is, on the target object.

5. Steps 2 through 5 are repeated to track the object continuously.

An illustration of this procedure is shown in Figure 2.1. It can be seen that the search window continuously shifts to the left towards the point with the highest density of high probability points.

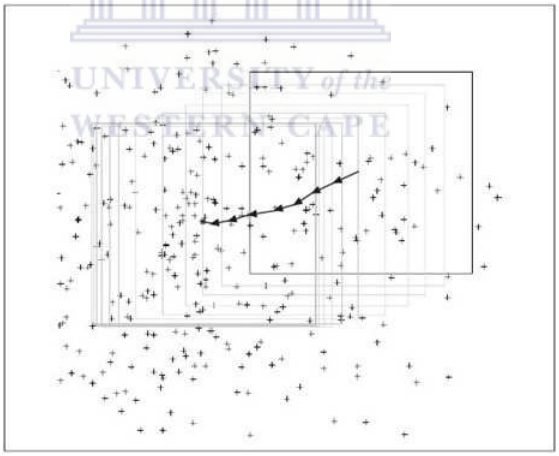


FIGURE 2.1: A demonstration of the operation of the mean-shift algorithm [16].

A principal drawback of the mean-shift algorithm is the static nature of the size and orientation of the search window. This limits its use in hand tracking applications since the size and orientation of the hand may change significantly.

In response to this drawback, Bradski [17] used the mean-shift algorithm as a basis for the camshift (Continuously Adaptive Mean Shift) algorithm. Camshift is an extension of the mean-shift algorithm that dynamically changes the size and orientation of the search window. It was originally designed for a perceptual user interface in which the camshift algorithm was used to track a user's face and identify its direction of movement

and rotation. Image capture was carried out with a single camera at a resolution of 160×120. The system was run on a 300 MHz Pentium II processor with 256 MB RAM and achieved a real-time performance of 30 FPS.

Camshift tends to be robust against transient occlusion as the search window will tend to first absorb the occlusion and then stick with the dominant distribution model when the occlusion passes. However, this procedure fails to track the hands when the hands pass over the face region and binds to the face region. This can be attributed to the high skin colour probability distribution present in the face region as compared to the hand.

Askar *et al.* developed a hand tracking algorithm that makes use of the centring and re-centring approach of the mean-shift algorithm [5]. However, their implementation uses an alternative method to initialize the search window as well as to determine the centre of mass of the search window. They use a prior knowledge-based skin detection method to segment skin pixels from non-skin pixels in the image. Only skin pixels are considered, resulting in the appearance of skin blobs in the image which were mainly the face and two hands. This is seen in Figure 2.2.

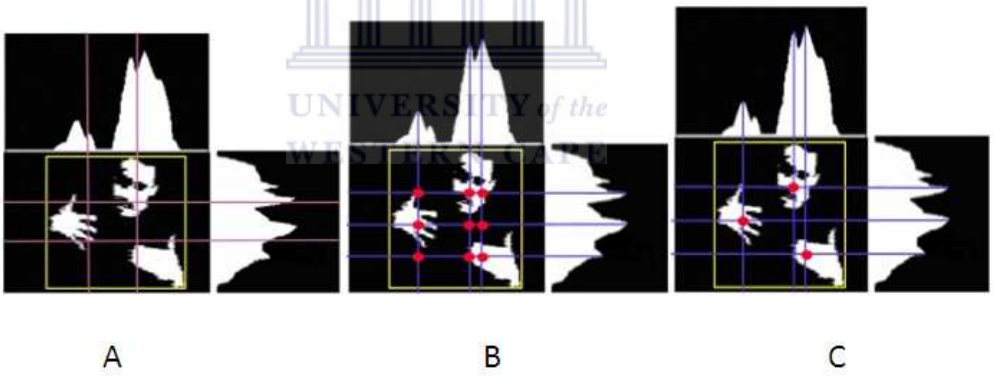


FIGURE 2.2: The histograms of the horizontal and vertical projection of Askar *et al.*'s method [5].

A combination of horizontal-vertical projection and neighbourhood analysis is used to search for the biggest blobs and so identify the position of the hands and head. A bounding box is then drawn around the hand blobs where the search window begins. This procedure is repeated to re-position the search window on the hands as they move. Figure 2.3 illustrates this tracking strategy. The centre of the bounding box around the blob is considered to be the centre of gravity of the blob in this method. The hand contours are segmented by using a region growing approach on the centres of gravity of the respective blobs.

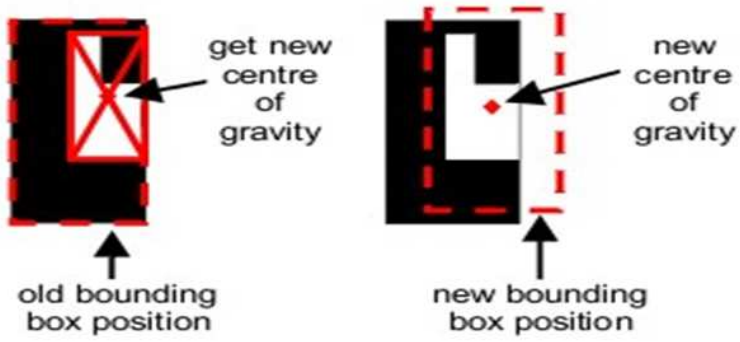


FIGURE 2.3: Tracking of the center of gravity [5].

This approach was tested on a 2 GHz Pentium IV PC on a colour frame sequence of resolution 576×720 and achieved a real-time performance of 25 FPS. It was found that the projection method has an inherent limitation: it is easily affected by noise [58]. Any random pixels that are determined to be skin pixels can easily affect the projection. This can be complicated further when a hand shape is used that produces a small skin blob such as a flat hand that is perpendicular to the camera.

It can therefore be seen that a major strength of this method is its ability to perform in real-time.

2.1.2 Particle Filters

Particle filters are sequential Monte Carlo methods for online learning within a Bayesian framework. A posterior probability distribution is represented by a set of random samples with associated weights. Each sample is also known as a particle. The weights and probability distribution are updated from one time measurement to the next.

There are various particle filters that differ in the procedures and measurements when using different sampling strategies, such as sequential importance sampling(SIS) particle filters, sampling importance re-sampling(SIR) particle filters, auxiliary sampling importance re-sampling(ASIR) particle filters and regularized particle filters. A detailed explanation of these particle filters is provided in [4].

Particle filters generally seek filtered estimates of X_k based on the set of all available measurements $Z_{1:k} = \{Z_i, i = 1, ..., k\}$ up to time k.

Let $\{X^i_{0:k}, w^i_k\}^{N_s}_{i=1}$ denote a random measure that characterizes the posterior distribution function(PDF) $p(X_{0:k}|Z_{1:k})$, where $\{X^i_{0:k}, i = 0, ..., N_s\}$ is a set of support points with associated weights $\{w^i_k, i, ..., N_s\}$ and $X_{0:k} = \{X_j, j = 0, ..., k\}$ is the set of all states up to time k. The weights are normalized such that $\sum_i w^i_k = 1$.

Draw N samples X_k^i from the proposed distribution q : $X_k^i \sim q(X_k|X_{k-1}^i, Z_k)$

Each particle is assigned a weight, w_k^i, according to:

$$w_k^i \propto w_{k-1}^i \frac{p(Z_k|X_k^i)p(X_k^i|X_{k-1}^i)}{q(X_k^i|X_{k-1}^i, Z_k)} \tag{2.1}$$

and the posterior filtered density $p(X_k|Z_{1:k})$ can be approximated as:

$$p(X_k|Z_{1:k} \approx \sum_{i=1}^{N_s} w_k^i \delta(X_k - X_k^i)) \tag{2.2}$$

An effective sample size $\widehat{N}_{\text{eff}}$ can be estimated by:

$$\widehat{N}_{\text{eff}} = \frac{1}{\sum_{i=1}^{N_s}(w_k^i)^2} \tag{2.3}$$

Particle filters have been successfully applied to visual tracking [48] and have been found to be particularly robust in cluttered environments [72, 97]. However, a major drawback of this method is the degeneracy phenomenon and the sample impoverishment problem. Both problems limit the performance of particle filters.

The degeneracy phenomenon causes inefficiency in sampling since, after a few iterations in a complex scene, most samples may have a very low likelihood. Their contribution to the posterior estimation may become negligible. This causes unnecessary computational costs. This can be seen in Figure 2.4. Also, the sample impoverishment problem causes low success rates in the estimation phase since less samples can be used to reflect the true probability distribution [4].

In response to the degeneracy phenomenon, Gordon *et al.* [86] introduced sequential importance re-sampling (SIR) particle filters. The SIR particles are propagated over time using a combination of sequential importance sampling and re-sampling steps. The re-sampling step statistically multiplies and/or discards particles at each time step to adaptively concentrate particles in regions of high posterior probability. These methods are very flexible and can easily be applied to non-linear and non-Gaussian dynamic models.

A recent study using SIR particle filters was conducted by Spruyt *et al.* [97]. Their study utilizes an off-line trained skin model and Gaussian mixture model (GMM) to reduce regions which are not represented in the skin colour space. In order to increase the speed and efficiency of the SIR particle filters, they combine motion and skin colour

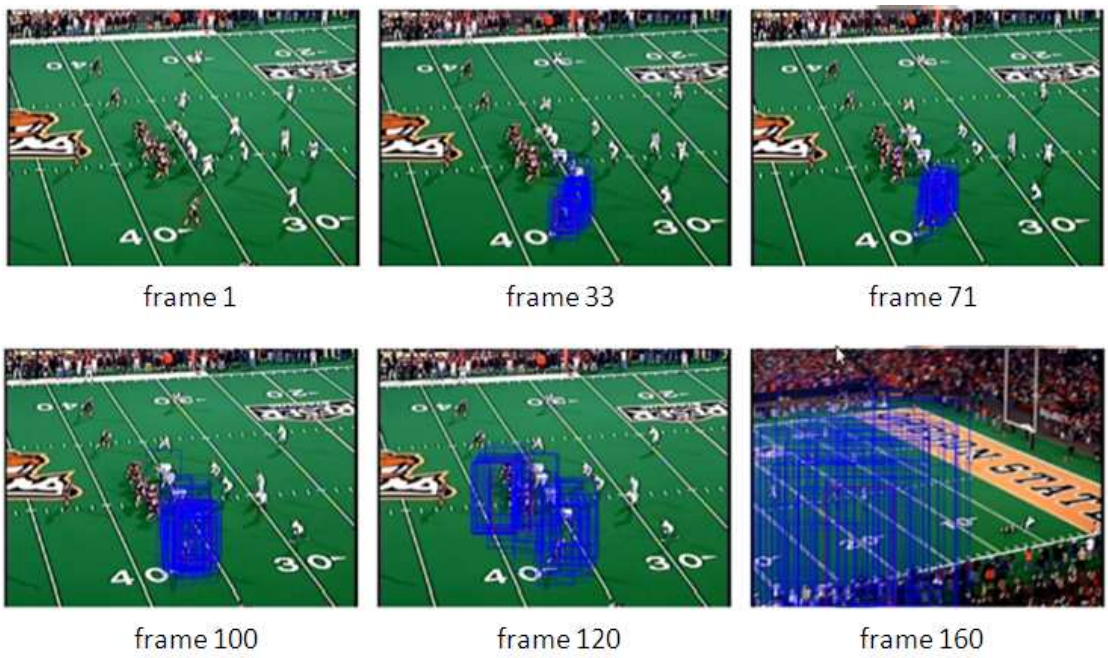


FIGURE 2.4: A bad tracking result from particle filters [41].

information. Thus, any colour distributions that are similar to that of the target are eliminated and do not affect the prediction of the SIR particle filters. They also use a region growing algorithm for hand shape segmentation.

However, the usefulness of particle filters in online hand gesture tracking is limited by the inefficiency of sampling and the computational complexity of the algorithm. The details of the inefficiency in sampling of SIR particle filters is explained in the next section since this drawback is common to SIR particle filters and methods that combine mean-shift with standard particle filters.

Rincon *et al.* [72] make use of particle filters for tracking objects in complex scenes. They present a new particle filter algorithm that uses two sampling techniques. This substantially improves the efficiency of the filter. Tracking the object is based on a proposed distribution which is determined using the a priori probability and information about the current observation. Their method samples pixels from the object to represent the colour distribution. Using non-parametric density methods, they estimate the probability of any observation in the colour space. Using the probability colour density image(PDI) and a sample likelihood evaluation of all particles, they estimate the target position in the next frame.

2.1.3 Combining Mean-Shift and Particle Filters

The mean-shift algorithm and particle filters differ in their search paradigms. The former is a stochastic and model-driven process while the latter is a deterministic and data-driven process. Both methods have their respective strengths and weaknesses. A number of studies emerged that proposed algorithms that combined the mean-shift algorithm and particle filters.

Shan *et al.* [91] proposed a new algorithm, the mean-shift embedded particle filter (MSEPF). In this study, the mean-shift algorithm is performed on each of the particles after they are propagated so that the particles are "herded" to nearby local modes with a large probability. The MSEPF produces a better estimation of the posterior estimation to the position of the target even with a smaller set of samples, thus reducing the number of computations required. Similar efforts on combining mean-shift with particle filter have also been reported in the works of Koichiro *et al.* [31], Maggio and Cavallaro [70], and Cai *et al.* [18].

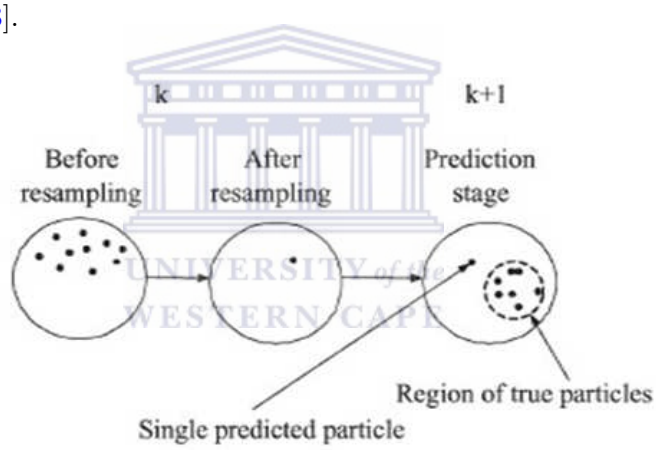


FIGURE 2.5: The example of re-sampling impoverishment [108].

A major advantage of this method is its reduction of the degeneracy problem. However, the mean-shift algorithm will inevitably make the samples too concentrated, aggravating another problem of particle filters – sample impoverishment. Sample impoverishment occurs when all particles collapse to a single point within a few iterations. The small or negligible difference between particles reduces the effectiveness of the algorithm to locate the moving object. This is especially severe when tracking a fast and randomly moving object in which the velocity and movement varies quickly [31, 45, 70]. Thus, the MSEPF begins to function similar to the mean-shift algorithm. This problem also occurs in the SIR particle filters. Figure 2.5 illustrates at time k, the resampling process in which samples are focused on the region which has the highest probability density distribution. If only a small number of particles satisfy this requirement, it is easy to lose

tracking at time $k+1$. This is because, in this case, not enough particles are available to estimate the correct position of the target.

2.1.4 Optical Flow

Optical flow is a method that tracks a pixel based on its intensity. The assumption that is made is that image measurements such as the intensity of the pixels in the image remain the same while their location changes over time. Mathematically, this is expressed as:

$$I(x+u, y+v, t+1) = I(x, y, t) \tag{2.4}$$

Where x and y are the x- and y- coordinates of the tracked pixel in the image, u and v are changes in the x- and y-coordinates, t is the time, and I is the intensity of the tracked pixel.

This assumption implies that the intensity of a pixel, having been displaced in the x and/or y direction over a time sequence remains the same. This implies that this value can be used to track the pixel.

Optical flow searches for the location of the required intensity value in consecutive frames to track it. Figure 2.6 illustrates this tracking technique. However, noise is able to severely affect the performance of this method. Any pixels that have the same intensity value as the tracked pixel can affect the tracking result. Additionally, the lighting of the image must be stable as any changes in intensity expectedly affect the result.

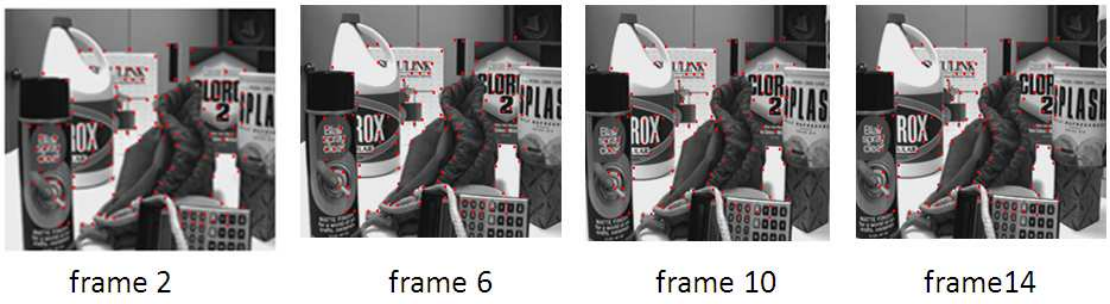


FIGURE 2.6: An implementation of optical flow tracking technique [11].

In order to reduce the interference of noise, the Lukas-Kanade (L-K) optical flow technique assumes small changes in location and constant flow of a tracked pixel in a local neighbourhood. The algorithm is applied in a sparse context and only relies on local information that is derived from a search window of known size surrounding each optical flow point [16]. The size of the search window is a variable that can be optimized. The use of smaller window sizes can cause the tracked pixel to easily fall outside the window

if the motion is too fast, thus losing track. The use of larger window sizes begins to introduce the same disadvantages as using no window at all, that is, sensitivity to noise.

The problem of determining an optimal window size led to the development of an enhanced L-K optical flow technique called the pyramidal L-K optical flow algorithm. This algorithm assumes a small window size – typically 3×3 to track the motion of a pixel by 1 pixel around its current location. It then handles the problem of large motions in the tracked object by creating a pyramid of increasingly smaller resolution copies of the image, as shown in Figure 2.7. An example of an image that has had its resolution halved 4 times is shown in Figure 2.8. Tracking is then carried out on the lowest resolution image. This ensures that even large motions in the original image appear as small motions in the lower resolution image, the size required by the search window. This informs the tracking of the image in the preceding level in the pyramid and ensures that tracking is not lost, despite a large motion. The motion estimate from the preceding level is taken as the starting point for estimating motion at the next layer down. This eventually propagates up to the original image in which tracking is now possible. A drawback of this method is that it cannot effectively track objects that undergo large deformations during global motions such as the hands [82].

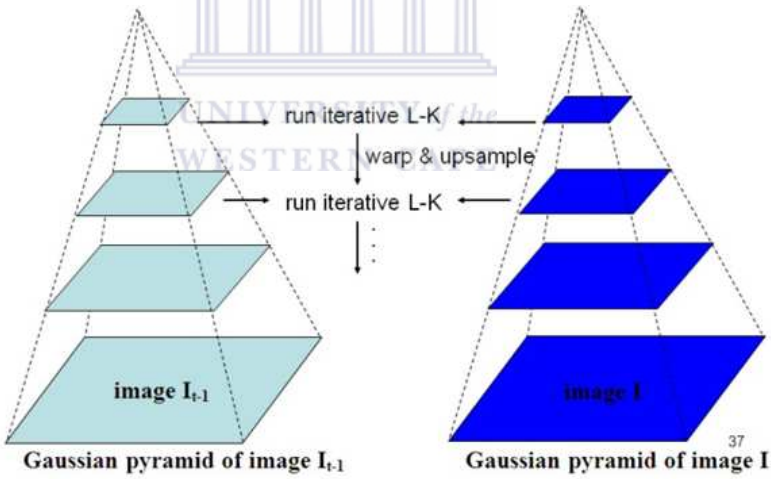


FIGURE 2.7: Pyramid L-K optical flow [82].

A method developed by Kolsh and Turk called "flocks of features" attempts to manage this problem and allows tracking despite vast and rapid deformations in the object as it moves [54]. This method uses a set of initial L-K optical flow points that are placed around a skin blob to be tracked. The main idea behind this method is that, similar to a flock of birds in flight, the points in the flock of features should always maintain a "safe" minimum distance from each other but should not wander too far from the flock either. As the object moves, optical flow is used to relocate the positions of each of the features in the flock. If relocation causes a violation of the constraint of the point

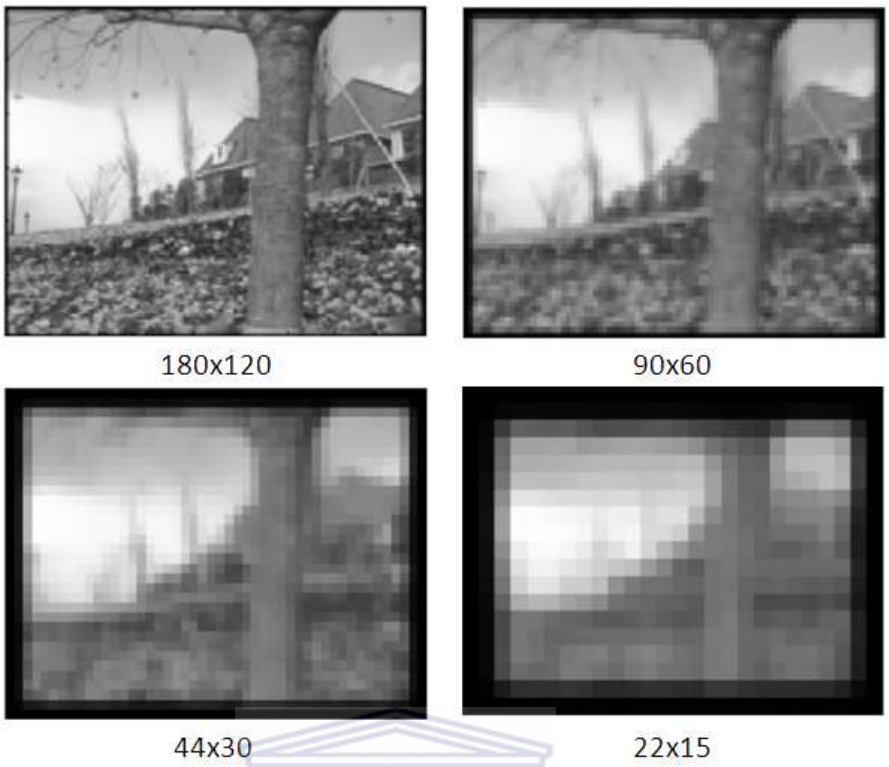


FIGURE 2.8: One image in different resolutions [82].

moving too close to other points or too far from the flock, the point is re-located to a position of high skin probability that does not violate the constraint [68, 92]. The hand position in each frame is determined by taking the median distance of the features in the flock. Their system operates at an average near-real-time performance of 12.7 FPS for a video input at a resolution of 720×480 RGB on a 3 GHz PC.

There are, however, a few limitations of this method: the target cannot move too fast and the velocity should remain constant when tracking in a complex background. The tracker easily loses track when the background contains features similar to the hand. Therefore, when the hand passes over the face or neck region, the tracker is attracted by the latter regions and loses track of the hand. Figure 2.9 is an illustration of the HandVu application [53] that uses flocks of features to track the hand. It can be seen that the tracker loses track in frame 80 when the hand passes over the cupboard with a similar colour distribution to the hand and frame 120 when it passes over the neck region.

2.1.5 Summary of Articulated Object Tracking

Rigid objects with a known shape can be tracked reliably in environments with an arbitrary background using model-based methods [12, 49]. However, model-based methods

FIGURE 2.9: An demo for hand tracking with flocks of features [57].

are infeasible and ineffective when the shape of an object varies vastly over time such as with natural hand motion. In such cases, most approaches resort to a model-free approach. The model-free tracking approach does not model the appearance of the target. Rather, it tracks the target using colour information.

The model-free approaches that are popularly used to track the hands include particle filters, L-K optical flow and mean-shift [17, 54, 97]. These strategies are often used for visual object tracking and their effectiveness in this area, where possible, has been evaluated. Each of these methods have respective strengths and weaknesses but all three have been seen to be able to operate in real-time. It is concluded that they are, for the most part, on equal footing. Choosing a particular method therefore depends on the specific application.

For the sign language application of this research, the mean-shift tracking algorithm is selected. It is a computationally inexpensive method, yet very effective. It has been seen that it operates faster than particle filters, optical flow and other combined methods, due to the implementation of a single hypothesis that climbs the density gradients to find the peak of probability distributions in the colour space. Specifically, the extension of mean-shift called camShift is selected since it has been seen to be more robust than mean-shift. The robustness of the algorithm on a complex background can be improved by combining the proposed method with methods such as skin detection and background subtraction.

2.2 Articulated Object Recognition

Recognition of articulated objects attempts to determine the shape of the object. Hand shape recognition generally uses either template matching or machine learning techniques. Template matching uses pre-constructed hand shape examples to compare and match with an observed image from a camera. The match between the examples and the observed image can be done by using the distance to edges technique [88, 94, 107] which is also called the segmented silhouettes technique [100]. Machine learning techniques

make use of pre-trained classifiers which are constructed from specific features extracted from a number of classified examples. There are many machine learning techniques that can be used to generate these classifiers such as support vector Machines (SVMs) and neural networks (NNs). There are a wide range of features that can be used in these classifiers such as shape contours and Fourier descriptors.

2.2.1 Template Matching Techniques

We first focus on template matching techniques. Shimada *et al.* used a distance to edges method to recognise hand shapes [94]. In their method, 256 points are sampled along the contour of the binary silhouette of the hand as seen in Figure 2.10. The distance from each point to the centre of gravity is computed. The distances are plotted resulting in the graph shown in Figure 2.10. The shape of the graph then determines the corresponding hand shape and is unique to that hand shape. To make the result invariant to in-plane rotations of the hand, the plotted distances are shifted to start with the most significant peak. Thereafter, the distances are plotted either clockwise or counter-clockwise – it is not clear which. The distances are also normalized to obtain scale invariance. Because the distances are measured relative to the centre of gravity, this measure is also invariant to translation. Their system was tested on a 6 node cluster [94] and achieved a real-time recognition rate of 30 FPS. However, the recognition accuracy of the system is not clear.

Schreer *et al.* used a similar method but used a different distance function [88]. For each contour point around the hand, the normal to the tangent of the point along the contour is determined. Then, following the normal, the distance from the point to the contour on the opposite side is determined and plotted. Figure 2.11 illustrates a hand shape and the graph of the distances produced from that hand shape. A classification into 13 different hand shapes can be performed using this method.

Their system was tested on a "state-of-the-art" PC [88] on a video feed at a resolution of 640×480 pixels and achieved a real-time performance of 25 FPS.

This method is a completely data-driven approach and does not need any training process or data-set of hand shapes. There are, however, limits to the detection rate of systems using this method. The silhouette of the hand is affected by the law of perspective projection. A major drawback of this system is that it is unable to recognise hand shapes in which the fingers are not distinctly separated and visible.

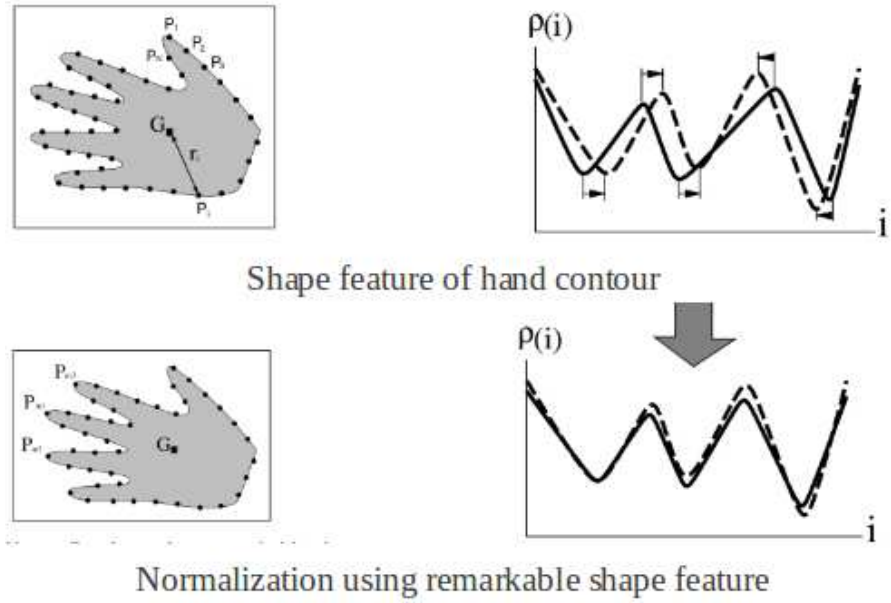
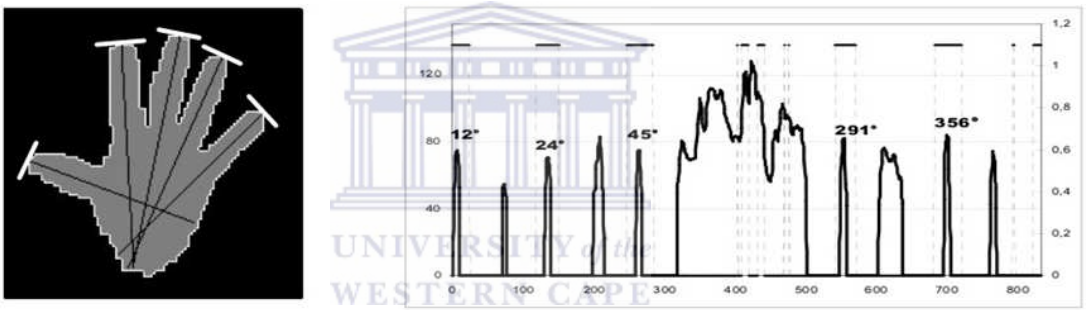


FIGURE 2.10: Shimada's hand shape estimation by using distance measurement [94].



a) Normal to the tangents from the fingers

b) Distance function from the silhouette along the contour for the finger

FIGURE 2.11: Schreer's hand shape recognition by using distance function [88].

2.2.2 Machine Learning Techniques

A second approach to hand shape recognition is the use of machine learning techniques. Stergiopoulou and Papamarkos [102] propose an extended neural network-based hand shape recognition system. The program starts by locating the hand using a skin colour distribution map in the YCbCr space [21]. Once the hand is located, they propose a new technique called the Self-Growing and Self-Organized Neural Gas (SGONG) neural network [9]. The SGONG is an innovative neural network that starts with a small number of neurons and grows according to the hands morphology to approximate its shape in a very robust way.

Figure 2.12 illustrates this procedure: Figure 2.12 (a depicts the SGONG starting with only two neurons; the number of neurons grows, as seen in Figure 2.12 (b and converges

in Figure 2.12 (c. In Figure 2.12 (c it is obvious that the grid of the output neurons has taken the shape of the hand [102].

Furthermore, the tips of the fingers are located by locating those neurons which have a small number of links with other neurons and are called root neurons. In Figure 2.12 (c, such characteristics can be found in the tip of five fingers. This step is very important for increasing the reliability of the system. Subsequently, the completion of the hand shape recognition process is achieved by using a likelihood-based classification method to determine the hand shape. The proposed hand shape recognition system has been trained to identify 31 hand gestures that consist of combinations of raised and not raised fingers. They conducted testing on their system. A total of 1800 tests using 180 hand shape images were carried out. The recognition rate, under controlled conditions – a static background and constant lighting conditions – was 90.45%. The computation time cost for each hand shape was around 1.5 seconds on a 3 GHz CPU Computer.

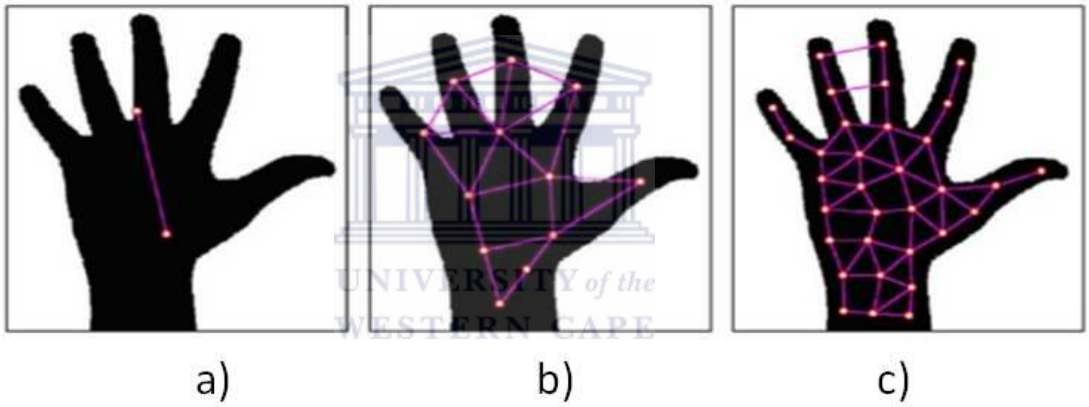


FIGURE 2.12: Stergiopoulou's hand shape recognition by using SGONG [9].

Sato *et al.* also used neural networks but devised a different method of hand shape recognition [87]. They trained the classifier for six hand shapes using a number of examples for each hand shape. In order to achieve a higher accuracy, they employ a normalization pre-process on these images. First, the hand region is located. It is translated so that the centre of gravity of the region is placed at the centre of the image. The region is then rotated based on the orientation of the principal axis of the region so that the axis is aligned with one of the image axes. Thereafter, the image is down-sampled to a resolution of 12×12 pixels. This step is necessary to reduce the computation cost and effects of small illumination variation. This normalized hand region is sent to the training phase and ensures that the system is invariant to the hand alignment in the image. This process is illustrated in Figure 2.13. The system uses two cameras which observe the hand motion. This is carried out to overcome the problems arising from self-occlusion.

FIGURE 2.13: Sato uses normalized data for hand shape recognition [87].

The system was tested on a Pentium II 450 MHz PC with 384 MB memory. The system was found to be able to process more than 20 FPS. A recognition accuracy of 85% overall was achieved in this experiment. The typical mode of failure was misclassification happening during transition from one hand shape to another. This result is to be expected since the neural network was not trained to recognise those intermediate hand shapes.

2.2.3 Summary of Articulated Object Recognition

Template matching methods can be used to achieve a real-time performance with a high recognition accuracy in certain situations; in order to extract complete and clear hand contours, these methods either need a simple background [88] or a special camera configuration [94]. A normalization pre-process is used to make such features scale and rotation invariant. Subsequently, a set of predefined criteria that uses a threshold or range is used to recognise the hand shape.

This set of criteria will only be effective in the same conditions under which it was constructed. In other conditions the recognition accuracy would deteriorate.

Therefore, while such methods are not complex and can operate in real-time with a high recognition accuracy, they are mostly only effective under controlled conditions.

On the other hand, machine learning methods recognise hand shapes using a training model. This model is trained on a number of examples. These examples can be obtained from a variety of background conditions. The greater the number and diversity of examples used for training, the greater the robustness of the system. Therefore, machine learning methods tend to be more flexible than template matching methods; they can be trained to operate under a variety of background conditions. The training process is carried out using a machine learning algorithm. As in the case of template matching methods, a normalization pre-process is used to make the features scale and rotation invariant.

Machine learning methods can be difficult to train and require many examples to achieve a robust system. Also, the performance of the system can be affected resulting in processing times that are less than real-time [102]. However, it has been shown that real-time performance can be achieved [87] and the result is much more robust than that of template matching methods.

Finally, a disadvantage of both methods is that they are only capable of recognizing a pre-defined and limited number of hand shapes. This means that such systems will not be able to recognise hand shapes that are formed as the hand transitions between one hand shape and another. The next section explains how estimation can be used to overcome this problem.

2.3 Articulated Object Estimation

Hand shape estimation uses a 3D model to approximate the exact configuration of the hand using a set of input features. The methods used to estimate hand shapes can be divided into two types. The first method matches the joints of the observed hand with the joints of a 3D hand model using fitting algorithms. The second method incorporates the use of hand shape recognition to match the shape of the observed hand with one in a pre-defined database which covers all possible hand shapes. This database is generated using a 3D model.

Subsection 2.3.1 and 2.3.3 describe the related work surrounding estimation using joint matching methods and hand shape recognition respectively.

2.3.1 Joint Matching Methods

This method of estimation works by extracting local image features and fitting a given 3-D physical model [55, 83]. In most cases the model consists of a set of linked rigid bodies. It is able to describe all possible variations in 3D [67]. The model is limited to only those hand shapes that are plausible. Plausible hand shapes are determined by using a kinematics model [56] or from the data collected using a data glove.[1] In each frame, image features such as the fingertips [55] or line segments on the observed hand [84] are extracted and fitted onto the 3D model. The resulting model provides an approximation of the exact 3D configuration of the hand.

Lee and Kunii [55] pioneered the use of a 3D model to analyse hand shapes. They employ a set of constraints to simplify the finger inverse kinematics and propose a model fitting

[1] CyberGlove

algorithm to fit the hand model to the observed image of the real hand. The matching is guided by hand constraints and the tracking of seven characteristic points on the hand. These characteristic points are illustrated in Figure 2.14. They use custom-made colour-coded gloves and two cameras that are positioned at different viewpoints to track these points. The exact estimation accuracy of their system is not clear.

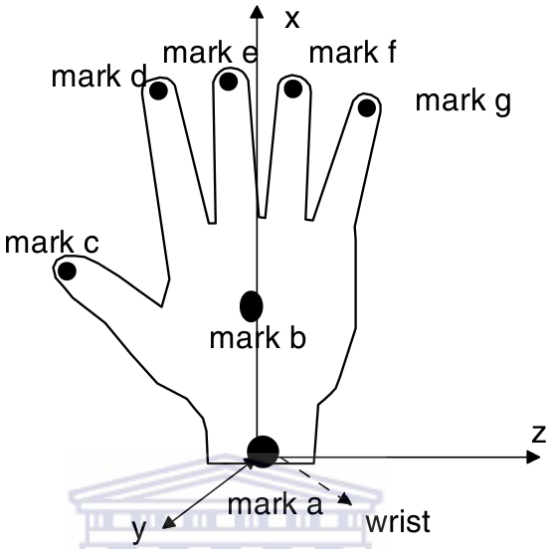


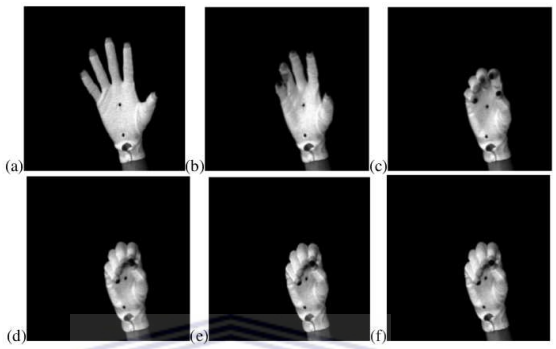
FIGURE 2.14: 7 characteristic points on the hand [55, 62].

On the basis of Lee *et al.*'s contribution, Lien *et al.* [62] proposed an optimized hand model fitting method to track the finger joints during hand motion using a stereo vision system – "IIs-head" [93] – and a custom-made colour-coded glove similar to that of Lee and Kunii [55].

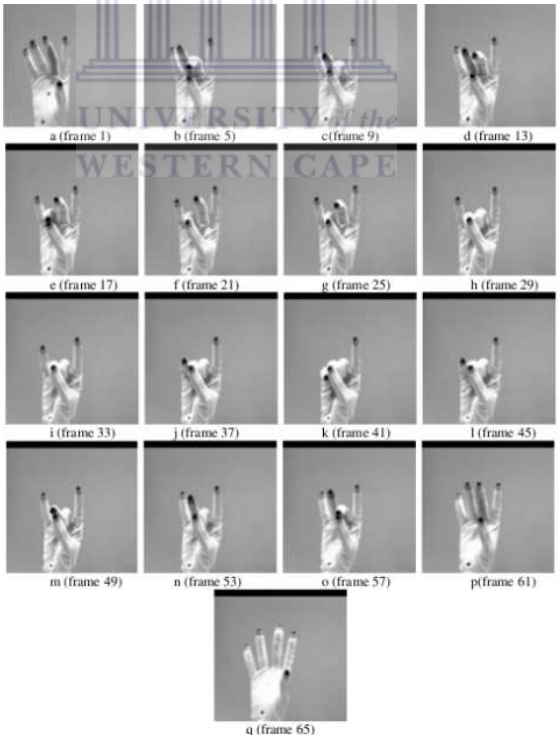
To generalize the application of the marker-based hand shape analysis system and overcome the scale inconsistency problem, a scale calibration process is carried out using an image sequence of hand-grasping movements.

Hand-grasping movements are executed by bending five fingers. The image sequence provides sufficient calibration information for the evaluation of the 3D hand model's scale fitness. During this process, ten scaling variables for the hand model are identified. The scale of the 3D hand models are calibrated using the three-dimensional positions of the seven markers. Once the three-dimensional hand model is calibrated, this hand model of known size is applied to analyse the input hand shapes. The temporal information of the hand motion is utilized to estimate the hand shape when occlusion of markers pasted on the hand occurs. This information includes all joint angles in the current and previous fitted hand models captured over time. The occluded hand shape is predicted by applying the differential values between the two hand models.

The system was tested on a AMD Athlon XP 2500+ 1.87 GHz PC and achieved an estimation rate of between 3.7 and 10 FPS. Two hand shape deformations were used to test their system: one without occlusion. These are illustrated in Figure 2.15. They achieved an estimation accuracy of 99% the hand shape without occlusion shown in Figure 2.15 a). The estimation accuracy of the hand shape with occlusion shown in Figure 2.15 b) is not so clear although a histogram is provided and illustrated in Figure 2.16. The estimation of more complex hand shape deformations is not possible with the current system.



a) Hand shape deformation without occlusion



b) Hand shape deformation with occlusion

FIGURE 2.15: Two hand shape deformations for testing.

Lu *et al.* [67] propose a method that uses a single camera and does not need for colour-coded gloves. In this system, three image cues are used for tracking: edges, shading

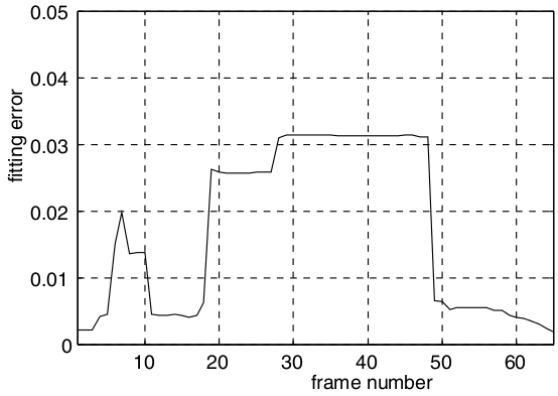


FIGURE 2.16: Error analysis for fitting the 3D model to a hand shape deformation with occlusion shown in Figure 2.15 b).

information and optical flow. Initially the 3D model is fitted to a pre-defined hand gesture. The Canny edge operator is used to extract the finger edges and a curvature-finding function is used to locate the base points. The edge of the hand is tracked by observing the change of base-point position between the current and next frame. Subsequently, the finger tip is tracked using optical flow applied to the area inside the edge. At each frame, visibility checking is carried out to match the corrected observed hand and the 3D model hand.

Variations in the shading of the hand as it rotates with respect to the light source is also used. This constraint combines the optical flow and shading constraints and degenerates to one of these when the other is not available [67]. They combine these cues to reduce or eliminate the errors arising from the assumption of brightness constancy. Their system is able to track the complex and articulated motions of the hand as the shading changes.

The system was tested on a 1 GHz Pentium 4 PC and achieved an estimation rate of 4 FPS. The estimation accuracy is not so clear although a set of visual results is provided and illustrated in Figure 2.17.

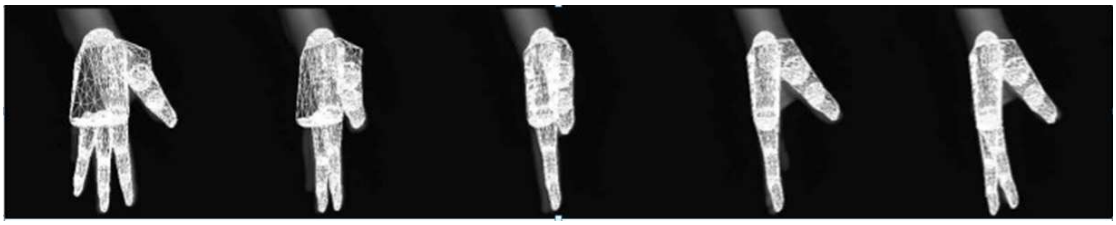


FIGURE 2.17: Lu's hand shape estimation result [67].

This method handles partial occlusions by means of visibility calculations. However, it is not a comprehensive solution to the problem in the case of single camera systems. The method does not explicitly handle occlusions.

2.3.2 Summary of Joint Matching Methods

Joint matching methods estimate the location of each joint or feature by projecting a 3D hand model onto the image space and comparing it with the image features. Kinematic constraints play an important role in hand shape estimation. These include: reduction of computation cost and solving the problem of partial occlusions. Ambiguity arises when multiple joints along the kinematic chain can produce similar limb movements.

Using multiple cameras and colour-coded gloves allows for the capturing of the posture of the hand in detail. This can be used to increase and optimize the system performance [55, 62]. However, the SASL project requires non-contact solutions. Therefore, these are not suitable for this research. Methods that do not use colour-coded gloves are also able to track a few features of the hand from a single view on a static background based on a robust tracking algorithm such as optical flow [67]. However, such methods often fail due to a large variation in the appearance of the hand and multiform self-occlusions made by the hand. They are also complex and struggle to achieve real-time performance.

2.3.3 Methods that Use Recognition For Estimation

It is challenging to track joint features using a single camera and directly match it with a 3D hand model. This section describes hand shape estimation methods that use hand shape recognition to match a synthetic hand image and an observed image using a single camera. Each synthetic hand image is labelled with parameters describing the hand shape and the positions of all articulations. The system consists of a large hand template database for describing all possible hand shapes.

Conventionally, a synthetic hand template database is populated by a computer graphics application or data glove. In the estimation phase, given an input image, the system retrieves the most similar hand template from the database, and the pre-stored configuration parameters are then used as candidate estimates for the parameters of the hand shape.

Athitsos and Sclaroff [8] followed a database retrieval approach. A database of 107,328 images consisting of 26 poses in 4128 views was constructed using a 3D model. Different similarity measures were evaluated, including chamfer distance, edge orientation histograms, shape moments and detected finger positions. A weighted sum of these was used as a global matching cost. Retrieval times of 3–4 seconds per frame on a 1.2 GHz PC were reported in the case of a neutral background with an accuracy of 84%. Figure 2.18 illustrates some of the results of their work. Therefore, pose restrictions are a part

of many full DOF hand pose estimation algorithms in order to avoid large amount of occlusions.

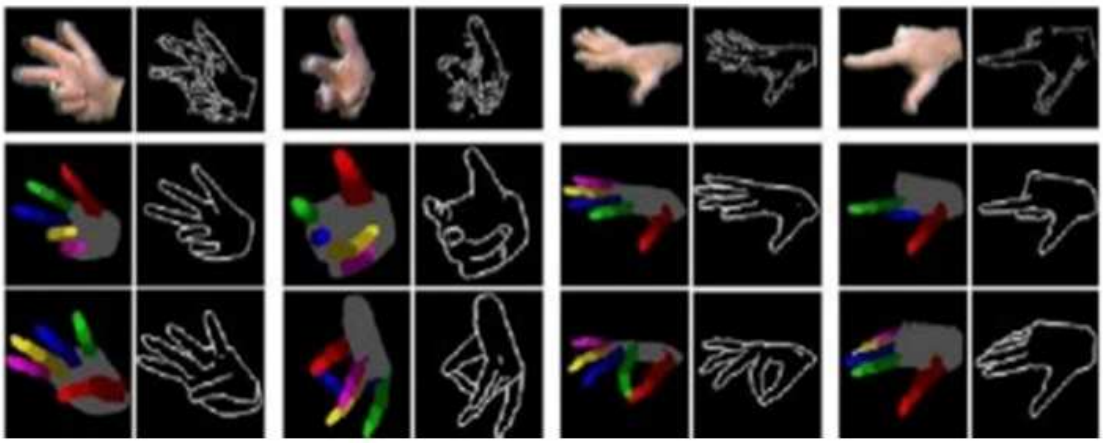


FIGURE 2.18: Athitsos and Sclaroff's hand shape estimation.

In order to avoid exhaustive searches in the database and achieve real-time processing, Shimada *et al.* [94] proposed a strategy that uses an adjacency map to reduce the search area in the database. For each frame, the search area is restricted to the neighbourhood of the estimated appearance in the previous frame. A transition network was proposed such that the search space can be limited. Figure 2.19 illustrates a possible transition network. Only templates that are likely successors to the previous state are evaluated. The network is learned from a series of example sequences. The details and drawbacks of their strategies that are used for matching the features between input hand shape and synthetic hand shape were described in Subsection 2.2.1.

Their system was tested on a 6 computer cluster framework running in parallel, with each PC in the cluster consisting of a single 600 MHz Pentium III CPU and 256 MB RAM. The system achieved a real-time processing rate of 30 FPS. However, the estimation accuracy of their system is not discussed.

If self-occlusions of the fingers result in a loss of tracking, it is difficult to recover the correct hand shape, even when a simple search is performed. This renders it unstable and less reliable. An alternative is to use a predefined hand shape to initialise the tracking process in the transition network. This natural sense of tracking makes it suitable for real-life problems, such as a sign language estimation system. A generalized way of solving the tracking initialization problem is to estimate possible configurations of the hand from the current image rather than using information from the previous result. This is done by hierarchical tree-based searching.

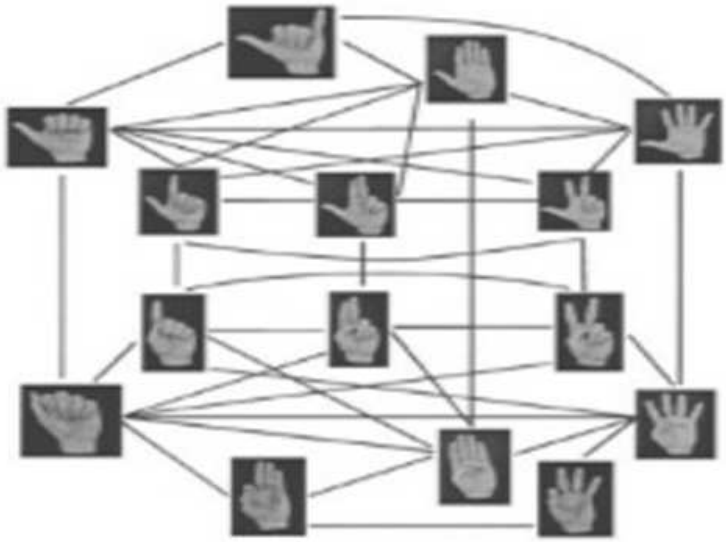


FIGURE 2.19: Shimada's transition network for hand shape estimation [94].

Thayananthan *et al.* propose a tree-based estimator using Bayesian filters [105]. Each node of the tree corresponds to a cluster of natural hand configurations which are collected using a data-glove. This approach uses a coarse-to-fine search approach by approximating the posterior distribution at multiple resolutions, without further evaluating sub-trees of the search space. On the basis of the search tree idea, Stenger *et al.* [100] proposed an alternative classification method which uses a multi-class cascade of classifiers for shape template matching. Unlike the normal use of boosting for single object detection, the cascade of classifiers is arranged in a tree order to recognise multiple object classes – hand configurations – hierarchically, as shown in Figure 2.20. Each weak classifier is trained to detect a single hand pose. If that pose is detected, the search continues to child classifiers that do a finer classification.

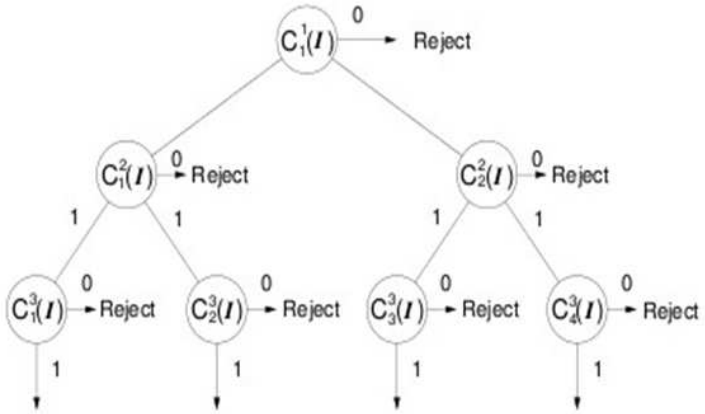


FIGURE 2.20: Tree based cascade of classifiers [100].

Referring to the cascade of classifiers C for multiple classes in a tree structure with levels indexed by I in Figure 2.20: similar objects are grouped together and the classifiers on

the leaves recognise single objects. A binary tree is shown but the branching factor can be larger than two.

The main advantage of boosting approaches are their speed. However, the number of classifiers needed grows exponentially with the dimensionality of the pose parameters, demanding a large amount of memory. Although high accuracy can be obtained, the computational cost of this system is still too high for real-time applications. Using a relatively small range of hand poses, an implementation of the method proposed by Stenger *et al.* was found to operate at a rate of 0.5 FPS on a 1 GHz Pentium 4 PC.

2.3.4 Summary of Methods that Use Recognition For Estimation

Methods that use recognition for estimation perform well and are robust to self-occlusion, since they directly match the intensity property between the input and the registered template. They use a single camera and do not need a colour-coded glove. This makes them very suitable to the system of this research since the sign language system requires non-contact solutions.

The related work highlights three parameters that should be considered: the size of the database to cover all possible hand shapes; the matching technique between the query and database; and an efficient search strategy.

In comparison to the nearest neighbourhood searching strategy [6, 32, 94], the approach proposed by Thayananthan *et al.* [105] and improved by Stenger [100] is able to obtain a high accuracy. However, these frameworks demand a large set of classes if a comprehensive range of recoverable poses is desired. The drawback is that labelling a large number of samples becomes a very time consuming process in the training phase. It also increases the computation time in the prediction phase.

2.4 Conclusion

In this chapter, the related work surrounding the three main steps required to extract hand shapes from a video sequence was discussed. These steps are: hand tracking, hand shape recognition and hand shape estimation.

Under hand shape tracking, three popular tracking algorithms and their related works were introduced. They are: mean-shift, particle filters and optical flow. For each method, extensions to the method were also discussed. It was found that all three methods, on average, performed the same. camshift is an extension algorithm of mean-shift.

It was found that this algorithm was slightly faster than other methods. Furthermore, it was found to have a robust performance on a complex background by combining methods such as motion detection and background subtraction. Therefore, it is selected for this project and used for tracking the hand.

The related work surrounding hand shape recognition was also discussed. Two methods of hand shape recognition were discussed: template matching methods and machine learning methods. It was found that machine learning methods have more potential to perform robustly on a complex background than template matching methods. This is because the training model that is used to determine hand shapes can be trained under different backgrounds and conditions. Therefore this method was selected for this research.

The related work in the field of hand shape estimation was also presented. It was also categorized into two methods: joint matching methods and methods that use hand shape recognition for estimation. Joint matching methods were found to be computationally expensive and mostly required custom-made gloves worn by the user to perform well. Therefore, these were not suitable for this research. It was found that the latter methods have a robust performance and do not need the aforementioned glove.

However, this method has the disadvantage that it requires a large number of examples if a comprehensive range of recoverable poses is to be achieved. This increases the computation time. Although high accuracies can be achieved, the computational cost of such a system is still too high for real-time applications. Thus far, a system that is perfect in hand shape estimation does not exist. The estimation method proposed by this research, and discussed in later chapters, is based on this method and extends it to overcome these drawbacks.

Chapter 3

Hand Tracking

Hand tracking is a fundamental step in hand shape recognition. For recognition to take place, the hand must first be located in the image sequences and segmented from the background. The hand is a complex deformable object. It is capable of carrying out fast and discontinuous motions. Chapter 2 compared colour-based tracking strategies such as particle filters, optical flow and mean-shift. The chapter concluded with the selection of the camshift algorithm as the primary tracking technique.

In this chapter, the contributions of this research to the field of automated hand tracking on complex backgrounds is presented. An overview of the proposed system is presented. Also, the background and theories of the techniques mentioned in this framework are presented in order to establish a clear understanding of the framework.

The remainder of this chapter is organized as follows: Section 3.1 presents an overview of the system design and implementation for hand detection and tracking proposed by this research. Sections 3.2 through 3.5 discuss: adaptive skin detection using face detection and histogram back projection; background subtraction; hand detection using hierarchical chamfer matching; and the camshift tracking algorithm. The chapter is concluded in Section 3.6.

3.1 System Design for Hand Tracking

The camshift tracking algorithm is often affected by noise in the background that has a similar colour probability distribution as the tracked object. To overcome this weakness, a combination of multiple cue methods is used to limit the noise on complex backgrounds. Only moving skin coloured objects in the video sequence are required. Therefore, a logical AND operation is performed between the motion cue image and the colour cue

image to produce a final image where the skin colour of the hands have been successfully extracted. This process is illustrated in Figure 3.1.

The skin cue image is determined in 3 steps: 1) A face is located in a scene. 2) A cascade classifier will determine the nose region and use its colour as the skin colour of the individual. 3) A skin cue image is computed using histogram back projection to analyse the Hue-Saturation (H-S) histograms of the skin region and the original image.

The motion cue image is obtained by using an optimized Gaussian mixture model (GMM) background subtraction method.

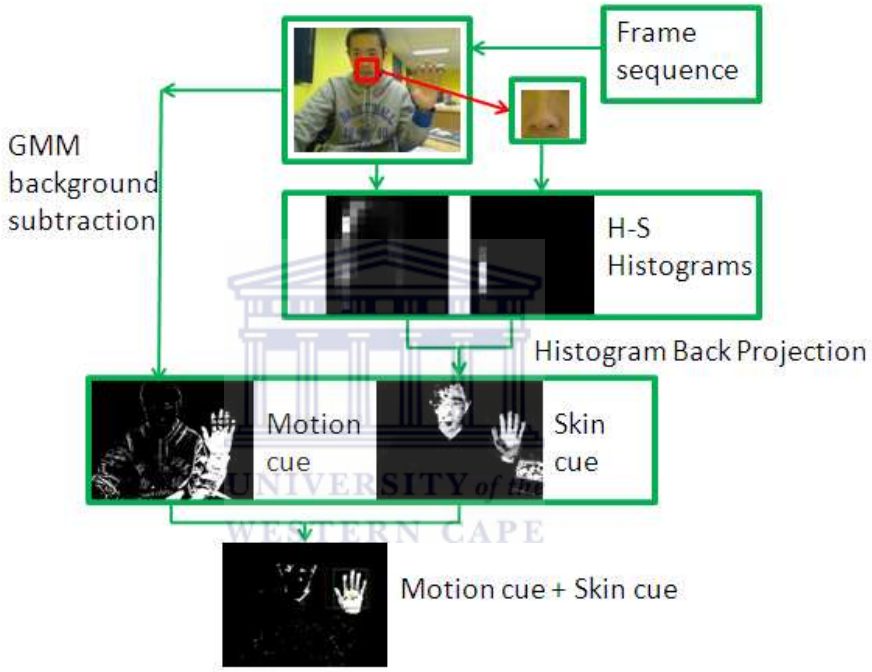


FIGURE 3.1: Combining multiple cues to filter out noise.

After producing the new image, through a combination of the motion and skin cue images, the chamfer Distance matching algorithm is used to locate the position of the hand in the image. The chamfer Distance matching algorithm is explained in Section 3.4.1. The Hierarchical chamfer Matching of the hand functions by detecting varying resolution sizes of the hand, ascending form small sizes to large sizes. In this system, the size of the face is used to obtain the optimal hand resolution size in the scene during initialization. The theory used is based on the proportions of the human body specified by the theory of Da Vinci [29]. This method improves the detection speed of the system vastly. It is important to note that the hand location is only determined once using the chamfer matching algorithm to obtain an initial region-of-interest (ROI) of the hand.

Thereafter, this region-of-interest is tracked continuously using camshift. The size of the ROI is specified as 120×100 pixels in the testing carried out. The skin pixels of the face

are eliminated from the resulting skin cue image using face detection since these pixels may affect the tracking when the hand approaches the face. The entire hand tracking system design and process is demonstrated in Figure 3.2.

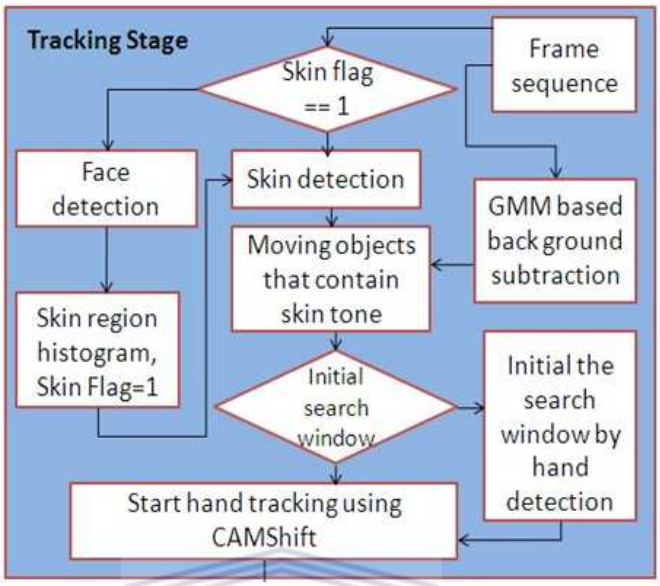


FIGURE 3.2: Hand tracking system design and implementation.

3.2 Adaptive Skin Detection Using Face Detection and Histogram Back Projection

In order to achieve versatile skin detection, the system is required to adapt to different skin tones. The face is located in the frame using face detection. The skin colour distribution of the region around the nose is extracted [2]. This distribution can be used to identify all skin pixels of a user in a frame for any skin colour.

The following subsections explain face detection, skin region extraction and histogram back projection.

3.2.1 Face Detection

The OpenCV library's implementation of the face detection technique is employed. This technique was initially developed by Viola and Jones [112]. The technique has 3 major sections:

1. The Haar features are detected by 4 types of rectangular masks.
2. The region image is converted to the integral image for rapid feature detection.

3. Cascade face detector is used to detect the face.

The following subsections detail each of these sections.

3.2.1.1 Haar Features Detection

Rectangular features are used to begin analysis on an image. These features are known as Haar features [112]. Figure 3.3 is an example of 4 types of Haar features. In each feature, the dark block is the same size as the light block.

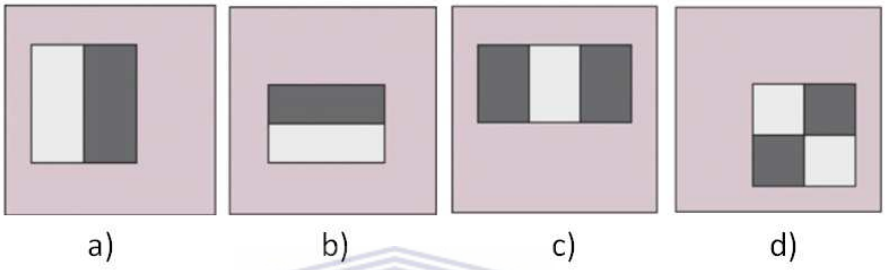


FIGURE 3.3: Four types of features used in Viola-Jones [112].

The procedure in the Haar features detection is as follows; the image is converted to grey scale so that the current image is defined as a set of intensity values from 0 to 255. These values are used to detect large-scale rectangular Haar features in the image. This is done by summing the intensity values in various rectangular blocks. The presence of a Haar feature is determined from the difference between the average dark-region pixel value from the average light-region pixel value. If the difference is greater than a specified threshold, it is said that feature is present. Haar features are computed over multiple image locations and multiple scales, as shown in Figure 3.4.

FIGURE 3.4: Haar features detection.

3.2.1.2 Efficient Computation Using the Integral Image Technique

Viola and Jones used a technique known as Integral Image [112] to efficiently and rapidly determine the presence or absence of hundreds of Haar features at every image location

and at several scales. The integral image is an intermediate representation of the image, which progressively adds together small units of a region. In this case, the small units referred to are pixel values. The integral value $P'(x, y)$ for each pixel is the sum of all pixels to its left and above it. Starting at the top left of the image and traversing to the right and down, the whole image can be integrated by proceeding row by row using the previously computed integral image values together with the current pixel value $P(x, y)$ in the row of the image (P) and calculating the next integral pixel value $P'(x, y)$ as follows [16]:

$$P'(x, y) = P(x, y) + P'(x - 1, y) + P'(x, y - 1) - P'(x - 1, y - 1) \tag{3.1}$$

An example that converts an image to an integral image is shown in Figure 3.5.

FIGURE 3.5: An example of a 7×5 image converted to a 7×5 integral image [16].

3.2.1.3 Cascade Face Detector

The face detector is trained by Adaboost which organizes it as a rejection cascade of nodes. In the rejection cascade, each node contains an entire boosted cascade of groups of binary classifiers trained on the Haar features from faces and non-faces. This is so that the weighted training error is minimized. Rejection cascades can greatly reduce the total computation time by quickly terminating the search in cases where the search region is not recognised by a weak classifier. A search window searches across the image and checks every location. The features in the search window of the image will be considered as a face when all the features pass the entire cascade structure.

The face region generates a large cluster of regions that are detected to be a potential face. The merge step first groups regions that contain a large amount of overlap and

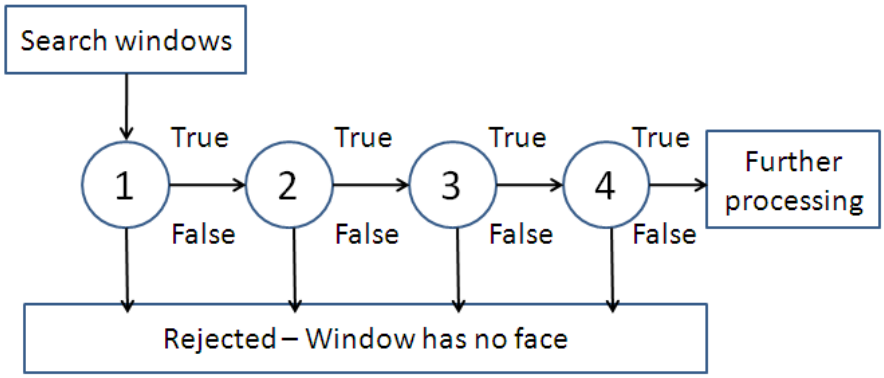


FIGURE 3.6: A cascade classifier.

then replaces all detected regions in the group with an average region. This region is the final detected face. This is illustrated in Figure 3.7.

FIGURE 3.7: Applying the cascade classifiers in each search window and then merging the detected regions of interest.

3.2.2 Skin Region Extraction

When expressing the skin region in an image, skin colour pixels were represented as white pixels and non-skin colour pixels as black pixels [2]. As long as a face can be detected in an image, the skin region of an individual can be extracted quickly and accurately. After carrying out face detection, a cascade nose detector was applied to only the face region. The result in shown Figure 3.8. The nose is used to extract the skin colour.

3.2.3 Histogram Back Projection

Back-projection is applied to the frames using the extracted skin colour so as to highlight the hands and eliminate all those objects which do not fall inside the skin colour distribution [103, 104]. Histogram back projection describes a probability distribution of skin pixels in an image depending on an H-S histogram that is obtained from the skin

FIGURE 3.8: Nose region extraction used as the skin colour.

region of the nose. H-S is derived from the Hue, Saturation and Value (HSV) colour space, which describes colour with individual values. According to the H-S histogram of the skin region, given C represents the colour of a pixel 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.

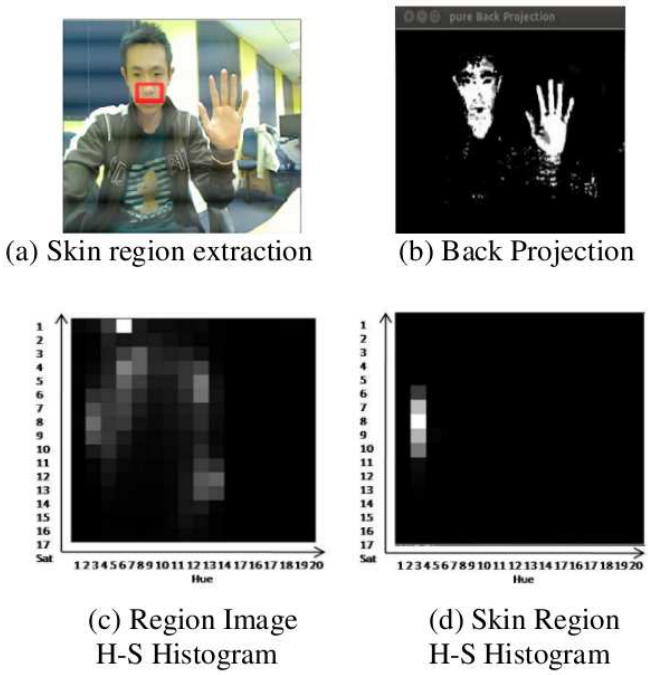
$$P(F|C) = \frac{P(F)}{P(C)} P(C|F) \tag{3.2}$$

This probability is then back-projected onto a grey scale image. Figure 3.9 illustrates an example of a back projected image. An intensity value of 255 indicates a high likelihood ratio of skin colour while a value of 0 indicates that the pixel has no skin colour.

This technique detects skin pixels effectively. However, it is often influenced by noise in the background that has a similar or higher skin colour probability than the tracked object. To overcome this, GMM background subtraction can be used as a pre-processing step in order to eliminate the tracking of false objects. This is explained in the next section.

3.3 Background Subtraction

Background subtraction is simply defined as the difference between the background image and an observed image. Background subtraction is a technique used to classify moving objects as objects belonging to the foreground while stationary objects are classified as objects belonging to the background [98]. This section will look at two methods of background subtraction. These are: simple background subtraction in Subsection 3.3.1 and statistical background subtraction in Subsection 3.3.2. Subsection 3.3.3 provides a comparison between the two methods.



(a) Skin region extraction (b) Back Projection

(c) Region Image H-S Histogram (d) Skin Region H-S Histogram

FIGURE 3.9: Projection and H-S histogram.

3.3.1 Simple Background Subtraction

Simple background subtraction uses a pre-defined background image from which each new frame is subtracted and the result, thresholded. The binary image highlights non-stationary regions in the image [74]. Large regions of non-stationary pixels are identified as being an object of interest. This approach can be described by the following equation:

$$I(x, y) - B(x, y) > \text{Threshold} \tag{3.3}$$

Where I denotes the pixel intensity in the observed image at position (x, y) and B denotes the pixel intensity in the background image. T is a threshold value that is usually determined empirically [90].

There is, however, a disadvantage when using a static background. It is easily affected by small changes in illumination or camera movements. In addition, a single threshold is not always suitable for every situation.

Frame differencing can efficiently solve these challenges by continuously refreshing the background image. Frame differencing is a technique that determines the background image by using recent frames. This approach is also referred to as an adaptive background subtraction approach. Each pixel value $F(x, y)$ in the background image is based on the pixel's recent history. If using t to describe the time sequence, frame differencing can be formulated as shown in the following equation:

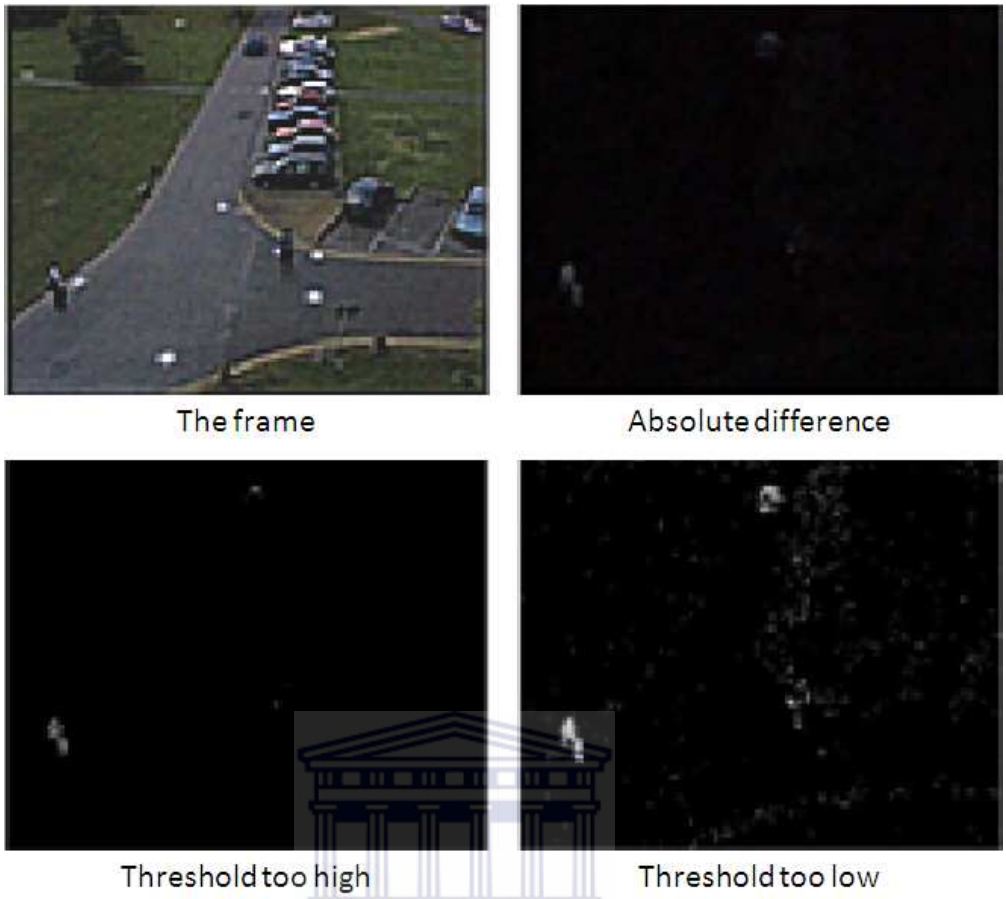


FIGURE 3.10: An example of simple background subtraction with a static threshold.

$$|F_t(x,y) - F_{t-1}(x,y)| > \text{Threshold} \tag{3.4}$$

In addition to this technique, the median value of the last n frames can be used to update the background image B in sequence [64]. The median value provides an adequate background image and it can be formulated as shown in the following equations:

$$|F_t(x,y) - B(x,y)| > \text{Threshold} \tag{3.5}$$

where

$$B(x,y) = \frac{\sum_{j=1}^{n} F_{i-j}(x,y)}{n} \tag{3.6}$$

This combination increases the stability of the background image.

3.3.2 Statistical Background Subtraction

Statistical background subtraction is a method that uses clustering algorithms to generate a converged background model. The background model dynamically reflects sudden illumination changes so as to only allow detection of highly active objects of interest. Mixture of Gaussians is a popular approach for background modelling to detect moving objects from static cameras [98]. A common optimisation scheme used to fit a GMM is the expectation maximisation (EM) algorithm. The EM algorithm is an iterative method that guarantees to converge to a local maximum in a search space. This search space is formed by using collected frames [98].

However, the method requires long periods of time to train at initialization, especially in busy environments. Based on such a framework, Bowden *et al.* [50] developed a method that reduces the training time using the expected sufficient statistics update equations. This method can efficiently initialize the training model. When the first L samples are processed, the system then switches to the EM algorithm.

Such statistical update equations yield a good performance on the background model estimation before all L samples are collected in the initial stages. This initial estimation improves the estimation accuracy. For the purpose of producing a stable background model, the initial estimation also allows for faster convergence.

The L-recent example update equations give priority over recent data. They are able to adapt to changes in the scene. In this approach, recent variations of each pixel are modelled by a mixture of k Gaussian distributions, k normally ranging from 3 to 5. To formulate this concept, the probability that a certain pixel has a value of X_N at time N can be written as [50]:

$$P(X_N) = \sum_{j=1}^{k} W_j \theta(X_N; \eta_j) \tag{3.7}$$

Where W_j is the weight parameter of the jth Gaussian component, μ_k is the mean value of k Gaussian distributions. $\theta(X_N; \eta_j)$ is the normal distribution of the kth component represented by [50]:

$$\theta(X_N; \eta_k) = \theta(X_N; \mu, \sum_k) = \frac{1}{|2\pi \sum_k|^{\frac{1}{2}}} e^{-\frac{1}{2}(X-\mu_k)^T \sum_k^{-1}(X-\mu_k)} \tag{3.8}$$

where $\sum_k = \sigma_k^2 I$ is the covariance of k^{th} component.

Different colours are modelled by different Gaussians. The weight parameters of the mixture Gaussian components represent the proportions of time that those colours remain in the scene. The background components are determined by an assumption that the background contains B most probable colours. The probable background colours stay longer and are more static.

Static objects which have a single colour tend to form tight clusters in the colour space while moving objects form wider clusters due to different reflecting surfaces during the movement. This measurement is known as the fitness value.

The k distributions are ordered based on a fitness value W_k/σ_k and the first B distributions are used as a model of the background of the scene where B is estimated as [50]:

$$B = \underset{b}{\operatorname{argmin}} \left(\sum_{j=1}^{b} W_j > T \right) \tag{3.9}$$

The threshold T is the minimum fraction of the background model. In other words, it is the minimum prior probability that the scene contains the background.

An update scheme is applied to allow the model to adapt to illumination changes and operate in real-time. Each new pixel value is compared with existing model components in order of fitness. The first model component that is matched is updated with the new observation. If no match is found, a new Gaussian component is added to the mean at the point.

Background subtraction is performed by marking a foreground pixel as any pixel that is more than 2.5 standard deviations away from any of the B distributions. The first Gaussian component that matches the test value will by updated by expected sufficient statistics equations [50]:

$$W_k^{N+1} = W_k^N + \alpha^{N+1}(P(\omega_k|X_{N+1}) - W_k^N) \tag{3.10}$$

$$\mu_k^{N+1} = \mu_k^N + \alpha^{N+1}(X_{N+1} - \mu_k^N) \tag{3.11}$$

$$\sum_k^{N+1} = \sum_k^N + \alpha^{N+1}((X_{N+1} - \mu_k^N)(X_{N+1} - \mu_k^N)^T - \sum_k^N) \tag{3.12}$$

where

$$P(\omega_k|X_{N+1}) = \begin{cases} 1; & \text{if } \omega_k \text{ is the first match Gaussian component} \\ 0; & \text{otherwise} \end{cases} \tag{3.13}$$

and

$$\alpha^{N+1} = \max(\frac{1}{N+1}, \frac{1}{L}) \tag{3.14}$$

ω_k is the kth Gaussian component and $\frac{1}{L}$ defines the time constant which determines change.

3.3.3 Summary and Conclusions of Background Subtraction Methods

In a given scene, if a single steady camera and a stationary background is used, simple background subtraction will suffice as a background subtraction technique. However, when the environment is complex, there are moving objects and the lighting conditions are uncontrolled, the static background subtraction method is not suitable for use.

The frame differencing approach can adapt quickly to changes in illumination and works well for a continuously moving object. However, this approach necessarily requires the object to be in constant motion. This is not true of the hands in sign language. For instance, when the hand performs small motions in a fixed position, the hand will be considered as a background object. If the background image is neither accurate nor active, background subtraction can lead to the detection of false objects. In addition, the median filter does not accommodate a rigorous statistical description and does not provide a deviation measure for adapting the subtraction threshold.

The GMM background subtraction technique can efficiently separate the background and foreground. It achieves a better performance than other background subtraction approaches for the purpose of hand tracking. In Figure 3.11, the desired foreground is a scene where only the hands are present. To adapt to changes in illumination, the only requirement of this background subtraction technique is that the user sits or stands still for approximately 3 – 4 seconds. In this period, the background model is computed and updated by the difference between the current frame and the current background model [59]. Thus, when the following frame is retrieved, the foreground pixels for the moving objects can be separated automatically from the image and illumination changes on the static object are still considered as part of the background. Therefore, GMM background subtraction is selected as a background subtraction technique for this research.

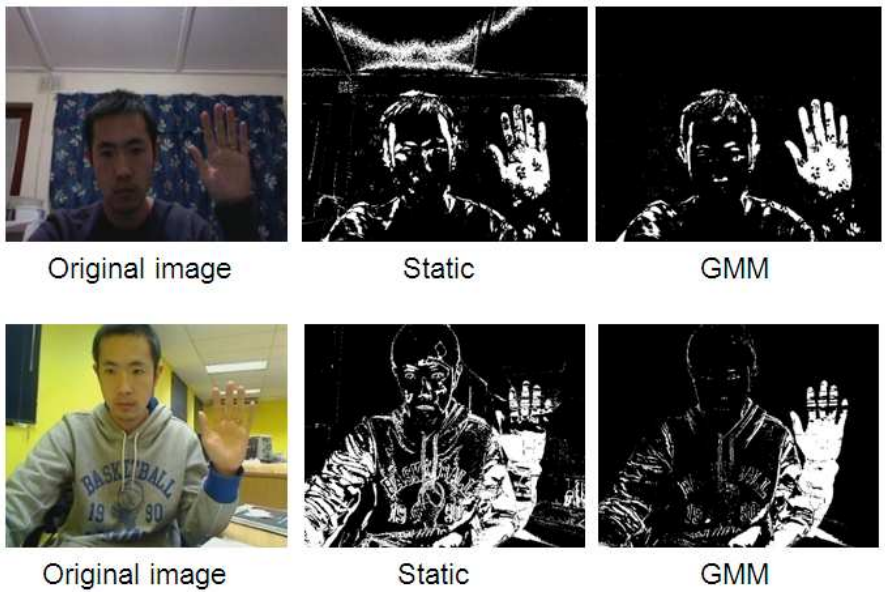


FIGURE 3.11: An comparison between simple background subtraction and model based background subtraction.

3.4 Hand Detection Using Hierarchical Chamfer Matching

This section explains the hierarchical chamfer matching technique. In the previous chapter, the camshift tracking algorithm was introduced. This colour-based tracking technique requires an initial window to locate the target once. Subsequently, the tracking window automatically tracks the moving target. In order to provide the algorithm with this initial window, hierarchical chamfer matching is used to detect the hand only once. The location of the hand is used to initialize the camshift tracker.

Subsection 3.4.1 describes the method of converting an edge image to a chamfer distance image, also known as a distance image. Subsection 3.4.2 and 3.4.3 both describe the methods that use templates to detect target objects in the chamfer distance image, but the latter can be applied to detect the target of different scales.

3.4.1 Chamfer Distance Transform

The distance transform of edge images offers a platform for template-based shape matching. This is true even in cases of an incomplete or unclear silhouette in the foreground. A distance transform is an operation that converts a binary edge image to an approximate distance image. Each non-edge pixel is given an intensity value ranging from 0 to 255. This value is a measurement of the distance to the nearest edge pixel. An example of this process is shown in Figure 3.12. This operation can be executed efficiently using different distance masks.

A 3×3 mask with value (3,4) distance transform is recommended by Borgefors [14, 15]. In Borgefors' research, the city block and the 3×3 mask with (3,4) distance transform were compared to the Euclidean distance and the use of the 3×3 with (3,4) distance transform was recommended. Using the Euclidean distance itself is usually not necessary as the edge points are influenced by noise. Furthermore, it is inefficient to compute exact distances from inexact edges. The 5×5 mask and 7×7 mask show better approximation accuracies in [28], but are more suitable for dense texture object matching such as street map matching.

FIGURE 3.12: An edge image converted to a distance image using the distance transform algorithm.

The 3×3 mask with (3,4) distance transform includes a forward mask and a backward mask and is illustrated in Figure 3.13. Two passes are made over the image by propagating the computed distance values across the image like a wave. The forward mask travels from left to right and from top to bottom; the backward mask then travels from right to left and from bottom to top.

FIGURE 3.13: The 3×3 mask with value ($d1$=4,$d2$=3) distance transform.

For each traversal, a distance mask is used for the propagation of the distance values. The approximation is expressed in the following equations:

Forward:

$$C_{i,j} = \min(C_{i-1,j-1} + 3, C_{i,j-1} + 4, C_{i+1,j-1} + 3, C_{i-1,j} + 4, C_{i,j}) \tag{3.15}$$

Backward:

$$C_{i,j} = \min(C_{i,j}, C_{i+1,j+1} + 3, C_{i,j+1} + 4, C_{i-1,j+1} + 3, C_{i+1,j} + 4) \tag{3.16}$$

3.4.2 Template Matching Using Chamfer Distance

Once the distance image has been computed, it is used to determine the similarity between templates of the target object and the input image. This process is known as chamfer distance matching. The template is a predefined binary hand contour image. The input image combines skin cues and motion cues. The binary image is transformed into an edge image. Subsequently, the distance image of the input image is computed.

In order to efficiently detect the hand in the image, the template image is overlapped onto the transformed image. It is passed over each pixel on the transformed image from left to right and top to bottom.

For every edge pixel in the template image, the corresponding pixel in the distance transformed image is summed, as shown in Figure 3.14. This summed value can also be understood as a distance measure. The location that contains the smallest sum value is considered as the target position.

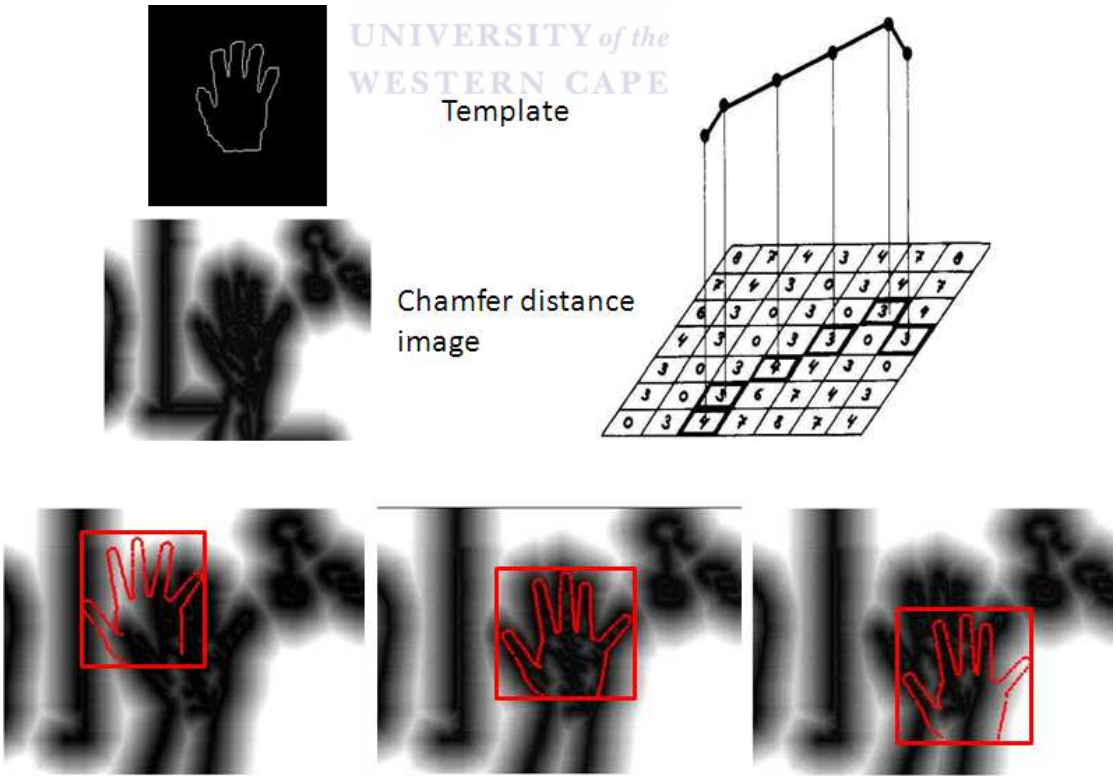


FIGURE 3.14: Template matching by chamfer distance [100].

Chamfer distance matching can be made more robust by using the mean of the thresholded distance. This approach reduces the effect of outliers and missing edges [106].

3.4.3 Hierarchical Template Matching for Hand Detection

Chamfer matching can not be used to detect the hand at different scales if only a single template is used. Multiple templates at different scales can be used to solve this problem. Borgefors [15] introduced hierarchical chamfer matching in which a coarse-to-fine search is performed using a resolution pyramid of the image. A number of templates that contain the target object at different scales are represented by a cluster prototype. This method was optimized such that the prototype is first compared to the query image using chamfer distance matching and only if the error is below a given threshold value are the templates within the cluster compared to the image. The use of a combined template hierarchy and coarse-to-fine approach in shape matching achieves very large speed-ups [37].

Another optimization to this method is to use the size of the face to estimate the size of the hand [59]. The optimal scale size of the hand can be determined in the scene at initialization. This method vastly improves the speed of the system at initialization by scaling it relative to the size of the face.

In this research, the location of the hand is determined only once using hierarchical template matching. The size of the face is used to estimate the size of the hand, thus speeding up processing. When the region-of-interest of the hand is found, it is tracked using the camshift tracking algorithm.

3.5 CAMShift Tracking Algorithm

Camshift is the preferred tracking algorithm for this project. Therefore, some theory on the algorithm is provided as a basis of overall understanding.

Camshift is based on the mean-shift algorithm. It is an effective and computationally efficient colour-based tracking technique. It ascends the density gradients to locate the peak of probability distributions [17]. It uses a search window positioned over a section of the distribution. The maximum distribution can be computed by an average computation within this search window. The search window is then shifted to the location of this maximum distribution and an average computation is repeated. This process is repeated until the algorithm obtains a local maximum and converges. Camshift tends to be effective in overcoming transient occlusions. This is because the search

window tends to first absorb the occlusion but reverts to the dominant distribution when the occlusion passes.

The pixel data of the video frames are represented as a probability distribution. Each pixel $I(u, v)$ in a frame is assigned a probability value $P(u, v)$, depending on its colour. P is a value which indicates how likely it is that the related pixel belongs to the target. Therefore, a target model of the desired object that may, for example, have been selected by the user, is created in the form of a 1D histogram. Bradski takes the H-channel of the HSV colour space to describe the target object by a range of colour hues. Depending on the range of the hue in the histogram, the probability value lies in the range $[0, 1]$. To increase performance, the probability distribution for the mean-shift algorithm is created within a search window. This is, in most cases, smaller than the image. The probability values are assigned to the pixels depending on their hue. 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 position of the target object in the actual frame. To calculate the maximum within the search window, statistical moments of zeroth and first order are used [3, 17].

A statistical moment of order p and q can generally be formulated as [17]:

$$m_{pq} = \sum_{(u,v)\in\sigma} P(u, v) \cdot u^p \cdot v^q \tag{3.17}$$

Where σ is the distribution. The zero-order moment m_{00} is given by [17]:

$$m_{00} = \sum_{(u,v)\in\sigma} P(u, v) \tag{3.18}$$

This corresponds to the integral over the distribution σ. Similarly, the moments of first order are given by [17]:

$$m_{10} = \sum_{(u,v)\in\sigma} P(u, v) \cdot u \quad , \quad m_{01} = \sum_{(u,v)\in\sigma} P(u, v) \cdot v \tag{3.19}$$

The position of the target object $c = (c_x, c_y)$ is then calculated as [17]:

$$c_x = \frac{m_{10}}{m_{00}} \quad , \quad c_y = \frac{m_{01}}{m_{00}} \tag{3.20}$$

After the position of the target object has been determined, the size of the search window is adapted for the next frame. This is achieved using the moments of the zeroth order and the maximum value in the distribution P_{max}.

$$w_s = s \cdot \sqrt{\frac{m_{00}}{P_{max}}} \quad and \quad h_s = 1.2 \cdot w_s \tag{3.21}$$

where w_s is the width and h_s is the height of the search window. Bradski [17] multiples the height with a factor of 1.2 since the human face is approximately elliptical. However, the hand is an articulated object. Based on the centre of the original tracking window, a new window with a specific size of 120×100 can locate the hand over any deformations in time.

3.6 Conclusion

This chapter discussed the hand tracking strategy used in this research. The procedures involved in this process were also discussed. These are: adaptive skin detection, GMM background subtraction, hand detection and hand tracking. Adaptive skin detection uses face detection and H-S histogram back projection to estimate a skin probability image – the skin cue image. A motion cue image is created using GMM background subtraction to form a new image that only highlights the pixels of the hand region. The skin and motion cue images are combined. Based on the new image, hierarchical chamfer matching detects the hand location and in the frame. The speed of this algorithm is increased by choosing a template that has a size similar to that of the face. Finally, the camshift tracking algorithm continuously and successfully tracks the hand motions.

Chapter 4

Hand Shape Recognition

Chapter 3 described the hand tracking method used in this research. The CamShift algorithm was used to track the hand and extract a region of interest (ROI) of size 120×100 pixels containing the hand.

This chapter explains the method of recognition of hand shapes employed by this research. The method of extracting contour features from the hand ROI and the use of these features to efficiently recognise the hand shape is also explained. In Section 4.1, an overview of the hand shape recognition system developed in this research is described.

This chapter is organized as follows: feature extraction and normalization is explained in Section 4.2; Section 4.3 compares two machine learning algorithms: neural network and support vector machine and chooses the more suitable one. Using the selected machine learning algorithm to implement recognition is presented in Section 4.4. Section 4.5 concludes the chapter.

4.1 System Design for Hand Shape Recognition

In order to gain a distinct feature from different hand shapes, the hand contour is used to represent the feature vector. Considering noise might still exist in the binary images resulting from the combination of skin and motion cues, the connected component analysis algorithm [34] is used to extract the contours of all blobs in the scene. The contour which has the largest number of contour pixels is considered to be the hand. In addition, a computational geometry algorithm [35] can be used to obtain an oriented minimum bounding box to segment the hand contours from the ROI tracked by camShift.

In order to overcome misalignment invariance, normalization is carried out by rotating and scaling the extracted hand region. The region is rotated based on the orientation of

FIGURE 4.1: Hand shape normalization.

the principal axis of the region, such that the axis is aligned to one of the image axes. Thereafter, the image is scaled to a resolution of 20×30 pixels. This step is essential when aiming to reduce the computational cost as well as the effects of variation in the size of the hand. An example of the results of this entire process is illustrated in Figure 4.1. Pre-processing allows for better generalization of features when using a machine learning algorithm for recognition.

Support Vector Machines (SVMs) are selected as the preferred machine learning technique for this research. This choice is justified in Section 4.3. LibSVM [22] is an open-source implementation of SVMs. It is used in this research as it provides a simple and effective means to train and predict data using SVMs.

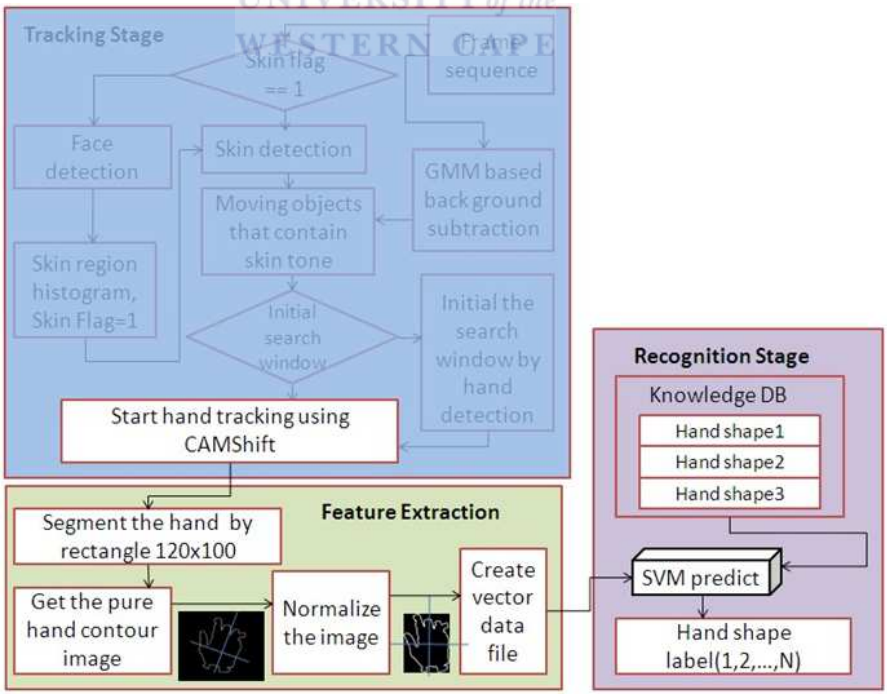


FIGURE 4.2: Process flow chart for feature extraction and hand shape recognition.

4.2 Feature Extraction and Normalization

In the following sections, the techniques that are used to extract the hand contour from the ROI in a clear and complete manner are described. Subsections 4.2.1 through 4.2.3 discuss: morphological operations, connected component analysis and feature normalization by applying a minimum bounding rectangle.

4.2.1 Morphological Operations

Background subtraction may generate holes in the resulting image when the background contains the same colour as the target and may generate noise when the illumination changes.

To complete the silhouette of the hands, four morphological operations are used, namely: dilation, erosion, opening and closing. The use of these operations ensures that noise is smoothed out and holes in the hand area of the silhouette are filled in.

These operations are based on mathematical morphology. They arise in a wide variety of contexts such as removing noise or joining disparate elements in an image. These operations are used on either the binary images or greyscale images. However, in this study only their use on binary images is discussed.

The following subsections describe these morphological operations.

4.2.1.1 Dilation and Erosion

Dilation and erosion are two fundamental morphological operations. The dilate operation can expand foreground regions. It is often used when attempting to find connected components, that is, large discrete regions of similar pixel colour or intensity. Dilation is necessary in many cases where a large region might be broken apart into multiple components after the background subtraction technique. The dilation will cause such components to merge together into one.

The erode operation can erode away the boundaries of regions as well as eliminate speckle noise in an image. It is illustrated in Figure 4.3. The larger regions that contain visually significant content are not affected [16].

These operations have two major elements: a binary input image and a 3×3 mask. As the 3×3 mask is scanned over the binary image, the maximal or minimal pixel value is computed by a pattern of elements relative to an origin at a certain pixel. This causes

FIGURE 4.3: Dilation and erosion.

bright regions within an image to grow or shrink. Considering the binary image only consists of 1s and 0s – "1" represents foreground pixels; "0" represents background pixels – the dilation operation can be defined as [65]:

$$A \ominus B = \{X | B_x \cap A \neq \emptyset\} \tag{4.1}$$

and the erosion process can be defined as [65]:

$$A \oplus B = \{X | B_x \subseteq A\} \tag{4.2}$$

where A is a binary image, dilated or eroded by a mask B. B_x is the set B translated by the vector X. The mask scans over the image. For the dilation process, if the 8 pixels that surround the centre of the 3×3 mask all equal "0", then the current pixel remains a"0" – it is background; otherwise the current pixel is set to "1" – it is foreground. For the erosion process, if the 8 pixels that surround the centre of the 3×3 mask are all equal to "1", then the current pixel remains a "1" – it is foreground; otherwise the current pixel is set to "0" – it is background.

4.2.1.2 Opening and Closing

Opening and closing are both derived from the fundamental operations of dilation and erosion. These two operators are important and normally applied to binary images.

Opening is defined as an erosion following by a dilation. It is similar to erosion and tends to remove some of the boundary pixels from the edges of foreground pixels and eliminates small clumps of undesirable foreground pixels. However, it is less destructive than erosion in general. The effect of the operator is to separate the foreground objects and to preserve large foreground regions that have a similar shape. The opening operation of a binary image can be defined as [40]:

$$A \circ B = (A \ominus B) \oplus B \tag{4.3}$$

where the set A represents a binary image and the set B represents the 3x3 mask.

Closing is opening performed in reverse. It is defined as a dilation followed by an erosion using the same mask for both operations. It is similar to dilation in that it tends to enlarge the boundaries of foreground regions in an image but it is less destructive of the original boundary shape [40]. The effect of the operator is to shrink holes in foreground objects while attempting to keep the original shape of the objects. The closing operation of a binary image can be defined as [40]:

$$A \bullet B = (A \oplus B) \ominus B \tag{4.4}$$

where set A represents as a binary image and the set B represents the 3x3 mask.

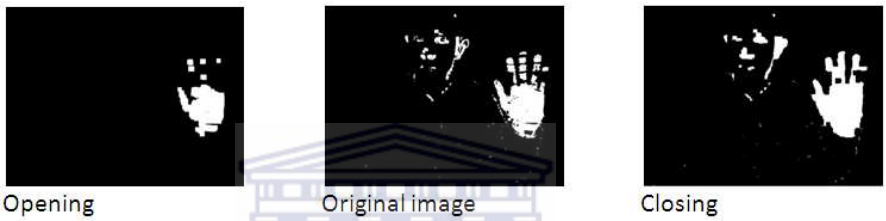


FIGURE 4.4: Opening and closing.

4.2.1.3 The Disadvantages of Morphology Algorithms

While morphology algorithms can be useful in limiting noise and filling in holes in the image, they have disadvantages.

In the foreground image, the pixels cluster around the area of each blob, which can potentially be the target object or noise in the image. The pixels of smaller unwanted blobs are sparsely distributed and are not clustered. The pixels considered to be noise can be shrunk or eliminated by applying erosion, and then applying dilation to rebuild the area of the remaining components that was shrunk in erosion. This technique works well for images which contain small amounts of sparsely distributed noise. However, in this process, the target blob is affected as well.

Iterating each of the two operations – opening and closing – many times would not be particularly useful. The opening or closing should only be performed once with the same mask [16] to remove small amounts of scattered noise. It can be used as an initial clean up on noise on the foreground image. The use of connected component analysis can further reduce noise. This is explained in the following subsection.

4.2.2 Connected Component Analysis

Connected Component Analysis is an algorithm that attempts to extract all contours of the blobs in an image. Subsequently, the biggest contour will be considered as the target object.

Connected components analysis [34] is implemented by applying 8-connectivity labelling operators to scan the binary image row by row and column by column. Figure 4.5 illustrates the 8-connectivity labelling operator [85].

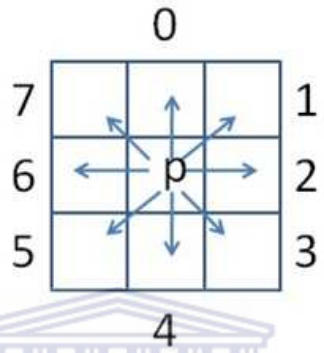


FIGURE 4.5: 8-connectivity labelling operator.

A point p is labelled when a foreground pixel, 255, is found. The p denotes the pixel to be labelled at any stage in the scanning process. The 8-connectivity labelling operator examines the 8 neighbours of p which have already been encountered in the scan based on this information. The labelling of p occurs as follows [33]:

1. If all the neighbours of p are of the value 0 then assign a label q.
2. If all the neighbours of p are of the value 255 then assign a label p.
3. If more than one of the neighbours are of the value 255, assign one of the labels to p and make a note of the equivalences.

After the completion of the scan, a secondary scan is made in which each label is replaced by a label assigned to its equivalent class. The foreground components have only two labels p and q belonging to the foreground and background respectively. All blobs that are connected to each other are labelled as a single foreground object [33]. After the connected component labelling, the boundaries of the blobs are approximated to polygon lines or to convex hulls to a clear boundary.

The foreground pixels are gathered in the connected components. Each contour of connected components is labelled with an index and a number of pixels. The remaining

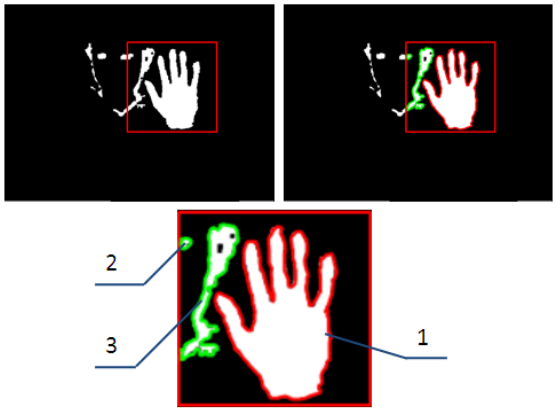


FIGURE 4.6: Connected components analysis to binary image of ROI.

smaller blobs are filtered out depending on the threshold on the size of each retrieved contour [1, 10]. In Figure 4.6, the connected components analysis on the binary image that is generated by combining skin cue and motion cue is illustrated.

Considering this technique is implemented in the ROI with a size of 120×100, the hand contour can be identified easily due to its larger size. The hand region labelled 1 would remain and the other small regions from labels 2 and 3 would be eliminated. The processing speed for this computation is very fast since it is a small region.

4.2.3 Feature Normalization by Applying a Minimum Bounding Rectangle

A computational geometry algorithm [35] can be used to obtain an oriented minimum bounding rectangle to segment the hand contours from the ROI of camshift. The minimum bounding rectangle with an arbitrary orientation contains the vertex set S of the geometric object.

The algorithm for a minimum bounding rectangle is based on a theorem by Freeman and Shapira: the minimum area rectangle enclosing a convex polygon has a side collinear with an edge of the polygon. The input is assumed to be a convex polygon, the vertices of which are given in clockwise order [35]. Figure 4.7 illustrates the process of finding the minimum bounding rectangle that encases the hand contour. It can be seen that the minimum bounding rectangle, depicted in the right-most image of Figure 4.7, has a side collinear with the hand.

The following rotating caliper algorithm determines the minimum bounding rectangle of the convex polygon, in this case the hand contour [35]:

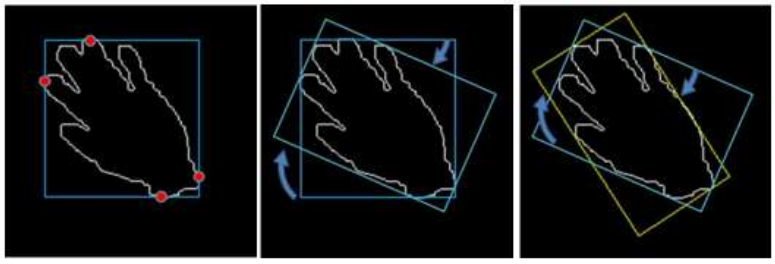


FIGURE 4.7: The process of determining the minimum bounding rectangle of the hand contour.

1. The hand contour p is to be encoded into a chain $a_1 a_2, \cdots, a_n$. This is also referred to as a closed curve or convex polygon.

2. A bounding rectangle encloses the given curve. This rectangle is formed by lines $x_1 = W_{\min}$, $y_1 = H_{\min}$, $x_2 = W_{\max}$, $y_2 = H_{\max}$. $W_{\min} = \min_j \sum_{i=0}^{j} a_{ix}$; $H_{\min} = \min_j \sum_{i=0}^{j} a_{iy}$; $W_{\max} = \max_j \sum_{i=0}^{j} a_{ix}$; $H_{\max} = \max_j \sum_{i=0}^{j} a_{ix}$, where $j = 0, \cdots, n$; the a_{ix} and a_{iy} are the x and y components of the vector denoted by a_i. Compute the area of the rectangle formed by the four lines and record it as the current minimum bounding rectangle.

3. Rotate the lines clockwise and encase the given curve. Find the coordinate parameters (x,y) of the all the chain points lying on these lines. If more than two chain points lie on one line, only the first and last are entered in the vertex list. Label the selected chain point with the number of the chain link and keep it in an ordered vertex list.

4. The current bounding rectangle and the previous bounding rectangle form two sets of callipers. Compute the area of the current bounding rectangle and compare it to the previous one using these callipers.

5. Update the current minimum bounding rectangle if necessary based on the new bounding rectangle.

6. Repeat steps 3–5 until the area of the rectangles corresponding to all pairs of successive vertices have been computed.

7. Output the minimum bounding rectangle.

The details of the geometric computation of this algorithm can be found in [34, 35].

In order to solve misalignment invariance, normalization is carried out by rotating and scaling the extracted hand region. The selected region is rotated according to the orientation of the principal axis of the region, in order to align the axis to one of the image axes. It is also essential to scale the resolution to a size of 20×30 pixels. This allows

for better generalization of features when training and testing using Machine Learning methods. This process has two advantages: the recognition accuracy of the Machine Learning methods is improved and the computational cost is reduced significantly.

4.3 Neural Network vs Support Vector Machine

This section presents two popular machine learning algorithms: neural networks (NNs) and support vector machines (SVMs). A comparative study of the two techniques is undertaken. Subsequently, the better of the two is adopted for the task of hand shape recognition in this research. Subsection 4.3.1 and 4.3.2 present the basic theories of NNs and SVMs. Subsection 4.3.3 gives a summary of this comparison.

4.3.1 Neural Networks

A neural network is a machine learning algorithm composed of interconnecting artificial neurons that simulate real biological neurons to solve artificial intelligence problems.

In principle, an artificial neuron is a mathematical algorithm that implements non-linear mapping. Consider a vector X that contains I input signals:

$$\vec{X} = (x_1, x_2, ..., x_I) \tag{4.5}$$

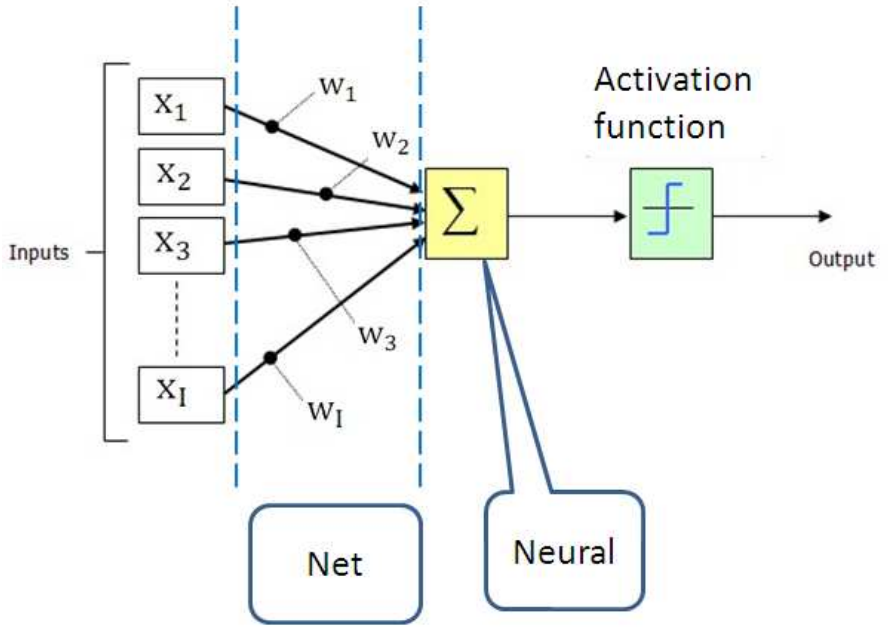


FIGURE 4.8: Artificial neural configuration [96].

Referring to Figure 4.8, each input signal x_i is assigned a weight w_i to strengthen or deplete the input signal. The artificial neuron computes the Net input signal, and uses an activation function F_{AN} to compute the output signal y given the Net input. The strength of the output signal is further influenced by a threshold value θ, also referred to as the bias.

The Net input signal to a neuron is usually computed as the weighted sum of all input signals:

$$\text{Net} = \sum_{i=1}^{I} x_i w_i \tag{4.6}$$

The neurons that compute the Net input signal as the weighted sum of input signals are referred to as summation units. An activation function then receives the Net input signal and determines the output of the neuron. Popular activation functions include: linear functions, step functions, Ramp functions and Gaussian functions. All these functions have respective weaknesses and strengths. Choosing a particular type of function depends on the specific problem.

In this discussion, the step function is considered as a basis of understanding since it is a basic function. This activation function is given by:

$$F(Net - \theta) = F(\sum_{i=1}^{I} x_i w_i - \theta) = \begin{cases} \beta_1; & \text{if Net} \geq 0 \\ \beta_2; & \text{if Net} < 0 \end{cases} \tag{4.7}$$

The w_i and θ are adjusted during the training process. The learning quality depends on the example data. The learning consists of adjusting weights and threshold values until a certain criterion is satisfied.

The process of training takes the form of providing the neuron with a data set consisting of input vectors and a desired output associated with each input vector. This data set is referred to as the training set. Supervised training then involves adjusting the weight vector $\vec{W}$ and threshold value θ such that the error between the real output of the neuron and the target output is minimized. The real output of the neuron is given by:

$$y = F(Net - \theta) \tag{4.8}$$

To simplify the learning equations, the input vector is augmented to include an additional input unit x_{i+1}, referred to as the bias unit. The value of x_{I+1} is always -1, and the

weight x_{I+1} serves as the value of the threshold. The Net input signal to the AN is then calculated as:

$$Net = \sum_{i=1}^{I} x_i w_i + x_{I+1} w_{I+1} = \sum_{i=1}^{I+1} x_i w_i \tag{4.9}$$

where $\theta = x_{I+1} w_{I+1} = -w_{I+1}$.

In the case of the step function, an input vector yields an output of 1 when $\sum_{i=1}^{I+1} x_i w_i \geq 0$, and 0 when $\sum_{i=1}^{I+1} x_i w_i < 0$.

A single neuron can only be used to realize linearly separable functions. For more complex problems, an interconnected network of neurons is required. The neurons are arranged in layers, with outputs from one layer feeding into inputs in the next layer. Multilayer perceptron networks consist of three layers of neurons, illustrated in Figure 4.8.

This network has an input layer, a hidden layer and an output layer. Each neuron may be connected to several neurons in the next layer where the output of the neuron in one layer feeds in as the input to neurons in the next layer. The input layer only distributes the input value to the next layer and is often referred to as a two-layer network. The output may consist of several neurons.

This implies that NNs are not limited to binary classification problems but are suitable for multi-classification problems as well [20, 96]. Multilayer perception properties include universal approximation of continuous non-linear functions, learning with input-output patterns and the ability to create advanced network architectures with multiple inputs and outputs [96].

However, multilayer perception properties introduce the following issues: NNs may not be able to find an optimal global solution to the problem due to the use of the back-propagation, rather locating a local minimum; and it may be difficult to determine the correct number of neurons required for an optimal neural network for a task at hand. It is important to note that even if the neural network converges, it may not result in a unique solution [75].

4.3.2 Support Vector Machines

The foundations of support vector machines (SVMs) were developed by Vapnik [111]. Using statistical learning theory, SVMs were originally developed to deal with binary classification problems, but have since been extended to support classification of multiple

classes [2]. SVMs are gaining popularity due to the many attractive qualities of the technique. The training procedure of the technique finds a global and unique solution using a hyperplane to separate the different classes in a higher dimensional space or feature space with a maximum margin. The separation achieved by the maximum margin overcomes the problem of converging on local minima. A kernel function is used to project the feature data in the input space to the feature space. This allows SVMs to use linear classification techniques to solve non-linear classification problems. During the training phase, the weight can be scaled very well since the training model only selects feature vectors that lie on the respective hyperplanes in the feature space – support vectors – to carry out the training process.

The following subsections describe major techniques involved in the use of SVMs. Subsection 4.3.2.1 presents maximum margin classification; Subsection 4.3.2.3 explains the kernel function; soft margin classification is introduced in Subsection 4.3.2.2; and Subsection 4.3.2.4 presents a method of using binary-class classification to be able to handle multi-class classification problems.

4.3.2.1 Maximum Margin Classification

Maximum margin classification is an important technique for classification optimization. It is an empirical performance requirement that provides the least change that causes a misclassification when a small error occurs in the location of the boundary between two classes[27]. Maximum margin classification attempts to find the global and unique optimal solution to solve the classification problem.

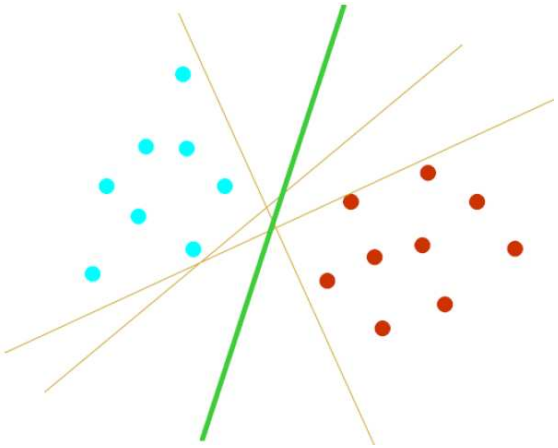


FIGURE 4.9: The optimal solution to classify the data in many hyperplanes [76].

Referring to Figure 4.9, there are many linear classifiers or hyperplanes that can separate a given set of data. However, only one of these hyperplanes – the green line – achieves maximum separation of the data.

The importance hereof is that a hyperplane drawn to separate that data may end up closer to one subset of the dataset than others, which is undesirable. Only the hyperplane that separates the two classes with the maximum margin in this space is the optimal solution. This hyperplane is also known as the maximum margin hyperplane or optimal hyperplane.

Maximum margin classification attempts to find an optimal hyperplane which separates the data set of one class from that of the other class in an exact manner. The optimal hyperplane passes the mid-point between the boundaries of these sets and is the most distant hyperplane from both sets. This hyperplane is expected to be the optimal classification of the sets.

The training vectors that lie on the boundaries are called support vectors. An illustration of the optimal hyperplane with the maximum margin is provided in Figure 4.10.

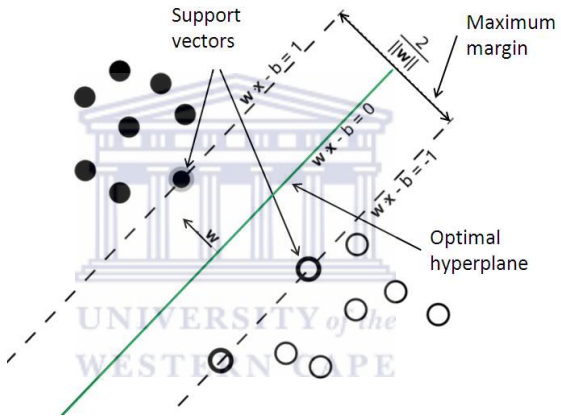


FIGURE 4.10: The optimal hyperplane with the maximum margin [76].

To present the use of the maximum margin to solve binary classification problems such as the illustration in Figure 4.10, let the training set S have n vectors as follows:

$$S = \{(x_i, y_i)|x_i \in \mathbb{R}^p, y_i \in \{-1, 1\}\}_{i=1}^{n} \tag{4.10}$$

Each point, x_i, can be a n-dimensional vector but is a 2-dimensional vector in this example. Each element in the vector is a positive real number. y_i is the class of the corresponding vector and indicates the class to which the vector x_i belongs.

Let D describe the distance between the boundaries of the two classes. The margin for the linear classifier shown in Figure 4.10 is given by:

$$D = \frac{2}{||\mathbf{w}||} \tag{4.11}$$

SVMs use a weight coefficient vector w and a bias term b to maximize the margin between the parallel hyperplanes that are as far apart as possible while still separating the data.

In order to maximise the margin, the term $||\mathbf{w}||$ has to be minimized. Considering the points still need to fall outside the margin, the w constraints are given by:

$$\mathbf{w} \cdot x_i + b \geq 1 \; for \; y_i = 1 \tag{4.12}$$

and

$$\mathbf{w} \cdot x_i + b \leq -1 \; for \; y_i = -1 \tag{4.13}$$

These two can be combined into one set of inequalities:

$$y_i(\mathbf{w} \cdot x_i + b) - 1 \geq 0, \; \forall \; i = 0, 1, 2, ..., n \tag{4.14}$$

The boundary hyperplanes of both of the classes can be described as:

$$\begin{cases} \text{a)} \quad \mathbf{w} \cdot x_i + b = 1 \\ \text{b)} \quad \mathbf{w} \cdot x_i + b = -1 \end{cases} \tag{4.15}$$

$\mathbf{w}$ and b are scaled using the support vectors x_i that lie on the respective boundary hyperplanes.

This procedure involves the following quadratic programming optimization problem: considering the norm of $\mathbf{w}$ – $||\mathbf{w}||$ – involves a square root, it is necessary to alter the equation by substituting $||\mathbf{w}||$ with $\frac{1}{2}||\mathbf{w}||^2$, the factor of $\frac{1}{2}$ being used for mathematical convenience. The optimization for minimizing $\mathbf{w}$ is defined as:

$$\begin{aligned} &\text{Minimize } \frac{1}{2}||\mathbf{w}||^2 \\ &\text{Subject } to \; y_i(\mathbf{w} \cdot x_i + b) - 1 \geq 0, \; \forall \; i = 0, 1, 2, ..., n \end{aligned} \tag{4.16}$$

The constraint formulation for each i in Equation 4.14 can be replaced by constraints on the Lagrangian multipliers themselves, that is, $\alpha_1, \alpha_2, ..., \alpha_n \geq 0$. As such, $\mathbf{w}$ and the constraint functions are simplified. The new constraint equations are multiplied by positive Lagrangian multipliers and subtracted from the objective function. The Lagrangian is defined as:

$$L_P = \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{n} \alpha_i (y_i(\mathbf{w} \cdot x_i + b) - 1) \tag{4.17}$$

It is then required to minimize L_P in Equation 4.16 with respect to $\mathbf{w}$ and b and, at the same time, cause the derivatives of L_P with respect to all the a_i to vanish, with all these subject to constraints $a_i \geq 0$. This is given by:

$$\min\{L_P\} = \min_{\mathbf{w},b}\{\frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{n} \alpha_i (y_i(\mathbf{w} \cdot x_i + b) - 1)\} \tag{4.18}$$

L_P is the primal Lagrangian formulation. In order to deal with the convex quadratic programming problem, L_P is translated into L_D to form a dual Lagrangian formulation which is referred as Lagrangian duality.

Requiring that the gradient of L_P with respect to $\mathbf{w}$ and b vanish gives the conditions:

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i x_i \tag{4.19}$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{4.20}$$

Substituting Equation 4.19 and Equation 4.20 into Equation 4.17 yields:

$$L_D = \sum_{i=i}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j x_i \cdot x_j \tag{4.21}$$

Therefore 4.16 can also reduce to a maximization function:

$$\max\{L_D\} = \max_{\alpha_i}\{\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j (x_i, x_j)\}$$
$$\text{Subject } to \sum_{i=1}^{n} \alpha_i y_i = 0 \text{ and } \alpha_i \geq 0, i = 1, 2, ..., n \tag{4.22}$$

The dual form Lagrangian formulation L_P reveals that the maximum margin hyperplane and the classification task are only functions of the support vectors – the training data that lie on the margin.

According to Equation 4.15, those training data points with non-zero α_i values will fall on the +1 or -1 plane. These are the data points that contribute to defining the decision boundary. If the other data points are removed and the classifier is retrained on the remaining data, the training will result in the same decision boundary. Support vectors with larger α_i are more important since they have a stronger influence on the decision boundary.

In order to conveniently handle $\mathbf{w}$ and other coefficients, the constraint formulation 4.14 is replaced by the Lagrangian formulations L_P and L_D.

The two Lagrangians are given different labels, "P" for primal and "D" for dual, to indicate that the two formulations are different: L_P and L_D arise from the same objective function but with different constraints and the solution is found by minimizing L_P and maximizing L_D.

4.3.2.2 Soft Margin Classification

Ideally an SVM analysis should produce a hyperplane that completely separates the feature vectors into two non-overlapping groups. Realistically, if the data has noise and outliers, perfect separation may not be possible. These noise and outliers may also cause overfitting which prevents the trained model from generalizing well. The soft margin method attempts to find a hyperplane that splits the examples as clearly as possible, while still maximizing the distance to the nearest clearly split data.

This is achieved by using a set of variables ξ, also known as slack variables, which measure the degree of misclassification of the data x_i. The cost function can be expressed as:

$$\begin{aligned} &\min_{\mathbf{w},b,\xi} \{\frac{1}{2}||\mathbf{w}||^2 + C \cdot \Sigma_{i=1}^{n} \xi_i\} \\ &\text{Subject } to\ y_i(\mathbf{w} \cdot x_i + b) \geq 1 - \xi_i \\ &\text{where } \xi_i \geq 0 \text{ and } 0 \leq \alpha_i \leq C \end{aligned} \tag{4.23}$$

The cost parameter C determines the trade-off between allowing rigid maximum margins to be enforced and tolerating the training errors, in order to allow some flexibility in separating the categories. In LibSVM, a grid search function is available, which can be used to find the best value for C and γ. The grid search function uses geometric steps to try each parameter value across the specified search range. If the model fit improves, then the centre of the search moves to a new point where the process is repeated. If there is no improvement, the step size will be reduced and the search will be repeated.

When the step size of the search is reduced to a specific tolerance level, then the pattern search will stop [2].

According to formulation 4.23, the solution to the optimisation problem of Equation 4.22 subject to $\sum_{i=1}^{n} \alpha_i y_i = 0$ and $\alpha_i \geq 0$ is given by the saddle point of the Lagrangian:

$$L_P = L(w, b, \alpha, \xi, \beta) = \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \alpha_i (y_i(\mathbf{w} \cdot x_i + b) - 1 + \xi_i) - \sum_{j=1}^{n} \beta_i \xi_i \quad (4.24)$$

where α and β are both the Lagrangian multipliers. The Lagrangian has to be minimized with respect to $\mathbf{w}$, b, x and maximized with respect to α and β. Referring to Equation 4.21, classical Lagrangian duality enables the primal formulation 4.24 to also be transformed to its dual formulation:

$$\max L(\alpha, \beta) = \max_{\alpha,\beta}(\min_{w,b,\xi} L(w, b, \alpha, \xi, \beta))$$
$$\text{Subject } to \sum_{i=1}^{n} \alpha_i y_i = 0 \text{ and } \alpha_i \geq 0, i = 1, 2, ..., n \quad (4.25)$$

4.3.2.3 Kernel Function

The description of linear classification mentioned previously is based on an assumption: the data is linear separable so a separating hyperplane can be used to divide the data. However, it is often the case that the data is far from linear and the datasets are inseparable. To allow for this, kernel functions are used to non-linearly map the input data to a higher-dimensional feature space where a hyperplane can be used to do the separation [73]. A very simple illustration of this is shown in Figure 4.11.

Let Φ be a transformation to a higher dimensional space, x_i be a training vector in the input space and x_j be the corresponding training vector in the higher dimensional space. The transformed space should satisfy the requirement that the distance is defined in the transformed space and that the distance has a relationship to the distance in the original space. The kernel function $K(x_i, x_j)$ that satisfies the above conditions is introduced. The kernel function satisfies [73]:

$$K(x_i, x_j) = \phi(x_i)\phi(x_j) \quad (4.26)$$

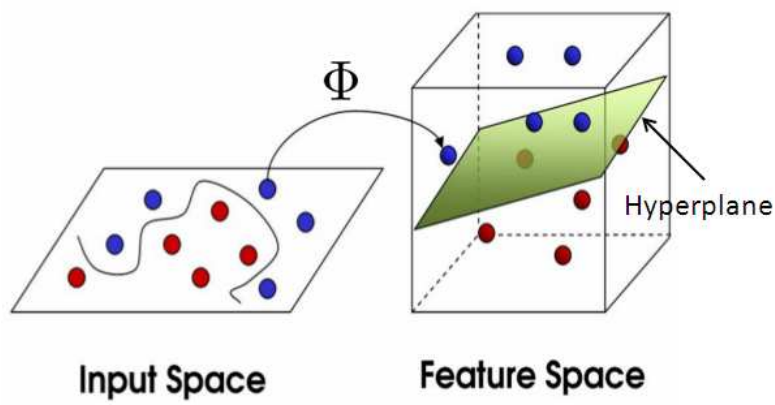


FIGURE 4.11: The use of kernel functions to separate inseparable data [73].

This equation indicates that the kernel function is equivalent to the distance between x_i and x_j measured in the higher dimensional space transformed by ϕ. Following this constraint, there are 4 kernels that are most used by SVMs for training and classification [73]:

1. Linear: $K(x_i, x_j) = (x_i)^T \cdot (x_j)$
2. Polynomial: $K(x_i, x_j) = (\gamma(x_i)^T \cdot (x_j) + r)^d$, where $\gamma > 0$
3. Radial basis function (RBF): $K(x_i, x_j) = exp(-\gamma \cdot ||x_i - x_j||^2)$, where $\gamma > 0$
4. Sigmoid: $K(x_i, x_j) = \tanh(\gamma \cdot (x_i)^T \cdot (x_j) + r)$, where $\gamma > 0$

where r, γ and d are kernel parameters [26].

The choice of kernel function is important as it influences the prediction capabilities of the SVM [39]. However, no standard method exists to find the most appropriate kernel [117]. Selecting an appropriate kernel is often a process of trial and error [26].

4.3.2.4 Multi-class Classification

By definition, SVMs are binary classifiers used for 2-class problems. However, they can still can be applied to multi-classification problems by using a variety of techniques. These techniques attempt to reduce the single multi-class problem into multiple binary classification problems. "One-versus-one" is one of the most common methods [44]. An brief explanation of this method follows.

Let L be the number of classes, $\frac{L(L-1)}{2}$ binary classifiers are trained using every binary pair-wise combination of the L classes. A classifier is trained for each distinct pair (u, v) and $u \neq v$. By using the data points in class u and v as positive and negative

examples, every classifier is trained to differentiate between the two classes. The max-wins algorithm [36] is used in order to combine the classifiers. This algorithm determines the correct class by selecting the class with the majority vote as voted by the classifiers [69].

4.3.3 Summary of Neural Network vs Support Vector Machine

In general, many researchers prefer to use neural networks to train knowledge-based classifiers for most tasks involving hand shape recognition and human body detection [80, 87, 118]. However, traditional neural network approaches suffer from local minima and over-fitting. The former is caused by the fact that the NN may not produce a unique and optimal solution to classify the dataset in the training phase. The latter is caused by the presence of too many dimensions and complexity in the training data. Traditional NNs do not have an effective strategy to control or mitigate this problem.

On the other hand, SVMs use the maximum margin strategy which always finds the unique optimal hyperplane to separate the dataset. They also use a kernel function which maps data points to a higher dimensional space to separate the dataset. Therefore SVMs are not encumbered by higher dimensional data such as image data. Moreover, SVMs use a cost function and a cost parameter to balance the complexity of the training data and the training error.

SVMs are claimed to be the best classification technique when given very limited or very complex data examples [111]. In this research, the SVM method is used for the task of recognition.

4.4 Recognition Using SVM

This section discusses the entire procedure used to implement the hand shape recognition sub-system of this research. Subsection 4.4.1 presents the approach for training the SVM. SVMs require examples for training. This is a costly process in terms of time. However, it is only required once as an off-line process. Once the classifier is trained, on-line recognition of new input data can be performed efficiently and accurately. To demonstrate that this system can be used to recognise hand shapes, 10 hand shapes were selected from a South African Sign Language (SASL) dictionary [42] and the system was trained on these hand shapes. These hand shapes are illustrated in Figure 4.12. In order to ensure that the resulting classifier is robust, Subsection 4.4.2 describes a method to test the classifier. Subsection 4.4.3 describes the use of the classifier to

recognise the selected hand shapes. Subsection 4.4.4 then details the modifications made to the recognition system to reduce the recognition errors arising from intermediate hand shapes.

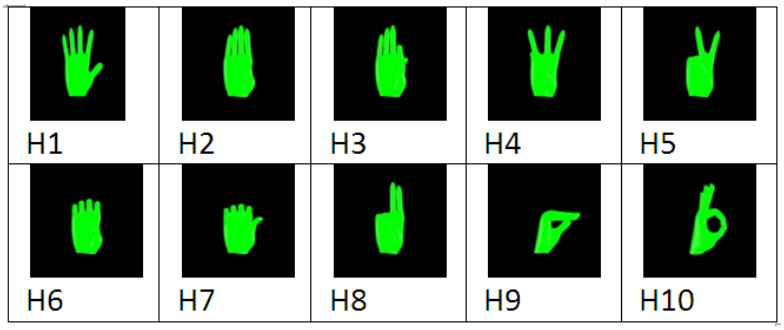


FIGURE 4.12: South African sign language hand shapes.

4.4.1 Training

In order to achieve a large set of examples for training a knowledge-based classifier, 10 videos, one for each of the 10 hand shapes, were recorded, with each video consisting of no less than 500 frames. In each video, the performer was asked to hold up the hand shape from start to stop. Using the methods explained in the previous section, the hand contours were automatically extracted from the video. Of the approximately 500 frames, 40 examples were manually selected as training data for each hand shape. This was followed by the feature normalization stage. The resolution of the hand contour image was scaled to a 20×30 pixel image as seen in Figure 4.13.

FIGURE 4.13: The normalized image of size 20×30 pixels for the hand shape labelled 1.

Finally, all the hand contour images were written to a feature data file. Each contour image was represented as a vector with a size of 600×1. In each element of the vector, a "1" was inserted to represent a contour pixel and a "0" for other pixels. The vectors of the same hand shape were given a common label. The data file is described in the format shown in Figure 4.14.

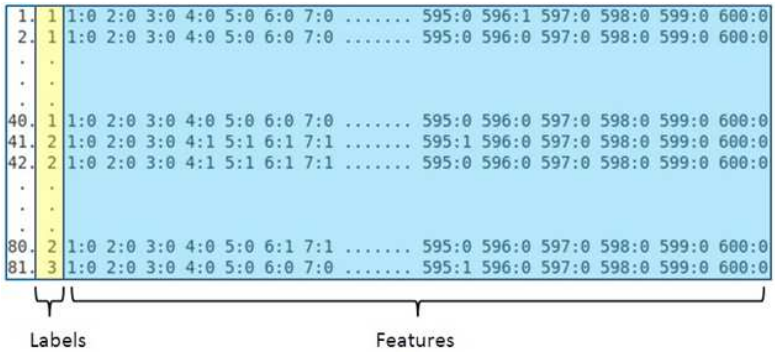


FIGURE 4.14: The data file for describing the features with labels in the training phase.

The following step in the system is to obtain the kernel parameters. Keerthi and Lin [51] have shown that the radial basis function (RBF) kernel can be used to solve the problems of a non-linear nature between attributes and multi-classes. They have also shown that the use of an RBF kernel with parameters C and γ and a linear kernel with only parameter C, provides comparable performance. The polynomial kernel on the other hand, in comparison to the RBF kernel, has more hyper parameters, thus increasing the complexity of the SVM [63]. Furthermore, it is also shown that the sigmoid kernel with certain parameters behaves similar to the RBF kernel [63]. Therefore, based on the evidence shown, it is reasonable to begin experimentation with the RBF kernel.

When determining the parameters for the RBF kernel, two parameters C and γ are required. In order to train the SVM effectively and predict the test data accurately, the best C and γ should be chosen for the given problem. To determine the best parameters, an exhaustive approach can be used by manually trying each C and γ combination, where every parameter is an exponentially growing sequence. An alternative approach is to use the grid-search function offered in LibSVM, which uses cross-validation to divide the training set into n equal subsets. The classifier is then trained on the $n-1$ subsets and the remaining subset is used for testing, for each parameter combination [63]. In this research, the latter approach is used to determine the best cross-validation accuracy for the given problem where the optimized C and γ parameters are chosen.

4.4.2 Testing

The testing phase follows the same feature extraction and normalization procedure as the training phase except that different examples are used in the training phase. The features, along with their labels, were stored in the data file, where each feature vector represents the corresponding hand contour. Using the trained model, the SVM predicted the labels for these vectors. In the testing phase the accuracy of the trained model of the SVM was identified. If an acceptable accuracy was obtained, the process moved on

to the recognition phase. If an acceptable accuracy was not obtained, more hand shape samples were obtained and used to re-train the SVM model.

4.4.3 Recognition

Once the knowledge-based data model was correctly generated, on-line recognition for new input data could be performed efficiently. First, this input data would be considered as an unknown hand shape image that was segmented from the hand detection step. The recognition process then followed the same feature extraction and normalization strategies as the training phase. However, only a single feature vector of the hand shape was written to the data file with a default label. Finally, the SVM recognised the hand shape by predicting the label of the feature vector.

4.4.4 Modification of Hand Shape Recognition to Avoid Intermediate Hand Shapes

Practically, recognition proved to be problematic in cases where the user moved between two known hand shapes. The system was unable to correctly classify intermediate hand shapes between two hand shapes that were known and the correctness of the recognition was affected. It was resolved to carry out recognition after every 3 frames. With this modification most intermediate hand shapes are skipped in the deformation between two known hand shapes. The remaining intermediate hand shapes are very similar to the next gesture and do not significantly affect the recognition accuracy. Moreover, this modification also significantly speeds up the performance of the entire system.

4.5 Conclusion

This chapter discussed the recognition aspect of the system proposed by this research. An overview of the proposed recognition sub-system was presented. A detailed discussion of the pre-processing procedure used was provided. This included the procedure used to extract and normalize features from images in the video sequence.

In selecting a suitable machine learning classification strategy to classify hand shapes using the normalized features, a detailed comparative study between SVMs and NNs was presented. SVMs were selected as the preferred strategy since they are able to generalize well on the training set and are very suitable for higher dimensional data such as images.

The implementation of the recognition system was also explained. This involved the training and testing of the SVM and subsequent recognition using the SVM. Modifications to the recognition framework to overcome recognition errors were also discussed.

Chapter 5

Hand shape Estimation

This chapter discusses the proposed method of estimating transitions between two recognised hand shapes using kinematics functions embedded into a 3D hand model.

Van Wyk [109] developed a methodology for the creation of a high quality 3D hand model in the modelling and animation tool Blender. The methodology included a method to fit the hand with kinematic constraints to ensure that only plausible hand shapes and motions could be simulated. This model is used in this research in a novel method to estimate intermediate hand shapes between two known hand shapes. Two recognised hand shapes are set as keyframes on the model and all intermediate hand shapes are then generated using interpolation functions. The constraints on the model ensure the prevention of implausible hand shapes during the transition.

In order to comprehensively describe the robustness of the framework, Section 5.1 provide an overview on kinematics and the kinematics functions that make it possible to simulate a realistic human hand using a 3D hand model. Section 5.2 introduces a powerful environment for 3D computer graphics modelling and animation, namely Blender.

The methodology developed by van Wyk [109] is reviewed in Section 5.3. Van Wyk demonstrated the effectiveness of his methodology by using it to create a high quality 3D hand model. In Section 5.4, the method proposed by this research is explained.

5.1 Hand Kinematics

Kinematics is a branch of classical mechanics that describes the motion of bodies or hands without consideration of the forces that cause the motion. In the current context, there are two methods that are used to pose the hand model: forward kinematics (FK) and inverse kinematics (IK).

Inverse kinematics involves transforming a bone lower in the tree and automatically calculating and assigning rotation angles to its parent and ancestor bones such that the bones remain connected and the transformation is valid [78]. Figure 5.1 b) depicts a 3D hand model and the bones placed inside it. The bones at the finger tips are IK-enabled bones. When each of these bones are moved, bones below them move with the them in such a way that they remain attached and the movement is plausible. This is illustrated in Figure 5.2.

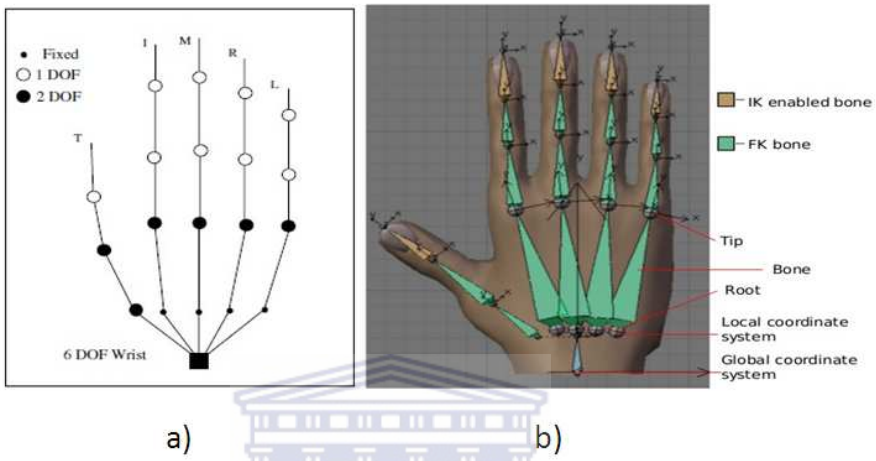


FIGURE 5.1: 3D hand model in Blender [55, 109].

When applying an IK rotational constraint to a distal bone, one can set rotational constraints for the whole chain of bones that is linked to it [109].

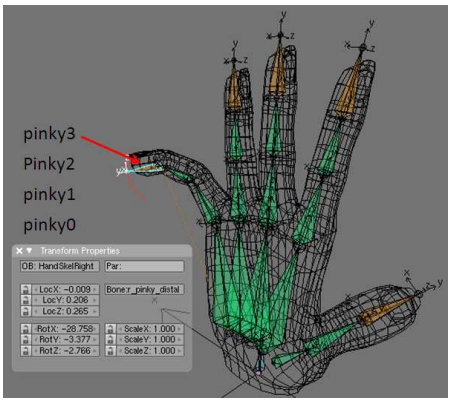


FIGURE 5.2: Illustration of IK applied to the tip of the pinky – pinky3.

Forward kinematics involves transforming each bone in the hand into the correct pose starting at bones higher up in the skeleton tree and working down the tree in the same manner. In Figure 5.1 b) all bones that are not at the tip are set as FK-enabled bones. In other words, rotating or translating these bones causes bones higher up in the tree to move accordingly and plausibly. A simple example of FK motion on the base of the pinky finger – pinky1 – can be seen in Figure 5.3.

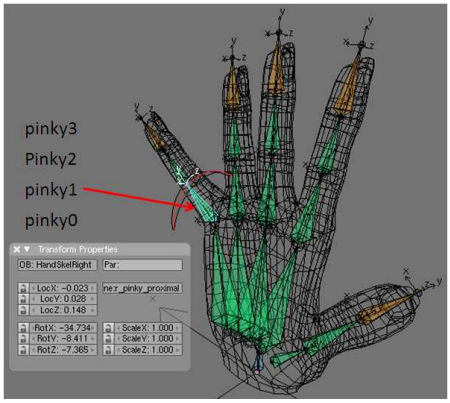


FIGURE 5.3: Illustration of IK applied to the base of the pinky – pinky1.

The next sections explain how IK and FK are used in this research.

5.2 Why Blender?

Blender is a free and open source generic interactive 3D modelling package developed by the Blender Foundation [13]. The large user community can be attributed to fact that Blender can be used for modelling, animation, rendering, compositing and the development of real-time interactive applications [13]. Moreover, Blender simplifies the task of 3D animation since one can interactively rotate and translate bones and create poses using both forward kinematics (FK) and inverse kinematics (IK). In this research, version 2.49 of Blender is used.

Blender includes many useful features such as [13]: a skeleton (armature) system with scale, rotation and translation constraints; constraint-capable forward and inverse kinematics; an embedded Python interpreter with an application programming interface (API); a state-of-the-art internal game engine with its own Python API and a visual game logic editor. With its advanced features and APIs, Blender can be a powerful interface for 3D programming and simulation with artificial intelligence. The Python APIs support information interaction between clients and servers on the Internet. Therefore, Blender has the potential to host virtual online communication and online games. By integrating the Python API with the Blender game engine, the development time of complex projects can be significantly reduced. An example of a Blender window configuration that demonstrates Blender's Python editor and the game engine's logic editor can be seen in Figure 5.4.

Blender's skeleton system is a tree of bones. A skeleton in Blender has a root bone with a global or object co-ordinate frame for the entire skeleton (skeleton space) and a local co-ordinate frame for each bone (bone space). Each bone has a root and a tip, with the

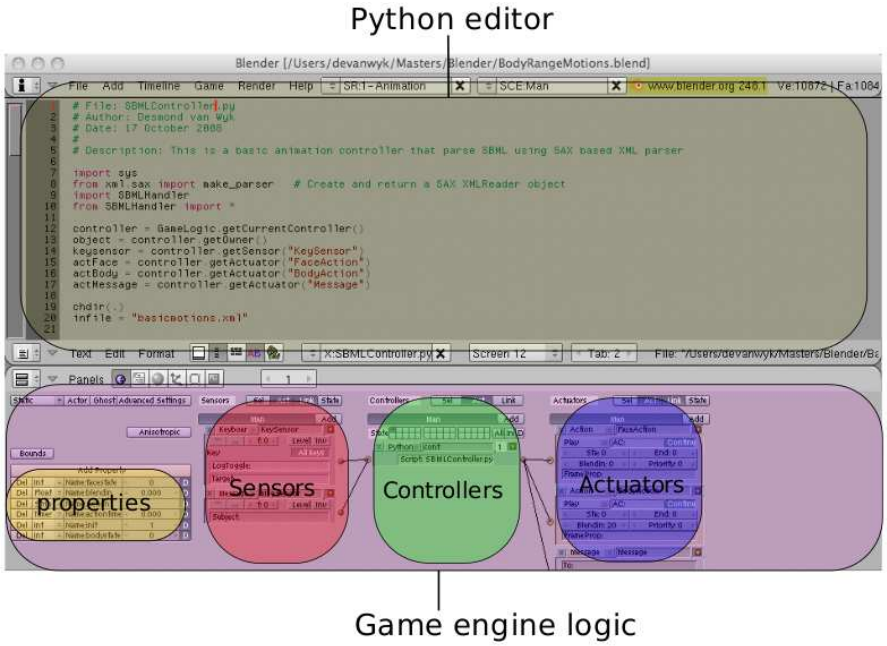


FIGURE 5.4: An example window configuration of Blender's Python editor and game engine logic [109].

co-ordinate frame situated at the root where rotation or articulation occurs. The root of a bone can thus be considered as a joint. A bone's co-ordinate system describes the bone's position in 3D space – X-, Y-, Z – co-ordinates. This system proves very useful for the purposes of this project.

Blender also has an advanced keyframe animation system with features such as constraints for its skeletal system, forward kinematics (FK), inverse kinematics (IK) and has several interfaces to aid in hand animation. Van Wyk developed a 3D hand model in Blender that is used in this research. His work is explained in the section that follows.

5.3 A Review of van Wyk's Work for 3D Hand Modelling and Animation

Van Wyk developed a methodology for the creation of a high quality 3D hand model in Blender. His model was acquired from MakeHuman [71]. MakeHuman is a parametric modelling tool that allows the creation of a high quality customized human model. Van Wyk's model was exported as a MakeHuman model and imported into Blender. He then created a skeleton for the model in Blender. The skeleton of the hand model was built according to the Humanoid Animation (H-Anim) standard [109].

H-Anim [47] is an open standard developed by the H-Anim working group [46]. It specifies a standard implementation for the skeleton structure of humanoid Avatars

parameterized using skeletal subspace deformation. The H-Anim specification defines a term "Level of Articulation" (LoA) that ranges from 0 to 3 and refers to standard quantities of joints (articulations) present in a skeleton structure. Higher LoA allow for more realistic motions, but require more bones and more complex skeleton structures. The skeleton of the 3D hand model developed by van Wyk can be seen in Figure 5.1 b).

The skeleton was then attached to the model – a process known as "rigging" – using the automatic rigging algorithm by Baran and Popović.

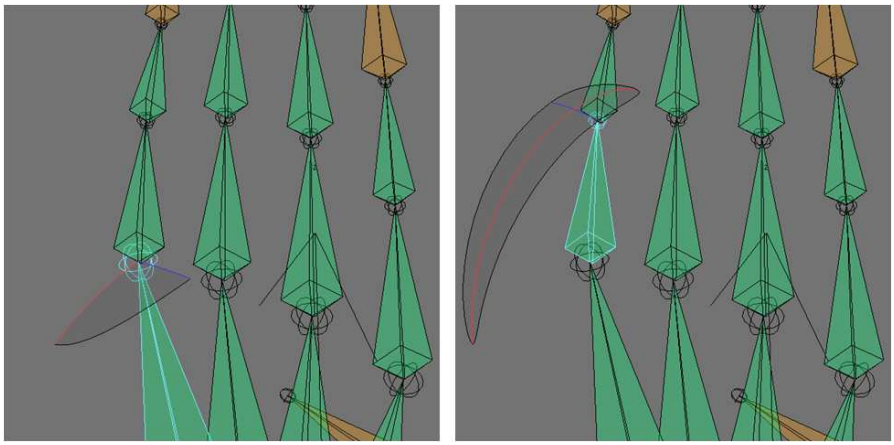
Van Wyk applied IK rotational constraints to distal phalanges – finger tips – and FK rotational constraints to all other bones. This can be seen in Figure 5.1 b).

When applying an IK rotational constraint to a distal bone, rotational constraints for the whole chain of bones that is linked to it can be set [109]. Figure 5.5 illustrates an example rotational constraint on a bone. Table 5.1 summarizes the rotational constraints placed on all the bones in the hand model. These constraints made the hand model capable of creating realistic human hand gestures and shapes.

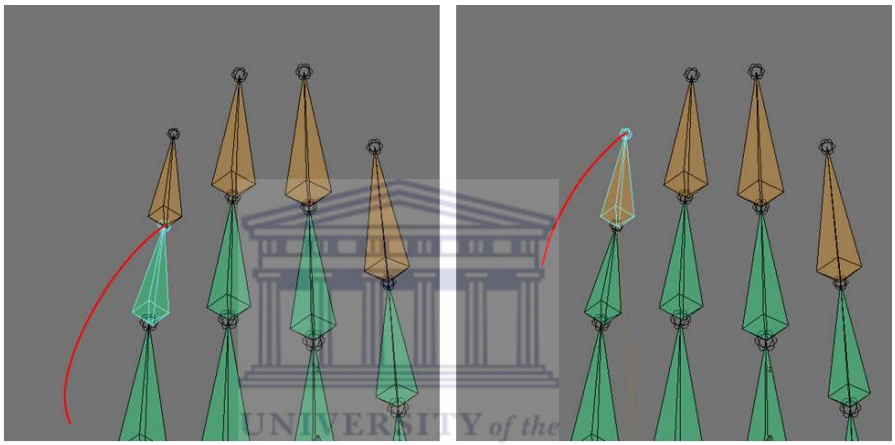
TABLE 5.1: constrains of rotation DOFs in degrees (∘) for bones in the hand [109].

Bone(joint)	**Limit X-axis**	**Limit Y-axis**	**Limit Z-axis**
thumb1	-180–20	-180–0	-90–120
thumb2	-85–30	0	-5–5
thumb3	-90–60	0	0
middle0 index0	0	0	0
ring0 pinky0	0	0	-5–5
middle1	-90–0	0	0
ring1	-115–25	-5–25	-15–15
pinky1	-115–25	-5–18	-25–15
pinky2 ring2 middle2 index2	-110–5	0	0
pinky3 ring3 middle3 index3	-90–15	0	0

By combining the Blender game engine, Python API and the 3D hand model, van Wyk developed a finger spelling alphabet animation controller in Blender. He created 26 stored keyframes that represent 26 letters in the finger spelling alphabet. Each



a) Constraints placed on bones pinky0 and pinky1, both represented as a range.



b) Constraints placed on bones pinky2 and pinky3, both represented as a red curve.

FIGURE 5.5: Constraints placed on the bones of the pinky.

keyframe was created by posing the virtual hand manually. It was then stored in the action-database or animation-database in Blender.

A keyboard sensor also forms part of the Blender game engine and can be used to capture keyboard input and pass it to a Python controller. An Action actuator is used by the Python controller to activate a pre-defined action within the action-database. When a letter of the keyboard is pressed, the virtual hand changes hand shape to the one that represents that finger spelling letter. The transition between different hand shape actions during the animation is smoothed using the kinematics functions embedded in the armature bones of the hand and the interpolation system embedded into the animation system.

Van Wyk [109] performed experiments on his model on a MacBook Pro with a 2.16 GHz Intel Core 2 Duo processor, 1 GB RAM and ATI Mobility Radeon X 1600 graphics card. The finger spelling animation system was tested by pressing various keys to activate the

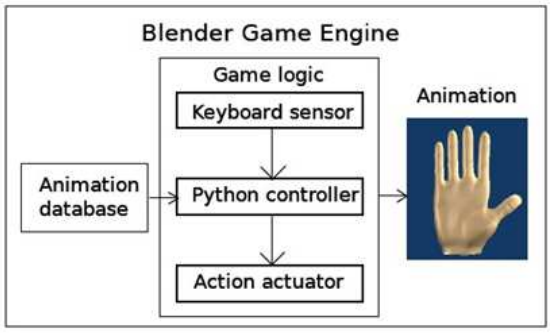


FIGURE 5.6: The finger spelling animation process [109].

corresponding finger spelling hand shapes. All animations were rendered at between 300–600 FPS.

5.4 Proposed System Design for Hand Shape Estimation

After tracking the hand in the video sequence and recognizing specific hand shapes, the recognised hand shape can be mapped from the 2D recognition to the 3D model. This makes it possible to obtain 3D co-ordinates of each of the joints. Also, interpolation in the animation system in Blender can be used to estimate smooth transitions in 3D space between two recognised hand shapes.

Figure 5.7 depicts the entire system with the estimation and recognition components highlighted.

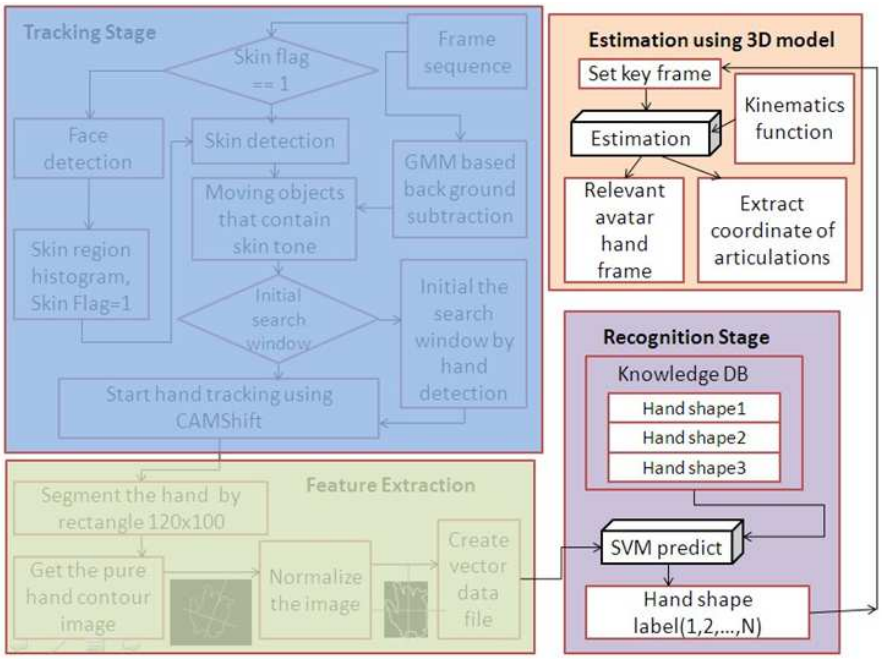


FIGURE 5.7: Process Flow Chart for hand shape estimation.

This section is broken into the following subsections: Subsection 5.4.1 discusses the method used to integrate the 3D hand model created by van Wyk with the hand shape recognition subsystem of this research; and Subsection 5.4.2 describes a strategy to overcome prediction errors by the SVM and demonstrates results of the proposed hand shape estimation method.

5.4.1 Linking the Recognition Subsystem with the Blender Game Engine for Hand Shape Estimation

The 3D hand model can be linked with the hand shape recognition subsystem of this research to perform hand shape estimation in real-time. It has been explained that van Wyk used the keyboard sensor to detect keyboard signals. The Blender game engine also has a sensor called the random sensor. This sensor can be configured to detect a signal from a pre-defined variable in the Python interface. A change in the value of this variable triggers the sensor to send its value to the actuator. In the current context, the value of the variable is continuously set to the hand shape detected by the recognition subsystem. When a signal is received, that is, the hand shape has changed, the result of the hand shape recognition subsystem is sent to the controller. Figure 5.8 illustrates the framework of hand shape estimation system.

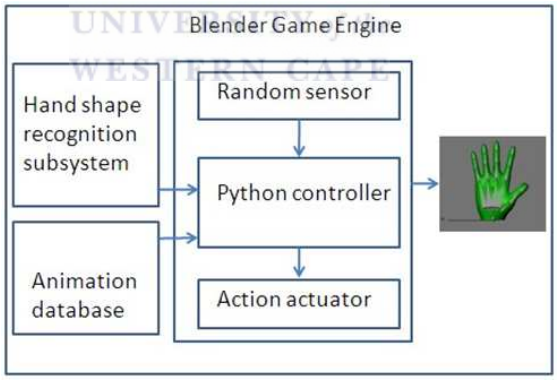


FIGURE 5.8: Hand shape estimation controller in the Blender game engine.

Two consecutive detected hand shapes are then set as keyframes in the animation system in the model. The model is transformed between the two hand shapes by means of an interpolation functions. This happens smoothly and correctly. Figure 5.9 is an example of the hand shape estimation that Blender carries out between two detected hand shapes $G(t)$ and $G(t+1)$. It should be noted that the animation system continues to correctly update the estimation process as long as the next detected hand shape from the recognition phase $G(t+1)$ is correct.

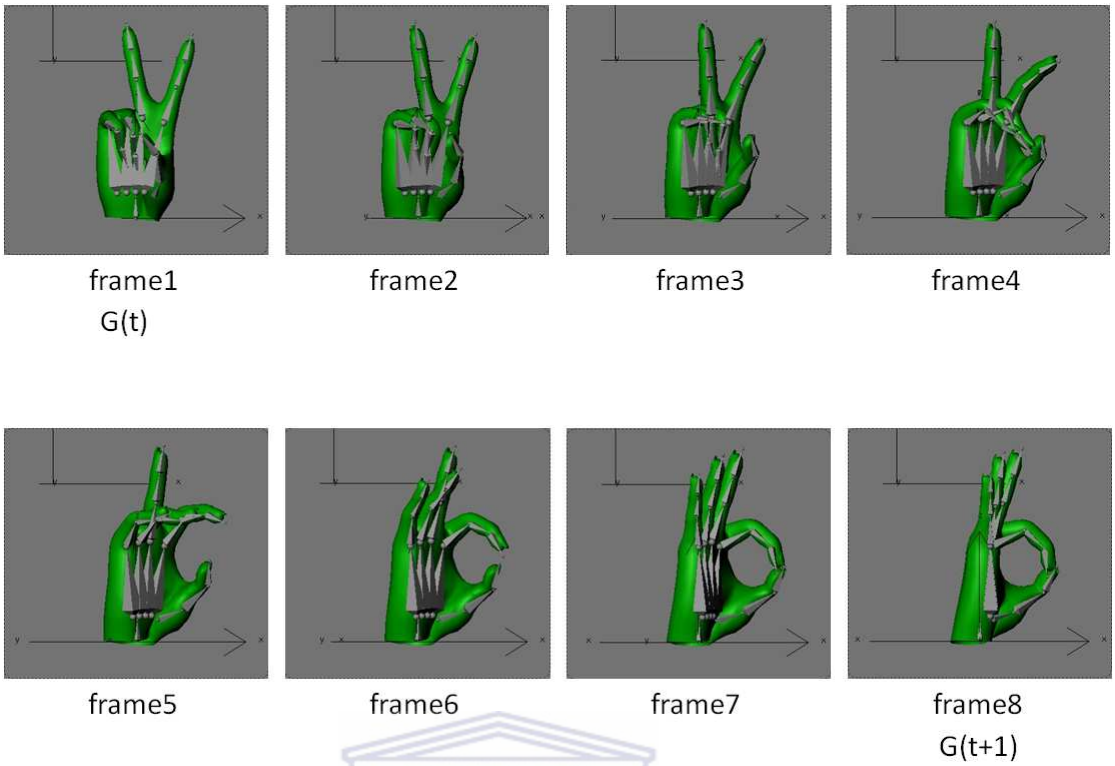


FIGURE 5.9: Smooth estimation of the transition between hand shapes $G(t)$ and $G(t+1)$ using Blender animation interpolation.

The framework is designed so that it relies on Blender's interpolation to fill in intermediate frames. This enables the recognition system to only focus on major hand shapes in SASL and ignore the wide variety of possible intermediate hand shapes. It, thus, avoids the need for large and infeasible training sets for the recognition system.

5.4.2 Modification of Hand Shape Estimation to Enhance Performance

The fact that the SVM is not 100% accurate implies that, in many cases, an incorrect classification is obtained. Consider the case where the user holds up an open hand. The SVM may be affected by various types of random noise and classify the input as various different, possibly erroneous, hand shapes. A modification was made to the estimation framework such that the hand shape will only be sent to the estimation process to be set as a key frame if three consecutive frames of the same classification are obtained. In all other cases, recognised frames are ignored. This significantly increases the accuracy of the estimation process.

Another modification was mentioned in Section 4.4.4 that described the recognition as only taking place on every third frame with the aim of skipping intermediate hand shapes between two known hand shapes. By combining these two modifications, the system is

able to operate at a rate of 18–25 frames per second. Figures 5.10 and 5.11 present two examples of the hand shape estimation process. It can be observed that a self-occlusion occurs in Figure 5.11 such that only the thumb and forefinger are visible. However, the system continues to operate correctly.

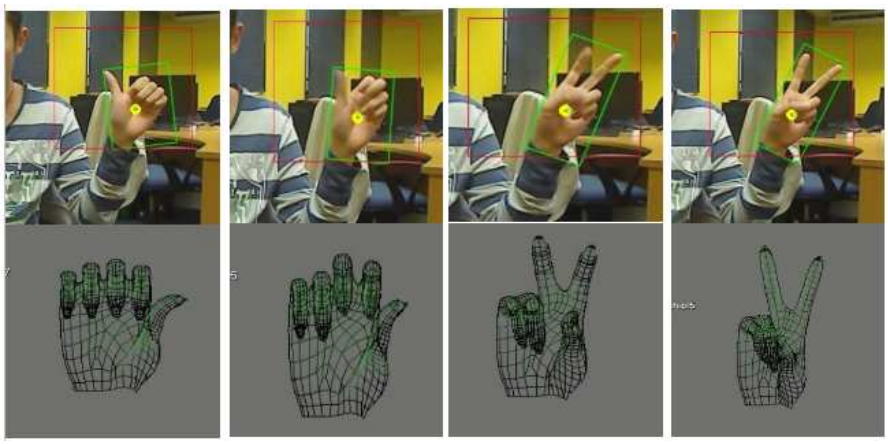


FIGURE 5.10: Hand shape estimation without self-occlusion [60].

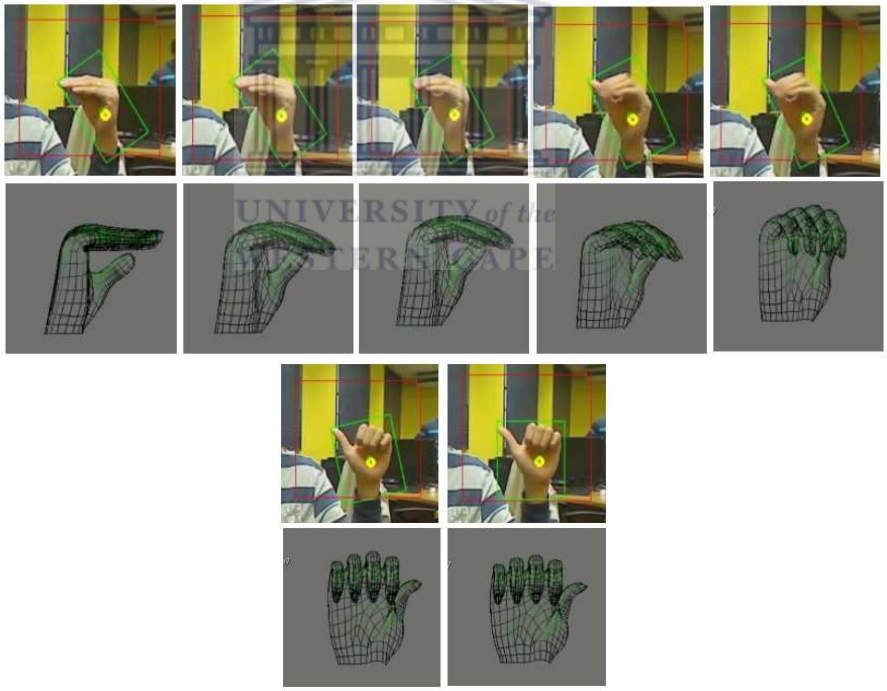


FIGURE 5.11: Hand shape estimation with self-occlusion [60].

5.5 Conclusion

This chapter provided an overview of the hand shape estimation method proposed by this research.

The chapter explained kinematics functions FK and IK which are used in 3D models to provide realism. The open-source modelling and animation tool Blender was then introduced. This tool was used extensively in the estimation system of this research. A justification for the use of this tool was presented.

Van Wyk's work was reviewed and his methodology was explained. His work provides a methodology for the creation of a high quality 3D hand model (and humanoid model) in Blender. His work also includes a method to fit the hand with IK and FK constraints with specific DOF-limits on bones to ensure that only plausible hand shapes and motions can be simulated.

It was explained that the Python API was used to link the hand model with the hand shape recognition sub-system of this research. Two recognised hand shapes are set as keyframes on the model and all intermediate hand shapes are then generated using interpolation functions in Blender. As long as the recognition of these two hand shapes is accurate, the estimation remains accurate as well. The constraints on the model ensure the prevention of implausible hand shapes during the transition. This architecture allows for a more efficient system with better accuracy than other related systems.

Modifications to this system were also explained. These modifications significantly reduce the effects of misclassification by the recognition system to ensure accurate estimation. The system proposed uses a strategy that does not require marked gloves to be worn and does not require a hand shape to be recognised at every frame.

Chapter 6

Experimental Results and Analysis

This chapter aims to assess the methodologies proposed in this research. Section 6.1 discusses the setup of the experiments carried out, such as, the equipment used and the environment in which testing was carried out. The sections that follow, Sections 6.2 to 6.5, detail the experimentation carried out to assess the three major parts of this research: hand tracking, hand shape recognition, and hand shape estimation. Section 6.6 concludes the chapter.

6.1 Experimental Setup

All experiments were carried out on a MacBook Pro with an Intel Core 2 Duo, 2.53 GHz CPU and 4 GB RAM, running the Ubuntu Linux 10.10 Operating System. A Logitech notebook Quick Cam web camera was used at a resolution of 640×480 pixels. Figure 6.1 depicts the locations at which the experiments were carried out as well as the equipment used. Two different backgrounds shown in Figures 6.1a – background A – and 6.1b – background B – were used in the experiments. Background A is a simpler and less noisy environment than background B. All testing was conducted on the left hand of test subjects.

The subjects were labelled A–F. For a clear understanding, Figure 6.2 illustrates the 6 subjects in Background A and Figure 6.3 illustrates the same subjects in Background B.

FIGURE 6.1: The two backgrounds used for testing.

FIGURE 6.2: The 6 subjects in background A.

FIGURE 6.3: The 6 subjects in background B.

6.2 Hand Tracking Testing

The hand tracking subsystem is an essential part of the system. All steps that follow tracking, such as feature extraction, hand shape recognition and hand shape estimation, are based on the ROI that is detected and provided by the hand tracking subsystem. Therefore, this subsystem is tested first.

The objective of the experiments in this section is to assess the rate of success of the hand tracking subsystem for different test subject and under different environments, that is, backgrounds A and B.

Subsection 6.2.1 describes the experimental procedure used to test the hand tracking subsystem and the criterion used to evaluate the testing results. Subsection 6.2.2 presents the results obtained and the analysis of those results.

6.2.1 Experimental Procedure

In this experiment, subjects B–F were used. Subject A was excluded from this experiment because the subject was not available at that time. Each subject was required to sit on the chair in front of the web-camera. The subjects were instructed to move their left hand. Such hand motion could include hand shape deformations, as instructed by the experimenter. Figure 6.4 illustrates one hand tracking test: a subject moves his hand across his face from one side of the frame to the other side. The tracking subsystem follows the motions of the left hand of the subject.

FIGURE 6.4: An example of the hand tracking test.

In order to test whether the hand tracking subsystem is robust, the testing was carried out under different conditions, similar to the testing strategy employed by Kolsch and

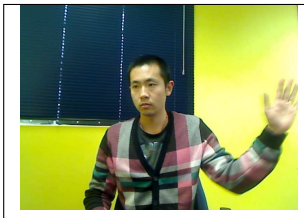
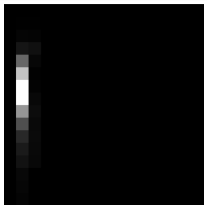
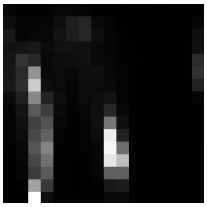
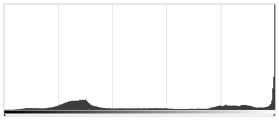
Turk [54]. Table 6.1 depicts the test subjects and the conditions under which the experiment took place. In each case, the table also presents whether or not the background contained pixels that were similar to the test subject's skin colour and whether or not the subject changed hand shapes during the test.

TABLE 6.1: Details of each subject.

Subject	Background	Skin background	Changing hand shapes
Subject B	Background B	yes	yes
Subject C	Background B	yes	no
Subject D	Background A	no	no
Subject E	Background A	yes	yes
Subject F	Background B	yes	no

The light conditions in the test location were found to change in the time taken to switch between test subjects. In order to clearly describe the testing environment for each subject, detailed information about the environments in each case is provided in Table A.1 in Appendix A. Table A.1 provides: the skin region and environment H-S histogram which jointly illustrate the similarity between skin colour of the subject and the background colour; and an intensity histogram which illustrates the lighting conditions at that time. An example of this table for Subject E is provided in Table 6.2.

TABLE 6.2: A detailed descriptions of the testing environment for Subject E.

Subject E in hand tracking testing	**Skin region H-S histogram**	**Environment H-S histogram**	**Environment intensity histogram**
			

For each subject, after the tracking window was initialized using hand detection, each frame captured was recorded to disk for a total of 10 seconds. This procedure was carried out three times for each subject in the same environment. Table 6.3 shows the total number of frames for each subject over the three testing iterations. It should be noted that the total number of frames for each subject varies. This is because the frame rate at which the tracking system ran varied. This is because the GMM background subtraction runs slower in a complex background than in a simple background. Moreover, illumination changes affect the frame rate as well.

TABLE 6.3: The total number of frames captured by the tracking subsystem over 30 seconds for each subject.

Subject	Total frames
Subject B	**963**
Subject C	**1098**
Subject D	**1146**
Subject E	**1065**
Subject F	**1002**

The criterion for a successfully tracked frame is defined as a frame that contains a tracking window of size 120×100 that encases the hand. This verification was done manually by the researcher.

6.2.2 Results and Analysis

Table 6.4 depicts the success rate of the hand tracking subsystem for each subject. A ratio of the tracked to the total number of frames can be used to evaluate the performance of the hand tracking subsystem for each subject in the corresponding testing conditions.

TABLE 6.4: Results of the hand tracking experiment.

Subject	Tracked frames / Total frames	Success rate (%)
Subject B	721 / 963	**74.9**
Subject C	783 / 1028	**76.2**
Subject D	1024 / 1146	**89.4**
Subject E	904 / 1065	**84.9**
Subject F	814 / 1002	**81.2**
Average		**81.3**

Referring to Table 6.4, the hand tracking subsystem achieves a success rate that ranges from 74.9% for Subject B to 89.4% for Subject D. The high performance achieved by Subjects D and E can be attributed to the fact that the backgrounds used in these experiments were less noisy than those applied to Subjects B, C, and F.

The experiments performed with Subjects C and F had similar conditions: they did not contain hand shape variations and were both conducted using background B. However, Subject F achieved a better performance than Subject C. This is attributed to the better lighting conditions for Subject F. This can be seen by comparing the light conditions of the two experiments using the corresponding intensity histograms in Table A.1 in Appendix A.

In analysing the frames after the experimentation, it was found that the hand motion was very fast in some cases. This fast motion caused blurring of the resulting extracted

frames. Figure 6.5 illustrates an example of a blurred hand. This blurring of the image changes the colour of the target. As a result, the tracking window can easily be grabbed by another region that contains a higher skin colour probability distribution than the blurred target, thus losing track. Therefore, it is not possible to avoid the loss of tracking due to very fast hand motions.

The ability to recover from this kind of error is very important. One solution to this problem is the use of a camera that captures frames at a faster rate. An example of such a camera is the spot action camera called HD Hero from GoPro [114]. The use of such a device is not available in this research. The solution adopted in this research arises from the nature of the camshift algorithm. When tracking is lost, the real target can still take back the tracking window by moving to the position where tracking had been lost. The target skin probability distribution of the real target is still higher than that of the region that grabbed the window and will take it back, as explained in Section 3.5. As such, the camshift algorithm effectively overcomes this problem.

FIGURE 6.5: A blurred hand caused by fast hand motion.

6.3 Training and Testing Data for Recognition and Estimation

In order to provide a clear understanding of the testing carried out, the training data and testing data are explained in this section.

Using the methods mentioned in Section 4.4.1, a dataset of real hand images was collected from three subjects labelled B, C and E and shown in Figure 6.6 in Background A. Referring to Figure 4.12, this dataset contains more than 1500 examples for each of the hand shapes H1 to H10. To produce a training set, 40 examples were randomly

selected from the dataset for each hand shape. Figure 6.6 illustrates the subjects and corresponding backgrounds used for training.

FIGURE 6.6: The three subjects and the corresponding backgrounds used for training.

For the testing phase, the six subjects A–F shown in Figure 6.2, were asked to perform each of the ten hand shapes H1–H10 once in succession. Each subject was asked to hold each hand shape for a few seconds (but no less than 9 frames) before transitioning to the next hand shape. This procedure took place once in Background A and once in Background B. This yielded 6 videos containing a sequence of hand shapes for each background, one per subject – a total of 12 videos. These videos were then used for hand shape recognition testing and hand shape estimation testing, as explained in subsequent sections.

6.4 Hand Shape Recognition Testing

This section describes the procedure and results of the experimentation of the hand shape recognition sub-system of this research.

Subsection 6.4.1 describes the approach used, and mentioned in Section 4.4.1, to select a suitable kernel function for the SVM used in the recognition sub-system. Section 6.4.2 describes the criterion that determines whether a recognised frame is correct or incorrect. This is used to assess the results of the experiments.

Subsequently, using this kernel function, two tests are described in Sections 6.4.3 and 6.4.4. The first test aims to obtain an indication of the robustness of the recognition sub-system for different test subjects. The second test aims to determine the success rate of the hand shape recognition in the two environments – background A and background B.

6.4.1 Determining a suitable kernel function and its corresponding parameters

This section presents a method to determine a suitable kernel for this research. In Chapter 4, the RBF kernel was identified as a suitable kernel for this system since it is recommended by Hsu *et al.* [43] and Keerthi and Lin [51]. It was also tested by Achmed [2] for the task of multi-classification using an SVM and achieved very good results.

Based on this, Hsu *et al.* propose a method of determining a suitable kernel [43]. They propose that an assessment of the RBF kernel should be carried out first. This can be done using the grid-search function mentioned in Chapter 4. An example is the grid-search function provided by LibSVM. If this kernel performs well, it will be adopted as the preferred kernel for the system. If it does not perform well, the performance of other kernels, such as the linear, polynomial and sigmoid kernels, will be evaluated.

This method is adopted by this research. The performance of the RBF kernel is assessed. If it performs well, the test concludes by adopting this kernel for this system. If it does not perform well, other kernels will be tested to determine the optimum kernel.

The results of the grid-search function on the training set mentioned in the previous subsection are depicted in Figure 6.7. The optimum parameters obtained were as follows: C was 0.03125, γ was 2. The accuracy rate of the kernel ranged from 98.8% before optimization to 99.5% after optimization. The small difference between the two accuracy rates indicates that the RBF kernel is very suitable and achieves a very high accuracy. Even without optimization, the system achieves a very high accuracy rate of 98.8%.

Therefore, it is unnecessary to test other kernels and the RBF kernel is adopted for use in this research.

In order to increase the accuracy rate of the system based on the optimization process, the SVM was re-trained using the optimized parameters of C and γ previously mentioned. The re-trained knowledge-based classifier was used for all subsequent tests.

6.4.2 Criterion for a Correctly Recognised Hand Shape

The following test involves comparisons between the inputs and the outputs of the hand shape recognition system. A criterion to analyse the input and output is presented in this section.

The recognition system uses a knowledge based classifier – SVMs – to predict labels (1-10) as outputs to input images. In order to easily compare inputs and outputs, 10

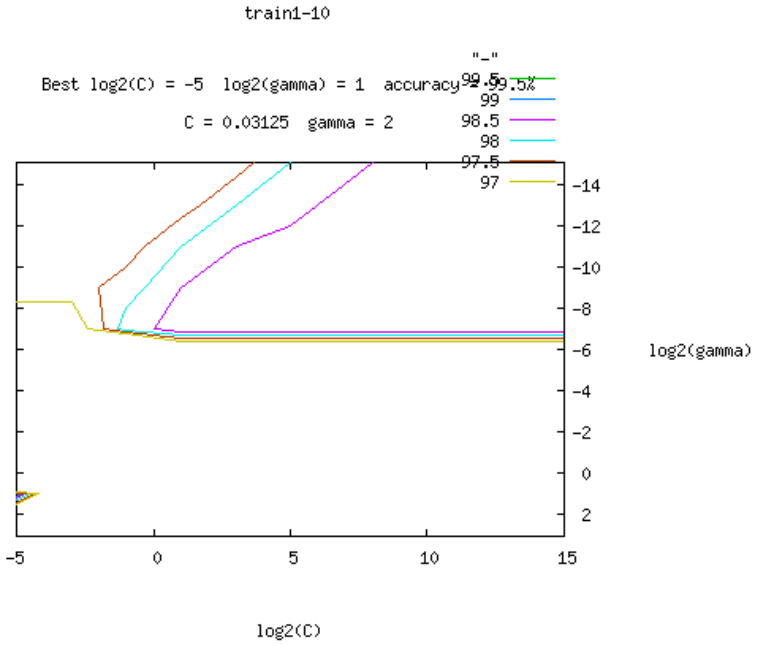


FIGURE 6.7: Results of the grid-search optimization function.

synthesized hand images are used to represent the labels of 10 hand shapes. During the testing, each real hand image corresponds to a synthesized hand image; the real hand image and the synthesized image are given the same index number as their name and saved in two folders. The criterion which defines a correctly recognised image is that the same hand shape can be found between a real hand image and a synthesized hand image.

The recognition sub-system outputs a label ranging from H1 to H10 given an input image containing an arbitrary hand shape that is to be recognised. During the test procedure mentioned previously, the recognition sub-system produced a label corresponding to each hand shape recognised in the video. Such classification is deemed correct if the hand shape that corresponds to the output label correctly matches the input hand shape. In order to simplify this process, the system was modified slightly for this test so that, for each hand shape recognised in the test video, it stored the original segmented hand image along with the image of the predicted label shown in figure 4.12 to disk with a common index. These two images were then manually compared. Correct classification was then defined as a match between the original hand image and the synthetic predicted image.

6.4.3 Test to Determine the Relationship between the Recognition Accuracy and Different Users

This section describes the experimentation carried out to assess the effect of varying the test subject on the recognition accuracy.

Different users of the system may perform the same hand shape with slight variations. For example, it was noted that the open hand shape differed significantly when performed by different users. Figure 6.8 depicts the contours of this hand shape as detected from different test subjects. Note that the images appear to be of the right hand because they are mirrored during capture. Even though the intended hand shape was the same, small variations led to very different recognition results.

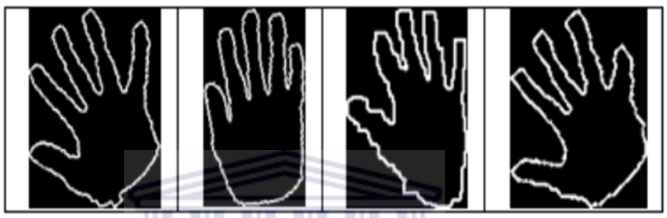


FIGURE 6.8: Contours of hand shape H1 – the open hand – as performed by different subjects.

One solution to this problem is to increase the number samples in the training phase. The following test assesses the degree to which the recognition accuracy is affected by variations in test subjects.

6.4.3.1 Experimental Procedure

The testing data collected on Background A was used in this experiment. 30 hand images for each of the 10 hand shapes for each of the 6 subjects were used – a total of 1800 images.

The criterion mentioned in Section 6.4.2 was used to determine correctly and incorrectly recognised images.

A Chi-Square statistical test was then applied to the results to produce a Chi-Square value and degree of freedom (df) value for each hand shape. The Chi-Square value and df value are then used to determine the corresponding P-Value as shown in Table B.1 of Appendix B.1. The P-Value is used to determine the statistical significance between recognition accuracy and subjects. The process used to determine the Chi-Square value and df is also provided in Appendix B.2 for completeness.

6.4.3.2 Results and Analysis

Table 6.5 summarizes the P-Values for each hand shape above the threshold of 0.05. It is observed that the P-value for every hand shape exceeds 0.1. Therefore varying the test subject has no significant effect on the recognition accuracy of the system. There is no relationship between the recognition accuracy and the test subject. The system is robust to variations in test subjects.

TABLE 6.5: Chi-Square test results for the test to determine the relationship between the recognition accuracy and test subject ($df = 5$).

Hand shape	Chi-Square	P-value
H1	3.8603	**0.5711**
H2	2.6732	**0.7502**
H3	1.6951	**0.8906**
H4	3.8741	**0.5682**
H5	1.6031	**0.9015**
H6	4.2234	**0.5173**
H7	7.3842	**0.1946**
H8	4.8612	**0.4332**
H9	2.4023	**0.7915**
H10	1.6033	**0.9019**

6.4.4 Hand Shape Recognition Testing on Backgrounds A and B

This section assesses the hand shape recognition accuracy on backgrounds A and B.

6.4.4.1 Experimental Procedure

The test data collected on Background A and B was used in this experiment. 30 hand images for each of the 10 hand shapes on each background for each of the 6 subjects were used – a total of 3600 images.

The criterion mentioned in Section 6.4.2 was used to determine correctly and incorrectly recognised images. The average recognition accuracy was then determined for each hand shape, subject and background.

6.4.4.2 Results and analysis

Tables 6.6 and 6.7 summarize the recognition accuracies obtained for each hand shape and subject for Backgrounds A and B respectively.

An analysis of the results under Background A was carried out. Referring to Table 6.6, high recognition accuracies are achieved by all test subjects for all 10 hand shapes on this background. The overall accuracy per subject ranged from 79.0% for Subject F to 87.0% for Subject C.

Analysing the accuracy results on a per-hand shape basis, it is observed that the accuracy ranged from 75.6% for hand shape H1 to 92.8% for hand shape H7. Referring to an earlier experiment carried out [59], the accuracy of hand shape H1 has increased from 55.0% to 75.6%. This was achieved by increasing the number of samples in the training phase from that experiment. Hand shapes H6 and H7 obtained the highest accuracies of 90.6% and 92.8%. Overall, an average accuracy of 83.3% was achieved on Background A.

An analysis of the results under Background B was also carried out. Referring to Table 6.7, good recognition accuracies are achieved by all subjects for all hand shapes. On a per-subject basis, the accuracy ranged from 71.3% for subjects C and F and 77.3% for subject E. The accuracy per hand shape ranged from 67.8% for hand shape H4 to 80.0% for hand shape H6 on a per-hand shape basis. The overall average accuracy on this background was 73.6%. This result is very encouraging given Background B was a very cluttered and noisy background.

The overall average accuracy of the system over both backgrounds was 78.5%. This result is very encouraging and indicates that the recognition sub-system is very accurate and robust to test subjects and background noise.

Analysing the causes of the difference between the accuracies achieved on Backgrounds A and B, it was found that two factors affected the result: varying light conditions and background noise in the form of regions that resembled skin in Background B. These factors make segmentation of a complete hand contour image in a frame very difficult. To overcome these factors, Shimada *et al.* [94] and Schreer *et al.* [88] either use a CCD camera on a static background or a simple background. The solution that can be adopted in this research is to include more examples during the training phase. This will aid the SVMs in predicting the correct hand shape given an unclear and incomplete hand contour image. Figure 6.9 illustrates examples of correctly recognised hand shapes from incomplete or unclear hand contours.

6.5 Hand Shape Estimation Testing

This section describes the procedure and results of the experimentation of the hand shape estimation sub-system of this research. Section 6.5.1 describes the criterion that

TABLE 6.6: Hand shape recognition accuracy on Background A.

Subject \ Hand shape	H1	H2	H3	H4	H5	H6	H7	H8	H9	H10	**Average**
Subject A	76.7	80.0	83.3	70.0	83.3	90.0	96.7	70.0	83.3	86.7	**82.0**
Subject B	80.0	83.3	86.7	83.3	86.7	96.7	96.7	76.7	76.7	80.0	**84.7**
Subject C	73.3	93.3	83.3	86.7	83.3	93.3	100.0	86.7	86.7	83.3	**87.0**
Subject D	83.3	80.0	86.7	73.3	86.7	83.3	86.7	86.7	83.3	83.3	**83.3**
Subject E	76.7	83.3	86.7	76.7	80.0	93.3	90.0	86.7	90.0	76.7	**84.0**
Subject F	63.3	83.3	76.7	83.3	76.7	86.7	86.7	76.7	80.0	76.7	**79.0**
Average	**75.6**	**83.9**	**83.9**	**78.9**	**82.8**	**90.6**	**92.8**	**80.6**	**83.3**	**81.1**	*83.3*

TABLE 6.7: Hand shape recognition accuracy on Background B.

Subject \ Hand shape	H1	H2	H3	H4	H5	H6	H7	H8	H9	H10	**Average**
Subject A	66.7	70.0	73.3	70.0	56.7	83.3	80.0	70.0	70.0	76.7	**71.7**
Subject B	76.7	80.0	66.7	70.0	80.0	76.7	80.0	76.7	70.0	76.7	**75.3**
Subject C	73.3	63.3	73.3	66.7	83.3	80.0	70.0	66.7	63.3	73.3	**71.3**
Subject D	63.3	80.0	73.3	63.3	70.0	83.3	86.7	83.3	70.0	73.3	**74.7**
Subject E	73.3	76.7	80.0	66.7	76.7	86.7	83.3	70.0	73.3	86.7	**77.3**
Subject F	63.3	70.0	76.7	70.0	63.3	70.0	76.7	70.0	76.7	76.7	**71.3**
Average	**69.4**	**73.3**	**73.9**	**67.8**	**71.7**	**80.0**	**79.5**	**72.8**	**70.6**	**77.2**	*73.6*

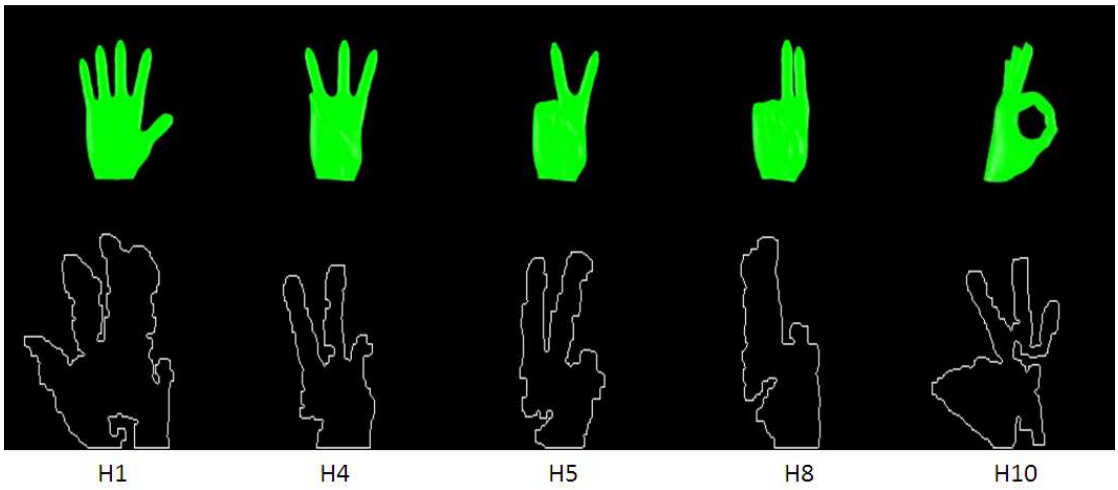


FIGURE 6.9: Examples of correctly recognised hand shapes from unclear and incomplete hand contour images on Background B.

determines whether an estimated frame is correct or incorrect. This criterion was used to analyse the estimated results.

Subsequently, two tests are described in Sections 6.5.2 and 6.5.3. The first test aims to determine the difference between the estimation accuracy as computed using two methods on the simple background A. The second test determines the estimation accuracy on the two backgrounds A and B, similar to the test carried out for hand shape recognition.

6.5.1 Criterion for a Correctly Estimated Hand Shape Transition

One technique that can be used to determine whether the transition between two hand shapes has been estimated correctly involves the use of a data glove. The joint locations and other 3D parameters of each estimated intermediate hand shape would be compared with the actual value obtained from the data glove. This would indicate whether or not the estimation process was done correctly [95]. However, this device is very costly and was not available for this research.

A second technique involves the use of manual observation to determine whether or not an intermediate hand shape has been correctly matched with the corresponding estimated image. This approach has been used in a number of other projects that involve hand shape estimation [8] and body motion estimation [2, 7]. This is the technique used in this research.

It should be noted that the number of frames of the actual transitions were different to the number of frames of the estimated transitions. This is because the speed at which subjects performed hand shape transitions differed to that of the estimation system.

The estimation system does not extract speed or timing information of transitions from the input. It only uses recognised hand shapes as key frames in the transition animation, and these are placed at a constant time apart. Additionally, this number also varied between test subjects depending on the speed at which each individual moved their hand. Therefore, the second technique involves comparing the frames of the actual transition with those of the estimated transition on a holistic basis. Frames of the estimated transition that are perceived to be correctly moving towards the next hand shape are marked as correct. Frames that are perceived to be moving towards a different and incorrect hand shape are marked as incorrect.

Figure 6.10 illustrates an estimated transition in which the actual hand shapes are H7 and H8. The estimated hand shape first moves correctly towards H8. These frames are marked as being correctly estimated. However, at frame 954 it begins to move away from H8 and towards an incorrect hand shape because the pinky and ring fingers begin to rise in these frames. These frames are marked as being incorrectly estimated. Thereafter, at frame 956, the hand shape is again seen to revert towards the correct hand shape – H8 – as the pinky and ring fingers fall. These frames are once again marked as being correctly estimated.

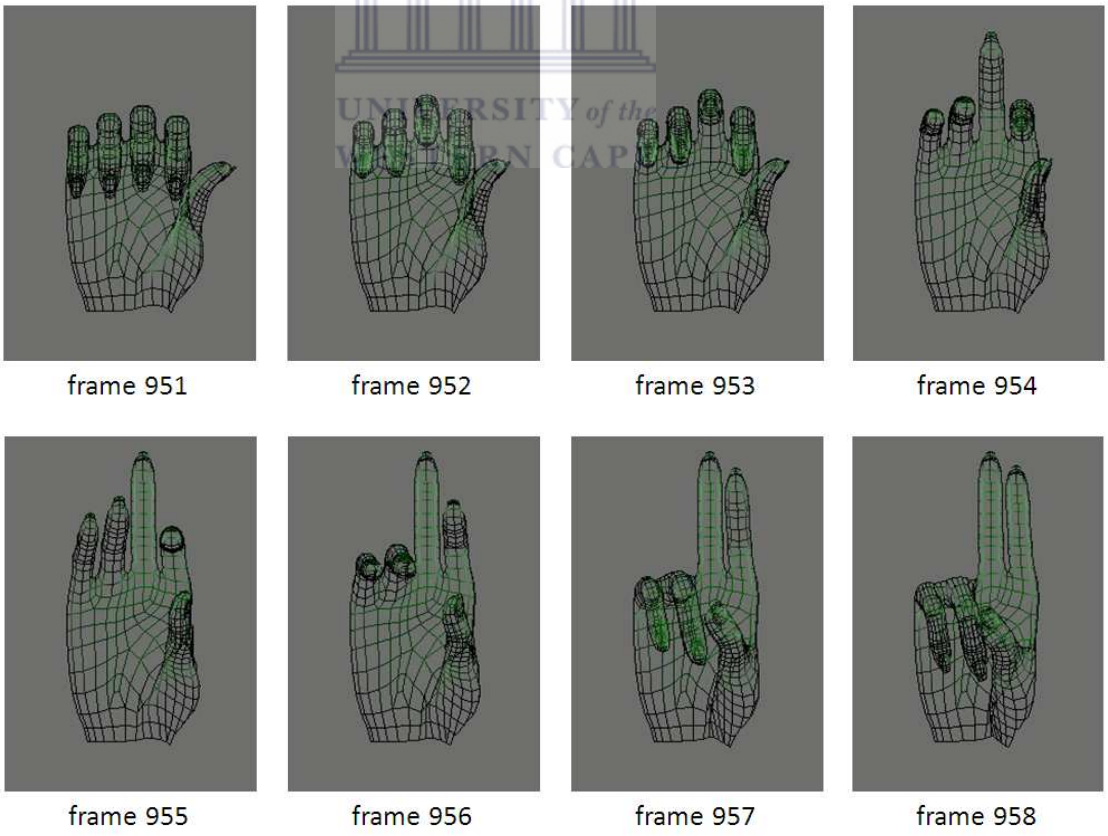


FIGURE 6.10: An estimated transition from H7 to H8.

6.5.2 Test to Compare the Estimation Accuracy of Two Methods of Estimation

This test attempted to determine whether there is a significant difference in the estimation accuracy of two different methods of hand shape estimation.

When a subject performs a hand shape transition on a simple background, there are two reasons that lead to an incorrect hand shape estimation result. The first reason is the presence and incorrect classification of intermediate hand shapes which are not recognised by the recognition system. The second reason is classification errors by the SVM, even in cases where a correct hand shape input is provided. In the Section 4.4.4 of Chapter 4 and Section 5.4.2 of Chapter 5, two methods used to overcome these problems were mentioned. These two methods, henceforth referred to as M1 and M2, are as follows:

M1: Recognise only every third frame and use the result as a keyframe for hand shape estimation.

M2: Recognise only every third frame and only use the result as a keyframe for hand shape estimation if three consecutively recognised frames yield the same hand shape.

The following subsections describe the comparative test carried out on these methods and the subsequent results and analysis of the results.

6.5.2.1 Experimental Procedure

9 hand shape transitions – T1 to T9 – were defined for this experiment. These transitions are detailed in Table 6.8. The table also shows the start and end hand shapes that characterize each transition. It was not feasible to test all possible transitions, such as H1 to H3–H10, H2 to H4–H10 etc, due to time constraints. However, transitions T1–T9 make a fair representation of all possible transitions, ranging from simple transitions such as T1 to complex transitions such as T8.

The testing data collected on Background A was used in this experiment. It has been explained that the number of frames between any two transitions performed by test subjects differed. This is because test subjects performed transitions at varying speeds. A complete list of the number of frames, in the test data, for each transition performed by each subject is shown in Table 6.9.

TABLE 6.8: Pre-defined hand shape transitions used in estimation testing.

Transition	Start Hand Shape	End Hand Shape
T1	H1	H2
T2	H2	H3
T3	H3	H4
T4	H4	H5
T5	H5	H6
T6	H6	H7
T7	H7	H8
T8	H8	H9
T9	H9	H10

TABLE 6.9: The number of frames for each transition as performed by each test subject.

Transition / Subject	T1	T2	T3	T4	T5	T6	T7	T8	T9	**Total**
Subject A	29	18	23	29	15	20	25	26	19	**204**
Subject B	17	13	16	10	12	15	16	20	12	**131**
Subject C	20	15	17	22	29	25	23	27	28	**206**
Subject D	27	26	20	17	25	19	29	26	30	**219**
Subject E	16	21	15	12	28	14	22	25	11	**164**
Subject F	21	11	19	23	26	17	15	20	18	**170**

The criterion mentioned in Section 6.5.1 was used to determine correctly and incorrectly estimated frames in each transition.

McNemar's test is a paired version of the Chi-Square test in which *df* is always 1. MacNemar's test was applied to the the number of correctly and incorrectly estimated frames obtained from the two methods.

Tables B.4 to B.12 in Appendix B.3 illustrate the Chi-Square and P-Value for each transition from T1 to T9 across all subjects.

6.5.2.2 Result and Analysis

Tables 6.10 and 6.11 summarize the results of the estimation accuracy achieved for each method. An analysis of the results for method M1 was carried out. Referring to Table 6.10, high estimation accuracies are achieved by all test subjects for all 9 transitions for this method. The overall accuracy per subject ranged from 64.7% for Subject A to 77.0% for Subject D.

Analysing the accuracy results on a per-transition basis, it is observed that the accuracy ranged from 64.4% for transition T3 to 94.5% for transition T6. Overall, an average accuracy of 72.5% was achieved using method M1.

An analysis of the results under method M2 was also carried out. Referring to Table 6.11, improved estimation accuracies are achieved by all subjects for all transitions. On a per-subject basis, the accuracy ranged from 85.4% for subjects A and 92.7% for subject E. The accuracy per transition ranged from 82.3% for transition T3 and 98.5% for transition T6. The overall average accuracy using this method was 88.5%. This result is very encouraging for both methods.

Figure 6.11 is a graphical comparison between the estimation accuracy of the two methods for each transition. It is observed that method M2 achieves much higher accuracies for all transitions than method M1.

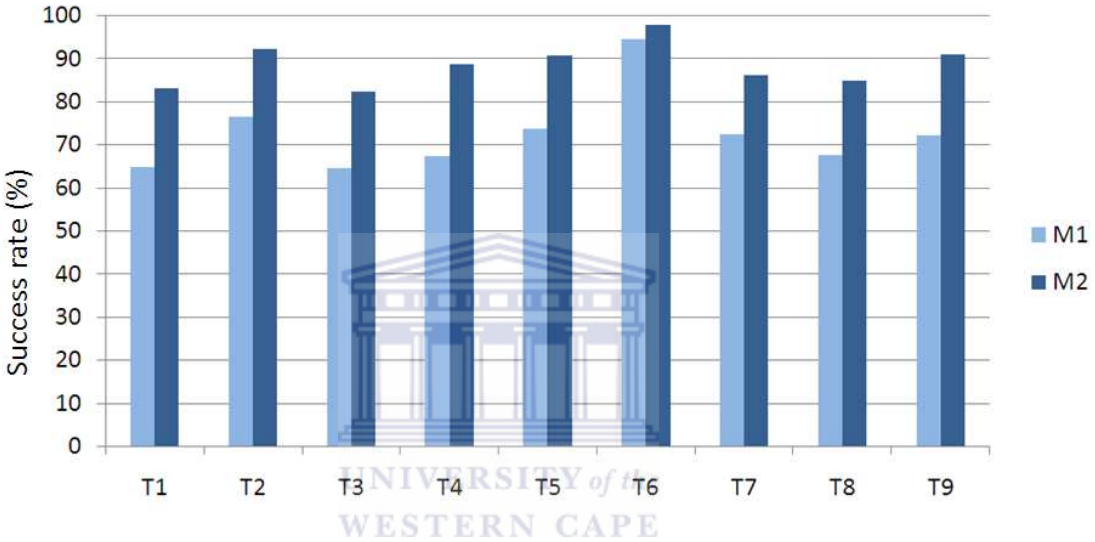


FIGURE 6.11: Estimation accuracy comparison for methods M1 and M2.

Table 6.12 summarizes the results of performing McNemars test on this data. It is observed from the table that the P-Value of transition T6 was greater than the acceptable significance level of 0.05. Therefore, this is the only transition in which there is no significant difference in estimation accuracy between the two methods M1 and M2. Therefore, the estimation accuracy achieved for transition T6 will be the same using both methods.

The other 8 transitions have P-Values that are less than the acceptable significance level. Transitions T1 and T4 obtain P-Values less than 0.0001 which indicates that there is an extremely significant difference between methods M1 and M2 for these transitions. The remaining 6 transitions obtain P-values that indicate that there is a very significant difference between two methods for these transitions.

Therefore, it is concluded that method M2 is more suitable than M1 and was adopted as the estimation method for the subsequent test.

TABLE 6.10: Success rate of hand shape estimation using method M1 on Background A.

Subject \ Transition	T1	T2	T3	T4	T5	T6	T7	T8	T9	**Average**
Subject A	51.7	72.2	56.5	69.0	66.7	85.0	60.0	57.7	63.2	**64.7**
Subject B	76.5	61.5	62.5	70.0	75.0	100.0	75.0	75.0	75.0	**74.5**
Subject C	65.0	73.3	58.8	72.7	69.0	100.0	78.3	66.7	75.0	**73.2**
Subject D	63.2	90.5	72.7	72.7	83.3	81.8	80.8	62.1	86.4	**77.0**
Subject E	66.7	76.5	72.7	60.0	76.9	100.0	75.0	76.9	66.7	**74.6**
Subject F	64.7	84.6	63.2	60.0	70.6	100.0	64.7	66.7	66.7	**71.2**
Average	**64.6**	**76.4**	**64.4**	**67.3**	**73.6**	**94.5**	**72.3**	**67.5**	**72.1**	*72.5*

TABLE 6.11: Success rate of hand shape estimation using method M2 on Background A.

Subject \ Transition	T1	T2	T3	T4	T5	T6	T7	T8	T9	**Average**
Subject A	89.7	88.9	78.3	79.3	73.3	100.0	84.0	80.8	94.7	**85.4**
Subject B	88.2	92.3	87.5	90.0	91.7	100.0	87.5	85.0	91.7	**90.4**
Subject C	85.0	86.7	76.5	95.5	86.2	100.0	78.3	88.9	89.3	**87.4**
Subject D	84.2	90.5	81.8	88.9	100.0	90.9	92.3	72.4	86.4	**87.5**
Subject E	80.0	94.1	90.9	93.3	92.3	100.0	91.7	92.3	100.0	**92.7**
Subject F	70.6	100.0	78.6	84.0	100.0	100.0	82.4	88.9	83.3	**87.5**
Average	**83.0**	**92.1**	**82.3**	**88.5**	**90.6**	**98.5**	**86.0**	**84.7**	**90.9**	*88.5*

TABLE 6.12: Results of McNemars test on the hand shape estimation accuracy results of methods M1 and M2.

M1	M2	Chi-Square	P-Value
T1	T1	18.893	**<0.0001**
T2	T2	9.389	**0.0022**
T3	T3	13.136	**0.0003**
T4	T4	16.531	**<0.0001**
T5	T5	11.130	**0.0008**
T6	T6	3.200	**0.0736**
T7	T7	14.063	**0.0002**
T8	T8	14.667	**0.0001**
T9	T9	14.450	**0.0001**

6.5.3 Hand Shape Estimation Testing on Backgrounds A and B

This section assesses the hand shape estimation accuracy of the system on backgrounds A and B. As such, the estimation accuracy on these backgrounds can be determined.

6.5.3.1 Experimental Procedure

The testing data collected on Background A and B was used in this experiment. The data used was the frames in these videos for transitions T1 to T9 explained previously. The criterion mentioned in Section 6.5.1 was used to determine correctly and incorrectly estimated frames in each transition. For the case of Background A, the required test had already been conducted in the experiment to determine the difference between methods M1 and M2. As stated previously, method M2 was adopted as the primary estimation method. Therefore the results summarized in Table 6.11 were used for this experiment. The experiment was conducted for Background B.

6.5.3.2 Result and analysis

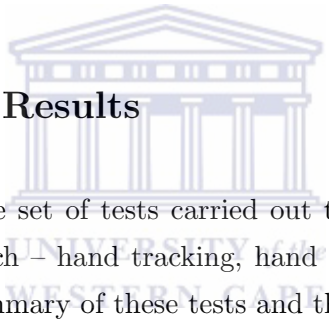
Table 6.13 summarizes the estimation accuracies obtained for each hand shape transition and on Background B. The same results for Background A have been provided in Table 6.11.

An analysis of the results under Background A was carried out. Referring to Table 6.11, high estimation accuracies are achieved by all test subjects for all 9 hand shape transitions on this background. The overall accuracy per subject ranged from 85.4% for Subject A to 92.7% for Subject E.

Analysing the accuracy results on a per-hand shape basis, it is observed that the accuracy ranged from 82.3% for transition T3 to 98.5% for transition T6. Transition T6 obtained the highest accuracy of 98.5%. Overall, an average accuracy of 88.5% was achieved on Background A.

An analysis of the results under Background B was also carried out. Referring to Table 6.13, encouraging estimation accuracies are achieved by all subjects for all hand shape transitions. On a per-subject basis, the accuracy ranged from 70.2% for subject D and 78.1% for subject E. The accuracy per hand shape transition ranged from 69.6% for transition T1 and 94.9% for transition T6 on a per-hand shape basis. The overall average accuracy on this background was 75.4%.

Given Background B was a very noisy background, this result is very encouraging. The overall average accuracy of the system over both backgrounds is 82.0%. This result is very encouraging and indicates that the estimation sub-system is accurate and robust to background noise.



6.6 Summary of Results

This chapter described the set of tests carried out to assess the performance of the 3 sub-systems of this research – hand tracking, hand shape recognition and hand shape estimation. Below is a summary of these tests and the results obtained:

- The hand tracking subsystem was tested on different test subjects performing varying hand shapes on different backgrounds. It was found that the system can track the hands very accurately. Overall, an impressive tracking accuracy of 81.3% was achieved across all subjects and backgrounds.
- A suitable kernel was then selected for the hand shape recognition sub-system. This was done using the grid-search function. It was found that the RBF function was suitable with a final accuracy of 99.5%. Based on this kernel, the system was then tested in two ways. The first test aimed to determine the relationship between the recognition accuracy and different users. It was found that varying the users had no significant effect on the recognition accuracy. The second test aimed to determine the recognition accuracy on two backgrounds: a simple background and a complex background. It was found that both backgrounds achieve a very high accuracy. As expected, the accuracy on the simple background – 83.3% – was higher than that on the complex background – 73.6%. An overall accuracy of 78.5% was achieved across both backgrounds.

TABLE 6.13: Hand shape estimation accuracy on Background B.

Subject \ Transition	T1	T2	T3	T4	T5	T6	T7	T8	T9	**Average**
Subject A	74.1	75.0	75.9	73.7	66.7	100.0	71.0	74.1	75.9	**76.2**
Subject B	65.5	85.0	70.8	70.0	86.2	93.3	69.2	72.4	83.3	**77.3**
Subject C	74.1	82.4	66.7	73.7	72.7	100.0	74.1	66.7	77.3	**76.4**
Subject D	65.0	60.0	76.5	63.6	75.9	76.0	69.6	66.7	78.6	**70.2**
Subject E	62.5	90.5	73.3	66.7	78.6	100.0	77.3	72.0	81.8	**78.1**
Subject F	76.2	63.6	73.7	73.9	73.1	100.0	66.7	75.0	66.7	**74.3**
Average	**69.6**	**76.1**	**72.8**	**70.3**	**75.5**	**94.9**	**71.3**	**71.1**	**77.3**	*75.4*

- The hand shape estimation sub-system was also tested in two ways. The first test aimed to determine the effect of two estimation methods on the estimation accuracy. The first method involved using the recognition result of only every third frame. The second method involved using the recognition result only if three consecutively recognised frames were consistent. It was found that, apart from the transition between hand shape H6 and H7, there was a significant difference between the two methods, with the second method performing better – 88.5% – than the first – 72.5%. Based on this, the second method was adopted as a basis for the subsequent experiments. The second test aimed to determine the estimation accuracy on two backgrounds: a simple and a complex background. It was found that a high overall estimation accuracy of 82.0% was achieved over both backgrounds. Again, as expected, the accuracy was much higher on the simple background – 88.5% – than the complex background – 75.4%.

6.7 Conclusion

In conclusion, 81.3% of frames are tracked correctly. The potential tracking losses in the remaining 18.7% of frames are compensated for effectively using method M2 – recognizing only every third frame and only using the result as a keyframe for hand shape estimation if three consecutively recognised frames yield the same hand shape. Using this method, it is possible to obtain 100.0% recognition and estimation accuracy as was the case in 9 estimation tests on Background A – a simple background (Table 6.11).

Therefore, the RBF kernel is the optimum kernel for this system. Method M2 is better than method M1. Using this method, the system is user-independent and tolerant to background noise.

It is concluded that the system performs at a very high level of accuracy and is robust to background noise and variations in test subjects.

Chapter 7

Conclusion

This research has contributed to 3 fields of hand based human computer interaction. These are: hand tracking, hand shape recognition and hand shape estimation.

A state-of-the-art hand tracking system was developed in line with the first objective of this research mentioned in Chapter 1. This system uses a multi-cue strategy to effectively reduce the effects of background noise. A skin cue image is produced by face detection and 2D histogram back projection and a motion cue image is produced by using GMM background subtraction. The combination of the skin and motion cue images effectively facilitates the detection of only the moving objects that contain skin. The use of face detection makes this multi-cue image adaptable to users of any skin tone. Based on the multi-cue image, hierarchical chamfer matching detects the hand location in a frame only once at system start. The speed of this algorithm is increased by choosing a template that has a size similar to that of the face. Finally, the camshift tracking algorithm continuously tracks the hand motion. It tracks the target by finding the highest probability distribution using a single hypothesis search strategy. It achieves a very fast implementation speed and is able to track regardless of any deformations of the hand during motion. The system is also able to recover from tracking losses caused by very rapid hand motions. The system achieves a very high accuracy.

A highly accurate novel hand shape recognition system was also developed as a response to the second objective of this research mentioned in Chapter 1. Using the image of the tracked hand, the system uses an advanced normalization strategy that ensures scale and orientation invariance of the hand. The system uses a support vector machine with the radial-basis function kernel to recognise hand shapes. The hand shape recognition system is highly robust and accurate and is able to recognise hand shapes against both simple and complex backgrounds. It was also shown that the system is invariant to skin

tone and different users. This has not been achieved in any known research studies to date.

Finally, a novel hand shape estimation system was developed in line with the third objective of this research mentioned in Chapter 1. The system relies on the hand shape recognition system and Blender to estimate intermediate hand shapes in the transition between recognised hand shapes. Smooth transitions are achieved by interpolating between recognised hand shapes using the embedded kinematics function in the 3D hand model developed in Blender, which is used in this research. This enables the recognition system to only focus on major hand shapes in SASL and ignore the wide variety of possible intermediate hand shapes. This avoids the need for large and infeasible training sets for the recognition system but is still able to produce them by means of estimation. This strategy is also a novel approach proposed and used in this research. The hand shape estimation system was also shown to achieve a very high and encouraging accuracy.

The entire system is highly accurate and robust and has satisfied the fourth objective of this research mentioned in Chapter 1. In response to the overall research question mentioned in Chapter 1, it is concluded that it is possible to accurately recognise and estimate South African Sign Language hand shapes for a person of any skin tone from a monocular video feed on an arbitrary background.

This research has contributed greatly to the SASL project and the field of hand based human computer interaction.

7.1 Directions for Future Work

While this research was able to produce an accurate and effective hand shape recognition system for SASL, 10 hand shapes were used as a proof of concept. In future, the system can be trained on all major SASL hand shapes. The system can also be integrated into the current machine translation framework prototype developed at the group, called iSign [38].

It was mentioned in a previous chapter that it was not possible to obtain a data glove during the course of this research. The data glove can be used to compare the data captured from actual users with the results of the hand shape estimation system. If found necessary, it is possible to improve on the animations of the estimated transitions by including more constraint functions determined from the data glove data.

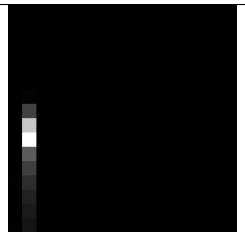
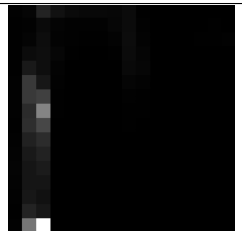
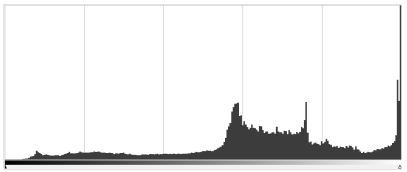
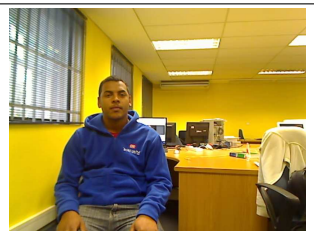
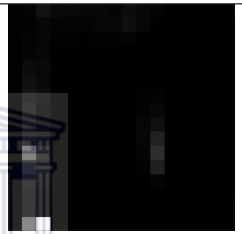
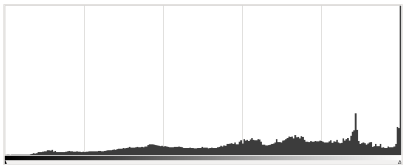
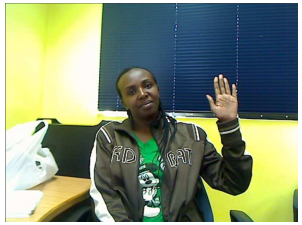
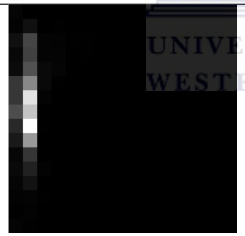
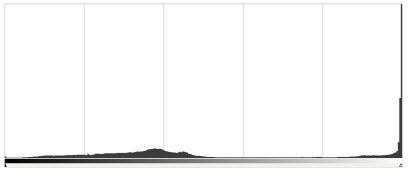
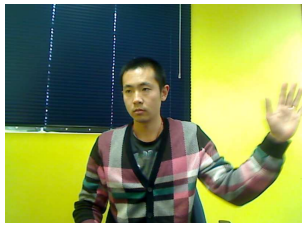
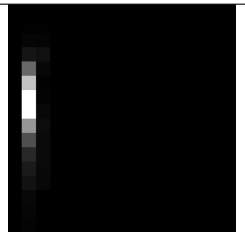
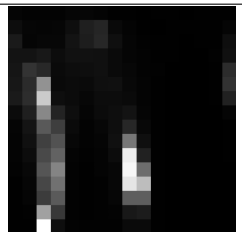
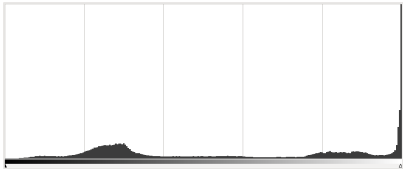
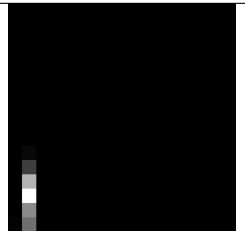
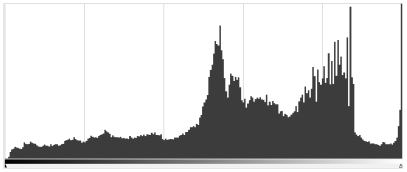
7.2 Concluding Remarks

The experience throughout the course of this research has been of great educational value to the researcher. It is hoped that this research will serve as a valuable asset in the advancement of sign language translation at the SASL group and be of value to the field of human computer interaction.

Appendix A

The Environment Used for Hand Tracking Testing

TABLE A.1: Description of the environment for each subject.

Subject	**Environment image**	**Skin region H-S histogram**	**Environment H-S histogram**	**Environment intensity histogram**
Subject B				
Subject C				
Subject D				
Subject E				
Subject F				

Appendix B

Chi-Square and McNemar's Test

B.1 Chi-Square Distribution

TABLE B.1: Chi-Square Distribution.

Degree of Freedom (*df*)	**Probability (P-Value)**										
	0.95	**0.90**	**0.80**	**0.70**	**0.50**	**0.30**	**0.20**	**0.10**	**0.05**	**0.01**	**0.001**
	Chi-Square										
1	0.004	0.02	0.06	0.15	0.46	1.07	1.64	2.71	3.84	6.64	10.83
2	0.10	0.21	0.45	0.71	1.39	2.41	3.22	4.60	5.99	9.21	13.82
3	0.35	0.58	1.01	1.42	2.37	3.66	4.64	6.25	7.82	11.34	16.27
4	0.71	1.06	1.65	2.20	3.36	4.88	5.99	7.78	9.49	13.28	18.47
5	1.14	1.61	2.34	3.00	4.35	6.06	7.29	9.24	11.07	15.09	20.52
6	1.63	2.20	3.07	3.83	5.35	7.23	8.56	10.64	12.59	16.81	22.46
7	2.17	2.83	3.82	4.67	6.35	8.38	9.80	12.02	14.07	18.48	24.32
8	2.73	3.49	4.59	5.53	7.34	9.52	11.03	13.36	15.51	20.09	26.12
9	3.32	4.17	5.38	6.39	8.34	10.66	12.24	14.68	16.92	21.67	27.88
10	3.94	4.86	6.18	7.27	9.34	11.78	13.44	15.99	18.31	23.21	29.59
	Nonsignificant								**Significant**		

B.2 Chi-Square Test

An example is provided on the method that was used to obtain the Chi-Square and P-Values for the experiment to determine the relationship between the recognition accuracy and different users. Table B.2 depicts the parameters that are required for the test and the method used to compute them.

TABLE B.2: The table used in the Chi-Square test illustrating the various parameters and their computation.

	Correct frames	Incorrect frames	Total
Subject A	a	b	$a+b=sum_{r1}$
Subject B	c	d	$c+d=sum_{r2}$
Subject C	e	f	$e+f=sum_{r3}$
Subject D	g	h	$g+h=sum_{r4}$
Subject E	i	j	$i+j=sum_{r5}$
Subject D	k	l	$k+l=sum_{r6}$
Total	$a+c+e+g+i+k=sun_{c1}$	$b+d+f+h+j+l=sum_{c2}$	$a+b+c+d+e+f+g+h+i+j+k+l=N$

Let i be the column number and j be the row number. df of table B.2 can be obtained using the following equation:

$$df=(i-1)(j-1)=(6-1)(2-1)=5 \tag{B.1}$$

Let $S=\{x_v|a,b,\cdots,l\}$ represent each date from a to l, table B.3 depicts the method used to obtain $O(x)$ and $E(x)$:

TABLE B.3: Observed and expected values.

Observed valve $=O(x)$	Expected value $=E(x)$
$O(a)=a$	$E(a)=(sum_{r1})(sum_{c1})/N$
$O(b)=b$	$E(b)=(sum_{r1})(sum_{c2})/N$
$O(c)=c$	$E(c)=(sum_{r2})(sum_{c1})/N$
$O(d)=d$	$E(d)=(sum_{r2})(sum_{c2})/N$
$O(e)=e$	$E(e)=(sum_{r3})(sum_{c1})/N$
$O(f)=f$	$E(f)=(sum_{r3})(sum_{c2})/N$
$O(g)=g$	$E(g)=(sum_{r4})(sum_{c1})/N$
$O(h)=h$	$E(h)=(sum_{r4})(sum_{c2})/N$
$O(i)=i$	$E(i)=(sum_{r5})(sum_{c1})/N$
$O(j)=j$	$E(j)=(sum_{r5})(sum_{c2})/N$
$O(k)=k$	$E(k)=(sum_{r6})(sum_{c1})/N$
$O(l)=l$	$E(l)=(sum_{r6})(sum_{c2})/N$

A Chi-Squared, or χ^2, value for table B.2 can be obtained by applying:

$$\chi^2=\sum_{u=1}^{v}(O(x_u)-E(x_u))^2/E(x_u) \tag{B.2}$$

Using the Chi-Square value and the df value, the corresponding P-Value in table B.1 is obtained. The P-Value can be applied to determine the significance between subject and accuracy for table B.2.

B.3 McNemar's Test

McNemar's test is a paired version of the Chi-Square test. The test was applied to the the number of correctly and incorrectly estimated frames obtained from the two methods M1 and M2 in chapter 6.

Let "1" represent the correctly estimated frames and "0" represent the incorrectly estimated frames. Tables B.4 to B.12 summarize the Chi-Square and P-Value for each transition from T1 to T9 across all subjects. The *df* is 1 for each table from T1 to T9. The P-Value is then obtained in the same manner as described in the previous section.

TABLE B.4: McNemar's test for T1.

		M1		
		"1"	"0"	Total
M2	"1"	72	26	**98**
	"0"	2	17	**19**
	Total	**74**	**43**	**117**
Chi-Square				18.893
P-Value				<0.0001

TABLE B.5: McNemar's test for T2.

		M1		
		"1"	"0"	Total
M2	"1"	73	2	**75**
	"0"	16	6	**22**
	Total	**89**	**8**	**97**
Chi-Square				9.389
P-Value				0.0022

TABLE B.6: McNemar's test for T3.

		M1		
		"1"	"0"	Total
M2	"1"	59	2	**61**
	"0"	20	16	**36**
	Total	**79**	**18**	**97**
Chi-Square				13,136
P-Value				0.0003

TABLE B.7: McNemar's test for T4.

		M1		
		"1"	"0"	Total
M2	"1"	76	4	**80**
	"0"	28	11	**39**
	Total	**104**	**15**	**119**
Chi-Square				16.531
P-Value				<0.0001

TABLE B.8: McNemar's test for T5.

		M1		
		"1"	"0"	Total
M2	"1"	68	3	**71**
	"0"	20	7	**27**
	Total	**88**	**10**	**98**
Chi-Square				11.130
P-Value				0.0008

TABLE B.9: McNemar's test for T6.

		M1		
		"1"	"0"	Total
M2	"1"	106	0	**106**
	"0"	5	2	**7**
	Total	**111**	**2**	**113**
Chi-Square				3.200
P-Value				0.0736

TABLE B.10: McNemar's test for T7.

		M1		
		"1"	"0"	Total
M2	"1"	86	0	**86**
	"0"	16	17	**33**
	Total	**102**	**17**	**119**
Chi-Square				14.063
P-Value				0.0002

TABLE B.11: McNemar's test for T8.

		M1		
		"1"	"0"	Total
M2	"1"	83	5	**88**
	"0"	28	17	**45**
	Total	**111**	**22**	**133**
Chi-Square				14.667
P-Value				0.0001

TABLE B.12: McNemar's test for T9.

		M1		
		"1"	"0"	Total
M2	"1"	70	1	**71**
	"0"	19	7	**26**
	Total	**89**	**8**	**97**
Chi-Square				14.450
P-Value				0.0001

Bibliography

[1] R. Abbott and L. Williams, "Multiple target tracking with lazy background subtraction and connected components analysis," *Machine Vision and Applications*, vol. 20, no. 2, pp. 93–101, 2009.

[2] I. Achmed, "Upper body pose recognition and estimation towards the translation of South African Sign Language," in *Masters Thesis, University of the Western Cape, Computer Science*, 2010.

[3] N. Artner, "A comparison of mean shift tracking methods," in *12th Central European Seminar on Computer Graphics*, 2008, pp. 197–204.

[4] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *Signal Processing, IEEE Transactions on*, vol. 50, no. 2, pp. 174–188, 2002.

[5] S. Askar, Y. Kondratyuk, K. Elazouzi, P. Kauff, and O. Schreer, "Vision-based skin-colour segmentation of moving hands for real-time applications," in *Proc. of 1st European Conf. on Visual Media Production (CVMP)*, 2004, pp. 524–529.

[6] V. Athitsos, J. Alon, S. Sclaroff, and G. Kollios, "Boostmap: A method for efficient approximate similarity rankings," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 2004, pp. 268–275.

[7] V. Athitsos and S. Sclaroff, "Inferring body pose without tracking body parts," in *International Conference on Computer Vision and Pattern Recognition*, 2000.

[8] ——, "An appearance-based framework for 3D hand shape classification and camera viewpoint estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Published by the IEEE Computer Society, 2002, pp. 45–50.

[9] A. Atsalakis and N. Papamarkos, "Color reduction and estimation of the number of dominant colors by using a self-growing and self-organized neural gas," *Engineering Applications of Artificial Intelligence*, vol. 19, no. 7, pp. 769–786, 2006.

[10] Y. Benezeth, B. Emile, H. Laurent, and C. Rosenberger, "Vision-based system for human detection and tracking in indoor environment," *International Journal of Social Robotics*, vol. 2, no. 1, pp. 41–52, 2010.

[11] S. Birchfield. (1997) KLT: An implementation of the Kanade-Lucas-Tomasi feature tracker. Source code: http://robotics.stanford.edu/~birch/klt [Accessed 1 November 2011].

[12] ——, "Elliptical head tracking using intensity gradients and color histograms," in *Computer Vision and Pattern Recognition, 1998. Proceedings. 1998 IEEE Computer Society Conference on.* IEEE, 1998, pp. 232–237.

[13] Blender. [Online] Available at http://www.blender.org [Accessed 1 November 2011]. Blender Foundation.

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

[15] ——, "Hierarchical chamfer matching: A parametric edge matching algorithm," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, no. 6, pp. 849–865, 1988.

[16] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library.* O'Reilly Media, 2008.

[17] G. Bradski, "Computer vision face tracking for use in a perceptual user interface," *Intel Technology*, 1998.

[18] Y. Cai, N. de Freitas, and J. Little, "Robust visual tracking for multiple targets," *Computer Vision–ECCV 2006*, pp. 107–118, 2006.

[19] A. Cavender, R. Ladner, and E. Riskin, "MobileASL:: intelligibility of sign language video as constrained by mobile phone technology," in *Proceedings of the 8th International ACM SIGACCESS Conference on Computers and Accessibility.* ACM, 2006, p. 78.

[20] A. CAWSEY, *Essence of artificial intelligence.* Prentice Hall, 1997.

[21] D. Chai and K. Ngan, "Face segmentation using skin-color map in videophone applications," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 9, no. 4, pp. 551–564, 1999.

[22] C. Chang and C.-J. Lin, "LibSVM: A library for Support Vector Machines," *ACM Transactions on Intelligent Systems and Technology*, 2011, [Online] Software Available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[23] K. Chutisant and S. Hideo, "Model-based hand tracking by chamfer distance and adaptive color learning using particle filter," *EURASIP Journal on Image and Video Processing*, vol. 2009, 2009.

[24] D. Comaniciu, V. Ramesh, and P. Meer, "Real-time tracking of non-rigid objects using mean shift," in *CVPR*. IEEE Computer Society, 2000, pp. 142–149.

[25] J. Connan, "Integration of signed and verbal communication: South African Sign Language recognition and animation," January 2011, [Online] Available at http://www.coe.uwc.ac.za/index.php/SASL.html.

[26] N. Cristianini, "Support vector and kernel machines," *Tutorial at ICML*, 2001.

[27] N. Cristianini and J. Shawe-Taylor, *An introduction to Support Vector Machines: and other kernel-based learning methods*. Cambridge Univ Pr, 2000.

[28] O. Cuisenaire, "Distance transformations: Fast algorithms and applications to medical image processing," PhD Dissertation, Communications and Remote Sensing Laboratory, Catholic University of Louvain, Belgium, October 1999.

[29] L. Da Vinci, *A Treatise on Painting*. Seventh Press, 2008.

[30] DeafSA, "Deaf Federation of South Africa," January 2011, [Online] Available at http://www.deafsa.co.za/index-2.html.

[31] K. Deguchi, O. Kawanaka, and T. Okatani, "Object tracking by the mean-shift of regional color distribution combined with the particle-filter algorithms," in *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, vol. 3. IEEE, 2004, pp. 506–509.

[32] H. Fillbrandt, S. Akyol, and K. Kraiss, "Extraction of 3D hand shape and posture from image sequences for sign language recognition," in *Proceedings of the IEEE International Workshop on Analysis and Modeling of Faces and Gestures*. IEEE Computer Society, 2003, p. 181.

[33] B. Fisher. Image processing learning resources: Connected components labeling. [Online] Available at http://homepages.inf.ed.ac.uk/rbf/HIPR2/label.htm [Accessed 1 November 2010]. HIPR2 Group.

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

[35] H. Freeman and R. Shapira, "Determining the minimum-area encasing rectangle for an arbitrary closed curve," *Communications of the ACM*, vol. 18, no. 7, pp. 409–413, 1975.

[36] J. Friedman, "Another approach to polychotomous classification," Department of Statistics, Stanford University, Technical Report, 1996, [Online] Available at http://www.stat.stanford.edu/reports/friedman/poly.ps.z.

[37] D. Gavrila, "Pedestrian detection from a moving vehicle," *Computer Vision–ECCV 2000*, pp. 37–49, 2000.

[38] 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, June 2010.

[39] S. Gupta, "Support Vector Machines based modelling of concrete strength," *International Journal of Computer Systems Science and Engineering*, vol. 3, no. 1, 2008.

[40] L. He, "Generation of human body models," Master's Thesis, The University of Auckland, April 2005.

[41] R. Hess and A. Fern, "Discriminatively trained particle filters for complex multi-object tracking," in *Computer Vision and Pattern Recognition*. MiaMi, USA: IEEE, 2009, pp. 240–247.

[42] S. Howard, *Finger talk–South African Sign Language Dictionary*. South Africa: Mondi, 2008.

[43] C. Hsu, C. Chang, and C. Lin, "A practical guide to support vector classification," National Taiwan University, Technical Report, 2003.

[44] 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.

[45] C. Hue, J. Le Cadre, and P. Pérez, "Sequential Monte Carlo methods for multiple target tracking and data fusion," *Signal Processing, IEEE Transactions on*, vol. 50, no. 2, pp. 309–325, 2002.

[46] H-Anim working group. [Online] Available at http://h-anim.org/ [Accessed 1 November 2011]. Humanoid animation working group.

[47] Information technology–Computer graphics and image processing–Humanoid animation (H-Anim), ISO/IEC FCD 19774:200x. [Online] Available at http://hanim.org/Specifications/H-Anim200x/ISO_IEC_FCD_19774/ [Accessed 1 November 2011]. Humanoid animation working group.

[48] M. Isard and A. Blake, "Condensation–conditional density propagation for visual tracking," *International journal of computer vision*, vol. 29, no. 1, pp. 5–28, 1998.

[49] ——, "A mixed-state condensation tracker with automatic model-switching," in *Computer Vision, 1998. Sixth International Conference on.* IEEE, 1998, pp. 107–112.

[50] P. KaewTraKulPong and R. Bowden, "An improved adaptive background mixture model for real-time tracking with shadow detection," in *Proc. European Workshop Advanced Video Based Surveillance Systems*, vol. 1, no. 3, 2001.

[51] S. Keerthi and C. Lin, "Asymptotic behaviors of Support Vector Machines with Gaussian kernel," *Neural Computation*, vol. 15, no. 7, pp. 1667–1689, 2003.

[52] R. Kennaway, "Experience with and requirements for a gesture description language for synthetic animation," *Gesture-Based Communication in Human-Computer Interaction*, pp. 449–450, 2004.

[53] M. Kolsch. (2010, January) HandVu: Vision-based hand gesture recognition and user interface. [Online] Available at http://www.movesinstitute.org/~kolsch/HandVu/HandVu.html.

[54] M. Kolsch and M. Turk, "Fast 2D hand tracking with flocks of features and multi-cue integration," in *Computer Vision and Pattern Recognition Workshop.* IEEE, 2004, pp. 158–158.

[55] J. Lee and T. Kunii, "Model-based analysis of hand posture," *Computer Graphics and Applications, IEEE*, vol. 15, no. 5, pp. 77–86, 1995.

[56] S. Lee and I. Cohen, "3D hand reconstruction from a monocular view," *Pattern Recognition*, vol. 3, pp. 310–313, 2004.

[57] D. Lelis. (2010, January) EHCI 6 degrees of freedom hand tracker. [Online] Available at http://code.google.com/p/ehci/wiki/HandTracking or http://www.youtube.com/watch?v=Rmh-mZFxWns.

[58] P. Li and J. Connan, "Numberplate detection using double segmentation," in *Proceedings of the 2010 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists.* ACM, 2010, pp. 386–389.

[59] P. Li, M. Ghaziasgar, and J. Connan, "Hand shape recognition and estimation for South African Sign Language," in *Proceedings of the Southern Africa Telecommunication Networks and Applications Conference.* SATNAC, 2011, pp. 184–189.

[60] P. Li. The website for hand tracking and estimation. [Online] Available at http://www.cs.uwc.ac.za/~lpei/index_sub_demo.htm [Accessed 1 November 2011].

[61] R. Liang and M. Ouhyoung, "A real-time continuous gesture recognition system for sign language," in *Proceedings of the 3rd IEEE International Conference on Automatic Face and Gesture Recognition*, 1998, pp. 558–567.

[62] C. Lien, "A scalable model-based hand posture analysis system," *Machine Vision and Applications*, vol. 16, no. 3, pp. 157–169, 2005.

[63] H. Lin and C. Lin, "A study on sigmoid kernels for SVM and the training of non-PSD kernels by SMO-type methods," Department of Computer Science and Information Engineering, National Taiwan University, Technical Report, March 2003, [Online] Available at http://www.csie.ntu.edu.tw/cjlin/papers/tanh.pdf.

[64] B. Lo and S. Velastin, "Automatic congestion detection system for underground platforms," in *Intelligent Multimedia, Video and Speech Processing, 2001. Proceedings of 2001 International Symposium on.* IEEE, 2001, pp. 158–161.

[65] L. Louw, "Automated face detection and recognition for a login system," Master's Thesis, University of Stellenbosch, March 2007.

[66] S. Lu, G. Huang, D. Samaras, and D. Metax, "Model-based integration of visual cues for hand tracking," in *Motion and Video Computing.* Published by the IEEE Computer Society, 2002, pp. 118–124.

[67] S. Lu, D. Metaxas, D. Samaras, and J. Oliensis, "Using multiple cues for hand tracking and model refinement," in *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, vol. 2. IEEE, 2003, pp. 443–450.

[68] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *International Joint Conference on Artificial Intelligence*, vol. 3, 1981, pp. 674–679.

[69] G. Madzarov, D. Gjorgjevikj, and I. Chorbev, "A multi-class SVM classifier utilizing binary decision tree," *Informatica: An International Journal of Computing and Informatics*, vol. 33, no. 2, pp. 225–233, 2009.

[70] E. Maggio and A. Cavallaro, "Hybrid particle filter and mean shift tracker with adaptive transition model," in *Proc. of IEEE Signal Processing Society International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2005, pp. 19–23.

[71] MakeHuman project website. [Online] Available at http://www.makehuman.org/ [Accessed 1 November 2011]. The MakeHuman team.

[72] J. Martinez-del Rincon, C. Orrite-Urunuela, and J. Herrero-Jaraba, "An efficient particle filter for color-based tracking in complex scenes," in *IEEE Conference on Advanced Video and Signal Based Surveillance*. London, UK: IEEE, 2007, pp. 176–181.

[73] C. McCulloch, D.and Campbell, "The kernel of SVM," [Online] Available at http://www.enm.bris.ac.uk/teaching/projects/2004_05/dm1654/kernel.htm [Accessed 1 November 2010], 2010.

[74] A. McIvor, "Background subtraction techniques," in *Proc. of Image and Vision Computing, Auckland, New Zealand*, 2000, pp. 147–153.

[75] T. Mitchell, "Machine learning." *McGraw-Hill*, p. 368, 1997.

[76] A. W. Moore, "Tutorial of SVM by andrew moore," [Online] Available at http://www.autonlab.org/tutorials/svm.html [Accessed 1 November 2010], 2010.

[77] S. Morrissey, "Assistive translation technology for deaf people: translating into and animating Irish Sign Language," in *11th International Conference on Computers Helping People with Special Needs*, Linz, Austria, 2008, pp. 9–19.

[78] T. Mullen, *Introducing character animation with Blender*. Sybex, 2007.

[79] N. Naidoo, "South African Sign Language recognition using feature vectors and Hidden Markov Models," Master's Thesis, University of the Western Cape, Computer Science, December 2009.

[80] S. Oh, S. Kim, and S. Oh, "Pedestrian collision warning systems using neural networks based on a single camera," in *15th World Congress on Intelligent Transport Systems*. New York, USA: ITS America, 2008, p. 12.

[81] V. Pavlovic, R. Sharma, and T. Huang, "Visual interpretation of hand gestures for human-computer interaction: A review," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 19, no. 7, pp. 677–695, 1997.

[82] F. Quek, "Unencumbered gestural interaction," *Multimedia, IEEE*, vol. 3, no. 4, pp. 36–47, 1996.

[83] J. Rehg and T. Kanade, "Visual tracking of high dof articulated structures: an application to human hand tracking," *Computer Vision–ECCV'94*, pp. 35–46, 1994.

[84] ——, "Model-based tracking of self-occluding articulated objects," in *iccv*. Published by the IEEE Computer Society, 1995, p. 612.

[85] M. Ren, J. Yang, and H. Sun, "Tracing boundary contours in a binary image," *Image and vision computing*, vol. 20, no. 2, pp. 125–131, 2002.

[86] D. Salmond, N. Gordon, and A. Smith, "Novel approach to nonlinear/non-gaussian bayesian state estimation," *IEE Proceedings-F Radar and Signal Processing*, vol. 140, no. 1, pp. 107–113, 1993.

[87] Y. Sato, M. Saito, and H. Koik, "Real-time input of 3D pose and gestures of a user's hand and its applications for HCI," *vr*, pp. 79–86, 2001.

[88] O. Schreer and S. Ngongang, "Real-time gesture recognition in advanced video-communication services," in *Image Analysis and Processing, 2007. ICIAP 2007. 14th International Conference on.* IEEE, 2007, pp. 253–258.

[89] V. Segers, "The efficacy of the eigenvector approach to South African Sign Language identification," Master's Thesis, University of the Western Cape, Computer Science, December 2009.

[90] S. Sen-Ching and C. Kamath, "Robust techniques for background subtraction in urban traffic video," in *Proceedings of SPIE*, vol. 5308, 2004, pp. 881–892.

[91] C. Shan, Y. Wei, T. Tan, and F. Ojardias, "Real time hand tracking by combining particle filtering and mean shift," in *6th IEEE international conference on Automatic face and gesture recognition.* Washington, USA: IEEE Computer Society, 2004.

[92] J. Shi and C. Tomasi, "Good features to track," in *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on.* IEEE, 1993, pp. 593–600.

[93] S. Shih, Y. Hung, and W. Lin, "Calibration of an active binocular head," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 28, no. 4, pp. 426–442, 1998.

[94] N. Shimada, K. Kimura, and Y. Shirai, "Real-time 3D hand posture estimation based on 2D appearance retrieval using monocular camera," in *Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems, 2001. Proceedings. IEEE ICCV Workshop on.* IEEE, 2001, pp. 23–30.

[95] T. Shiratori, H. Park, L. Sigal, Y. Sheikh, and J. Hodgins, "Motion capture from body-mounted cameras," in *ACM Transactions on Graphics (TOG)*, vol. 30, no. 4. ACM, 2011, p. 31.

[96] D. Skapura, *Building neural networks*. Addison-Wesley, 1996.

[97] V. Spruyt, A. Ledda, and S. Geerts, "Real-time multi-colourspace hand segmentation," in *17th International Conference on Image Processing (ICIP)*. IEEE, 2010, pp. 3117–3120.

[98] C. Stauffer and W. Grimson, "Adaptive background mixture models for real-time tracking," in *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, vol. 2. IEEE, 1999.

[99] B. Stenger, P. Mendonça, and R. Cipolla, "Model-based hand tracking using an unscented kalman filter," in *Proc. British Machine Vision Conference*, vol. 1, 2001, pp. 63–72.

[100] B. Stenger, A. Thayananthan, P. Torr, and R. Cipolla, "Hand pose estimation using hierarchical detection," *Computer Vision in Human-Computer Interaction*, pp. 105–116, 2004.

[101] ——, "Model-based hand tracking using a hierarchical bayesian filter," *IEEE transactions on pattern analysis and machine intelligence*, pp. 1372–1384, 2006.

[102] E. Stergiopoulou and N. Papamarkos, "A new technique for hand gesture recognition," in *Image Processing, 2006 IEEE International Conference on*. IEEE, 2006, pp. 2657–2660.

[103] M. Stricker and M. Swain, "The capacity of color histogram indexing," in *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*. IEEE, 1994, pp. 704–708.

[104] M. Swain and D. Ballard, "Color indexing," *International journal of computer vision*, vol. 7, no. 1, pp. 11–32, 1991.

[105] A. Thayananthan, B. Stenger, P. Torr, and R. Cipolla, "Learning a kinematic prior for tree-based filtering," in *Proc. British Machine Vision Conference*, vol. 2, 2003, pp. 589–598.

[106] ——, "Shape context and chamfer matching in cluttered scenes," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1. IEEE Comput. Soc, June 2003, pp. 127–133.

[107] J. Triesch and C. von der Malsburg, "Classification of hand postures against complex backgrounds using elastic graph matching," *Image and Vision Computing*, vol. 20, no. 13-14, pp. 937–943, 2002.

[108] P. Vadakkepat and L. Jing, "Improved particle filter in sensor fusion for tracking randomly moving object," *Instrumentation and Measurement, IEEE Transactions on*, vol. 55, no. 5, pp. 1823–1832, 2006.

[109] D. Van Wyk, "Virtual human modeling and animation for sign language visualization," Master's Thesis, University of the Western Cape, Computer Science, December 2008.

[110] L. van Zijl and D. Barker, "South African Sign Language Machine Translation System," in *Proceedings of the 2nd international conference on Computer graphics, virtual Reality, visualisation and interaction in Africa*. ACM, 2003, p. 52.

[111] V. Vapnik, *The Nature of Statistical Learning Theory*. Springer Verlag, 2000.

[112] 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*, vol. 1, December 2001, pp. 511–518.

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

[114] HD Hero sport action camera. [Online] Available at http://gopro.dlvr.co.za/ [Accessed 1 November 2011]. Woodman Labs.

[115] Y. Wu, G. Hua, and T. Yu, "Tracking articulated body by dynamic markov network," in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*. IEEE, 2003, pp. 1094–1101.

[116] Y. Wu, J. Lin, and T. Huang, "Capturing natural hand articulation," in *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, vol. 2. IEEE, 2001, pp. 426–432.

[117] L. Yi, "KernTune: Self-tuning linux kernel performance using Support Vector Machines," Master's Thesis, The University of the Western Cape, November 2006.

[118] L. Zhao and C. Thorpe, "Stereo-and neural network-based pedestrian detection," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 1, no. 3, pp. 148–154, 2000.