UNIVERSITY OF THE WESTERN CAPE

Faster Upper Body Pose Recognition and Estimation Using Compute Unified Device Architecture

by

Dane Brown

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

in the
Faculty of Science
Department of Computer Science

Supervisor: Mehrdad Ghaziasgar
Co-supervisor: James Connan

February 2013

Declaration

I, Dane Brown, declare that this thesis “Faster Upper Body Pose Recognition and Estimation Using Compute Unified Device Architecture” is my own work, that it has not been submitted before for any degree or assessment at any other university, and that all the sources I have used or quoted have been indicated and acknowledged by means of complete references.

Signature: .... Date: ....

"When natural inclination develops into a passionate desire, one advances towards his goal in seven-league boots."

Nikola Tesla

Abstract

The SASL project is in the process of developing a machine translation system that can translate fully-fledged phrases between SASL and English in real-time. To-date, several systems have been developed by the project focusing on facial expression, hand shape, hand motion, hand orientation and hand location recognition and estimation. Achmed developed a highly accurate upper body pose recognition and estimation system. The system is capable of recognizing and estimating the location of the arms from a two-dimensional video captured from a monocular view at an accuracy of 88%. The system operates at well below real-time speeds. This research aims to investigate the use of optimizations and parallel processing techniques using the CUDA framework on Achmed's algorithm to achieve real-time upper body pose recognition and estimation. A detailed analysis of Achmed's algorithm identified potential improvements to the algorithm. A re- implementation of Achmed's algorithm on the CUDA framework, coupled with these improvements culminated in an enhanced upper body pose recognition and estimation system that operates in real-time with an increased accuracy.

Keywords

Pose Recognition and Estimation, Graphics Processing Unit, Compute Unified Device Architecture, Face Detection, Skin Detection, Background Subtraction, Morphological Operations, Haar Features, Support Vector Machine, Blender.

Acknowledgements

First and foremost, I thank my Lord and Saviour, Jesus Christ, for giving me the strength, wisdom and perseverance throughout my life. Also thank You for giving me this wonderful opportunity to study at the University of the Western Cape in beautiful Cape Town, South Africa.

I would like to thank my parents for giving me the opportunity to further my education. I am indebted to my parents. They have taught me the value of life and have always supported me. They have motivated me to excel at everything and not to easily get discouraged by the negative sources of life.

I would especially like to thank my supervisor Mr Mehrdad Ghaziasgar for his outstanding guidance, inspiration and encouragement. Thank you for being so understanding and patient with me. Your guidance is invaluable. I would also like to thank my co-supervisor Mr James Connan for his guidance and support from afar.

I would like to thank Prof. Richard Madsen for his guidance on my statistical analysis. I would also like to thank my lab mates Ibraheem Frieslaar, Warren Nel, Diego Mushfieldt and Imran Achmed. Our lab was balanced with research and fun because of your auras. Imran thank you for your generous advice and assistance.

I would like to thank the University of the Western Cape for the opportunity to study here. I thank my lecturers for educating me and helping me acquire knowledge in the field of computer science, mathematics and other fields in life. I would especially like to thank Mr James Connan and Mr Reg Dodds for their different ways of sculpting my brain, effectively aligning it with unique problem solving attributes.

Thank you Selina for your friendship, motivation and support through all these years. It really means a lot to me.

Contents

List of Figures

List of Tables

Abbreviations

CPU	**C**entral **P**rocessing **U**nit
CT	**C**omputed **T**omography
CUDA	**C**ompute **U**nified **D**evice **A**rchitecture
DAG	**D**irected **A**cyclic **G**raph
DNA	**D**eoxyribo**n**ucleic **A**cid
DOF	**D**egree **O**f **F**reedom
FPS	**F**rames **P**er **S**econd
GHz	**G**iga**h**ertz
GMM	**G**aussian **M**ixture **M**odels
GPGPU	**G**eneral-**P**urpose **C**omputing on **G**raphics **P**rocessing **U**nits
GPU	**G**raphics **P**rocessing **U**nit
HSV	**H**ue **S**aturation **V**alue
LibSVM	**L**ibrary of **S**upport **V**ector **M**achines
LLE	**L**ocally **L**inear **E**mbedding
MCMC	**M**arkov **C**hain **M**onte **C**arlo
MRI	**M**agnetic **R**esonance **I**magery
PSO	**P**article **S**warm **O**ptimization
RAM	**R**andom **A**ccess **M**emory
RNA	**R**ibo**n**ucleic **A**cid
RR	**R**idge **R**egression
RVM	**R**elevance **V**ector Memory
SASL	**S**outh **A**frican **S**ign **L**anguage
SQL	**S**tructured **Q**uery **L**anguage
SVM	**S**upport **V**ector **M**achines
VCM	**V**ector **C**oherence **M**apping

Chapter 1

Introduction

1.1 Background and Motivation

Effective communication is essential for daily interaction, whether it is formal or informal. Examples are buying groceries and bus tickets or communicating ideas in the workplace. Over six hundred thousand Deaf[1] South Africans struggle to communicate with the hearing [55].

Hearing individuals are typically uneducated or unfamiliar with sign languages and are partially or completely unable to communicate with the Deaf. A common misconception amongst the hearing is that Sign Language is a universal language through which all Deaf individuals communicate. The fact is that various sign languages exist, with most countries having their own unique sign language [29, 30]. There can be as much diversity in sign languages as there are in spoken languages. Another misconception is that what is perceived to be the universal Sign Language is the signed equivalent of spoken languages [1, 29]. Research has shown that sign languages are fully fledged languages on their own, with an entirely different set of grammatical and syntactic structure to spoken languages [55]. An example is British Sign Language, which is entirely different from English. This has also led to the incorrect assumption that Deaf individuals can read and write in spoken languages.

In South Africa the official language for the Deaf is South African Sign Language (SASL) [55]. The Deaf in South Africa have limited access to educational services and socio-economic privileges compared to hearing people. This caused segregation between the Deaf and hearing in society. The Deaf in South Africa suffer from poverty and typically cannot afford education [29]. Skilled interpreters can be hired as a medium between

[1]The social group that are completely unable to communicate in spoken languages.

Deaf and hearing individuals [55]. However, their services tend to be costly and scarce. The number of interpreters relative to that of Deaf individuals in South Africa is very small [30, 40]. In some cases privacy can be a problem when using interpreters. A Deaf person might prefer doctor patient confidentiality in private consultations and not be comfortable with an interpreter.

The "Integration of Signed and Verbal Communication: South African Sign Language Recognition and Animation" project [29] at the University of the Western Cape, also called the SASL project, aims to develop a real-time machine translation system that can translate between English and SASL. The system involves two distinct procedures: translating from SASL to English; and translating from English to SASL. The procedure relevant to this research is the translation of SASL to English, which involves the extraction of semantic information from a video that consists of a Deaf individual communicating in SASL.

Research has shown that five parameters characterize sign language gestures [34, 51]. These parameters are facial expressions, hand shape, hand motion, hand orientation and hand location. The SASL project has conducted research into the recognition of these five parameters from SASL videos. Whitehill developed a robust facial expression recognition system [104]. Li developed a hand shape estimation system [51]. Naidoo [61] and Rajah [72] developed hand motion recognition systems. Achmed developed a hand location recognition system, also referred to as upper body pose recognition and estimation in the literature [1]. Achmed focused on achieving high recognition accuracy using a computationally intensive algorithm. The aim of his part of the project was to successfully estimate the positions of the arms and joints in 3D using the 3D humanoid avatar developed by Van Wyk [100]. The system was solely focused on achieving a high-accuracy recognition methodology. It is known that the system runs slower than real-time speed. This research aims to re-implement and optimize the system using parallel processing techniques such that it achieves both a high accuracy and real-time performance.

Real-time performance is key to the interactive communication of the machine translation system. The processing speed of the system affects the response time of the system to the user. A higher processing speed helps present a result to the user faster. Hence, a better response time. Achmed [1], Li [51], Naidoo [61] and Rajah [72] investigated methods of improving the response time by selectively dropping frames possibly at the expense of accuracy. To date, a trade-off was made between the response time and the accuracy in terms of the number of frames processed. This research aims to eliminate this trade-off.

1.2 Research Question

The following research question can be specified based on the previous section: "Can optimization and parallel processing techniques on the CUDA framework be applied to Achmed's methodology to achieve real-time performance at a sustained or improved estimation accuracy?"

1.3 Research Objectives

1. Conduct an analysis of Achmed's algorithm to identify which components require optimization.
2. Re-implement the identified components using optimization or alternative techniques.
3. Re-implement the entire system using parallel processing techniques on the Graphics Processing Unit (GPU).
4. Conduct testing to determine whether real-time performance has been achieved.
5. Conduct testing to determine whether the accuracy has been sustained or improved.

1.4 Premises

- It is assumed that the sign language videos to be used in training and testing will consist of the entire upper body of the signer facing the web camera. During a sign language conversation, breaking eye contact with the signer is considered rude and disrespectful [55, 83]. This assumption is therefore justified.
- It is assumed that the signer will stand in front of an arbitrary background and not require any special equipment such as data gloves or coloured markers. This is a requirement of the SASL project to attempt to provide the most natural feel to the system.

1.5 Thesis Outline

The remainder of the thesis is arranged as follows:

Chapter 2: *Pose Recognition and Estimation*: This chapter reviews existing literature on pose recognition and estimation. It explores the various approaches and algorithms used to attain varying accuracies. The strengths and weaknesses of the various approaches are discussed.

Chapter 3: *Compute Unified Device Architecture*: This chapter reviews studies which compare the processing speed of applications as implemented on the CPU and GPU. The studies first deal with pose recognition and estimation applications in the literature as it is the most relevant to this research. It is followed by image processing applications and general-purpose processing applications.

Chapter 4: *Image Processing in Learning-Based Pose Recognition and Estimation*: This chapter discusses the components that can form part of the learning-based system in detail. This discussion forms the basis of the optimization process used in subsequent chapters.

Chapter 5: *Design and Implementation of the Faster Upper Body Pose Recognition and Estimation System*: The chapter discusses the proposed upper body pose recognition and estimation system. It discusses the analysis, optimization and re-implementation of Achmed's system to produce the faster upper body pose recognition and estimation system.

Chapter 6: *Experimental Results and Analysis*: This chapter discusses the testing carried out to determine whether the proposed system achieves real-time speed and a sustained or improved accuracy.

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

Chapter 2

Pose Recognition and Estimation

The majority of pose recognition and estimation systems require additional equipment to be pre-attached to the person's body [51]. Examples include data gloves, data suits and coloured markers. The use of such equipment makes these systems unnatural, conspicuous, expensive and impractical. The SASL project aims to avoid the use of such equipment and provide a natural feel to the machine translation system.

The project has therefore tended towards newer approaches that make use of computer vision, image processing and machine learning which can eliminate the pre-requisite of attached markers. These approaches can be categorized as follows: model-based, example-based and learning-based approaches. Where possible the results, accuracy, strengths and weaknesses of the various algorithms that have been used to perform pose recognition and estimation are mentioned.

The rest of the chapter is organized as follows: Section 2.1 defines pose recognition and estimation within the scope of this research; Sections 2.2 to 2.4 discuss the various systems according to the approach used; a summary of the chapter follows.

2.1 Definition

Pose recognition and estimation is defined as the process of recognizing and estimating the position and orientation of a human body in a single frame or over multiple frames [57]. In the case of multiple frames, the term tracking is used.

The objective of pose recognition and estimation is to determine the set of angles for each degree of freedom (DOF) of the joints in the human body model with respect to its local or relative coordinate system. Data captured from a single camera is represented in 2D with respect to a world coordinate system and later estimated in 3D using a 3D human body model with respect to its local coordinate system.

2.2 Model-Based Approaches

Model-based approaches assume an explicitly known parametric human body model and estimate the pose by comparing the test image on the known image positions for every body part [2, 14, 16]. Angles and lengths amongst the body parts are commonly used parameters for these models. These approaches often have the problem of a high computational cost due to their high configuration complexity. In some cases the computational cost can be exponential. The computational cost can be reduced by limiting the number of DOF and using symmetry. Top-down and bottom-up models are the two categories used in model-based pose estimation and are explained in the following two subsections.

2.2.1 Top-Down Methods

Top-down methods execute a brute-force match on the performed pose by comparing the high-dimensional pose space, kinematic structure and corresponding constraints directly to the model [43]. A complex cost function measures the similarity of the predicted pose to the actual observed pose. The aim of the cost function is to find a match based on the optimal pose hypotheses that minimize the cost function because it is exponential relative to time.

Most top-down methods use an initialization procedure that determines the initial pose to estimate the next frame. To achieve accurate results they minimize projection errors of kinematic models by using numerical optimization [89], generating a large number of pose hypotheses [48] and sufficiently fine sampling. Probability sampling uses the probability distribution to search entire body configurations. There are a number of different probability sampling techniques used. One example is Markov Chain Monte Carlo (MCMC). Human body models are roughly represented by link-joint models. This is composed of 2D/3D geometric primitives such as cylinders and rectangles that are fitted to measure similarity [38].

Taylor [94] investigated the top-down method by assuming that the corresponding points between the articulated object and the model are known. Another assumption made was that the relative lengths of each segment in the model are known. The relationship between the points in space and the projections onto the articulated object can be

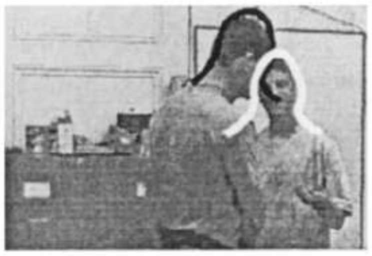


FIGURE 2.1: The tracking of multiple indistinguishable body parts by MacCormick and Blake's system [53].

modelled as a scaled orthogonal projection [1, 94].The resulting geometric constraints were used to estimate the performed pose. The testing procedure and results of Taylor's work are not clear from the literature.

Parameswaran and Chellappa's [67] research also used geometric constraints to estimate the pose of an individual. The isometric approximation is used by assuming that the body part proportions of all humans are approximately the same. Another assumption made in their research was that the amount of torso twist is negligible such that the distance between the shoulder joints are fixed. The head orientation is computed allowing the epipolar geometry of the image to be recovered, thus determining the 3D joint positions. Synthetic and real people were used to test the approach. Explicit results are not given, but the method was found to have poor accuracy when tested on real people compared to synthetically generated humans. This was attributed to the reduction in the definition of shoulders, hips and other joints of real humans by their clothes.

MacCormick and Blake's research resulted in the ability to track multiple indistinguishable body parts using a probabilistic exclusion principle [53]. This prevents image data from contributing to similar hypotheses for two or more body parts. They also proposed an efficient sampling method to improve the 3D-joint-position estimation accuracy. A partition sampling algorithm that uses particle filters with a simple wire-frame model was used. The algorithm reduces the computational cost of extra dimensions and is insensitive to background noise. This divides the wire-frame model into constituent body parts that can be determined individually. The test procedure and accuracy results are unclear from the literature. Figure 2.1 illustrates the results visually.

2.2.2 Bottom-Up Methods

Bottom-up methods do not use the whole body model to fit the observed pose [33]. Instead they fit constituent body part models, which are represented by cylinders, rectangles or feature points and use geometric constraints between the parts. The time complexity is much lower than that of the top-down approach. The model contains a

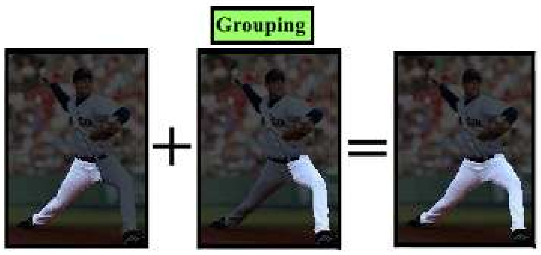


FIGURE 2.2: The grouping of two legs by Srinivasan and Shi's system [90].

list of body parts that are first identified and then pruned. Geometric constraints are used as a guide to assemble the full body pose that fits the model.

Srinivasan and Shi proposed a bottom-up shape parsing method that contains more complete partial body parts than previous bottom-up parsing methods [90]. Multiple image segmentations are parsed at each level to enable a robust initialization. The research is novel as it combines more than one body part and treats it as a bigger body part. The use of larger body parts has limitations such as its use of a fixed bottom-up parsing method that always moves from the legs upward. They describe their results as qualitatively good, but quantitatively poor. This means that the system is generally able to recognize the overall pose, but the error between the actual and estimated locations of constituent body parts is high. The test procedure and results are unclear from the literature, but Figure 2.2 illustrates the grouping of two legs.

Ioffe and Forsyth proposed that a human can be located in an image by finding candidate body segments and grouping the segments according to kinematic properties [41]. They state that pruning the search for body segments produces an efficient way of finding possible body parts. A probabilistic framework is used on the body parts located to limit the number of possible assemblies to more closely resemble humans. The test procedure consists of 120 positive images and 86 negative images. Their results show a 10% false positive rate and 49% false negative rate. The high false negative rate is attributed to the fact that the segment search is not able to find all relevant body segments, which negatively impacts the probabilistic framework. The grouping of segments according to kinematic properties is found to be effective, but requires a better body segment model.

Mori *et al.* proposed an approach that efficiently assembles body parts using low-level segmentation [60]. The body segment model is based on the Normalized Cuts algorithm [84], which produces candidates for limbs and torsos that are verified by a variety of cues. The constraints used to prune away incorrect body part combinations are symmetry, scale, and the position and colour of clothing. For future work they suggest using artificial intelligence (AI) heuristic search methods, such as the best-first search method. Furthermore, they suggest combining it with an example-based approach for a better

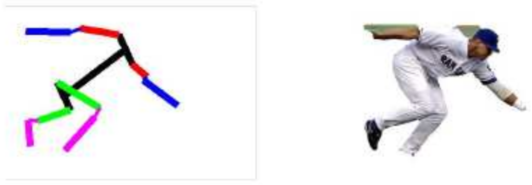


FIGURE 2.3: The detection of constituent limb segments by Mori *et al.*'s system [60].

result. The test procedure consists of 62 images of baseball players from sports news photographs. In 89% of the test images at least three of the eight limb segments are correctly estimated. One example of their system in action is illustrated in Figure 2.3.

2.2.3 Comparing Top-Down and Bottom-Up Methods

The two model-based approaches share a common challenge of initialization. The top-down approach requires suitable initialization to minimize projection errors of kinematic models, either by using numerical methods [89] or by generating a large number of pose hypotheses [48]. This helps to produce accurate results and reduce the search time. However, it can easily be trapped into local minima.

Bottom-up approaches require strict initialization of a particular body part, otherwise a fixed parsing method is used such as the work done by [90]. The bottom-up approach breaks up the human body into its constituent parts to efficiently handle its high dimensionality. Search space is significantly reduced and a greater number of poses can be recognized because of the low storage requirements [75]. The bottom-up approach does not easily converge to local minima and is generally less computationally intensive, but achieves a very poor accuracy when body part detectors fail.

Recent attempts have been made to eliminate the weaknesses and combine the strengths of these two model-based approaches described in the next subsection.

2.2.4 Combining Top-Down and Bottom-Up Methods

Hua *et al.* implemented a data-driven belief propagation Monte Carlo algorithm, which combines bottom-up and top-down visual cues within a rigorous statistical framework for efficient Bayesian inference [38]. The test procedure and results are not explicitly stated. Similarly, Lee and Cohen [48] proposed a data driven algorithm based on the Markov Chain Monte Carlo (MCMC) method, which introduces proposal maps to efficiently consolidate 3D pose candidates during the search. The test procedure and results are also not explicitly stated. Zhang *et al.* performed a hybrid strategy that utilizes the top-down MCMC method with a bottom-up deterministic search [106]. The test procedure

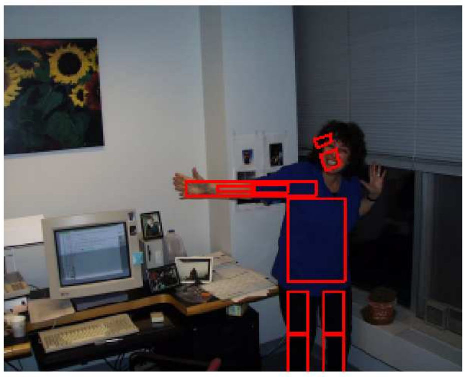


FIGURE 2.4: Felzenswalb and Huttenlocher's body part detectors based on colour [26].

contains 90 images of a person walking, 150 images of a person dancing and 100 images of people performing random poses. The hybrid strategy finds candidate poses with an accuracy of 40% from the 340 test images.

Gupta *et al.* combined the top-down approach and bottom-up approach to achieve efficient pose estimation with self-occlusions using multiple cameras [33]. The approach is based on 2D likelihoods and epipolar geometric constraints to search for likelihood regions in 3D human body space. Their results indicate a 96% correct body part detection rate when the joint error has a tolerance level of 50% of the limb length.

Felzenszwalb and Huttenlocher also combined the top-down and bottom-up model-based approaches using a collection of body part detectors to match a pose based on colour [26]. The distance transformation is used to find the global configuration of these body parts and to optimize a cost function to determine the pose. This significantly reduces the search complexity and has recently been used with belief propagation for 3D body tracking from multiple views [86]. However, it has not been found to have real-time performance. The test procedure is not clear. However, accuracy results are provided visually in Figure 2.4.

2.3 Example-Based Approaches

Example-based approaches use a large database of pose and image features that are trained with their corresponding pose or coordinates. Given a query image, the database returns one or more candidate poses with the closest matching features [14, 26, 58].

Example-based approaches require a solution to perform computationally expensive queries efficiently and accurately. Shakhnarovich *et al.* developed the parameter-sensitive hashing algorithm that indexes approximate nearest neighbours in the database that have similar features to the given query image [82]. Contour features are extracted

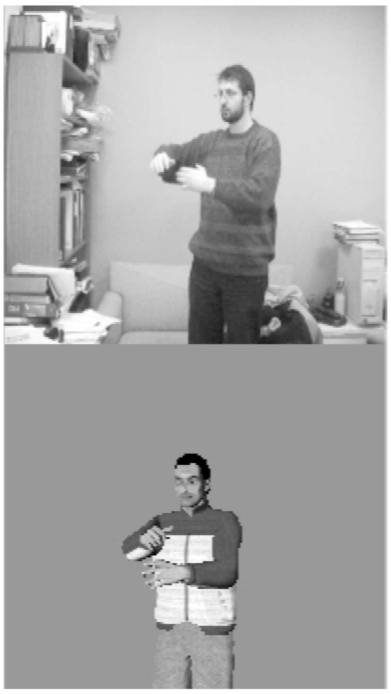


Figure 2.5: 3D pose estimated by Shakhnarovich *et al.*'s system [82].

using edge direction histograms. The training set consisted of 150000 images, which were rendered using a humanoid model in POSER. Only visual results are provided. In Figure 2.5, the estimated pose is illustrated below the input image. Athitsos and Sclaroff proposed similar work, but focused only on the estimation of hand poses [6].

Mori and Malik [59] proposed an approach that stores a number of exemplar 2D views of a human figure, locates the joint positions and estimates the body configuration and pose in 3D space. The stored images acquired from the CMU Motion of Body Database consists of individuals walking on a treadmill from multiple viewpoints. The stored views are manually marked and labelled. A shape context matching algorithm proposed by Belongie *et al.* is used with a kinematic chain-based deformation model in order to match the query image to the stored examples [10]. Alternatively matching can also be performed using an order structure algorithm proposed by Sullivan and Carlsson [92]. When a match occurs, the corresponding joint locations are transferred to the test shape. The 3D body configuration and pose is estimated from the test shape containing the joint coordinates of the body, using Taylor's algorithm [94]. The system was tested using a separate set of images of individuals performing the same actions. Only visual results are presented and show that their deformation-based approach performs well when the query image contains vivid edges similar to the stored image. This is especially true around the arms. The joint location process fails when the edges are substantially different.

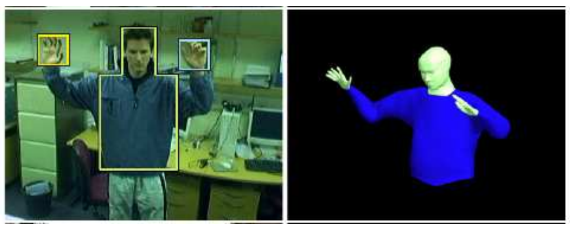


FIGURE 2.6: Micilotta's 3D pose estimation system [56].

The Chamfer Distance algorithm has proved to be effective in addressing the edge matching problem when applied to various shape comparison fields. Micilotta *et al.* conducted research using the exact Chamfer Distance method [56]. Several human upper body movements are stored as 3D body configurations. The body movements are distributed into three databases, namely, hand position, silhouette and edge map. An example of each database that has the highest matching score is used to reconstruct and estimate the pose based on the 3D configurations. Their reconstruction method enables the edges of the body parts to be defined vividly. The Chamfer Distance Transformation is applied to the matching process to identify the pose from the database. Their test results are displayed visually in Figure 2.6.

Cao *et al.* used the approximate Chamfer Distance to identify poses at a higher processing speed, but at a slightly reduced accuracy when compared to using the exact Chamfer Distance [15]. Eigen approximations are used to represent the distance transform in low-dimensional sub-space. They used a database containing 14964 images with numerous pose and angles of a 3D model. They compared their proposed implementation to the exact Chamfer Distance method and achieved better performance relative to time and memory usage, but at the expense of accuracy.

Achmed [1] proposed a similar approach to [56] and [15] that stores a number of exemplar 2D views of the upper body, locates the corresponding joint positions and estimates the body configuration in 3D space. The proposed implementation is specifically applied to sign language recognition and translation. His results show that a good face detection method is essential to increase the overall accuracy of the joint positions. Canny edge detection is used to enhance the edges before the Chamfer Distance is approximated. A 3D model, developed by Van Wyk [100] using Blender, was used to generate a wide range of example poses. The system was tested on 15 distinct SASL signs performed by six individuals. The approach achieved an overall recognition accuracy of 65%.

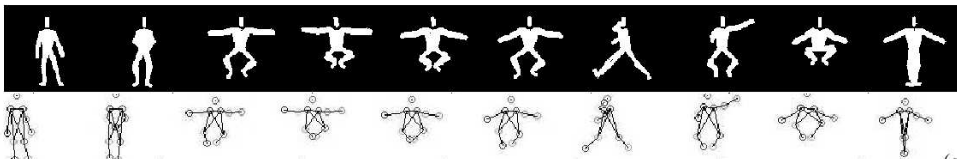


FIGURE 2.7: Estimated 2D joint configurations by Rosales and Sclaroff [79].

2.4 Learning-Based Approaches

Learning-based approaches extract features from images containing poses, representing them as vectors [14]. A trained regression function uses these vectors and predicts the pose by mapping the data of the image from feature space to pose space. This approach differs considerably to the other two approaches as it does not assume an explicit 3D body model. A variety of advanced machine learning techniques exist, each suited to specific applications and environments. Learning-based approaches are particularly appealing because of their potential to operate at high speeds compared to the previous two approaches, even as high as real-time. Some of the image features that are used include concatenated coordinates of sampled boundary points [31], Haar-like features generated by AdaBoost [103] and multi-scale edge direction histograms [22]. A set of example poses are trained to represent the relationship between the original image and the generated pose using regression functions. Many regression functions can be used, including AdaBoost, BoostMap [6], Relevance Vector Machines (RVMs) and Support Vector Machines (SVMs) [78].

Rosales and Sclaroff [79] proposed work that recovers body poses from single images using a non-linear supervised framework that maps image silhouettes to 2D body joint configurations. The mapping is done using a Specialized Mappings Architecture containing a feedback matching function. The image silhouettes and 2D body joint configurations are acquired using 3D motion capture data. Training of the system is carried out using the Expectation Maximization algorithm, which fits a Gaussian Mixture Model to cluster the 2D joint configurations. Joint clusters are used to train an inverse mapping between the image silhouette moments and 2D joint configurations. The last step is feedback matching, which reconstructs the joint configuration back to the visual cue space using the most probable configuration. Visual results of their work are presented in the literature. A sample of these results is provided in Figure 2.7, which contains an estimated pose below its corresponding image silhouette.

Agarwal and Triggs [2] proposed a tracking framework that does not assume an explicit body model and does not require the manual labelling of joint positions. Instead it uses sparse Bayesian non-linear regression of joint angles and a histogram of shape context descriptors to extract silhouette shapes. Pose data is created using regression on both

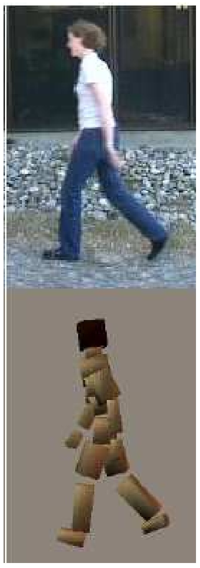


FIGURE 2.8: Pose estimation using a 3D POSER model by Agarwal and Triggs [2].

linear and kernel-based functions using either ridge regression (RR), RVMs or SVMs. The 3D modelling software, POSER, is used to render the data into a pose animation for a set of training and testing images. The test results indicate that the SVM classifier achieves higher accuracies than the RVM and RR classifiers. On average, the error in the estimated joint angle over all joints was found to be 5.91° for the SVM, 5.95° for the RR and 6.01° for the RVM. A sample of their visual results is illustrated in Figure 2.8.

Chen *et al.* [17] proposed a learning-based pose recognition and estimation system with an Implicit Shape Model-based human detector, proposed by Leibe [49]. The human detector identifies and divides a human silhouette in an image into segments using a segmentation mask and canny edge detection. The RR and RVM methods are used to train and test the data. The testing was performed using 50 frames of real human images and 50 frames of humanoid images generated by POSER. It is stated that 20% of poses are mis-estimated, although their criterion for determining a correctly estimated pose is not clear.

Grochow *et al.* proposed an inverse kinematics system that learns from previously seen poses [32]. An objective function is maximized based on the suitability of a pose. The Scaled Gaussian Process Latent Variable Model machine learning algorithm is used to represent the probability distribution across a wide variety of poses. An advantage of the system is that it can recognize unseen poses. However, poses similar to those in their training set are preferred. Visual results of their estimation technique are provided in the literature, illustrated in Figure 2.9.

Achmed [1] proposed a learning-based pose recognition and estimation system towards

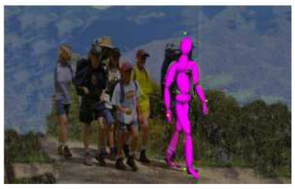


FIGURE 2.9: 3D humanoid superimposed on a walking human by Grochow *et al.* [32]

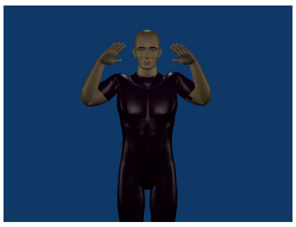


FIGURE 2.10: Upper body pose recognition and estimation by Achmed [1].

the sign language translation system of the SASL project. A novel skin detection algorithm was used. The Hue value of the HSV colour space was used to represent the skin as it is robust to dynamic lighting conditions. He proposed that the nose colour is closely representative of the average colour of skin on the human body. The pixels surrounding the nose are represented as a histogram. Various image processing techniques such a face detection, skin detection, background subtraction and morphological operations are used in the feature extraction process. The result is an image containing only the moving skin pixels of the arms. A SVM uses the pixel data obtained from the resulting image to determine the location of the wrists. The location of the wrists are mapped on to a 3D human model, developed by Van Wyk [100], using Blender. Chapter 5 explains Achmed's implementation in more detail. The system was trained and tested on 15 distinct SASL signs performed by six individuals. The system achieved an overall estimation accuracy of 88%. Figure 2.10 illustrates a sample 3D estimated sign.

2.4.1 Combining Model-Based and Learning-Based Approaches

Thayananthan *et al.* [95] used Tipping and Faul's [97] bottom-up approach with a sparse RVM regression classifier, similar to Sminchisescu *et al.* [89] and Agarwal and Triggs's work [2]. The system matches a set of image shape templates against the edge map of an input image. The results are mapped on to state space in a one-to-many configuration. The results are visually represented in Figure 2.11, illustrating an estimated pose below its corresponding input image. It is stated that the system achieves a high accuracy, but is computationally intensive.

Jaeggli *et al.* used Locally Linear Embedding (LLE) dimensionality reduction to model body poses in low-dimensional space [42]. A non-linear dynamic model is trained on

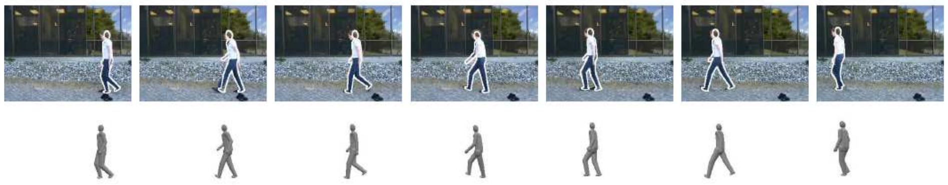


FIGURE 2.11: 3D estimation of a human walking by Thayananthan *et al.* [95]

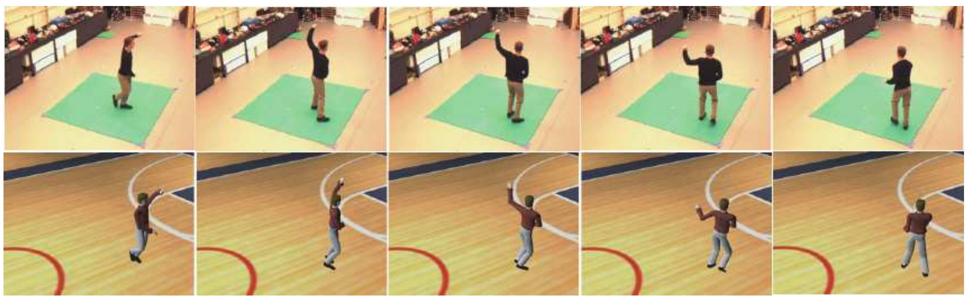


FIGURE 2.12: 3D model of a human dancing by Ren *et al.* [74]

possible body poses. The training set consisted of 4000 different images of people walking. The RVM regression classifier made use of a Gaussian kernel. Their estimation accuracy is not provided. It is stated that ground-truth results will be provided in the future.

Roberts *et al.* used probabilistic region templates to detect body parts [77]. The likelihood ratio is trained using the appearance distribution of the background and foreground. The result is compared to various dimensionalities by merging the top-down and bottom-up approaches. Their results are encouraging when visually inspected.

2.4.2 Combining Example-Based and Learning-Based Approaches

These two approaches are generally combined by storing exemplar images and training a model to efficiently search for a pose that is similar to the query image. Very little research has been conducted on this approach.

Ren *et al.* proposed a system that can produce a 3D model of a human dancing by selecting local features from 2D silhouette images to estimate the body configurations and yaw orientation of the user [74]. Haar-like features are used to compute feature vectors from silhouette images. The Haar-like features are trained on a set of hashing functions using AdaBoost. This allows for quick yaw estimation based on the silhouette, but it is limited due to its dependence on a domain-specific database. The system was tested using the top 20 matches in the hash. The visual results of the first five is illustrated in Figure 2.12.

2.5 Summary

This chapter presented a literature survey of pose recognition and estimation systems. The systems were categorized as using one of three approaches: model-based, example-based and learning-based approaches. The relative strengths and weaknesses of these three approaches were mentioned. Combined approaches were also discussed. Such approaches were combinations of two of the three approaches.

Achmed's work was also discussed. He produced both an example-based and learning-based system, specifically suited to recognizing and estimating SASL poses. The learning-based system achieved a much higher accuracy than the example-based system.

Chapter 3

Compute Unified Device Architecture

Real-time performance is crucial to the sign language translation system proposed by the SASL project due to the interactive nature of the system. The Graphics Processing Unit (GPU) is a parallel computing device designed for graphics rendering. However, it has evolved into a general-purpose processor capable of performing computations faster than a typical consumer Central Processing Unit (CPU) in the past decade [46]. This is attributed to the architecture of the GPU which contains hundreds of cores, capable of running millions of threads concurrently. This is far greater than the number of threads that a high-end CPU can initiate concurrently, typically between four and twelve.

This chapter begins by introducing the computing framework used in this research, called Compute Unified Device Architecture (CUDA); see Section 3.1. This is followed by a survey of studies which focus on investigating the use of CUDA to achieve increased processing speeds. The studies have been categorized according their relevance to this research in Section 3.2, which begins by discussing those studies that have investigated the use of CUDA to achieve increased processing speeds in pose recognition and estimation in Section 3.2.1. This is followed by less relevant studies involving other image processing applications in Section 3.2.2. Finally, of least relevance are studies involving general-purpose computation applications in Section 3.2.3. Where possible comparisons of the processing speed of CPU and GPU implementations are provided and discussed. The purpose of this survey is to determine whether the use of CUDA on the GPU can be used to achieve superior performance to that of CPU implementations.

The chapter is then concluded.

3.1 Compute Unified Device Architecture

Access to the GPU can be achieved using two types of frameworks, namely, computing frameworks and shading frameworks. Computing frameworks provide the ability to perform computations similar to a CPU, but with greater multi-threading capabilities on the GPU [46]. Shading frameworks make use of the shader units on the GPU to create and manipulate 3D environments for use in applications such as games [4].

Compute Unified Device Architecture (CUDA) is a propriety computing framework that is designed for NVIDIA GPUs [65]. It provides an interface for general-purpose computing on graphics processing units (GPGPU). The computations carried out by such applications are no different to those carried out on the CPU except that they are able to do so with greater multi-threading capabilities.

OpenCV is a state-of-the-art open source computer vision library [8, 12]. It provides many efficient image processing functions mainly aimed to achieve real-time computer vision. Originally OpenCV performed computations using the CPU, but has recently included support for the CUDA Application Programming Interface on the GPU. This has the potential to significantly speed up the image processing functions of OpenCV. The CUDA framework is adopted for use in this research.

The GPU hardware consists of a collection of multiprocessors. The multiprocessors execute a common program instruction on different data, which is known as the Single Instruction Multiple Data architecture [46, 80]. Each processor core contained in a multiprocessor communicates through the shared device memory. The software used by the CUDA programming model extends the C/C++ programming language.

The CUDA programmer controls the interface between the host code and the device code, which run on the CPU and GPU respectively. Host code should contain the sections of code that exhibit little or no data parallelism. The sections of code that exhibit a rich amount of data parallelism should be implemented in device code [64].

The device code is structured into kernels on the host illustrated in Figure 3.1. The host issues a kernel call to execute the device code that can be executed in a loop, be isolated as a function and work independently on different data. This results in the conversion of a sequential program to a data independent multi-threaded program in device memory.

These hardware thread contexts are grouped into warps and executed using the multiprocessors in a lock-step manner. Each warp contains 32 threads and is controlled at the hardware level. This limitation in the number of threads prevents the efficient use of the GPU in image processing and other GPGPU programs that require more threads.

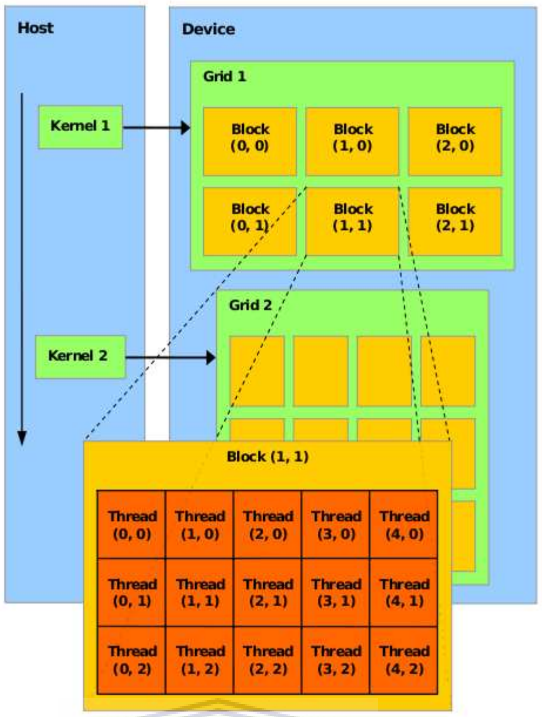


FIGURE 3.1: GPU Hardware Level [80]

Figure 3.1 illustrates the concept of blocks that can be used to overcome this limitation. Each block can group as many as 1024 threads. Similar to blocks, grids are used to increase the limit to several thousand threads. Multiple warps are assigned to each block or grid in a lock-step manner controlled by a near-zero overhead hardware scheduler to hide the memory latencies and pipeline stalls by intelligently switching between different warps [46, 80].

The result of the computation performed on the device in parallel is sent to the host memory.

3.2 Related Work

This section discusses studies which focus on investigating the use of CUDA to achieve increased processing speeds. They are categorized as investigating the effects of the GPU in the following applications: pose recognition and estimation applications, general image processing applications and general-purpose computation applications.

3.2.1 Pose Recognition and Estimation on the GPU

Bayazit *et al.* created a human gesture recognition system and used the CUDA framework to speed up the processing speed of the AdaBoost machine learning algorithm [9]. The system makes use of the optical flow algorithm, face detection algorithm and the AdaBoost algorithm. It extracts motion features from the optical flow estimates. Face detection is used to centre the user in each frame for normalization and the image is resized to 30×40 pixels before it is sent to the classifier. The optical flow algorithm is run in parallel with face detection on two separate CPU threads. Robust head movement is sacrificed by not synchronizing the threads as a trade-off for increased processing speed. The AdaBoost classifier uses a subset of the motion features in the resulting frame to classify the performed gestures on the GPU.

A comparison was carried out to determine the difference in processing speed of the system when running AdaBoost on the CPU and GPU. The test system had an Intel Xeon dual core CPU and an NVIDIA 9800 GX2 GPU. The data set that was used consisted of seven gestures, namely: punch-left, punch-right, sway, wave-left, wave-right, waves and idle. The time taken for AdaBoost to classify 8192 weak learners using the GPU implementation was, on average, 0.02 seconds while that of the CPU implementation was, on average, 0.09 seconds. The GPU implementation achieved an average speed up factor of 4.5. Although the CPU utilizes two cores when running the optical flow and face detection algorithms, it only runs AdaBoost on a single core. This places the CPU at a disadvantage in this comparison when it is considered that the GPU implementation of AdaBoost runs on multiple cores. It is stated that CUDA will be used to port the face detection and optical flow algorithm to the GPU in future. This will ensure that all of the algorithms run on the GPU.

Model-based pose recognition and estimation systems that run on the GPU can be implemented using a shader framework. Kyriazis *et al.* proposed such a system, with a focus on hand tracking. The system does not use the computational capabilities of GPUs, preferring to use the Direct3D shader framework [47].

The system uses multiple visual cues such as colour images and depth maps to track the hands. An input image is pre-processed with background subtraction and edge detection. A search is performed using the Particle Swarm Optimization (PSO) algorithm [101], which continuously updates to select the particle that exhibits the best position and velocity inside a swarm of particles. The MapReduce scheme proposed by two engineers at Google, Dean and Ghemawat [23], generates hypotheses based on the PSO, intermediate key/value pairs and a reduce function. The reduce function merges all the intermediate values associated with the same intermediate key. Feature mapping computes the occupancy of pixels from the position map, edges from the normal map and discrete layers from the depth map to produce a 3D model of the hand. For testing purposes, a CPU implementation was created and a comparison was performed between the GPU implementation and the CPU implementation. Testing was conducted using a 580 GTX GPU and an i7 950 CPU. It is stated that the GPU implementation achieved a 2–10 times faster processing speed than the CPU implementation when tracking the hands.

Park *et al.* used an example-based pose estimation algorithm that was designed specifically for the GPU using the CUDA framework [68]. Their system did not focus on estimating human poses. Instead it focused on poses of objects from any viewing angle. It uses a database of object poses, which are all matched against an input image using an error function. The error function uses distance transforms which measure the difference between the corresponding silhouette and the edges of the input image to the images in the database. The processing is performed using the CUDA framework on the GPU. The database of 2048 images consists of a collection of three different objects namely: pipe, bolt and elbow.

For testing purposes, 210 different images were chosen, consisting of 20 pipes, 30 bolts and 20 elbows. A CPU implementation was created and a comparison was performed between the GPU implementation and the CPU implementation. Testing was conducted using a 280 GTX GPU and an Intel i7 Q9550 quad core CPU. Results show an average accuracy of 96% and an average processing speed of 3 frames per second (FPS) for the GPU implementation. The same accuracy was achieved for the CPU implementation, but at a much slower average processing speed of 0.1 FPS. The GPU implementation therefore achieved a processing speed that was 30 times faster than the CPU implementation. This shows that a huge performance increase can be achieved using the GPU.

It should, however, be noted that the CPU implementation, in this case, was placed at a disadvantage as only one core on the CPU was utilized, but multiple cores were utilized on the GPU. It is stated that symmetry will be used to reduce pose search space in an effort to get speeds that are closer to real-time in the future.

3.2.2 Image Processing on the GPU

This subsection discusses the use of the GPU to increase the processing speed of general computer vision applications.

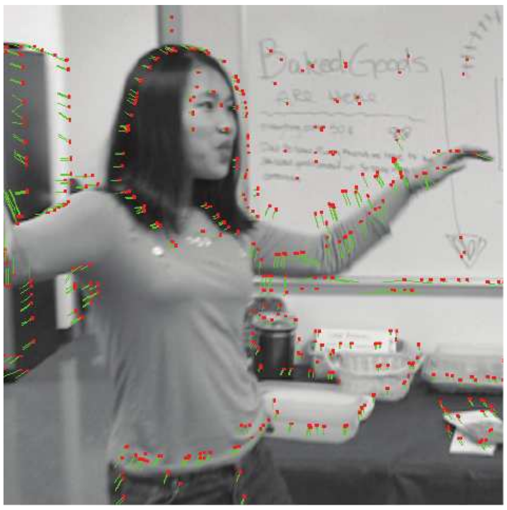


Figure 3.2: VCM algorithm applied to a video of a girl dancing by Huang *et al.* [39]

Huang *et al.* used the CUDA framework to implement Vector Coherence Mapping (VCM) on the GPU [39]. The VCM algorithm is used to extract motion fields from an input image. The algorithm is of a parallel nature. It is robust to noise and effective in gesture motion tracking. VCM is an excellent test candidate for determining the increased performance that the GPU offers in algorithms that exhibit code parallelism. The GPU implementation of the algorithm was compared to a CPU implementation of the system. The test system consisted of a single core Intel Pentium 4 CPU and an NVIDIA 8800 GTS GPU. The test input image sequence consisted of a human performing a dance. An unknown sequence of gestures performed during the dance were tracked using the VCM algorithm. The processed image is displayed in Figure 3.2, where the length of the green lines indicate the magnitude of motion and the red arrows indicate the direction of motion. The GPU implementation attained an average processing speed of 3 FPS. It is remarked that the processing speed of the CPU implementation was 41 times slower than the GPU implementation.

The CUDA framework was also used by Borovikov to produce a GPU-based image processing application that detects and tracks the pupil of the eye, which is a foundation to detect and track the limbus of the eye [11]. The GPU implementation used the CUDA framework and was compared to the CPU implementation which used the OpenCV libraries without CUDA support. The implementation makes use of a custom blob detector that is based on detecting circles using Hough transforms. A low-pass filter and an adaptive threshold are applied to the resulting image based on the one-dimensional Hue histogram, which contains the pixel data of the Hough circle. This is an iterative process. The radius of the detected circle decreases until it converges to the smallest Hough circle, assumed to be the pupil. Tests were conducted to measure the speed at which each iteration took place before convergence. The test system consisted of a single core Intel Xeon CPU and an NVIDIA 260 GTX GPU which were compared. It is stated that the GPU implementation was found to be 40 times faster than the CPU implementation.

Zhuge *et al.* proposed a GPU-accelerated version of fuzzy connected image segmentation [107]. Their research is used to solve the problem of carrying out fuzzy connected image segmentation on batches of radiology exams. This is a computationally intensive process. The Magnetic Resonance Imagery (MRI) scan and the Computed Tomography (CT) scan are examples of radiology exams that require such processing. A GPU-accelerated implementation that makes use of the CUDA framework was proposed in this work.

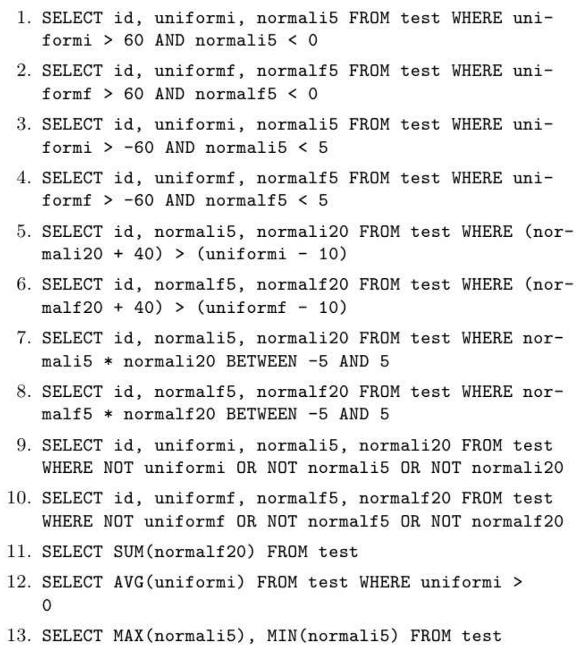
Fuzzy connected image segmentation computes fuzzy affinity relations and uses them to keep track of a fuzzy object. The fuzzy affinity relations contain voxel pairs for the tracking of the fuzzy objects using Dijkstra's shortest path algorithm [18]. A comparison in the time taken to process Computed Tomography (CT) scans was performed between the CPU implementation and GPU implementation. The performance of the CPU implementation was optimized by utilizing all the cores. The test system consisted of an Intel Xeon quad core CPU and three 580 GTX GPUs. The GPUs were combined for greater performance. The GPU implementation took 1.94 seconds and the CPU took 27.88 seconds to process the test set which contained an unknown number of CT images. The GPU achieved an approximate speed up factor of 14. It was stated that the speed up factor increases as the number of CT images increase.

3.2.3 General Purpose Computing on the GPU

GPGPU applications use the GPU to perform tasks that the CPU would otherwise perform. This can be used to improve the processing speed of general-purpose computer programs that exhibit code parallelism.

Bakkum and Skadron used the CUDA framework to accelerate the SELECT query in a Structured Query Language (SQL) database on the GPU [7]—SQLite. This was compared to the CPU implementation of the database. Both the CPU and GPU implementations load all data to be used into memory to prevent latencies associated with disk access. The CPU implementation is multi-threaded for improved performance. Both of these techniques are adopted for use in this research as they can potentially reduce hardware-related latencies.

In the GPU implementation the SELECT query was accelerated by assigning each row in a table to a thread. A limited number of aggregation functions, essential for performing a SELECT query on integer values, were implemented on the GPU. Examples are COUNT,



```
 1. SELECT id, uniformi, normali5 FROM test WHERE uni-
    formi > 60 AND normali5 < 0
 2. SELECT id, uniformf, normalf5 FROM test WHERE uni-
    formf > 60 AND normalf5 < 0
 3. SELECT id, uniformi, normali5 FROM test WHERE uni-
    formi > -60 AND normali5 < 5
 4. SELECT id, uniformf, normalf5 FROM test WHERE uni-
    formf > -60 AND normalf5 < 5
 5. SELECT id, normali5, normali20 FROM test WHERE (nor-
    mali20 + 40) > (uniformi - 10)
 6. SELECT id, normalf5, normalf20 FROM test WHERE (nor-
    malf20 + 40) > (uniformf - 10)
 7. SELECT id, normali5, normali20 FROM test WHERE nor-
    mali5 * normali20 BETWEEN -5 AND 5
 8. SELECT id, normalf5, normalf20 FROM test WHERE nor-
    malf5 * normalf20 BETWEEN -5 AND 5
 9. SELECT id, uniformi, normali5, normali20 FROM test
    WHERE NOT uniformi OR NOT normali5 OR NOT normali20
10. SELECT id, uniformf, normalf5, normalf20 FROM test
    WHERE NOT uniformf OR NOT normalf5 OR NOT normalf20
11. SELECT SUM(normalf20) FROM test
12. SELECT AVG(uniformi) FROM test WHERE uniformi >
    0
13. SELECT MAX(normali5), MIN(normali5) FROM test
```

FIGURE 3.3: 13 queries used in the testing procedure [7].

SUM, MIN, MAX and AVG. Similarly, a limited number of opcodes were implemented on the GPU, such as ADD, OR and BITAND. The CPU implementation used the standard SQLite application.

The test system consisted of an Intel Xeon X5550 quad core CPU and an NVIDIA Tesla C1060 GPU. The data set consisted of five million rows with an id column, three integer columns, and three floating point columns. The test set was generated with the GNU scientific library's random number generation functionality. A test was performed using the 13 queries depicted in Figure 3.3. The GPU implementation took an average of 0.045 seconds to perform the queries and the CPU took an average of 2.2737 seconds. This results in a processing speed that was 50 times faster on the GPU implementation. However, when taking the extra 0.018 seconds that the GPU needs to transfer the result back to the CPU into account, it achieves an average speed up factor of 36.

Rizk and Lavenier used the CUDA framework to increase the processing speed of the folding algorithm, which analyzes Ribonucleic Acid (RNA) and Deoxyribonucleic Acid (DNA) structures [76]. The computationally intensive algorithm uses dynamic programming to solve a function. The function recursively measures the energy of the structure according to its sequence, length and type.

A comparison was performed on a number of different CPUs and GPUs. The CPUs used were the Intel Xeon X5430 quad core, Core2 duo 6700 and Pentium 4 3 GHz edition, henceforth referred to as Xeon, C2 and P4 respectively. Additionally, two implementations of the Xeon were compared, one which used only a single thread and

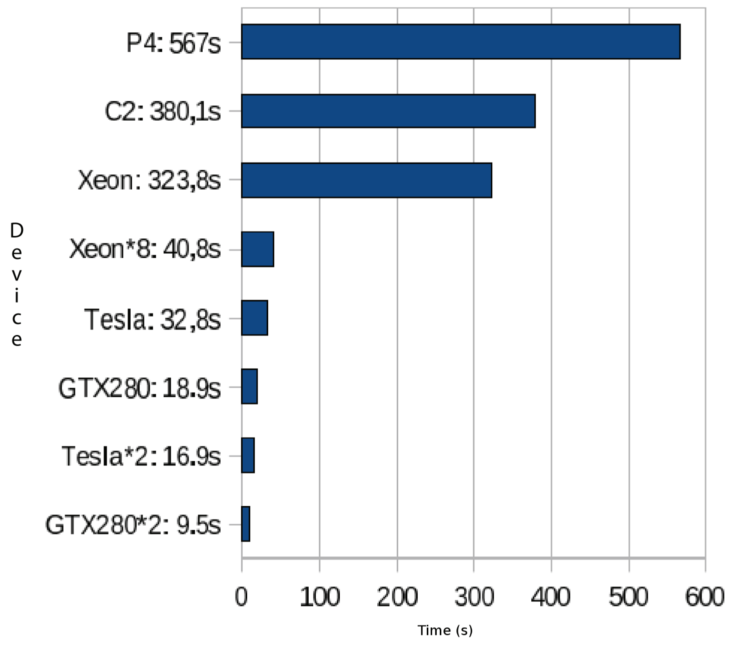


FIGURE 3.4: Comparisons of computation time of GPU and CPU implementations of the folding algorithm by Rizk and Lavenier [76].

the other which used 8 threads, denoted as Xeon*8. The GPUs used were the NVIDIA Tesla C870 and the NVIDIA GTX280, henceforth referred to as Tesla and GTX280 respectively. Two implementations of each of these GPUs were compared, one which used a single GPU and one that used two identical GPUs, denoted as Tesla*2 and GTX280*2. The total computation time in seconds of 40000 randomly generated RNA sequences of length 120 for each CPU and GPU is illustrated in Figure 3.4. It should be noted that a shorter computation time indicates a higher processing speed.

The fastest GPU implementation performs 4 times faster than the fastest CPU implementation. In fact, even the slowest GPU completes processing 8 seconds faster than the fastest CPU. It is clear from the results that the GPU implementations perform better than the CPU implementations. Furthermore, it should be noted that Xeon*8 achieved a significantly faster processing speed than Xeon, an approximate speed up factor of 8. This shows that multi-threading can be used to achieve significantly better processing speeds.

3.3 Conclusion

In this chapter, a background on the CUDA framework was discussed. This was followed by a survey of studies which focused on investigating the use of CUDA to achieve increased processing speeds. The studies were categorized according to their relevance to this research. Discussed were implementations involving: pose recognition and estimation, general image processing and general-purpose computation.

Of note is the discussion of Bakkum and Skadron's work which revealed the technique of loading all the data to be used into memory to reduce disk access. This technique is adopted in this research. Furthermore, Rizk and Lavenier's work revealed that multi-threading a system can lead to a significant increase in processing speed.

It is concluded that the CUDA framework on the GPU can be used to significantly increase the processing speed of computationally intensive applications. It is adopted for use in this research.

Chapter 4

Image Processing in Learning-Based Pose Recognition and Estimation Systems

This chapter discusses the components that form part of the learning-based approach used in the pose recognition and estimation methodology used in this research. The components of the feature extraction process are discussed in Section 4.1. These components include: face detection, skin detection, background subtraction and morphological operations. Once the relevant features have been acquired they are represented as vectors and used to train and test a SVM. Section 4.2 provides a detailed discussion on SVMs.

4.1 Feature Extraction Techniques

4.1.1 Face Detection

Face detection can serve as the foundation for image processing in learning-based systems for three reasons [9, 61]:

1. It identifies when an individual is present before the camera.
2. It makes it possible to normalize an image sequence by repositioning the individual such that he/she is in the centre of the frame at all times.
3. It is used as a reference point to find other points of interest on an individual's body.

A popular implementation of face detection uses the Viola-Jones object detection framework. The framework uses Haar classifiers to build a boosted rejection cascade of nodes to achieve a high positive detection rate [54]. To make this possible, a low rejection rate multi-tree classifier based on AdaBoost is used at every node in the cascade. Their framework can be described in four stages [12]:

1. The computation of Haar-like wavelet features used as input.
2. The computation of an integral image to accelerate the computation of Haar-like wavelets.
3. The use of a statistical boosting algorithm based on AdaBoost that characterizes nodes.
4. The organization of weak classifier nodes as a rejection cascade.

The following subsections discuss these four stages.

4.1.1.1 Haar-like Wavelets

Haar-like wavelets are single wavelength square waves that have one high and one low interval [103]. They consist of pairs of rectangles that have identical size and shape, are either light or dark, and are either vertically or horizontally adjacent. Haar-like wavelets consist of three types of features: a two-rectangle feature, a three-rectangle feature or a four-rectangle feature. The features used specifically for the face detection method are illustrated in Figure 4.1.

The two-rectangle features, depicted in blocks A and B of Figure 4.1, are determined by taking the difference between the sum of the pixels within each of the two rectangular regions in each case. The three-rectangle feature, depicted in block C of Figure 4.1, is computed by summing the pixels within the two outside rectangular regions and subtracting it from the sum of the pixels in the centre rectangular region. The four-rectangle feature—block D of Figure 4.1—is computed by taking the difference between the diagonal pairs of rectangles. In each case a threshold is applied to the result. The thresholded result indicates whether or not each feature is present.

Haar-like wavelet features are computed at multiple image locations and scales. Integral image representation is used to efficiently compute these features.

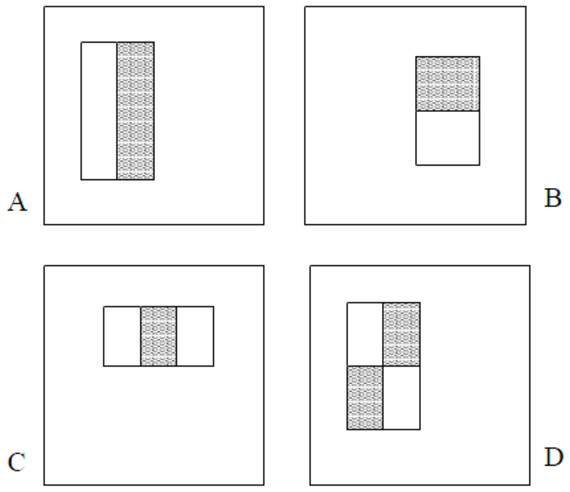


FIGURE 4.1: The Haar-like wavelet features used in the Viola-Jones face detection method [103].

4.1.1.2 Integral Image

Integral images are used to efficiently determine the presence or absence of hundreds of Haar-like wavelet features at every image location and at several scales. The original image is converted to an integral image by taking the sum of all the pixels to the left and above a corresponding pixel. Starting at the top left pixel of image I, proceeding row by row, each integral pixel value $I'(x,y)$ in the integral image I' is computed recursively by the formula [103]:

$$I'(x,y) = I(x,y) + I'(x-1,y) + I'(x,y-1) - I'(x-1,-1) \tag{4.1}$$

4.1.1.3 AdaBoost

The Viola and Jones [12, 103] face detection method uses a modified AdaBoost algorithm that selects a small set of features and trains the classifier. This learning method creates a strong classifier by combining many weak classifiers. A weak classifier recognizes more features than it rejects. Weights are assigned to each weak classifier, a process known as boosting. The best weak classifier is selected at each boosting interval. The result is a strong classifier consisting of a weighted combination of classifiers. Figure 4.2 illustrates an example of the features selected by AdaBoost.

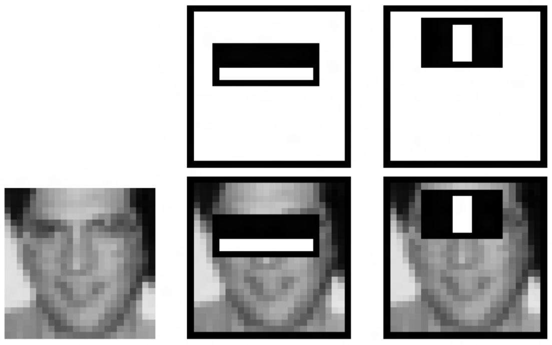


FIGURE 4.2: Features selected by AdaBoost for face detection [12].

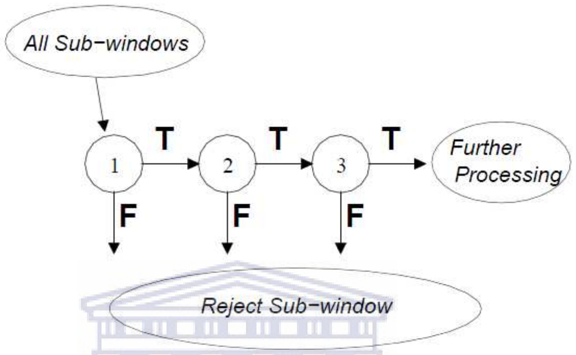


FIGURE 4.3: The detection process for rejecting regions in the image [12].

4.1.1.4 A Rejection Cascade of Weak Classifier Nodes

The Viola and Jones face detection method organizes weak classifiers in a cascade structure. A cascade significantly increases the processing speed of the face detection method by quickly eliminating background regions and focusing on promising regions in the image. The promising regions appear to be object-like and are set aside for further processing. These regions are selected such that the heavier weighted classifiers are selected first in the cascade for a faster elimination process. The process of elimination has the structure of a degenerate decision tree.

The initially selected classifier—the classifier with the heaviest weight—is applied to the promising regions in the image. If the classifier returns a positive result, it indicates that a possible face has been detected. The process is repeated on a sequence of increasingly complex classifiers. Processing on the region ends immediately when a negative result is obtained. A face exists in the image region when a positive result is obtained through all the classifiers. The face detection process is illustrated in Figure 4.3.

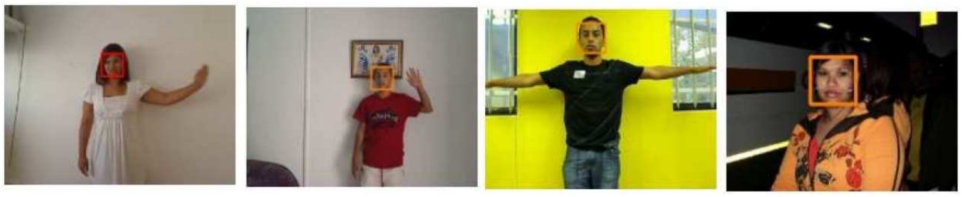


FIGURE 4.4: Examples of true positives for Viola-Jones face detection on a random test set [1].

4.1.1.5 Testing and Results on the Face Detection Method

The face detection accuracy of the Viola-Jones algorithm was evaluated by Achmed using a frontal face test set consisting of 1047 images randomly selected from the Internet [1]. The images in the test set have varying background complexities and camera properties. An accuracy of 88.9% was obtained. Some of the results are illustrated in Figure 4.4. The results show a high accuracy, which is especially desirable when using face detection as the foundation of the feature extraction process.

4.1.2 Skin Detection

Skin detection identifies the pixels in an image as either skin or non-skin pixels [13]. This process is the basis of a number of applications involving the detection of the human body and is especially useful in hand detection and tracking [37, 51]. It is robust to partial occlusions, rotations and the scaling of body parts [44]. The fact that most skin tones are distinct from the colours of most other objects can be used to help detect and track specific body parts [62]. However, detecting skin pixels can be non-trivial depending on factors such as illumination, the viewing angle and various camera properties. Creating a skin filter consists of the following three steps [44, 62, 102]:

1. An appropriate colour space needs to be selected to represent the pixels in the image.
2. A suitable classification algorithm needs to be used to model skin pixels.
3. Each pixel needs to be classified as either being a skin or non-skin pixel.

Colour spaces are used to mathematically represent colours in various ways [105]. There are a wide variety of colour spaces, but many have common properties. For this reason only the most widely used colour spaces will be discussed.

4.1.2.1 RGB Colour Space

RGB stands for Red-Green-Blue and is the default colour space used in computer graphics [102]. It uses a combination of red, green and blue pixel values to represent the colour of a single pixel. Other colour spaces are obtained by performing a linear or non-linear transformation on the RGB colour space. The transformation can be visualized as a cube consisting of red, green and blue on the three perpendicular axes, respectively.

The RGB colour space is simple to use, but it is not perceptually uniform. This means that the colours that humans perceive do not correspond to the actual colour value [102, 105]. Furthermore, the red, green and blue channels are highly connected, and luminance and chrominance data is not separated. Colour-based recognition algorithms are not likely to be as robust when using this colour space.

4.1.2.2 Normalized RGB Colour Space

The normalized RGB colour space is obtained by applying the following normalization formula to the default RGB colour space:

$$r = \frac{R}{R + G + B}, \quad g = \frac{R}{R + G + B}, \quad b = \frac{R}{R + G + B} \tag{4.2}$$

where r, g and b are the normalized red, green and blue pixel values, respectively, and R, G and B are the red, green and blue pixel values from the RGB colour space, respectively. It should be noted that the sum of the normalized pixel values is 1, as follows:

$$r + g + b = 1 \tag{4.3}$$

Since the sum is a constant, the third component can be omitted as it does not hold significant information. This reduces the space dimensionality [44, 102]. The remaining components, r and g, are less sensitive to lighting changes in the normalized RGB colour space.

4.1.2.3 HSV Colour Space

HSV stands for Hue-Saturation-Value and is also known as HSI and HSL. The HSV colour space has proven to be reliable for skin detection and is based on human colour perception [45, 87, 105]. It describes colour in terms of Hue, Saturation and Value. The Hue component defines the dominant colour of an area and the Saturation component measures the amount of dominant colour of an area in proportion to its illumination. The Value component stores the brightness information of a colour. A non-linear transformation is performed to map the RGB colour space to the HSV colour space. It is formulated as follows [102]:

$$V = max_{r,g,b} \tag{4.4a}$$

$$S = \frac{max_{r,g,b} \; - \; min_{r,g,b}}{V} \tag{4.4b}$$

$$H = \begin{cases} \frac{g-b}{6(max_{r,g,b} - min_{r,g,b})}, & \text{if } V = r \\ \frac{2-r+b}{6(max_{r,g,b} - min_{r,g,b})}, & \text{if } V = g \\ \frac{4-g+r}{6(max_{r,g,b} - min_{r,g,b})}, & \text{if } V = b \end{cases} \tag{4.4c}$$

where H, S and V are the Hue, Saturation and Value components, respectively, r, g and b are the normalized red, green and blue pixel values, respectively, and $max_{r,g,b}$ and $min_{r,g,b}$ are the maximum and minimum between the normalized red, green and blue pixel values, respectively. An interesting property of the Hue component is that it is unaffected by illumination changes [88, 105].

4.1.2.4 YCbCr colour space

The YCbCr colour space is commonly used by European television studios as well as in video and image compression schemes such as MPEG and JPEG [102]. A linear transformation is used on the RGB colour space to obtain the YCbCr colour space. The luminance component Y is the colour value and is computed by taking the weighted sum of the RGB pixel values. The chrominance components known as Cr and Cb are computed by subtracting the luminance component from the red and blue pixel values. It is formulated as follows:

$$Y = 0.299R + 0.587G + 0.114B \tag{4.5a}$$

$$Cr = R - Y \tag{4.5b}$$

$$Cb = B - Y \tag{4.5c}$$

where Y represents the luminance component, Cr, Cb represent the chrominance components and R, G and B are the red, green and blue pixel values from the RGB colour space, respectively. This colour space is also suitable for skin detection as it separates luminance and chrominance components.

4.1.2.5 TSL Colour Space

TSL stands for Tint-Saturation-Lightness [44]. It is a colour space that is obtained by performing a transformation on the normalized RGB colour space. If $r' = r - \frac{1}{3}, g' = g - \frac{1}{3}$, the TSL colour space can be formulated as follows [102]:

$$T = \begin{cases} \frac{\arctan(\frac{r'}{g'})}{2\pi} + \frac{1}{4}, & \text{if } g' > 0 \\ \frac{\arctan(\frac{r'}{g'})}{2\pi} + \frac{3}{4}, & \text{if } g' < 0 \\ 0, & \text{if } g' = 0 \end{cases} \tag{4.6a}$$

$$S = \sqrt{\frac{9(r'^2 + g'^2}{5})} \tag{4.6b}$$

$$L = 0.299R + 0.587G + 0.114B \tag{4.6c}$$

where T, S and L are the Tint, Saturation and Lightness pixel values and r' and g' are variants of the normalized red and green pixel values given by:

$$r' = r - \frac{1}{3}, g' = g - \frac{1}{3} \tag{4.7}$$

4.1.2.6 An appropriate colour space for skin detection?

Two factors are taken into consideration when selecting a colour space suitable for skin detection [44, 85, 102, 105]:

- The colour space must aid the separation of skin and non-skin pixel values.
- It must also address the problem of dynamic lighting conditions that interfere with the colour distribution.

Four studies have been performed by researchers to investigate the effectiveness of various colour spaces in skin detection algorithms [44, 85, 102, 105]. Kakumanu *et al.* [44] and Shin *et al.* [85] concluded that there was no significant improvement to the skin detection process when using a non-RGB colour space as compared to using the RGB colour space. Furthermore, they concluded that eliminating the brightness component does not improve the discrimination between skin and non-skin pixels. However, they suggest that the training data used in the classification process may benefit from the elimination of the brightness component.

On the other hand, Zarit *et al.* [105] and Vezhnevets *et al.* [102] suggest that an appropriate colour space should be chosen based on the input format of the image as well as whether the post-processing steps require a specific colour space. They also state that the HSV colour space has been proven to aid the skin detection process.

It is therefore unclear whether a non-RGB colour space should be used in skin detection. Many researchers choose a particular colour space without justifying their choice. However, many researchers [20, 21, 25, 88] agree with Forsyth and Fleck [28] that the Hue component in the HSV colour space has a colour range that effectively represents any human skin colour. Human skin colour is formed by the combination of haemoglobin, carotene and melanin [98]. Haemoglobin carries the oxygen in the red blood cells and forms a pink-red colour in the skin. Carotene is mostly found in the palms and soles with a vivid yellow-orange colour. Melanin is the primary factor of skin colour. There are two types of melanin, namely, pheomelanin, which is red and eumelanin, which is dark brown. The Hue component in the HSV colour space represents the combination of these colours very well [20, 21, 25, 88]. Determining the optimal colour space is beyond the scope of this research, but based on the above research, it can be concluded that the Hue component can effectively represent the skin colour of every race and skin tone.

4.1.2.7 Skin Model

Many researchers make use of a static skin model. This technique fails to identify skin pixels when applied to diverse skin tones [3, 69]. Studies show that the skin diversity in South Africa and other sub-Saharan African countries is the highest in the world [73]. An adaptive skin model is therefore necessary.

Achmed proposed a dynamic solution that uses an individual's nose colour to identify and continuously update the skin colour distribution of that particular individual [1]. A 10×10 pixel area at the centre of the nose is used because it is typically void of non-skin pixels, such as shadows, eyes or spectacles, that are present on the face. The Hue values of the area around the centre of the nose are represented as a histogram, which functions as a look-up table for skin pixel values.

The histogram groups these pixel values, also known as data points, into bins. The bin width determines the number of data points that are assigned per bin. For example, if

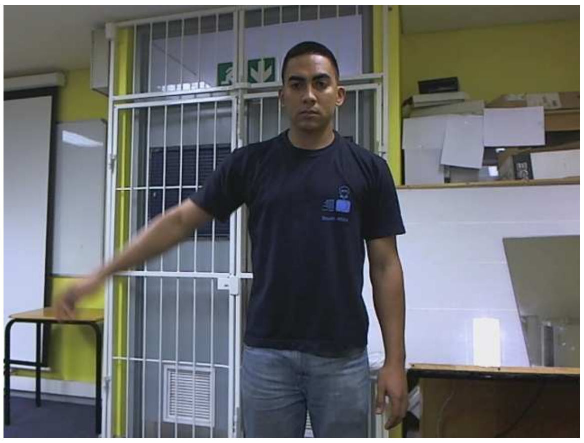


FIGURE 4.5: Input Image.

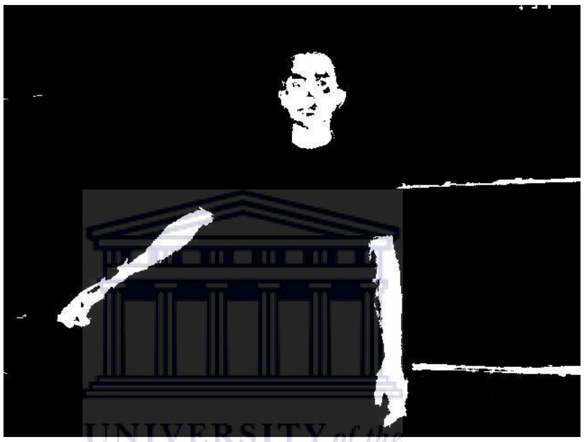


FIGURE 4.6: Skin Image.

the bin width is 8, then the first bin will contain the pixel values that fall into the range 0–7, the second bin will contain pixel values that fall into the range 8–15 and so on.

The Hue histogram is back-projected on to the original image to form a new greyscale image. The greyscale image consists of intensity values ranging from 0 to 255. The value 255 indicates the highest likelihood ratio of skin colour while a value of 0 indicates the highest likelihood ratio that the pixel is of non-skin colour in the histogram. A pre-determined threshold is used to binarize the image into skin and non-skin classes. Achmed [1] and Li [51] determined that a threshold of 60 is satisfactory.

Figure 4.6 is the result of applying the histogram back-projection method to the image depicted in Figure 4.5. Noise pixels can be observed in Figure 4.6 in the form of the two horizontal parallel lines on the right side of the figure. This is caused by certain objects, such as furniture and leather, being falsely identified as skin. This problem can be addressed by combining the skin image with a background subtracted image.

4.1.3 Background Subtraction

Background subtraction separates the background from the foreground in a sequence of images [50]. In this research the foreground image consists of the arms of the person standing in front of the camera performing sign language. The background subtraction algorithm should satisfy the following three conditions in order to effectively obtain the objects of interest [81]:

1. It should not fail under dynamic lighting conditions.
2. Moving background objects such as tree leaves, rain or snow should not be detected as part of the foreground image.
3. It should be robust to sudden changes in the scene.

The subsections below discuss well-known background subtraction techniques.

4.1.3.1 Static Background Subtraction

Static background subtraction uses a static reference image as the base of subtraction, commonly the first frame in the sequence. Each frame in the image sequence is subtracted from the reference image. A threshold is applied to the result to separate the background from the foreground. This process can be described as a binary classification technique where each pixel in the current image either belongs to the background or foreground class [24]. For each pixel $I(i,j)$ at position (i,j) in the current image I, label l is assigned to the pixel where $l \in \{background, foreground\}$. Label $foreground$ is assigned to the pixel if the following equation is satisfied:

$$I(i,j) - R(i,j) \ > \ T_h \tag{4.8}$$

where $R(i,j)$ is the pixel at position (i,j) in the reference image and T_h is a threshold determined empirically [51]. Label *background* is assigned if the following equation is satisfied:

$$I(i,j) - R(i,j) \ \leq \ T_h \tag{4.9}$$

The object of interest is highlighted using this mask, with background pixels set to black. This background subtraction technique is applied to an individual moving his right arm in Figure 4.5. The resulting image is illustrated in Figure 4.7.

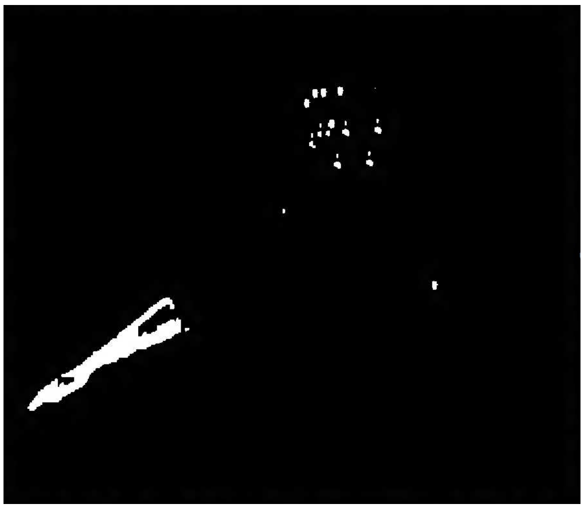


FIGURE 4.7: Result of static background subtraction.

4.1.3.2 Frame Differencing

Frame differencing is similar to static background subtraction. The key difference is that the reference image is continuously updated throughout the image sequence. The image that precedes the current image in the image sequence is commonly used as the reference image. A pixel $I_2(i, j)$ at position (i, j) in the current image I_2 is labeled as *foreground* when the following equation is satisfied:

$$| I_2(i, j) - I_1(i, j) | > T_h \tag{4.10}$$

where $I_1(i, j)$ is the pixel at position (i, j) in the dynamic reference image I_1 and T_h is a threshold determined empirically [51].

4.1.3.3 Gaussian Mixture Models

Gaussian Mixture Models (GMMs) model the background pixels as a mixture of adaptive Gaussians [91]. The history of a pixel (i, j), at time t, across image sequence I can be formulated as:

$$\{I_1, \ldots, I_t\} = \{I(i, j, x) : 1 \leq x \leq t\} \tag{4.11}$$

Given k Gaussian distributions, each pixel can be modelled by a mixture of these distributions. The probability that a pixel may have a value I_t at time t can be evaluated using the following formula:

$$P(I_t) = \sum_{x=1}^{k} W_{x,t} \times \eta(I_t, \mu_{x,t}, \Sigma_{x,t}) \tag{4.12}$$

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

$$\eta(I_t, \mu_{x,t}, \Sigma_{x,t}) = \frac{1}{(2\pi)^{\frac{n}{2}} \mid \Sigma_{x,t} \mid^{\frac{1}{2}}} e^{\frac{-1}{2}(I_t-\mu_{x,t})^T \Sigma_{x,t}^{-1}(I_t-\mu_{x,t})} \tag{4.13}$$

where $\mu_{x,t}$ is the mean and $\Sigma_{k,t} = \sigma_{k,t}^2 \mathbf{I}$ is the covariance of the k-th Gaussian component and $\mathbf{I}$ is the identity matrix. The number of distributions, k, are ordered based on the fitness value $\frac{W_{x,t}}{\sigma_{x,t}}$ and the background of the scene is modelled using the first M distributions where M is estimated as:

$$M = \text{argmin}_m (\sum_{x}^{m} W_{x,t} > T_h) \tag{4.14}$$

where T_h is the threshold, which is the minimum segment of the background model. After the background has been updated, the foreground is detected by labelling any pixel found to be more than 2.5 standard deviations away from any one of the M distributions as foreground. If the test value matches the x-th Gaussian component, then it is updated as follows:

$$W_{x,t} = W_{x,t-1} \tag{4.15a}$$

$$\mu_{x,t} = (1-\rho)\mu_{x,t-1} + \rho I_t \tag{4.15b}$$

$$\sigma_{x,t}^2 = (1-\rho)\sigma_{x,t-1}^2 + \rho(I_t - \mu_{x,t})^T(I_t - \mu_{x,t}) \tag{4.15c}$$

$$\rho = \alpha\eta(I_t \mid \mu_k, \Sigma_k) \tag{4.15d}$$

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

$$W_{x,t} = (1-\alpha)W_{x,t-1} \tag{4.16a}$$

$$\mu_{x,t} = \mu_{x,t-1} \tag{4.16b}$$

$$\sigma^2_{x,t} = \sigma^2_{x,t-1} \tag{4.16c}$$

If none of the components match the test value, then the component with the lowest probability is replaced by a new one with a low weight parameter, a high variance and the current value as its mean. When the Gaussian distributions are evaluated, pixels that do not match are classified as foreground and grouped using 2D connected component analysis.

4.1.3.4 A comparison of the Background Subtraction Techniques

The different background subtraction techniques have relative strengths and weaknesses. Selecting a suitable technique depends on its effectiveness towards a particular application. Comparisons of the accuracy of different background subtraction techniques are not common, especially on a per-application basis. It is currently not possible to compare the accuracy of the different techniques as there is presently no work on unbiased background subtraction benchmarks on different applications. However, the strengths and weaknesses of each technique have been highlighted [70, 91] and are summarized in Table 4.1.

Static background subtraction has a high processing speed, but is severely affected by a non-static background. It provides information about the location of motion. GMMs have a low processing speed compared to the other two techniques, but they adapt to a changing background and provide more information about the objects in the scene, that is, both the location and intensity of motion. Frame differencing operates at a high speed and is robust to a non-static background. It, however, provides less information about the objects in the scene, that is, the location of motion only.

These strengths and weaknesses can be used to select a suitable technique depending on the specific application.

Technique	Strengths	Weaknesses
Static Background Subtraction	• Simplicity • High processing speed	• Accuracy is affected by the object of interest's speed and the frame rate • Can be affected by a slower moving object in a scene that is not the object of interest • Background model does not update
Frame Differencing	• Simplicity • High processing speed • Background model can update at every frame	• Accuracy is affected by the object of interest's speed and the frame rate • Can be affected by a slower moving object in a scene that is not the object of interest
GMMs	• Rich source of information • The threshold for each pixel adapts with respect to time • Additional objects can merge with the existing background model with time	• Slower processing speed • Fails when lighting suddenly changes • Accuracy and background adaptation is highly dependent on the selected parameters

Table 4.1: A comparison of the background subtraction techniques.

4.1.4 Morphological Operations

The morphological operations used in image processing are based on mathematical morphology that uses a non-linear approach to image enhancement based on set theory and the geometry found within images. A structuring element is used as input to the relevant operation which is applied to a binary image that requires enhancement. The

0	1	0
1	1	1
0	1	0

FIGURE 4.8: 3×3 structuring element [3]

structuring element is an image processing element that is specified by a pattern of elements relative to an origin at the centre pixel [3, 12]. A 3×3 pixel structuring element is depicted in Figure 4.8.

The structuring element is scanned over the pixels in an image and its elements are compared to the pixel values in a way that is similar to a mask. Each operation has a defined set of rules applied to achieve a desired enhancement. These rules are applied based on the comparison of the values of the structuring element and the image pixels. Two basic and useful morphological operations are erosion and dilation. Opening and closing are two additional morphological operations that are derived from erosion and dilation. Other morphological operations exist, such as thinning, thickening and medial axis transform, but are not dealt with because they are not used in this research. The interested reader is referred to [27] for a further reading on these operations. The following subsections discuss the erosion, dilation, opening and closing operations.

4.1.4.1 Erosion

Erosion eliminates boundary image regions [3, 12]. It causes image regions to shrink and black pixel regions to grow. It can be used to remove small isolated noise regions in an image. When each pixel in the structuring element corresponds to a foreground pixel in the image, these pixels remain foreground pixels, otherwise they are set to background pixels. Erosion of a binary image by a structuring element in set A and set B is represented by the symbol $\ominus$ and can be defined as:

$$A \ominus B = \{x \mid (B)_x \subseteq A\} \tag{4.17}$$

where B_x is the set B translated by the vector $\boldsymbol{x}$.

4.1.4.2 Dilation

Dilation expands boundary image regions [3, 12]. It causes image regions to grow and black pixel regions to shrink. It can be used to fix regions in an image that lack pixel continuity. It is the exact opposite of erosion. This means that when dilation is applied to an image, it will have the same result as erosion on the inverse of that image. When each pixel in the structuring element corresponds to a background pixel in the image, these pixels remain background pixels, otherwise they are set to a foreground pixels. Dilation of a binary image by a structuring element in set A and set B is represented by the symbol $\oplus$ and can be defined as:

$$A \oplus B = \bigcup_{b \in B} A_b = \{x \mid (B^s)_x \cap A \neq 0\} \tag{4.18}$$

where B^s denotes the reflection of the set B and $(B^s)_x$ is B^s translated by the vector $\boldsymbol{x}$.

4.1.4.3 Opening

Opening is a morphological operation that is the application of erosion followed by dilation on an image [3, 12]. This operation is used to smooth the contours of objects, reduce fine noise and enhance the features in an image. Opening of a binary image by a structuring element in set A and set B is represented by the symbol $\circ$ and can be defined as:

$$A \circ B = (A \ominus B) \oplus B \tag{4.19}$$

4.1.4.4 Closing

Closing is a morphological operation that is the application of dilation followed by erosion on an image [3, 12]. This process fills large background regions surrounded by foreground pixels—features resembling large holes—in the image but can, in turn, introduce additional noise into the image. Closing of a binary image by a structuring element in set A and set B is represented by the symbol $\bullet$ and can be defined as:

$$A \bullet B = (A \oplus B) \ominus B \tag{4.20}$$

A property that closing and opening share is that both operations are idempotent, which means that repeating the operations on an image has no effect on the image and contributes to unnecessary computation time [3, 52].

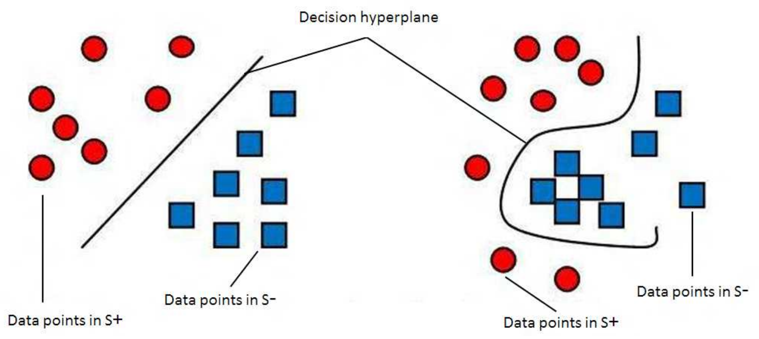


FIGURE 4.9: (*a*) Linear Classification. (*b*) Non-Linear Classification [63].

4.2 Support Vector Machines

Support Vector Machines (SVMs) have been used extensively in pattern recognition problems [63, 104]. An SVM is a machine learning tool that is derived from statistical learning theory that classifies data into one of two classes. Its classification mechanism has been extended to support multiclass problems.

SVMs offer several advantages over other classifiers [104]. One significant advantage is that the training time is not affected by the high dimensionality of feature vectors from large images. Another advantage is the power and flexibility provided by kernel functions. It is possible to change from the default linear kernel to the other alternative kernels such as the radial basis function, polynomial, sigmoid or other newer kernels. The use of alternative kernels can help spread out data points in the training model more evenly. Furthermore, kernels make it possible to use linear classification techniques to solve non-linear classification problems.

SVMs aim to maximize a mathematical function given a collection of data points [63]. Data points that consist of two classes can be separated by finding a boundary that separates those two classes. Consider a set S of M training points expressed as $S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_M, y_M)\}$. Letting $i \in \{1, 2, \ldots, M\}$, each x_i is a data point in R^n and each $y_i \in \{-1, 1\}$ is the label that corresponds to the data points, divided into a positive and a negative class.

Consider that the two classes $S^+ = \{x_i \mid y_i = 1\}$ and $S^- = \{x_i \mid y_i = -1\}$ are linearly separable in R^n. This results in at least one boundary that can be formed between them [63]. This boundary is referred to as the decision boundary, which is determined by training the SVM and is illustrated in Figure 4.9(a).

In a higher-dimensional space, the decision boundary takes the form of a plane, illustrated in Figure 4.10. It is referred to as the decision hyperplane and defined by the following equation:

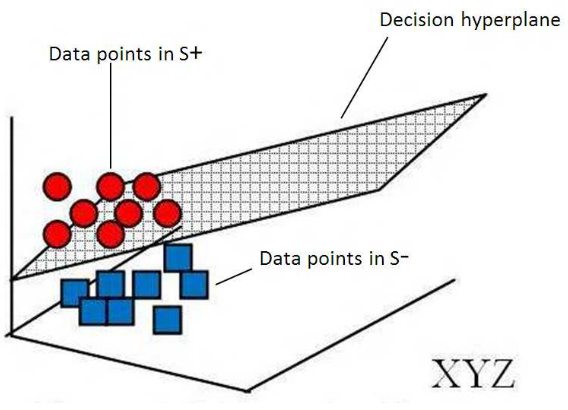


Figure 4.10: Linear classification using a hyperplane.

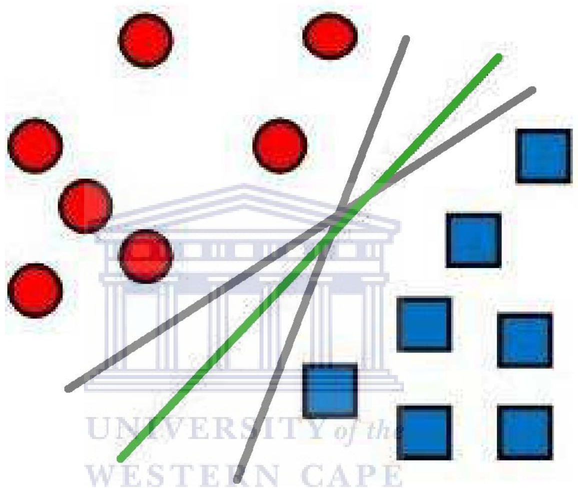


Figure 4.11: Various decision boundaries on the data set.

$$f(x) = \boldsymbol{w} \cdot x + b = 0; \boldsymbol{w} \in \mathbb{R}^n, b \in \mathbb{R} \tag{4.21}$$

where $\boldsymbol{w}$ is the normal vector and b is the interim term. Vector $\boldsymbol{w}$ of the decision hyperplane is defined as a linear combination of x_i with weights α_i as follows:

$$\boldsymbol{w} = \sum_{i=1}^{M} \alpha_{\alpha i} x_i y_i \tag{4.22}$$

As illustrated in Figure 4.11, many decision boundaries can be drawn to separate the data set. However, only one solution, the green line in Figure 4.11, achieves maximum separation between sets S^+ and S^-. SVMs aim to determine this solution, known as the optimal hyperplane. Using this hyperplane allows an SVM to classify new data

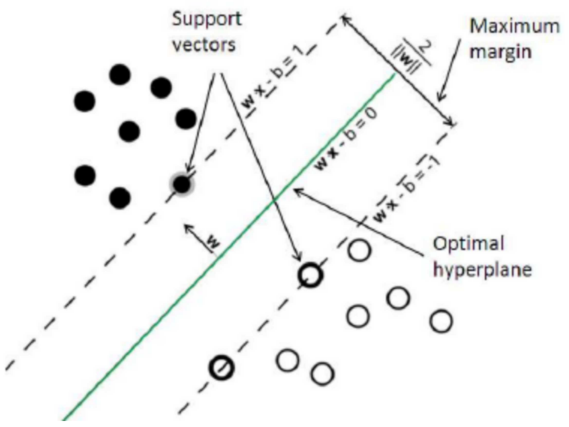


FIGURE 4.12: Optimal hyperplane and maximum margin [63].

points more accurately. The optimal hyperplane passes through the mid-point of sets S^+ and S^- and ensures that the distance between the two sets, known as the margin, is maximized.

The data points of sets S^+ and S^- that are located on the boundaries of the margin are known as the support vectors. A simple rescale of $\boldsymbol{w}$ for all x_i that are support vectors holds that:

$$\boldsymbol{w} \cdot x_i + b = 1 \tag{4.23a}$$

$$\boldsymbol{w} \cdot x_i + b = -1 \tag{4.23b}$$

The distance d between the decision boundary and the margin can be expressed as:

$$d = \frac{2}{|| \boldsymbol{w} ||} \tag{4.24}$$

The optimal hyperplane has the following two features: it clearly separates the data points of sets S^+ and S^-; and it achieves the maximum distance to the nearest data point from both classes. The first feature dictates that all data points should be classified correctly [99]. Hence, the parameters $\boldsymbol{w}$ and b of the hyperplane are to be estimated such that:

$$y_i(\boldsymbol{w} \cdot x_i + b) \geq 1 \; for \; y_i = 1 \tag{4.25}$$

and

$$y_i(\boldsymbol{w} \cdot x_i + b) \leq 1 \; for \; y_i = -1 \tag{4.26}$$

These two equations can be combined to give:

$$y_i(\boldsymbol{w} \cdot x_i + b) - 1 \geq 0,\ \forall\, i = 0, 1, 2, \cdots, N \tag{4.27}$$

The second feature dictates that the margin should be as large as possible. Maximizing the distance equation is the same as minimizing $\frac{||\boldsymbol{w}||}{2}$. Therefore, $f(\boldsymbol{w}) = \frac{1}{2} \mid\mid \boldsymbol{w} \mid\mid^2$ should be minimized. Following this, the optimal hyperplane can be found by solving the optimization problem defined as:

$$\text{Minimize } \frac{1}{2} \mid\mid \boldsymbol{w} \mid\mid^2 \tag{4.28}$$

subject to

$$y_i(\boldsymbol{w} \cdot x_i + b) - 1 \geq 0, \forall\, i = 0, 1, 2, \cdots, N \tag{4.29}$$

This problem can be solved, given the Lagrange multipliers $\alpha_1, \alpha_2, \cdots, \alpha_N \geq 0$ and the saddle point of the Lagrange function:

$$L(\boldsymbol{w}, b, \alpha) = \frac{1}{2} \mid\mid \boldsymbol{w} \mid\mid^2 - \sum_{i=1}^{N} \alpha_i(y_i(\boldsymbol{w} \cdot x_i + b) - 1) \tag{4.30}$$

Therefore, using the Lagrange function, the optimization problem can be expressed as:

$$\text{Maximize } \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_i (x_i, x_j) \tag{4.31}$$

subject to

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \textit{ and } \alpha \geq 0, i = 0, 1, 2, \cdots, N \tag{4.32}$$

The optimal hyperplane discriminant function under this formulation is:

$$f(x) = \sum_{i \in S} \alpha_i y_i (x_i x) + b \tag{4.33}$$

where S is the subset of support vectors corresponding to positive Lagrange multipliers.

Non-linear problems require more complex structures to find a hyperplane. In the case of Figure 4.9 (b) the data points are unevenly distributed and non-separable compared to those in Figure 4.9 (a). These are the cases where classes are not linearly separable and the constraint of equation 4.27 cannot be satisfied. A solution to these cases is a cost function that combines the margin maximization and minimization or error criteria.

This is achieved by using slack variables ξ_i which measure the degree of misclassification of the data x_i. The cost function can be expressed as:

$$\text{Minimize}_{\boldsymbol{w},b,\xi} \frac{1}{2} \mid\mid \boldsymbol{w} \mid\mid^2 + C \cdot \sum_{i=1}^{M} \xi_i \tag{4.34}$$

subject to

$$y_i(\boldsymbol{w} \cdot x_i + b) \geq 1 - \xi_i \tag{4.35}$$

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

The parameter C determines the trade-off between the amount of error to be tolerated and the margin maximization. Mercer's theorem [96] is used in the mapping space, so that the dot product of the vectors can be equally formed as a function of dot products of the corresponding vectors in the current space [63]. This equivalence can be expressed as:

$$\begin{aligned} K(x_i, x_j) &= \phi(x_i) \cdot \phi(x_i) \\ &= (x_i, x_i^2) \cdot (x_j, x_j^2) \\ &= x_i x_j + x_i^2 x_j^2 \\ &= x_i \cdot x_j + (x_i, x_j)^2 \end{aligned} \tag{4.36}$$

where the kernel function is represented by $K(x_i, x_j)$. This expression is true if and only if the following condition holds true for any function g:

$$\int g(x)^2 \, \mathrm{d}x \; is \; finite \implies \int K(x,y) g(x) g(y) \, \mathrm{d}x\mathrm{d}y \geq 0 \tag{4.37}$$

Without knowing the explicit form of ϕ, any data can be linearly separated in the higher dimensional space by simply selecting an appropriate kernel function. Thus, the dual optimization problem can be defined as:

$$\text{Maximize} \sum_{i=1}^{M} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{M} \alpha_i \alpha_j y_i y_i (x_i, x_j) \tag{4.38}$$

subject to

$$\sum_{i=1}^{M} \alpha_i y_i = 0 \; and \; \alpha \geq 0 \tag{4.39}$$

It should be noted that drawing a complex curve is not suitable for separating data. An alternative is to find an optimal hyperplane in the feature space that separates the data clearly and allows an SVM to accurately classify new test data. The decision function therefore becomes:

$$f(x) = \sum_{i \in S} \alpha_i y_i (x_i x) + b \tag{4.40}$$

where S is the set of support vectors.

4.2.1 Kernel Functions

In cases where the data is not linearly separable, a suitable hyperplane that separates the classes is required. To achieve this, a kernel function is used to map the data from the current space onto a higher-dimensional feature space. The following basic kernels, based on Mercer's theorem, are used by the SVM for training and classification, where r, d and γ are kernel parameters [35]:

- Linear: $K(x_i, x_j) = (x_i)^T \cdot (x_j)$
- Polynomial: $K(x_i, x_j) = (\gamma(x_i)^T (x_j) + r)^d, where\ \gamma > 0$
- Radial Basis Function: $K(x_i, x_j) = exp(-\gamma \cdot \| x_i - x_j \|^2), where\ \gamma > 0$
- Sigmoid: $K(x_i, x_j) = \tanh(\gamma \cdot (x_i)^T \cdot (x_j) + r), where\ \gamma > 0$

The choice of kernel is important because it affects the prediction accuracy of the SVM [19]. There is no standard on how to choose an optimal kernel. A process of trial and error is often used to choose the kernel most suited to an application.

4.2.2 A Comparison of Multi-Class SVM Techniques

SVMs are inherently binary classifiers that are intended towards problems involving two classes. However, they can be used in problems involving more than two classes using a variety of techniques that have been proposed in the literature [36]. These techniques generally take the form of a combination of binary classifiers and a decision strategy to determine which class the input pattern belongs to. Three of the most common techniques are explored in the following subsections [36].

4.2.2.1 One-vs-Rest

Consider an M-class problem. This technique separates the data points of every class $i \in \{1, 2, \ldots M\}$ from the data points of the remaining classes. The data points from all classes except class i are combined to form a single class. This results in a binary classifier with a label representing class i and another label representing the remaining classes. Repeating this procedure for every class $i \in \{1, 2, \ldots M\}$ results in a total M classifiers.

The testing phase consists of a test pattern that is presented to all M classifiers. Class i with the maximum output value is determined as the predicted label. This technique results in long training and testing times due to the potentially large number of data points in each combination pair of classes.

4.2.2.2 One-vs-One

This technique trains $\frac{M(M-1)}{2}$ binary classifiers using every binary pair-wise combination of M classes. The Max-Wins algorithm is used to combine the classifiers. Every classifier is trained to differentiate between two classes using the data points in those classes as positive and negative examples.

The testing phase uses the Max-Wins algorithm to determine the class with the majority of votes. This technique results in a shorter training time than the One-vs-Rest technique due to the number of data points in the combination of classes being smaller. However, it has longer testing times than the One-vs-Rest technique due to the large number of classifiers involved.

4.2.2.3 Directed Acyclic Graph

Platt *et al.* introduced the Directed Acyclic Graph (DAG) SVM algorithm [71]. The training phase is similar to the One-vs-One technique, where $\frac{M(M-1)}{2}$ binary classifiers are trained using every binary pair-wise combination of the M classes. The testing phase is based upon a rooted binary DAG that consists of $\frac{M(M-1)}{2}$ internal nodes and M leaves.

For example, assume a 4-class problem with $i \in \{1, 2, 3, 4\}$. Figure 4.13 illustrates the DAG constructed using the pair-wise binary classifiers. Beginning at the root node, classes 1 and 4 are compared. If the input pattern is classified as class 1, then it means that class 4 was rejected. Thus, from this node onwards, it will not be necessary to classify against class 4 again. Hence, after $M - 1 = 3$ steps, a single predicted class will remain.

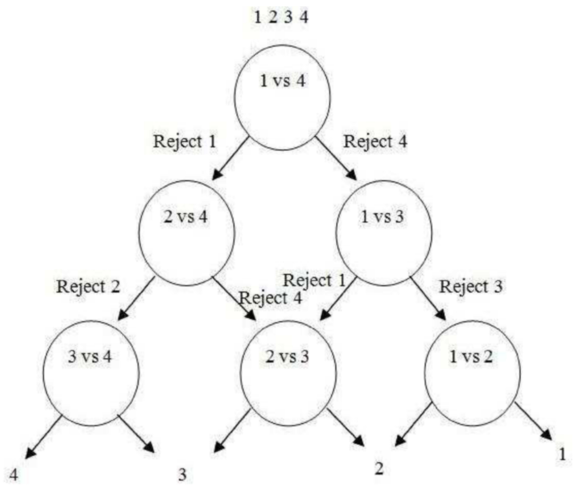


FIGURE 4.13: Directed Acyclic Graph (DAG) of a 4-class problem. At each node a class is rejected until a single class remains.

This technique results in much shorter training times compared to the One-vs-Rest technique. Furthermore, it results in much shorter testing times compared to the One-vs-One technique. Thus, this technique is a suitable technique for multi-class SVMs.

4.3 Summary

In this chapter the components that form part of the learning-based approach used in the pose recognition and estimation methodology used in this research were discussed.

Various techniques used in feature extraction were discussed. The Viola-Jones face detection algorithm was explained. A background into skin detection techniques was provided. A discussion of the various colour spaces and skin models that can be used, in this regard, were discussed. A description of the various background subtraction techniques was provided. Each technique was shown to have strengths and weaknesses. These need to be taken into account when selecting a technique suitable to a specific application. Lastly, morphological operations towards feature enhancement were examined. Four operations—Erosion, Dilation, Opening and Closing—were examined.

Subsequently, a detailed discussion of SVMs was carried out. The discussion detailed the theory behind classification technique used by SVMs. Various kernels that can be used were described. Finally, a comparison of the techniques used to extend the classification capability of SVMs to multi-class problems was carried out.

The next chapter describes the use of these techniques to achieve upper body pose recognition and estimation.

Chapter 5

Design and Implementation of the Faster Upper Body Pose Recognition and Estimation System

This chapter discusses the design of the faster learning-based pose recognition and estimation system proposed in this research. At the highest level of abstraction, the system can be viewed as taking place in two phases, depicted in Figure 5.1. The first phase involves the use of image processing techniques for the extraction of features from an input image sequence. The second phase involves classification on the extracted features using an SVM.

Sections 5.1 and 5.2 describe the analysis, optimization and re-implementation of the feature extraction phase of Achmed's algorithm. Thereafter, both the original and modified algorithms are re-implemented using the CUDA framework on the GPU, described in Section 5.2. Section 5.3 tables the four implementations. Section 5.4 describes the training and testing of the SVM.

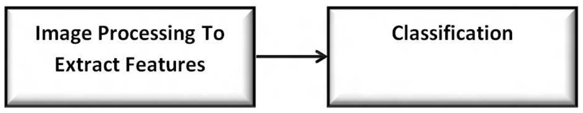


FIGURE 5.1: Image processing for feature extraction.

5.1 Optimization of Achmed's Upper Body Pose Recognition and Estimation Algorithm

This section discusses the optimization of the original upper body pose recognition and estimation algorithm by Achmed. The original algorithm runs on the CPU. For reference purposes this implementation is, henceforth, referred to as OrigCPU. The performance of the individual components are analyzed and the optimizations that can be made to the individual components are stated.

Subsection 5.1.1 provides a description of the original algorithm. The system is analyzed to determine inefficient and ineffective areas of the algorithm that require optimization in Subsection 5.1.2. The analysis is used to produce a modified and more efficient version of the algorithm in Subsection 5.1.3. The modified algorithm is described in Subsection 5.1.4. A performance comparison between each component of OrigCPU and the modified algorithm is carried out in Subsection 5.1.4.

5.1.1 The OrigCPU Algorithm

The OrigCPU algorithm is depicted in Figure 5.2. The face detection component is the foundation of the feature extraction procedure and is used to obtain the centre of the face in the current image. The centre of the face is used for two purposes:

1. To obtain a colour distribution that is representative of the individual's skin.
2. For normalization purposes. The individual is repositioned in the image such that he/she is in the centre of the frame.

The nose region is positioned around the centre of the facial frame. A 10×10 pixel area around the centre of the nose is extracted as illustrated in Figure 5.3. The Hue values of this area are represented as a histogram, which functions as a look-up table for skin pixel values. This method was discussed in the previous chapter. The histogram is back-projected on to the original image to produce a greyscale image in which regions that are more likely to be skin appear brighter. As per Achmed [1] and Li's [51] work, the result is thresholded with the value of 60 to obtain the binary skin image illustrated in Figure 5.4.

The GMM background subtraction technique is used to highlight the moving foreground in the current image as illustrated in Figure 5.5. The result of the background subtraction is combined with the result of the skin detection using a logical And operation.

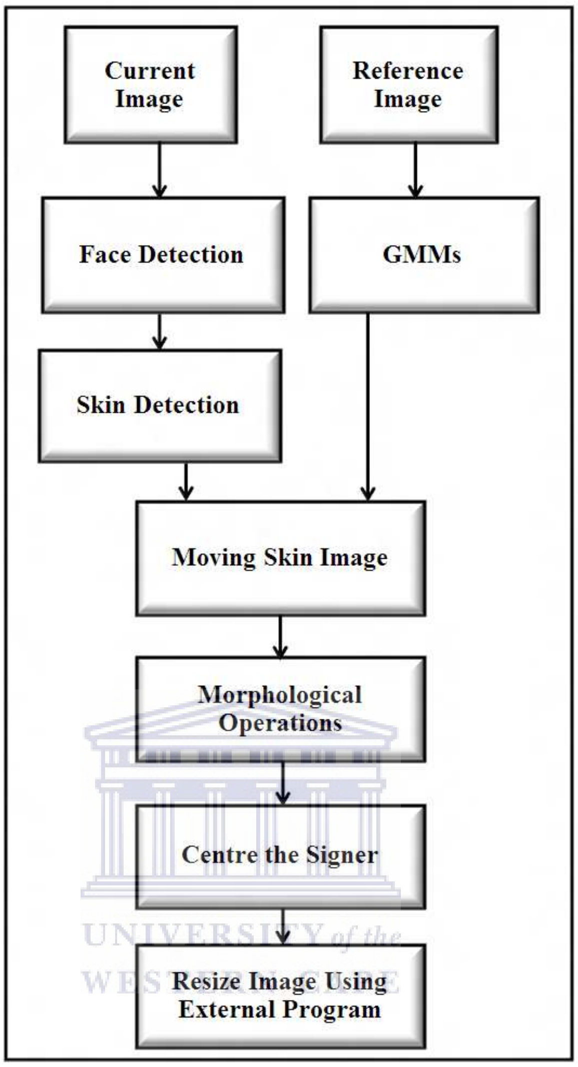


FIGURE 5.2: Original upper body pose recognition and estimation algorithm.

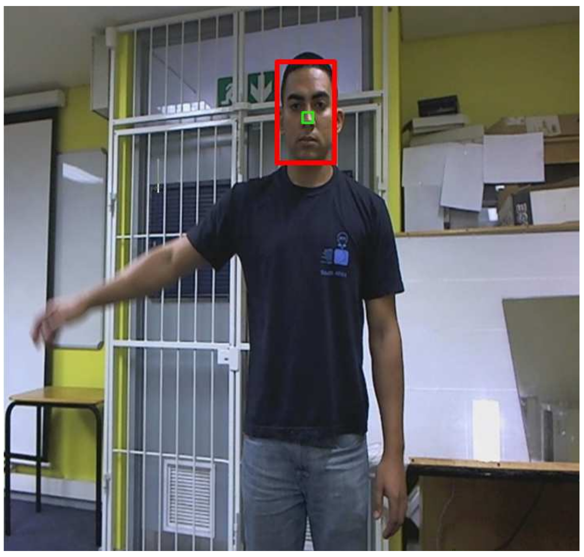


FIGURE 5.3: Face detection and nose region.

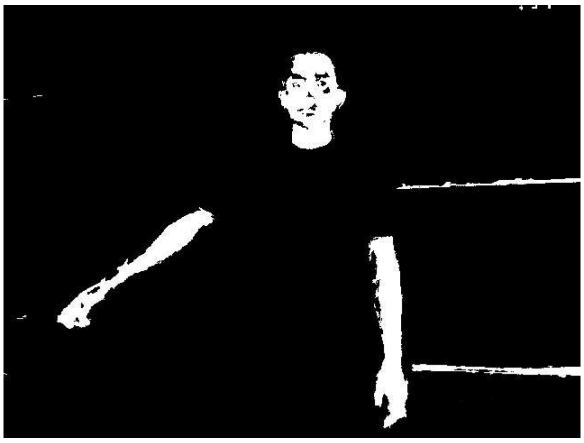


FIGURE 5.4: Skin Image.

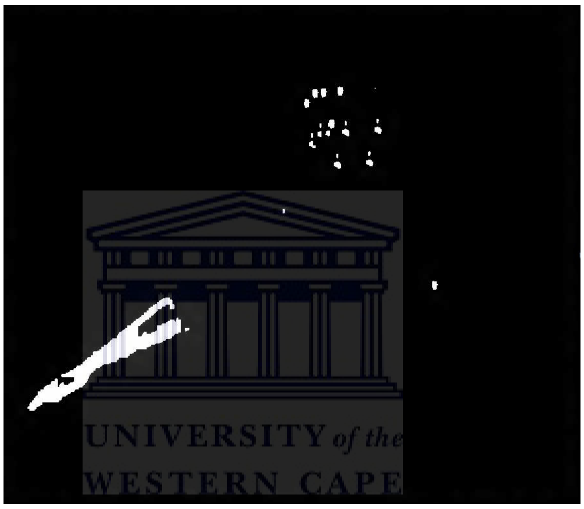


FIGURE 5.5: GMM background subtraction.

This technique highlights only the skin pixels that have moved, henceforth referred to as the moving skin image and depicted in Figure 5.6. Stationary pixels that were falsely detected as skin, such as skin-coloured furniture, are eliminated using this technique. Additionally, moving objects that are non-skin-coloured are also eliminated using this technique. The majority of noise in the image is eliminated. The feature extraction technique is very robust.

No further processing is performed when the result of the background subtraction contains less than a certain number of pixels. Achmed found 7000 to be the optimal number of pixels [1].

It can be observed in Figure 5.6 that the arm of the individual has rough contours and a large hole—a big discontinuity of white pixels in the arm. The following morphological operations are applied to remove such unwanted features from the image: Erosion, Opening and Dilation, in that order. Erosion is applied, using a 17×17 rectangular structuring element, to remove isolated noise regions from the image. The rough

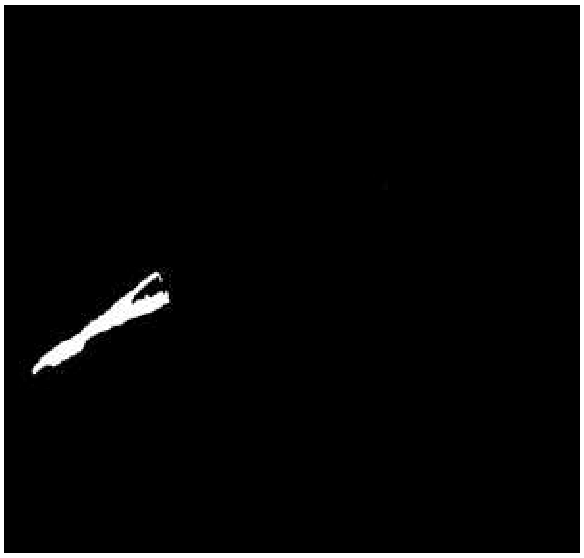


FIGURE 5.6: The results of a skin image superimposed on the objects of interest to obtain the moving skin image.

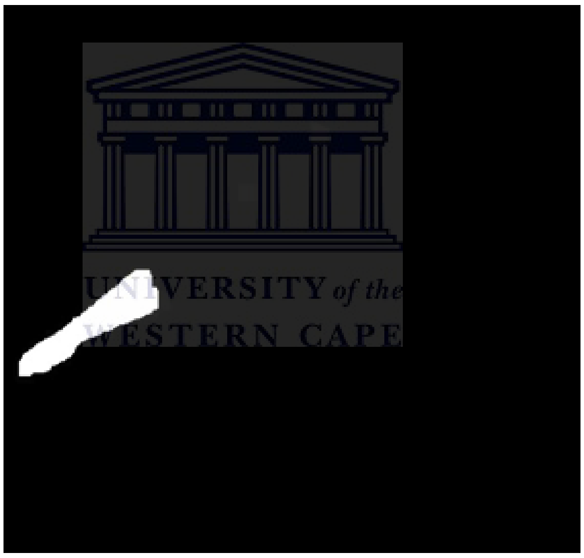


FIGURE 5.7: Enhanced moving skin image with noise removed.

contours and any remaining noise are removed by applying Opening with a 21×21 rectangular structuring element. Dilation is applied to the resulting image, using a 13×13 rectangular structuring element, to produce the enhanced image illustrated in Figure 5.7.

The location of the face is used to normalize the moving skin image. The normalization process shifts the moving skin image vertically and horizontally such that the face in the current frame is aligned with the first facial frame in the sequence.

The normalized image is resized. The input images used have a size of 640×480 pixels. The more pixels an image contains, the greater the amount of detail, which in turn allows for more accurate extraction of features. However, training a large number of features results in very long SVM training and testing times. An image size of 640×480 amounts to 307 200 features per image. When training on 1500 images, a total of 460 800 000 features are obtained. An efficient way to reduce the number of features, while retaining the essence thereof is to reduce the size of the image [12].

Each image is resized to 40×30 pixels using an external program called Convert. This takes place by averaging every 16×16 pixels into a single pixel. The resized image contains the feature vectors of the input image. It is written to a data file to be used by the SVM.

This system was implemented on the CPU. In this implementation, the entire input image sequence is first loaded into primary memory before the feature extraction phase. This is carried out to avoid the latencies associated with transferring data between the hard disk and memory. Also, each step in the process is executed on all cores of the CPU using Threaded Building Blocks (TBB). Making use of these two optimizations ensures that the CPU runs at its full capacity. This ensures that a fair comparison between the CPU and GPU implementations is carried out in the next chapter.

5.1.2 Analysis of Individual Components

The performance of each component in OrigCPU was measured in average FPS. Videos of 14 SASL signs performed by six test subjects were used as input to the OrigCPU implementation. The test subjects had varied skin tones. Two subjects had light skin, two had dark skin and the remaining two were in between. The average FPS of each component, per subject, per sign, was recorded. The complete set of results is provided in Tables A.3 of Appendix A. The results are summarized in Table 5.1.

Referring to Table 5.1, it is clear that the face detection component is a major bottleneck, averaging at 9 FPS. Background subtraction is also observed to be relatively slow, averaging at 28 FPS. Additionally, the resize component uses an external program that continuously accesses the hard disk. Eliminating this factor has the potential to provide a performance speed up.

The skin detection component operates at a high speed, averaging at 201 FPS. However, optimizing the bin width has the potential to improve the skin detection accuracy, hence, the overall accuracy of the system [12].

Component	FPS
Face Detection	9
Skin Detection	201
Background Subtraction	28
Morphology Operations	60
Resize	104

Table 5.1: Analysis of OrigCPU.

5.1.3 Optimization of Individual Components

5.1.3.1 Face Detection

The face detection component in Achmed's implementation searched for any number of possible faces in the image. Secondly, the initial face detection search size was set to the size of the frame and continuously reduced until a face was located. Using the first assumption set forth in Chapter 1, two optimizations can be made. It can safely be assumed that there will only be, at most, one face in the frame and that the initial search size will only be a fraction of the size of the frame. The processing speed of the Viola-Jones face detection method can be significantly improved using the following optimizations:

- Ending the search when the first face, which is the biggest face, is detected.
- Setting the initial search size to a fraction of the frame size. Setting the initial search size to an eighth of the frame size was found to be sufficient.

5.1.3.2 Skin Detection

The skin detection method was discussed in the previous chapter. The bin width used to generate the histogram representing the skin colour can affect the skin detection accuracy. Similar to Tabassum *et al.* [93], Oliveira and Conci [66], and Almohair *et al.* [5], the percentage of true positive and true negative skin pixels in a series of skin images, resulting from the use of a varied parameter—in this case, the bin width—was taken as the measure of skin detection accuracy. Manual true positive and true negative counts of skin pixels in videos of six different subjects performing the SASL sign "Away" at different bin widths were obtained. The bin widths used ranged from 4 to 32, in increments of 4.

Bin Width	**Mean True Positive Count (%)**	**Mean True Negative Count (%)**
4	93	15
8	90	91
12	78	96
16	64	98
20	54	98
24	51	98
28	44	98
32	40	98

Table 5.2: Summary of skin detection accuracy results at different bin widths.

Tables A.1 and A.2 in Appendix A contain the complete set of percentage true positive and true negative counts obtained at each bin width for each subject. These results are summarized in Table 5.2.

It can be observed in Table 5.2 that as the bin width increases the true positive count appears to decrease. It appears that the detection of true skin pixels deteriorates as the bin width increases. A similar but opposite effect is observed with the true negative count. It can be seen that the true negative count increases as the bin width increases, but seems to stabilize at a bin width of around 12, at which the true negative count does not have any notable increase.

This trend can be attributed to the fact that using a higher bin width causes a greater range of pixel values to be grouped into fewer bins—a significant loss of detail [12]. An increasing loss of detail eliminates an increasing amount of noise in the image, which is potentially limited and small to begin with, until it reaches the point where very little noise exists and thus no notable change in the true negative rate. On the other hand, an increasing amount of abundantly available actual skin pixels continue to be eliminated from the image as detail is lost. The optimization of the bin width strives to achieve a balance between the detection of actual skin pixels and elimination of the noise pixels.

A bin width of 8 registers the highest combination of the true positive and true negative skin detection count. A bin width of 8 was used as the bin width of the modified skin detection component. It is expected that the optimized skin detection component will help produce more enhanced features in frames with a larger amount of exposed moving skin. Motions of the arms in the plane of the captured frame are easily observable. Such movements expose large amounts of moving skin. These skin regions can be highlighted more accurately by the optimized skin detection component. Fewer noise pixels will be highlighted. On the other hand, the fact that depth information is not provided by the camera in the setup implies that movements of the arms towards and away from the camera may not be observable, appearing as a slow scaling of the object at best. Such movements may lead to the concealment of moving skin regions. The optimized skin detection component may not improve the detection accuracy of skin in such frames beyond that of the original skin detection component.

Examples of signs with a large amount of exposed moving skin are “curtains” and “wide”. The sign ”curtains” is performed in SASL by lifting both hands above the shoulders and moving them inward and outward in a waving motion in the plane of the frame. The sign ”wide” is performed in SASL by lifting both arms, outstretched, up the sides body in the plane of the frame to form a ‘T’ shape with the arms and body.

An example of a sign with a small amount of exposed moving skin is “run”. The sign “run” is performed in SASL by simulating the movement of the arms during running. Most of the motion takes place towards and away from the camera.

5.1.3.3 Background Subtraction

GMMs are used in OrigCPU for background subtraction. GMMs were discussed in detail in Chapter 4. They are robust and provide information about the location of moving objects in the frame as well as the intensity of their motion. However, this comes at the expense of performance. The intensity of motion does not form part of the feature vectors used in the classification phase in Achmed’s algorithm. Only the information on the location of moving objects is used. Therefore, the use of this technique is wasteful of processing speed.

Alternative background subtraction techniques were discussed in Chapter 4 and a comparison of these techniques was performed. Based on the discussion, a background subtraction technique that is better suited to Achmed’s algorithm is frame differencing. Frame differencing provides information on the location of moving objects at a much higher processing speed than GMMs. It is also more robust to a dynamic background than simple background subtraction techniques.

Frame differencing was implemented as the background subtraction technique in the modified upper body pose recognition and estimation system.

5.1.3.4 Resizing the Image

An external image resizing program called Convert is used to resize the image after morphological operations have been applied in Achmed’s implementation. This program resizes the image with minimal loss in detail by resampling the image using the pixel area relation method. In order to use the program, the image is required to be written

Component	**OrigCPU FPS**	**ModCPU FPS**	**Speed-Up Factor**
Face Detection	9	105	11
Skin Detection	201	200	1
Background Subtraction	28	372	13
Morphological Operations	60	60	1
Resize	104	1691	16

Table 5.3: Performance analysis of OrigCPU and ModCPU.

to the hard disk. The program is executed to resize the image. The result then has to be loaded back into memory to be used by the rest of the algorithm. The transfer of data between memory and the hard disk is a major source of latencies [64].

A more efficient method of resizing the image is the use of OpenCV's native resize function. This function can be passed appropriate parameters to perform the same operation as Convert [12]—resampling the image using the pixel area relation method. The use of this method is expected to drastically reduce latencies resulting in a much higher processing speed.

OpenCV's native resize function was implemented for the resizing of the image after morphological operations have been applied.

5.1.4 Modified Upper Body Pose Recognition and Estimation System

This subsection discusses the modified upper body pose recognition and estimation system, henceforth referred to as ModCPU. The modified algorithm is depicted in Figure 5.8. It can be observed from the figure that the optimized components discussed in the previous subsection are used.

A performance comparison was carried out between OrigCPU and ModCPU to assess the effectiveness of the optimizations made. Identical to the analysis of OrigCPU, described in Section 5.1.2, the performance of each component in ModCPU was measured in average FPS. The same videos were used as input to the ModCPU implementation. The average FPS of each component, per subject, per sign, was recorded. The complete set of results is provided in Tables A.3 and A.4 of Appendix A. The results of the test for both OrigCPU and ModCPU, along with the ratio of the ModCPU to OrigCPU performance for each component—the speed up factor—are summarized in Table 5.3. It is to be noted that a speed-up factor of more than 1 indicates an improvement in processing speed and the higher the speed-up factor, the more substantial the improvement in the processing speed. A speed-up factor of less than 1 indicates a loss in processing speed.

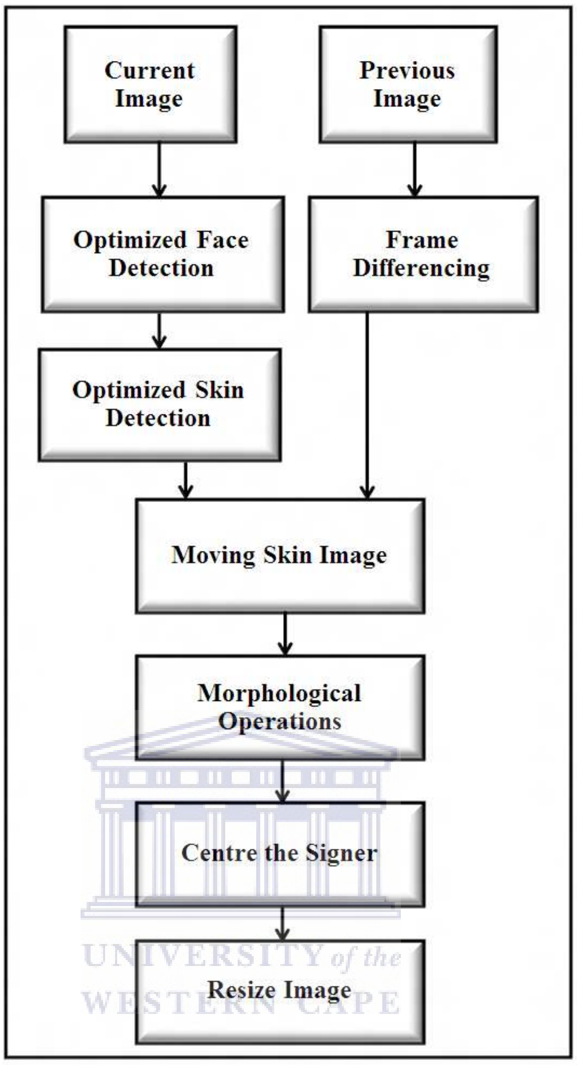


FIGURE 5.8: Modified upper body pose recognition and estimation algorithm.

It is clear from Table 5.3 that the face, skin and resize components of ModCPU are considerably faster than those in OrigCPU. Each of these optimized components perform no less than 11 times faster than its original counterpart, but as high as 16 times faster. As per expectation, the skin component, which was optimized for greater accuracy, does not register an increase in processing speed. The morphological operations component also registers no increase in processing speed.

A test was carried out to determine whether the difference between the performance of each component of OrigCPU and ModCPU was significant. A series of paired t-tests were conducted to examine difference between OrigCPU and ModCPU for each sign. The results of these tests are summarized in Table 5.4.

It is clear from Table 5.4 that the face, skin and resize components of ModCPU are significantly faster ($p < 0.0001$ in every case) than those in OrigCPU for every sign.

Sign	Face		Background		Skin		Morphological		Resize	
	Std Dev	P-Value	Std Dev	P-Value	Std Dev	P-Value	Std Dev	P-Value	Std Dev	P-Value
Away	2.6903	< 0.0001	8.8524	< 0.0001	16.4963	0.9032	2.5172	0.6165	12.3194	< 0.0001
Bye	2.5130	< 0.0001	17.7000	< 0.0001	11.6844	0.1511	3.3278	0.3563	9.4283	< 0.0001
Cracker	2.5413	< 0.0001	9.6446	< 0.0001	7.3299	0.0190	2.2335	0.8581	12.5945	< 0.0001
Curtains	3.3241	< 0.0001	10.5142	< 0.0001	17.7582	0.2606	2.0843	0.3637	10.7811	< 0.0001
Dress	1.8380	< 0.0001	14.1372	< 0.0001	10.3874	0.3275	4.8867	0.4377	14.0467	< 0.0001
Eat	3.3333	< 0.0001	12.4968	< 0.0001	16.0963	0.3162	3.1502	0.6171	13.8320	< 0.0001
Left	2.2430	< 0.0001	12.3901	< 0.0001	10.1392	0.9016	1.8721	0.0544	12.7285	< 0.0001
Light	3.2342	< 0.0001	12.0148	< 0.0001	27.2046	0.8759	4.4227	0.3569	9.9330	< 0.0001
Love	2.8901	< 0.0001	12.1828	< 0.0001	16.9674	0.6540	2.9405	0.1846	13.5952	< 0.0001
Right	2.9891	< 0.0001	11.1133	< 0.0001	21.8679	0.4984	4.0816	0.1244	12.4866	< 0.0001
Run	1.8573	< 0.0001	10.5980	< 0.0001	20.8965	0.6409	5.2603	0.7827	7.5917	< 0.0001
We	3.0816	< 0.0001	13.7493	< 0.0001	19.9982	0.6465	1.8803	0.0025	11.0711	< 0.0001
Why	2.6238	< 0.0001	15.0753	< 0.0001	16.9896	0.1709	4.6262	0.6317	4.6670	< 0.0001
Wide	2.0139	< 0.0001	12.1682	< 0.0001	11.4122	0.6338	2.3640	0.4485	13.1319	< 0.0001

TABLE 5.4: Performance analysis of OrigCPU and ModCPU for each component.

The performance optimizations made to these components were appropriate and highly successful.

As per expectation, the difference in performance between OrigCPU and ModCPU for the skin and morphological operations components is not significant.

5.2 Performance Enhancement Using CUDA

Both OrigCPU and ModCPU were re-implemented on the CUDA framework. The resulting implementations run on the GPU and are, henceforth, referred to as OrigCUDA and ModCUDA, respectively. The algorithms were ported to the GPU as follows:

1. A thread was assigned to each Haar cascade in the face detection component. This enables the face detection method to search multiple image sections for faces in parallel.

2. Background subtraction and face detection were configured to run simultaneously on separate parallel threads.

3. During background subtraction, each pixel in the current image and reference image was assigned a separate thread. The absolute difference is taken between the all the corresponding pixels of the current image and reference image in parallel.

4. Skin detection, similar to Background subtraction, had a thread assigned to each pixel. All pixels are processed in parallel.

5. The process of combining the background subtraction and skin detection components to produce the moving skin image was similarly parallelized by assigning each pixel of the respective components to a separate thread.

Implementation	Description
OrigCPU	Achmed's algorithm, running on the CPU.
ModCPU	The modified algorithm, running on the CPU.
OrigCUDA	Achmed's algorithm, running on the GPU.
ModCUDA	The modified algorithm, running on the GPU.

TABLE 5.5: Summary of Implementations.

Additionally, the resize component of ModCUDA was parallelized. This was done by assigning a separate thread to each pixel of the image. Each thread averages the 16×16 pixels in its vicinity in parallel. It was not possible to port the resize component of OrigCUDA as it uses an external program.

5.3 Summary of Implementations

The four implementations of the upper body pose recognition and estimation system are summarized in Table 5.5.

5.4 Training and Testing Phases

Subsection 5.4.1 discusses the procedure used to train the SVM. The trained SVM is used to predict the correct location of the wrists and estimate the performed sign using Blender in the testing phase described in Subsection 5.4.2.

5.4.1 Training Phase

The training set consisted of two individuals, male and female, performing 14 SASL signs described in Table 5.6. These signs are the same signs used by Achmed.

The procedure used to train the SVM is depicted in Figure 5.9.

Each frame of each video in the training set was processed using the entire feature extraction procedure mentioned previously. Starting with each image of size 40×30 pixels, resulting from the feature extraction procedure, a data file is created consisting of the pixel values of the image. Each pixel in the image is taken as a feature vector and is assigned an index, as illustrated in Figure 5.10. An image of size 40×30 pixels contains a total of 1200 feature vectors. Referring to Figure 5.10, the first feature vector, in this case, has an index of 1 and a value of 0. The last feature vector has an index of 1200 and a value of 1.

Sign	**Description**
Away	Move the right hand to and fro away from right side of the body.
Bye	Waving with the right hand inwards to the left and outwards to the right above right shoulder.
Cracker	Moving hands from the chest away from each other to the sides.
Curtains	Moving both hands towards the face and outwards again above respective shoulders.
Dress	Moving hands from the chest downwards. When reaching below the hips, move hands away from the body.
Eat	Moving both hands towards the mouth and mimic eating using chopsticks.
Left	Raise left hand away from the left side of the body.
Light	Raise right hand above right shoulder just above the head.
Love	Cross arms in the middle of the upper chest.
Right	Raise right hand away from the right side of the body.
Run	Moving both hands on the side of the chest imitating a running movement.
We	Move right hand to the left side of the chest and across to the right shoulder.
Why	Move right hand to the left side of the chest and tap twice against chest.
Wide	Raise right and left hand away from the sides of the body.

TABLE 5.6: The 14 SASL signs used in training and testing.

A label is assigned to each set of 1200 feature vectors, which groups the resulting features of that frame into a specific training class and indicates the position of both wrists in that frame. The positions of both wrists need to be identified. A structured method of assigning a label to the position of each wrist is to superimpose a grid on the training image. The grid consists of 168 equally sized squares and covers the entire pose space as illustrated Figure 5.11.

Each square is a quarter of the size of the face. The number of blocks is limited to only cover the pose space. Each block is assigned to a class in the SVM and is assigned to a set of feature vectors if the wrist is observed to be in that block. This yields a total of 168 classes. The top-left block is assigned the label 1, increasing towards the right and downwards, with the bottom-right block being assigned the label 168. In Figure 5.11, the wrist of the right hand is observed in block 32, as indicated by the cross, and is therefore assigned label 32. Both wrists are assigned a label.

Data scaling is another form of preparation of data for the SVM. Scaling the data avoids features with a greater numeric range from dominating features with a lower numeric

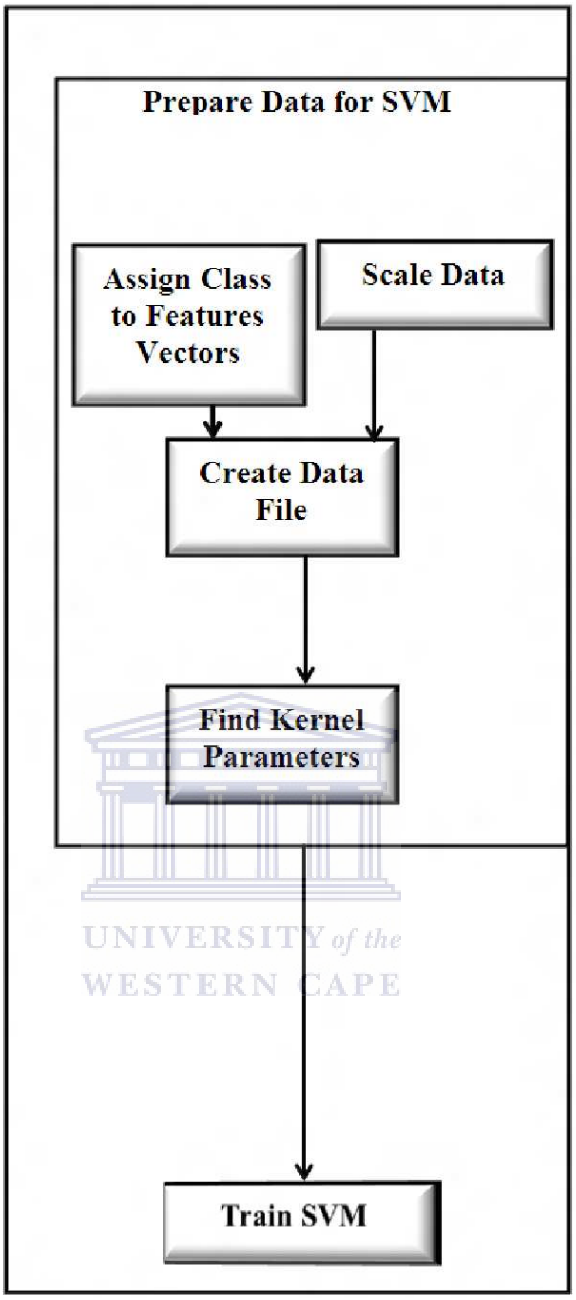


FIGURE 5.9: Procedure used to train the system.

```
1:0 2:0 3:0 4:0 5:0 6:0 7:0 8:0 9:0
10:0 11:0 12:0 13:0 14:0 15:0 16:0
17:0 18:0 19:0 20:0 21:0 22:0 23:0
24:0 25:0 26:0 27:0 28:0 29:0 30:0
31:0 32:0 33:0 34:0 35:0 36:0 37:0
38:0 39:0 40:0 41:0 42:0 43:0 44:0
45:0 46:0 47:0 48:0...............
1188:1 1189:1 1190:1 1191:1 1192:1
1193:1 1194:1 1195:1 1196:1 1197:1
1198:1 1199:1 1200:1
```

FIGURE 5.10: Data file without labels.

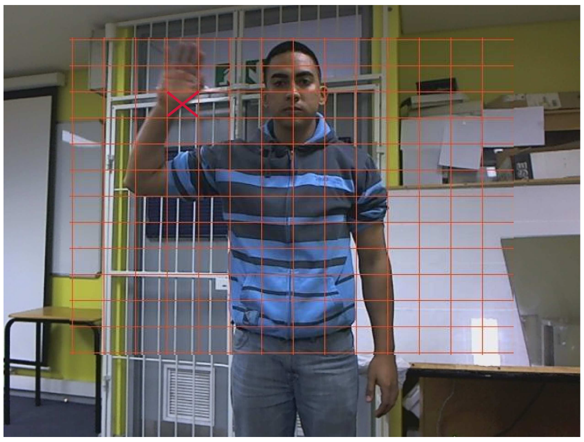


FIGURE 5.11: Superimposed grid in pose space.

range [36]. Thus, a pixel with a value of 255 is converted to 1 and a pixel with a value of 0 is left unchanged. This limits the range of feature vectors to [0,1].

The SVM can be trained to predict test data more effectively by determining the optimal C and γ RBF kernel parameters for the given problem. A brute-force approach that can be used is the trial and error of each C and γ combination, where each parameter is an exponentially growing sequence. A structured alternative uses the grid-search function in LibSVM, which uses cross-validation. The cross-validation method divides the training set into n equally-sized subsets, where the classifier is trained on $n - 1$ subsets and tested on the remaining subset for each parameter combination [35]. This highlights the combination of parameters with the best cross-validation accuracy.

The result of running the LibSVM grid-search function on the training data is depicted in Figure 5.12. The optimum parameters obtained were as follows: C was 512 and γ was 0.000122. The accuracy rate of the kernel was optimized from 88% before optimization to 91% after optimization. The small difference between the two accuracy rates indicates that a high accuracy can be achieved with the RBF kernel even without optimization.

The final format of the training data file is illustrated in Figure 5.13. Each line of the file consists of: the class representing the right-hand wrist; the class representing the left-hand wrist; and the list of feature vectors. Figure 5.13 depicts two lists of feature vectors. The first list represents the right-hand wrist in block 134 and the left-hand wrist in block 132. The second list represents the right-hand wrist in block 132 and the left-hand wrist in block 138.

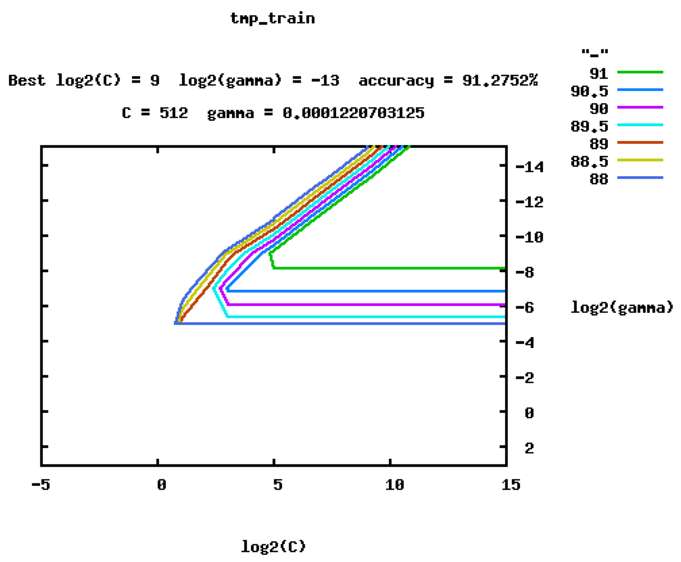


FIGURE 5.12: Grid-search optimization results.

```
134,132 1:0 2:0 3:0 4:0 5:0 6:0 7:0
8:0 9:0 10:0 11:0 12:0 13:0 14:0 15:0
16:0 17:0 18:0 19:0 20:0 21:0 22:0
23:0 24:0 25:0 26:0 27:0 28:0 29:0
30:0 31:0 32:0 33:0 34:0 35:0 36:0
................ 1188:1 1189:1 1190:1
1191:1 1192:1 1193:1 1194:1 1195:1
1196:1 1197:1 1198:1 1199:1 1200:1
132,138 1:0 2:0 3:0 4:0 5:0 6:0 7:0
8:0 9:0 10:0 11:0 12:0 13:0 14:0 15:0
16:0
```

FIGURE 5.13: Data file with labels in the training phase.

5.4.2 Testing Phase

The testing phase aims to predict the wrist position labels given a set of feature vectors. The predicted labels are used to estimate the upper body pose. This procedure is illustrated in Figure 5.14.

For each frame in a test video, the exact procedure used to generate the data file in the training phase is applied, except that the list of feature vectors is assigned default labels of 0. This is illustrated in Figure 5.15. The SVM is expected to predict the actual labels, which indicate the location of the right-hand and left-hand wrists.

The system uses the 3D humanoid model by Van Wyk [100]. The model was developed and runs in the 3D modelling software called Blender. The positions of the wrists of the 3D humanoid model are set to the corresponding locations of the predicted labels. Human-realistic kinematic constraints are used to automatically position all other joints

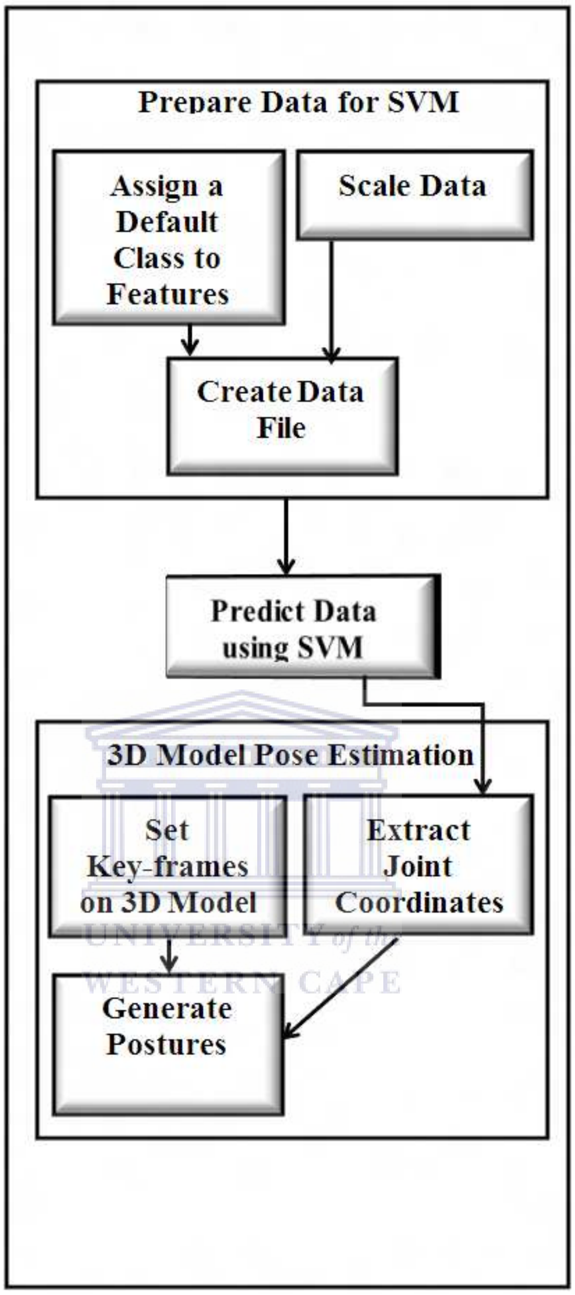


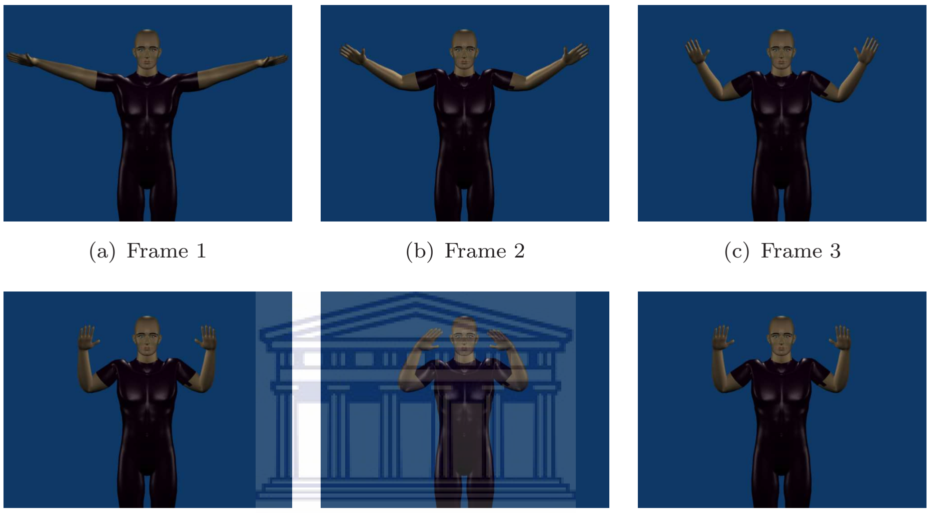
Figure 5.14: Procedure used in the testing phase.

```
0,0 1:0 2:0 3:0 4:0 5:0 6:0 7:0 8:0
9:0 10:0 11:0 12:0 13:0 14:0 15:0 16:0
17:0 18:0 19:0 20:0 21:0 22:0 23:0
24:0 25:0 26:0 27:0 28:0 29:0 30:0
31:0 32:0 33:0 34:0 35:0 36:0 37:0
38:0 39:0 40:0 41:0 42:0 43:0 44:0
45:0 46:0 47:0 48:0 49:0 50:0
1182:0 1183:0 1184:0 1185:0 1186:1
1187:1 1188:1 1189:1 1190:1 1191:1
1192:1 1193:1 1194:1 1195:1 1196:1
1197:1 1198:1 1199:1 1200:1
```

Figure 5.15: Data file with labels in the testing phase.

relative to the position of the wrists in a manner that is feasible and realistic. The system, thus, estimates the upper body pose in 3D.

The estimated upper body pose in each frame is set as a key frame in a Blender animation. A particularly useful feature in Blender is that key-frames can be dynamically created. Once all the key-frames have been created, Blender automatically interpolates between key-frames. This ensures that there are no discontinuities in the final estimated animation of upper body poses. Figure 5.16 illustrates the estimated result of the first 6 frames of the SASL sign "Curtains".


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

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

FIGURE 5.16: 3D estimated upper body pose sequence for the sign "Curtains".

5.5 Summary

This chapter discussed the implementation of the faster upper body pose recognition and estimation system. An analysis, optimization and re-implementation of the feature extraction phase of Achmed's algorithm was carried out. The analysis was used to determine areas of the algorithm where the efficiency and effectiveness could be significantly increased. The optimizations were implemented to produce a modified and more efficient version of the algorithm. A performance comparison between each component of the original and the modified algorithm was carried out. The comparison revealed that the three optimizations aimed at improving performance were highly successful. Thereafter, both the original and modified algorithms were re-implemented using the CUDA framework on the GPU. The above procedure yielded four implementations which are to be compared in the next chapter.

The classification phase used the extracted features for the training and testing of the SVM. Before training the SVM, the data obtained from the extracted features were scaled, labels were assigned to the set of feature vectors and the optimal kernel parameters were determined using a grid-search function. The trained SVM was used to predict the correct label corresponding to the location of the wrists to estimate the performed sign using Blender.

Chapter 6

Experimental Setup and Analysis of Results

This chapter assesses the four implementations of the faster upper body pose recognition and estimation system to determine whether the research objectives set out in Chapter 1 have been achieved. The objectives set out were to achieve a real-time performance at a sustained or improved accuracy.

Section 6.1 describes the setup of the experiments carried out, including the data set collected and used. Section 6.2 describes the testing carried out to determine whether each of the four implementations meet the accuracy criterion mentioned. A comparison in accuracy is carried out. Section 6.3 describes the testing carried out to determine whether the performance criterion was met by the four implementations. The experimental procedure and metrics used are discussed in both cases. A comparison between the four implementations was carried out in each case. A summary of the results and the conclusions drawn from these results close the chapter.

6.1 Experimental Setup

All experiments were carried out on a PC containing an Intel i7 2600k 3.8 GHz quad core CPU, an NVIDIA 580GTX GPU and 8 GB RAM, running the Kubuntu 12.04 x64 operating system. A Logitech C910 web camera was used at a resolution of 640×480 pixels at a frame rate of 15 FPS. The same 14 SASL signs used by Achmed were used in experimentation. Figure 6.1 depicts the location where the web camera was used to capture 14 SASL signs. 30 test subjects, each performing the 14 SASL signs are henceforth labelled as Subject 1–30. The test subjects were chosen such that a wide

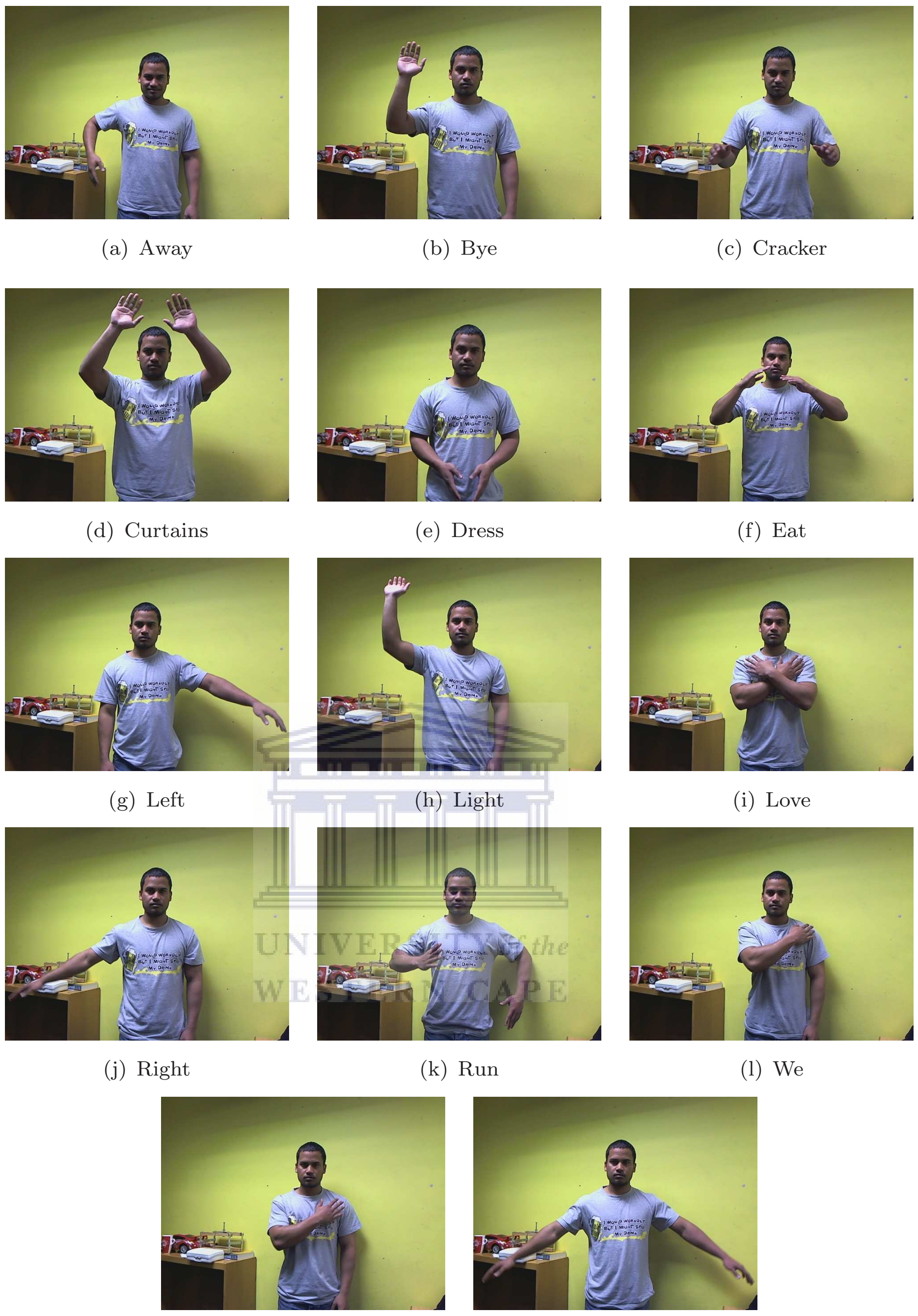
Sign	**Description**
Away	Move the right hand to and fro away from right side of the body.
Bye	Waving with the right hand inwards to the left and outwards to the right above right shoulder.
Cracker	Moving hands from the chest away from each other to the sides.
Curtains	Moving both hands towards the face and outwards again above respective shoulders.
Dress	Moving hands from the chest downwards. When reaching below the hips, move hands away from the body.
Eat	Moving both hands towards the mouth and mimic eating using chopsticks.
Left	Raise left hand away from the left side of the body.
Light	Raise right hand above right shoulder just above the head.
Love	Cross arms in the middle of the upper chest.
Right	Raise right hand away from the right side of the body.
Run	Moving both hands on the side of the chest imitating a running movement.
We	Move right hand to the left side of the chest and across to the right shoulder.
Why	Move right hand to the left side of the chest and tap twice against chest.
Wide	Raise right and left hand away from the sides of the body.

Table 6.1: The 14 SASL signs used in experimentation.

variety of skin colours ranging from white to black were represented. Nine of the subjects were females and the rest were males.

6.1.1 Data Set

No standard set of SASL poses exists upon which experimentation can be based. SASL consists of various poses each conveying a different meaning and each being equally important. As the proposed system is a subset of an automatic sign language translation system, it is important that the signs that are performed can be recognized as official SASL signs. Achmed chose 14 SASL signs from the "Fulton School for the Deaf" SASL Dictionary [34]. The chosen signs do not cover the entire sign language vocabulary, but effort was made to cover the vocabulary to a large extent. The signs performed represent a variety of wrist locations, with signs performed on the far left and far right of the body well represented. Table 6.1 provides a description of each of the 14 signs. Each SASL sign is depicted by a random frame in Figure 6.1.



(a) Away (b) Bye (c) Cracker

(d) Curtains (e) Dress (f) Eat

(g) Left (h) Light (i) Love

(j) Right (k) Run (l) We

(m) Why (n) Wide

Figure 6.1: The 14 SASL signs used in experimentation.

6.1.2 Collection of Videos

Each subject was required to stand facing towards the camera in such a way that his/her upper body and hands were in the frame. The subjects were instructed to move their arms according to the required sign, which was shown to them prior to them performing the sign. Each subject performed 14 signs at their preferred motion speed and a 3 second time period was given to them to first prepare themselves before the frames were recorded. Using 30 subjects resulted in a total of 420 sign videos consisting of a variable number of frames, ranging from 51 to 161.

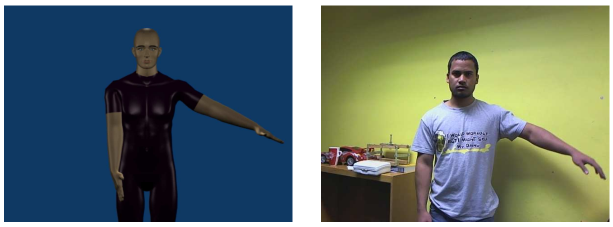
6.2 Accuracy Testing

This section describes the tests that aimed to compare the accuracy of the four implementations. The aim of the comparison was to determine whether an improved or sustained accuracy was achieved by the modified algorithm in accordance with the research question. The following subsections discuss the criterion for a correctly recognized sign, used as a metric for accuracy testing, followed by the procedure used to conduct the experiments and the accuracy test results and analysis.

6.2.1 Criterion for a Correctly Recognized Sign

Each input frame maps onto an estimated output frame in Blender. The estimated frame is generated by setting the positions of the wrists of the 3D humanoid avatar to the wrist positions predicted from the frame. There are many ways to measure the correctness of the estimated result such as performing a comparison of the joint angles and/or positions between the input and estimated frames. An alternative method is to perform a visual comparison between the input and estimated frames.

A point that becomes relevant in deciding between these methods is to consider the goal of the SASL project, which is the translation between SASL to English. The project is intended to create animations to be viewed by human beings. Therefore, it is considered sufficient for a visual comparison to indicate whether a match has been obtained. The estimated outcome is dichotomized as either being a match or non-match on a per-frame basis. Figure 6.2 illustrates a potential match.



(a) Input frame (b) Estimated frame

FIGURE 6.2: Visual comparison for a potential match.

Assessor	**Signs**
Assessor 1	Bye, Curtains, Eat, Light, Right, We, Wide
Assessor 2	Away, Crackers, Dress, Left, Love, Run, We, Wide

TABLE 6.2: Signs inspected by the assessors.

6.2.2 Experimental Procedure

The 420 sign videos from the 30 test subjects were used as input to the four implementations. In each case, the output of the system was analyzed using the criterion for a correctly recognized sign. In order to obtain an unbiased result two independent assessors were used to perform the visual comparisons. Table 6.2 identifies the signs that were assessed by each assessor. Each assessor was instructed to inspect every input frame and the corresponding estimated frame and determine whether there was a match between the frames.

Table A.5 in Appendix A contains the total number of frames in each sign video performed by each subject.

The complete set of results of the number of matches for OrigCPU and ModCPU are provided in Tables A.6 and A.7, respectively, in Appendix A. It was found that both implementations running the original algorithm—OrigCPU and OrigCUDA—and both implementations running the modified algorithm—ModCPU and ModCUDA—achieved exactly the same results. For this reason, these results have been omitted. The comparison of accuracies is, therefore, carried out between the original algorithm and the modified algorithm, referred to as "Orig" and "Mod" for the purposes of the current experiment. Table 6.3 depicts the average number of matches for each sign, across all test subjects, as a percentage of the total number of frames in each case.

Analyzing Table 6.3, the accuracies range from 78.72% to 93.71% for Orig and 83.61% to 97.33% for Mod. Overall Mod achieves a higher average accuracy of 93.08% compared to 87.35% for Orig.

A statistical test was used to determine whether the difference in accuracy between Orig and Mod for each sign was significant. The accuracy values are bounded above by 100 and the values are skewed for most of the signs. For this reason, the non-parametric Signed Rank Test was identified as being appropriate to use to test for significant differences between the algorithms. Table 6.4 summarizes the results of the test. It can be seen from the table that there are significant differences in favour of Mod ($p < 0.001$) for 9 out of the 14 signs, highlighted in the table. The accuracy of Mod for the remaining 5 signs is no less than that of Orig.

Sign	**Orig(%)**	**Mod(%)**
Away	90.15	90.41
Bye	87.46	97.33
Crackers	78.72	84.07
Curtains	85.54	95.66
Dress	89.29	90.66
Eat	90.54	95.05
Left	89.69	98.80
Light	92.72	95.82
Love	93.71	93.05
Right	85.23	96.17
Run	83.92	83.61
We	87.39	90.69
Why	82.60	94.68
Wide	85.93	97.10
Average	87.35	93.08

TABLE 6.3: Mean accuracy of the original and modified algorithms per sign.

The observation that 9 out of the 14 signs register significant increases in accuracy with the modified algorithm and the remaining 5 do not is attributed to the differences in the amount of exposed skin between the two groups of signs. The 9 signs that perform better all expose larger amounts of moving skin. The majority of the motion in the remaining 5 signs occurs towards and away from the camera, therefore exposing small amounts of moving skin. As expected and explained in the previous chapter, signs that expose larger amounts of moving skin are expected to achieve a higher accuracy due to the optimized skin detection component. However, the frame differencing background subtraction technique was chosen over GMMs. Therefore, more tests need to be conducted to further investigate the components that improve the accuracy.

Table 6.5 depicts the average number of matches for each test subject, across all signs, as a percentage of the total number of frames in each case. Analyzing Table 6.5, the accuracies range from 84.45% to 90.86% for Orig and 87.89% to 99.60% for Mod. The standard deviation in these results for Orig and Mod is 1.81% and 2.94%, respectively. These values are very small and indicate that both algorithms are robust to variations in test subjects.

In conclusion, the accuracy of the modified algorithm is at least as high as the original algorithm, although significantly higher in 65% of the signs. It is also as robust to variations in test subjects as the original algorithm. This result is extremely encouraging and shows that the optimizations made were appropriate and successful. The accuracy of the original algorithm, which was very high, has been improved significantly.

Sign	**Test Statistic**	**P-Value**
Away	−0.5	1.0000
Bye	215.5	< 0.0001
Cracker	136.5	0.0017
Curtains	200.5	< 0.0001
Dress	41	0.3599
Eat	159.5	0.0001
Left	201	< 0.0001
Light	134.5	0.0038
Love	−1	0.9814
Right	198.5	< 0.0001
Run	9.5	0.8489
We	54.5	0.2453
Why	220.5	< 0.0001
Wide	212.5	< 0.0001

TABLE 6.4: Results of the Signed Rank Test performed between the original and modified algorithms.

6.3 Performance Testing

This section describes the tests that attempted to answer the remaining section of the research question: can optimization and parallel processing techniques on the CUDA framework be used to achieve real-time upper body pose recognition and estimation based on Achmed's methodology? The following subsections describe the criterion for real-time performance, the experimental procedure and analysis of results of the performance testing.

6.3.1 Criterion for Real-Time Performance

This system aims to render estimated frames in real-time. For this to appear to be real-time to the user it must render frames at 15 FPS. Therefore, the system must be able to process input frames at 15 FPS.

Therefore, the criterion for real-time performance is a processing speed of at least 15 FPS.

6.3.2 Experimental Procedure

The 420 sign videos from the 30 test subjects were used as input to the four implementations. The system was modified to measure the time it took to process each frame and write the measured values to a text file, iteratively for all 420 sign videos. The values

Test Subject	**Orig**	**Mod**
Subject 1	90.76	92.31
Subject 2	88.14	92.63
Subject 3	89.83	97.26
Subject 4	87.48	90.35
Subject 5	88.22	91.35
Subject 6	86.81	91.68
Subject 7	90.86	93.65
Subject 8	85.04	92.85
Subject 9	88.03	90.12
Subject 10	84.45	91.63
Subject 11	89.72	88.84
Subject 12	87.47	92.07
Subject 13	89.19	91.22
Subject 14	85.97	91.81
Subject 15	85.88	91.81
Subject 16	87.19	95.17
Subject 17	85.48	98.68
Subject 18	86.32	98.66
Subject 19	90.75	99.60
Subject 20	88.19	93.58
Subject 21	87.08	96.28
Subject 22	88.07	96.26
Subject 23	85.40	95.55
Subject 24	88.13	92.13
Subject 25	86.09	92.96
Subject 26	86.16	90.68
Subject 27	86.08	89.74
Subject 28	85.64	91.65
Subject 29	87.17	87.89
Subject 30	84.84	93.92

Table 6.5: Mean accuracy of the original and modified algorithms per test subject.

written to the text file were obtained as follows: the time taken to process each frame was inverted to obtain the processing speed in FPS at that frame—the instantaneous processing speed. This procedure was repeated for all 420 sign videos using each of the four implementations.

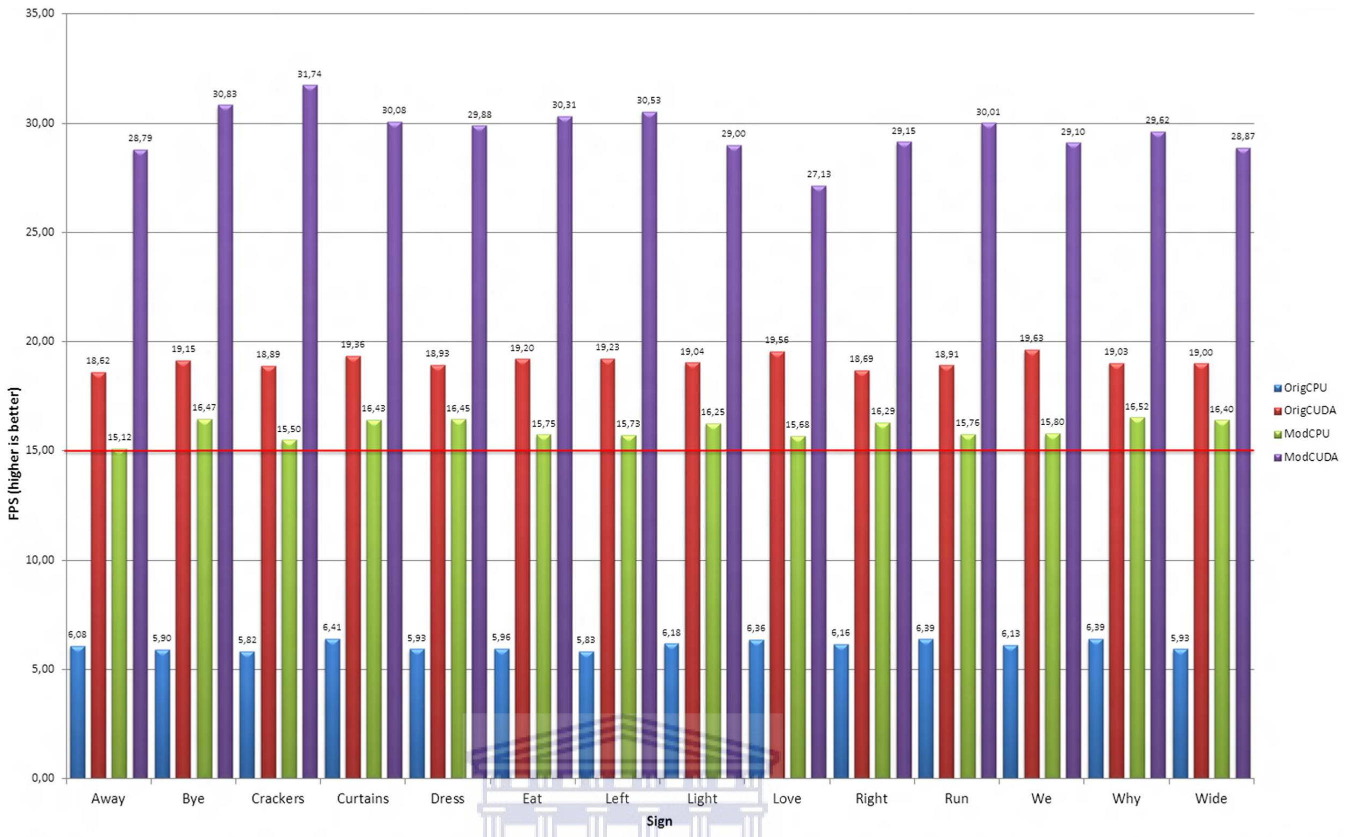


FIGURE 6.3: Comparison in mean performance of the 4 implementations.

6.3.3 Results and Analysis

Figure 6.3 is a graphical representation of the overall mean performance. It is clearly evident from the graph that ModCUDA has the highest performance out of all four implementations. It is approximately 5 times faster than the slowest performing implementation—OrigCPU—and approximately 1.5 times faster than the second fastest implementation—OrigCUDA. Three out of four implementations pass the criterion for real-time performance. They achieve a mean processing speed of at least 15 FPS. ModCUDA achieves a processing speed that is approximately double the requirement. The remaining two operate at, or slightly above, the required processing speed. OrigCPU fails this requirement, operating at a mean processing speed of approximately 6 FPS.

Table 6.6 summarizes the mean performance of all implementations, per sign, over all subjects. The performance ranges of the implementations are 5.90 to 6.41 for OrigCPU, 15.12 to 16.12 for ModCPU, 18.62 to 19.57 for OrigCUDA and 27.13 to 31.75 for ModCUDA.

A repeated measures analysis of variance was carried out to compare the processing speed of the four implementations, the results of which are summarized in Table 6.7. There are differences in the variability of the performance of the different methods. This

Sign	**OrigCPU**		**ModCPU**		**OrigCUDA**		**ModCUDA**	
	Mean	**Std Dev**	**Mean**	**Std Dev**	**Mean**	**Std Dev**	**Mean**	**Std Dev**
Away	6.08	1.70	15.12	3.12	18.62	2.13	28.79	5.98
Bye	5.90	1.61	16.47	2.90	19.15	2.39	30.83	7.22
Crackers	5.82	1.57	15.50	3.28	18.89	2.44	31.75	6.86
Curtains	6.41	1.51	16.43	3.24	19.36	2.22	30.08	6.96
Dress	5.93	1.65	16.45	2.84	18.93	1.86	29.88	6.86
Eat	5.96	1.52	15.75	3.20	19.20	2.36	30.31	6.17
Left	5.83	1.49	15.73	3.23	19.23	2.53	30.53	6.68
Light	6.18	1.32	16.26	3.49	19.04	2.21	29.00	6.54
Love	6.36	1.62	15.68	2.92	19.57	2.37	27.13	5.04
Right	6.16	1.55	16.29	3.33	18.69	2.19	29.14	6.17
Run	6.39	1.59	15.76	3.39	18.92	2.29	30.01	6.67
We	6.13	1.45	15.80	3.48	19.63	2.30	29.10	6.69
Why	6.39	1.42	16.52	2.51	19.03	2.43	29.62	5.27
Wide	5.93	1.54	16.40	2.67	19.00	2.32	28.88	6.23
Average	6.10		16.01		19.09		29.65	

TABLE 6.6: Mean performance of all implementations, per sign, over all subjects.

Implementation	**Estimate**	**Z-Error**	**Value**	***Pr > Z***
ModCUDA	41.11	2.84	14.46	< 0.0001
OrigCUDA	5.19	0.36	14.37	< 0.0001
ModCPU	9.69	0.67	14.42	< 0.0001
OrigCPU	2.37	0.17	14.16	< 0.0001

TABLE 6.7: Results of the repeated measures analysis performed among the four implementations.

variability was accounted for in the analysis. The results show that each implementation is significantly different ($p < 0.0001$) from all of the others. It can be stated with confidence that the descending order of performance is: ModCUDA → OrigCUDA → ModCPU → OrigCPU. Therefore, the use of the CUDA framework and parallel processing techniques had the most significant impact on processing speed, since the two implementations that used these techniques had the highest performance. This shows that the use of the CUDA framework, alone, is effective in providing an increase in processing speed. Following this are the optimizations made, with the two implementations that were optimized performing better than the original algorithms.

The results of the repeated measures analysis of variance also indicate that there is no significant difference among the signs, where $p = 0.85$. This shows that while the performance of every implementation is significantly different, there is a low standard deviation amongst all the signs.

In terms of differences among test subjects, the random effects test was conducted, the results of which are summarized in Table 6.8. The estimated random effects for different test subjects range from approximately -0.03 to +0.03. These values are insignificant

Mean	**Std Dev**	**Minimum**	**Maximum**
1.87379E-15	0.02	-0.03	0.03

TABLE 6.8: Estimated random effects for test subjects.

relative to the overall mean performances, which are in the vicinity of 6 FPS for OrigCPU and 30 FPS for ModCUDA.

It has been shown that three of the four implementations obtain or exceed a mean processing speed of 15 FPS. This indicates that the three implementations generally operate in real-time. However, this does not provide an indication of whether or not the system operates at 15 FPS at every instant. Falling below this value—15 FPS—at any time during processing causes a backlog of frames that require processing. This causes the system to appear unresponsive to the user at that time.

As an example, consider the case where 10 seconds of video are captured from the camera and processed. If the system processes the first 5 seconds of the video at a constant rate of 10 FPS and the remaining 5 seconds of the video at a constant rate of 20 FPS, the mean processing speed indicates a real-time performance of 15 FPS. This indicates that, on average, the system was responsive. However, during the first 5 seconds, a backlog of 5 frames accumulates every second, a total of 25 frames. The responsiveness of the system suffers during this period. In such cases, the minimum instantaneous processing speed provides an accurate indication of whether or not the system has performed in real-time over the entire duration of processing.

Figure 6.4 illustrates the speed at which the slowest frame was processed—the minimum instantaneous processing speed—per sign across all subjects. Obtaining this value involved collecting the frames of all subjects for each sign and comparing the speed at which each individual frame was processed. The frame which was processed at the slowest speed was taken as the minimum instantaneous speed. The red line in the figure indicates the minimum required processing speed of 15 FPS which is necessary to ensure real-time performance.

It can be observed that the only implementation that consistently achieves a minimum FPS of at least 15 FPS is ModCUDA. In fact, ModCUDA achieves a minimum speed which is higher than the requirement—approximately 19 FPS. All other implementations fail this criterion for real-time performance. OrigCPU, ModCPU and OrigCUDA achieve a minimum instantaneous processing speed of approximately 2, 6 and 11 FPS, respectively. To put this into perspective, it should be considered that operating at 2 FPS means that a backlog of 13 out of the 15 frames read in every second—87% of the frames—is accumulated. OrigCUDA accumulates the least backlog of frames among

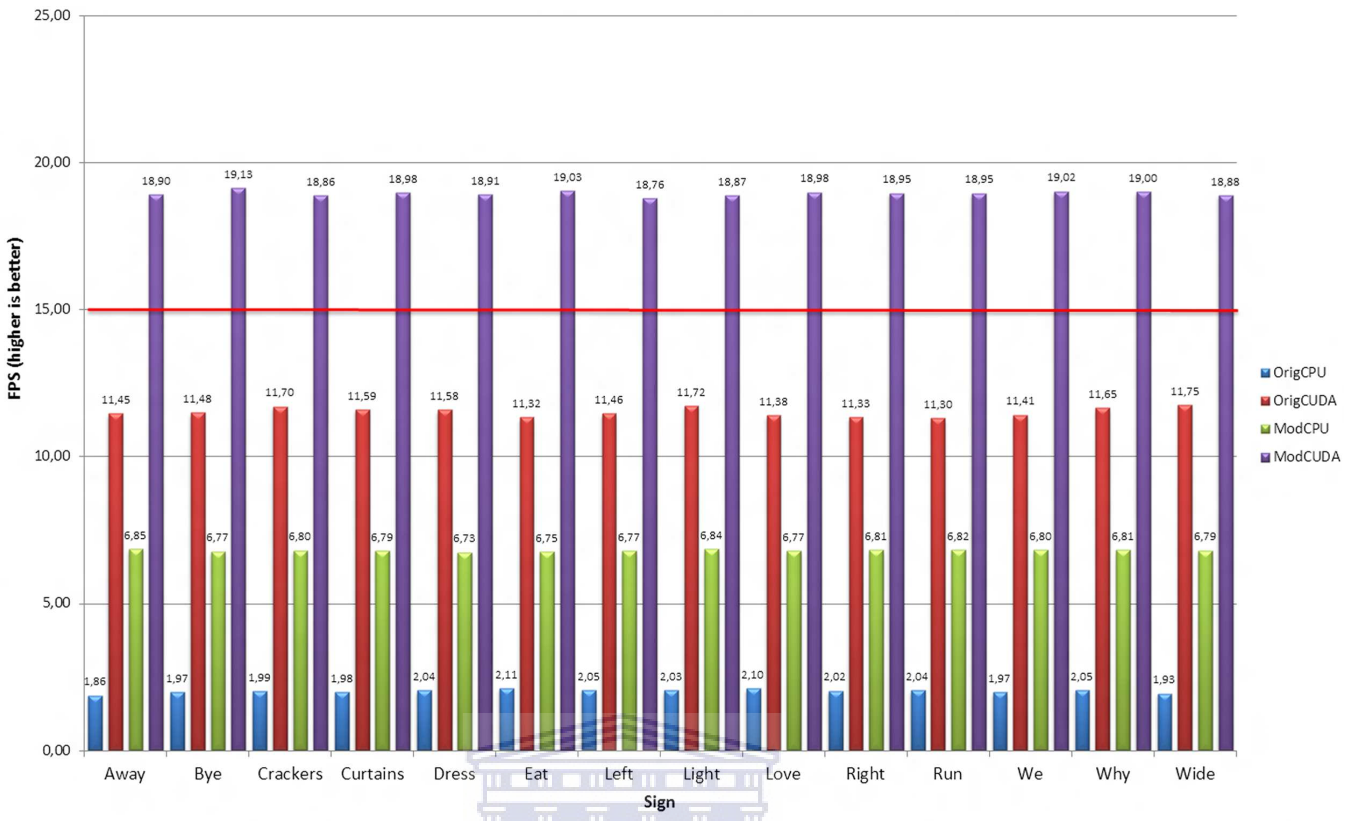


FIGURE 6.4: Minimum instantaneous processing speed of the four implementations.

the three implementations, but still accumulates a backlog of 4 frames per second while operating at this speed. ModCUDA does not suffer from any backlog of unprocessed frames.

Therefore, the only implementation that achieves real-time performance is ModCUDA.

6.4 Summary and Conclusion

This chapter assessed the four implementations of the proposed upper body pose recognition and estimation system. The experimental setup, the data set and the collection of the videos was discussed. The two types of testing conducted were accuracy testing and performance testing. The experimental procedure and metrics used were discussed in both cases.

In accuracy testing, a comparison was made between the the original algorithm and the modified algorithm. On average the modified algorithm achieved a 6% better accuracy than the original algorithm. In performance testing, comparisons were made amongst the four implementations.

It was found that the accuracy of the two modified implementations, ModCPU and ModCUDA, and the two original implementation, OrigCPU and OrigCUDA, were identical. The modified algorithm was found to perform significantly better than the original algorithm in 9 of the 14 signs—an improved accuracy. In the remaining signs, the accuracy of the modified algorithm was no different to that of the original algorithm—a sustained accuracy. Both algorithms were found to be robust to variations in test subjects.

With regards to performance, 3 out of the 4 implementations achieved a mean processing speed above the 15 FPS requirement. OrigCPU failed this requirement. ModCUDA achieved the highest mean processing speed, which was approximately double the requirement and 5 times higher than OrigCPU. This is a significant and extremely successful increase in processing speed over Achmed's original implementation. Furthermore, only ModCUDA achieved a minimum instantaneous processing speed above the 15 FPS requirement. It achieved a minimum instantaneous processing speed well above the requirement—approximately 19 FPS. This is also an extremely successful increase in processing speed over Achmed's original implementation. All three other implementations failed this requirement. Only ModCUDA satisfies the requirement of real-time performance.

The results showed that the use of both parallel processing techniques on the CUDA framework and optimization of Achmed's algorithm are effective in achieving increases in processing speed. It was shown that the use of both parallel processing techniques on the CUDA framework was more effective than the optimizations made to the original algorithm in speeding up the system. However, a combination of both of these techniques was necessary in order to achieve real-time performance.

Therefore, in response to the research question posed in Chapter 1, it can be concluded that optimization and parallel processing techniques on the CUDA framework are highly effective in increasing the processing speed of Achmed's methodology to achieve real-time performance at an improved estimation accuracy in 65% of the signs tested, and a sustained estimation accuracy in the remaining 35% of the signs tested.

Chapter 7

Conclusion

This research has made several crucial contributions to the field of sign language recognition and estimation.

The most significant contribution was the provision of parallel processing techniques and CUDA support to achieve a truly real-time upper body pose recognition and estimation system. It was shown definitively that the use of parallel processing techniques and the CUDA framework provide a considerable improvement in processing speed over and above the performance improvement provided by the optimizations made. This is a major milestone for the machine translation system of the SASL project. Real-time performance is key to the realization of an interactive system and is one of the key requirements of the project. With the conclusion of this research, this crucial requirement has been met.

Another important contribution made was a methodology which can be used to optimize image processing algorithms. A detailed performance analysis of Achmed's upper body pose recognition and estimation system was carried out to determine potential sources of delay in the algorithm. The methodology of this analysis can be used as a guideline by other researchers attempting to analyze similar algorithms. The results of the analysis can also serve as an invaluable source of information when attempting to carry out a further optimization of Achmed's algorithm.

Another significant contribution made was an optimized upper body pose recognition and estimation system with enhanced accuracy and processing speed. The results of the analysis were used to identify potential improvements to the algorithm. This resulted in the creation of optimized face detection, skin detection, background subtraction and resize components. These components provide substantial increases to both the processing speed and accuracy.

7.1 Directions for Future Work

Three directions for future work are provided in the following three subsections.

7.1.1 Porting the Existing SASL Systems to the GPU

The methodology used in this research can be used to increase the processing speed of the existing SASL systems discussed in Chapter 1. This can be used to provide sufficient processing power for the integration of the these systems into one fully-fledged SASL recognition system.

7.1.2 Utilizing Multiple GPUs

Multiple GPUs can be combined to further increase the parallel processing capabilities of the CUDA framework. This can allow the system to make use of more computationally intensive algorithms and provide a general increase in speed.

7.1.3 Investigating the OpenCL Framework

OpenCL has recently developed partial support for OpenCV. This is expected to reach a level of maturity in the near future. The system can be ported to run on OpenCL. The performance of the CUDA and OpenCL frameworks can be compared in this regard.

7.2 Concluding Remarks

The researcher has gained an enormous amount of experience throughout the period of research. Conducting this research served as a reinforcement of the vast capabilities of computer vision. It is hoped that the knowledge passed on in this research will serve as a foundation and aid to further advancements within the SASL project.

Appendix A

Additional Test Results

Bin Width	True Positive Count (%)							
	Subject 1	**Subject 2**	**Subject 3**	**Subject 4**	**Subject 5**	**Subject 6**	**Mean**	**Std Dev**
4	96	89	96	99	81	98	93	6
8	92	85	91	95	76	98	90	7
12	80	80	65	83	65	97	78	11
16	60	60	59	63	56	88	64	11
20	31	57	52	50	45	90	54	18
24	41	51	46	44	36	87	51	17
28	31	45	39	30	33	86	44	19
32	28	37	33	29	32	81	40	19

Table A.1: True positive percentages for skin detection at various bin widths.

Bin Width	True Negative Count (%)							
	Subject 1	**Subject 2**	**Subject 3**	**Subject 4**	**Subject 5**	**Subject 6**	**Mean**	**Std Dev**
4	15	15	15	15	15	15	15	0.12
8	92	91	91	91	91	91	91	0.33
12	96	96	96	96	96	96	96	0.18
16	98	98	98	98	97	98	98	0.11
20	98	98	98	98	98	98	98	0.12
24	98	98	98	98	98	98	98	0.10
28	98	98	98	98	98	98	98	0.11
32	98	98	98	98	98	98	98	0.11

Table A.2: True negative percentages for skin detection at various bin widths.

Component	Test Subject	Away	Bye	Crackers	Curtains	Dress	Eat	Left	Light	Love	Right	Run	We	Why	Wide	Average
	Subject 1	8.11	9.68	8.06	9.50	9.24	8.24	9.00	9.40	8.97	10.04	9.16	9.85	9.11	8.14	9.04
	Subject 2	8.59	10.03	9.21	9.78	8.17	8.45	9.58	8.70	9.34	8.01	8.79	9.84	8.08	8.35	8.92
	Subject 3	9.91	8.04	8.00	8.01	9.00	8.82	9.91	8.66	8.14	9.65	8.34	9.82	9.04	8.13	8.82
Face Detection	Subject 4	9.55	9.16	9.25	9.63	8.54	8.05	9.04	9.41	8.97	8.93	9.97	9.66	9.77	8.28	9.16
	Subject 5	9.16	9.74	9.91	9.34	8.34	10.00	8.63	9.95	8.87	8.55	8.84	8.78	9.49	9.33	9.21
	Subject 6	8.95	8.36	9.66	9.06	8.22	9.98	9.55	8.46	9.49	9.99	8.24	8.35	8.89	9.00	9.01
	Average	9.05	9.17	9.01	9.22	8.58	8.92	9.29	9.10	8.96	9.20	8.89	9.38	9.06	8.54	9.03
	Subject 1	188.83	218.20	209.43	203.50	210.28	206.44	200.22	183.92	206.03	186.02	215.43	180.31	195.99	208.39	200.93
	Subject 2	210.33	212.84	194.06	185.10	188.76	186.47	209.41	192.34	207.10	218.84	219.38	187.25	181.17	219.91	200.93
	Subject 3	210.23	191.75	205.17	219.05	189.85	194.50	202.45	180.03	180.84	182.57	183.94	206.86	188.59	219.38	196.80
Skin Detection	Subject 4	207.17	204.58	207.67	197.40	197.31	181.63	202.51	206.07	188.10	191.82	218.41	215.20	190.56	217.69	201.86
	Subject 5	182.35	191.72	217.50	212.58	203.48	202.56	211.53	213.32	217.06	193.88	213.35	217.90	196.45	217.30	206.50
	Subject 6	204.66	205.04	216.57	191.74	189.52	204.14	209.14	206.83	205.77	191.55	192.80	213.87	203.36	191.11	201.86
	Average	200.60	204.02	208.40	201.56	196.53	195.96	205.88	197.08	200.82	194.11	207.22	203.57	192.69	212.29	201.48
	Subject 1	26.62	27.52	28.59	29.18	26.18	28.62	25.66	28.15	29.71	28.73	27.66	27.83	25.80	27.38	27.69
	Subject 2	28.01	29.70	26.03	28.42	27.94	29.05	29.75	27.75	28.42	25.22	26.47	25.94	27.64	28.50	27.77
	Subject 3	28.28	28.44	25.52	29.91	25.97	29.11	29.08	27.15	27.73	29.74	25.30	27.44	28.47	27.96	27.86
Background Subtraction	Subject 4	25.28	29.27	25.34	28.29	28.97	26.36	26.71	26.91	25.41	26.45	29.66	28.84	26.67	26.13	27.16
	Subject 5	29.77	29.31	29.63	28.06	27.75	25.15	27.96	28.72	29.27	27.04	25.87	27.00	26.78	26.16	27.75
	Subject 6	29.44	25.25	29.12	29.72	29.52	29.46	28.01	28.49	25.82	29.71	25.40	26.23	26.16	25.05	27.67
	Average	27.90	28.25	27.37	28.93	27.72	27.96	27.86	27.86	27.73	27.82	26.72	27.21	26.92	26.86	27.65
	Subject 1	61.57	58.01	59.66	57.57	63.47	57.56	62.54	55.05	56.70	57.30	57.04	59.79	63.54	58.87	59.19
	Subject 2	59.31	58.49	59.38	59.61	62.60	64.67	59.31	56.15	60.24	60.50	63.18	58.78	57.73	63.66	60.26
	Subject 3	61.04	59.06	57.53	57.61	62.07	62.19	60.18	60.54	64.75	57.72	60.59	56.45	60.02	62.63	60.17
Morphological Operations	Subject 4	61.24	58.56	56.50	55.55	62.05	60.89	60.16	59.65	60.55	64.47	60.80	55.79	59.97	58.99	59.65
	Subject 5	59.57	62.70	57.64	55.61	56.77	60.17	58.22	63.84	57.36	63.40	59.38	57.11	56.12	64.97	59.49
	Subject 6	58.56	61.14	62.60	64.80	64.70	64.11	55.35	61.75	59.99	60.50	56.40	55.54	59.97	62.20	60.54
	Average	60.21	59.66	58.89	58.46	61.94	61.60	59.29	59.50	59.93	60.65	59.57	57.24	59.56	61.89	59.88
	Subject 1	101.29	102.72	98.26	105.29	107.49	106.55	105.59	99.39	108.08	103.25	107.62	98.92	98.90	102.05	103.24
	Subject 2	103.87	98.47	107.88	98.31	108.14	108.64	104.62	98.44	103.99	101.43	98.93	102.90	99.32	107.45	103.03
	Subject 3	105.23	100.39	105.40	108.53	105.11	105.66	104.82	103.60	103.21	101.41	104.99	102.30	106.66	103.61	104.35
Resize	Subject 4	103.22	107.56	107.66	98.09	108.03	106.54	98.40	107.17	106.18	105.02	107.61	101.17	108.45	108.54	105.26
	Subject 5	106.07	98.77	106.99	102.31	101.16	103.39	101.83	108.27	100.06	108.65	102.86	105.27	101.05	98.85	103.25
	Subject 6	98.57	98.71	104.46	103.79	108.27	103.11	103.87	107.31	100.65	104.27	105.48	108.83	100.29	104.09	103.69
	Average	103.04	101.10	105.11	102.72	106.37	105.65	103.19	104.03	103.70	104.00	104.58	103.23	102.45	104.10	103.80

Table A.3: The average FPS for OrigCPU of each component, per subject, per sign.

Component	Test Subject	Away	Bye	Crackers	Curtains	Dress	Eat	Left	Light	Love	Right	Run	We	Why	Wide	Average
	Subject 1	102.70	105.84	102.02	106.93	105.81	100.88	105.35	102.82	108.57	102.53	106.44	101.12	102.38	105.89	104.23
	Subject 2	109.13	105.60	101.13	101.84	105.55	100.04	106.80	108.12	108.52	102.64	100.74	101.01	103.60	104.09	104.20
	Subject 3	104.48	109.53	100.80	107.18	105.27	102.82	104.01	100.98	103.70	109.36	103.80	102.17	101.79	100.14	104.00
Face Detection	Subject 4	103.29	104.17	106.02	102.32	109.76	107.15	104.17	105.22	107.19	100.87	103.23	105.61	103.51	103.97	104.75
	Subject 5	106.61	107.11	108.06	101.00	106.54	108.86	108.17	101.71	101.58	102.08	102.69	105.28	101.33	106.49	104.82
	Subject 6	107.45	103.12	106.63	100.64	107.29	102.55	102.97	106.95	109.70	107.13	102.07	106.79	108.00	105.30	105.47
	Average	105.61	105.89	104.11	103.32	106.70	103.72	105.24	104.30	106.54	104.10	103.16	103.66	103.43	104.31	104.58
	Subject 1	214.43	190.52	195.88	185.16	193.87	207.82	209.99	216.02	213.70	190.36	191.84	203.53	210.39	213.11	202.62
	Subject 2	191.13	196.28	190.05	180.99	182.94	198.73	214.51	189.91	193.20	186.10	199.89	203.63	198.81	200.02	194.73
	Subject 3	197.22	191.98	194.92	181.55	202.51	201.80	186.70	216.38	189.52	216.69	212.29	183.11	217.05	214.13	200.42
Skin Detection	Subject 4	206.65	207.34	207.14	207.78	183.52	217.18	208.77	186.46	195.81	203.18	196.37	209.02	209.28	216.26	203.91
	Subject 5	192.54	187.99	196.18	199.76	199.97	202.10	201.31	182.37	183.79	208.01	218.75	193.31	204.60	210.95	198.69
	Subject 6	196.43	201.55	204.98	198.97	188.79	192.01	210.75	202.32	209.09	199.42	198.78	204.91	182.50	205.14	199.69
	Average	199.73	195.94	198.19	192.37	191.93	203.27	205.34	198.91	197.52	200.63	202.99	199.58	203.77	209.93	200.01
	Subject 1	371.27	390.40	364.97	364.09	375.82	369.01	361.22	372.16	373.52	382.08	372.20	383.86	388.54	353.35	373.03
	Subject 2	382.34	365.12	388.78	366.42	364.42	389.41	374.63	368.02	388.98	372.45	384.08	355.07	386.05	372.55	375.59
	Subject 3	362.39	390.82	375.34	383.66	390.22	390.31	356.74	375.04	368.32	367.97	356.20	350.84	359.04	378.40	371.81
Background Subtraction	Subject 4	384.70	356.58	381.75	376.04	371.70	379.53	351.46	386.12	377.94	376.09	363.14	375.92	357.54	356.22	371.05
	Subject 5	380.99	352.60	378.77	352.38	352.41	363.11	386.04	351.63	362.42	351.78	376.67	380.74	369.75	382.87	367.30
	Subject 6	381.58	378.80	370.27	375.28	385.38	361.02	360.32	366.08	390.55	361.79	361.20	377.48	387.88	374.34	373.71
	Average	377.21	372.38	376.65	369.64	373.33	375.40	365.07	369.84	376.95	368.69	368.92	370.65	374.80	369.62	372.08
	Subject 1	61.58	63.00	59.68	55.12	64.03	63.85	64.49	63.70	59.42	58.00	57.39	61.81	60.60	56.46	60.65
	Subject 2	62.26	63.20	58.51	61.33	56.13	64.85	58.23	61.32	56.53	55.11	56.74	61.31	59.25	59.89	59.62
	Subject 3	64.68	55.71	55.38	61.26	63.71	60.06	61.38	62.74	58.91	60.87	61.44	63.33	63.87	63.83	61.23
Morphological Operations	Subject 4	60.14	59.47	55.28	57.40	57.67	58.79	63.73	58.80	58.64	56.96	55.12	60.18	57.07	56.87	58.29
	Subject 5	56.48	61.32	61.76	56.16	62.02	62.14	62.42	60.73	57.20	58.80	58.47	61.11	64.67	64.91	60.59
	Subject 6	59.44	63.54	63.73	64.58	58.01	64.01	56.98	60.68	57.81	55.71	64.48	61.45	57.67	64.60	60.90
	Average	60.76	61.04	59.06	59.31	60.26	62.28	61.20	61.33	58.09	57.57	58.94	61.53	60.52	61.09	60.21
	Subject 1	1685.24	1694.03	1676.64	1677.66	1672.25	1707.32	1671.51	1685.32	1685.65	1671.52	1703.38	1694.79	1689.95	1685.02	1685.74
	Subject 2	1675.11	1692.37	1696.52	1684.08	1696.37	1674.04	1707.02	1706.22	1703.79	1690.57	1688.13	1694.83	1694.28	1684.97	1692.02
	Subject 3	1676.00	1680.20	1705.97	1691.24	1704.24	1672.62	1698.90	1706.49	1709.94	1700.41	1681.81	1685.59	1701.93	1675.19	1692.18
Resize	Subject 4	1670.38	1681.89	1690.21	1675.49	1704.25	1676.72	1689.57	1690.62	1680.77	1686.59	1686.84	1674.56	1707.16	1704.97	1687.14
	Subject 5	1699.39	1691.45	1679.94	1705.39	1701.65	1675.91	1686.63	1695.89	1678.53	1675.53	1692.37	1678.46	1705.94	1704.18	1690.80
	Subject 6	1694.05	1697.87	1709.37	1694.43	1709.76	1689.58	1699.92	1704.01	1696.30	1679.49	1684.63	1707.07	1696.08	1701.46	1697.43
	Average	1683.36	1689.63	1693.11	1688.05	1698.09	1682.70	1692.26	1698.09	1692.50	1684.02	1689.53	1689.22	1699.23	1692.63	1690.89

Table A.4: The average FPS for ModCPU of each component, per subject, per sign.

Sign	**Away**	**Bye**	**Crackers**	**Curtains**	**Dress**	**Eat**	**Left**	**Light**	**Love**	**Right**	**Run**	**We**	**Why**	**Wide**	**Total**
Subject 1	128	103	92	134	90	90	113	117	86	104	86	81	68	102	1394
Subject 2	151	68	89	112	91	57	98	98	73	64	86	91	68	92	1238
Subject 3	124	99	67	92	99	93	69	108	54	93	71	54	91	81	1195
Subject 4	157	84	77	75	90	69	56	91	62	61	92	79	56	52	1101
Subject 5	114	101	116	92	86	156	56	65	64	78	65	63	52	84	1192
Subject 6	167	123	94	102	103	90	66	104	55	61	84	75	62	68	1254
Subject 7	93	97	78	99	76	103	62	96	58	56	73	53	77	61	1082
Subject 8	51	128	103	103	104	141	96	96	57	68	105	92	75	75	1294
Subject 9	74	97	77	138	64	89	65	63	57	79	92	64	61	76	1096
Subject 10	157	73	73	87	94	88	58	84	54	67	88	70	71	71	1135
Subject 11	76	81	94	61	61	55	54	52	65	52	64	65	63	160	1003
Subject 12	107	144	73	133	113	115	88	53	54	54	74	54	63	56	1181
Subject 13	103	160	81	94	91	119	53	74	65	60	83	63	73	59	1178
Subject 14	167	106	83	101	86	101	101	101	51	84	92	53	53	135	1314
Subject 15	98	118	94	90	89	78	67	56	96	63	82	73	74	96	1174
Subject 16	145	59	72	78	105	138	76	69	57	64	86	56	73	82	1160
Subject 17	137	61	93	58	61	75	92	93	63	54	55	57	76	99	1074
Subject 18	134	83	78	62	76	65	93	51	94	54	64	75	84	82	1095
Subject 19	93	70	67	55	56	79	95	96	96	73	54	74	73	52	1033
Subject 20	53	90	63	63	70	61	96	58	96	66	80	74	68	98	1036
Subject 21	145	82	79	105	86	77	63	71	68	53	100	67	51	69	1116
Subject 22	129	86	61	71	71	75	91	59	63	63	61	73	85	68	1056
Subject 23	96	74	62	74	75	74	54	52	62	63	66	52	89	92	985
Subject 24	84	96	57	76	88	95	58	77	56	95	69	61	63	53	1028
Subject 25	88	67	127	64	58	57	75	51	62	73	51	52	76	95	996
Subject 26	75	64	126	59	59	141	56	67	67	52	72	57	59	99	1053
Subject 27	106	75	73	67	68	62	62	67	53	80	56	75	58	57	959
Subject 28	75	75	76	69	69	55	52	68	78	74	73	51	58	98	971
Subject 29	73	93	93	63	61	109	99	78	76	57	52	52	57	66	1029
Subject 30	58	58	98	117	79	107	99	78	65	161	69	54	59	63	1165
Total	3258	2715	2516	2594	2419	2714	2263	2293	2007	2126	2245	1960	2036	2441	33587

Table A.5: Frames per sign video, per subject.

Sign	Away	Bye	Crackers	Curtains	Dress	Eat	Left	Light	Love	Right	Run	We	Why	Wide	Total
Subject 1	120	90	80	120	78	85	113	113	86	98	67	69	61	90	1270
Subject 2	143	63	74	104	85	53	78	86	72	57	70	81	54	75	1093
Subject 3	124	86	56	75	91	81	69	101	54	79	67	49	72	68	1072
Subject 4	139	79	65	60	78	60	51	78	58	46	74	70	53	48	961
Subject 5	101	93	93	83	78	142	49	59	63	69	49	59	44	72	1053
Subject 6	135	105	70	96	96	76	62	97	47	48	71	68	54	60	1086
Subject 7	84	80	69	89	71	91	59	84	56	54	66	45	68	61	977
Subject 8	44	106	87	77	93	119	78	89	50	63	85	78	58	68	1095
Subject 9	65	81	59	114	56	81	64	58	57	64	82	60	58	57	955
Subject 10	132	62	58	70	87	77	47	73	49	51	78	68	50	57	961
Subject 11	70	76	68	57	54	53	46	49	62	49	50	61	52	151	899
Subject 12	95	122	59	122	105	93	79	48	50	45	70	42	51	53	1034
Subject 13	97	148	69	88	75	101	47	65	64	57	74	55	65	48	1052
Subject 14	152	91	59	78	76	89	83	99	50	69	71	53	42	117	1128
Subject 15	91	95	66	77	80	76	60	55	82	49	68	56	64	83	1004
Subject 16	138	54	59	73	85	121	71	65	51	54	76	44	59	66	1016
Subject 17	137	52	71	45	55	75	83	93	54	43	46	41	59	74	931
Subject 18	134	79	55	53	63	59	81	46	93	41	55	67	60	68	955
Subject 19	93	59	59	49	53	72	77	86	96	62	42	74	67	51	940
Subject 20	47	85	52	55	65	57	87	54	85	57	71	64	54	81	914
Subject 21	140	66	55	91	79	74	62	69	65	41	86	51	43	58	979
Subject 22	129	78	51	57	64	69	82	53	62	53	53	53	73	60	937
Subject 23	85	65	51	62	65	67	45	45	59	58	53	40	63	85	842
Subject 24	74	82	40	66	75	87	55	73	56	86	61	57	55	42	908
Subject 25	65	59	100	50	55	47	74	50	57	60	39	48	66	79	849
Subject 26	61	52	93	47	55	140	53	59	61	44	55	53	45	92	911
Subject 27	90	62	55	59	64	61	53	61	48	72	46	62	48	44	825
Subject 28	62	68	55	62	57	45	42	66	72	67	61	48	47	79	830
Subject 29	60	84	70	53	50	102	93	77	66	53	45	46	49	52	900
Subject 30	51	50	77	89	70	100	83	74	57	123	54	51	42	57	977
Total	2959	2371	1976	2222	2158	2451	2027	2127	1882	1809	1887	1711	1676	2098	29354

TABLE A.6: Matches per sign video, per subject for Orig.

Sign	**Away**	**Bye**	**Crackers**	**Curtains**	**Dress**	**Eat**	**Left**	**Light**	**Love**	**Right**	**Run**	**We**	**Why**	**Wide**	**Total**
Subject 1	120	90	80	120	78	85	113	110	86	100	70	70	65	102	1289
Subject 2	138	66	69	110	87	55	98	95	57	64	76	74	67	90	1146
Subject 3	124	99	54	92	94	93	69	97	54	93	69	54	91	81	1163
Subject 4	142	81	59	71	83	68	54	81	61	57	63	67	53	47	987
Subject 5	108	101	96	82	82	147	55	61	57	76	46	52	49	81	1092
Subject 6	151	121	74	92	103	85	64	99	48	56	72	62	60	65	1151
Subject 7	81	92	78	99	71	92	59	87	57	54	65	45	71	61	1012
Subject 8	48	127	80	99	104	125	89	94	51	68	91	74	72	75	1197
Subject 9	60	97	58	127	53	86	64	58	54	73	86	48	59	69	993
Subject 10	149	70	62	83	76	81	56	83	49	60	82	57	65	69	1042
Subject 11	69	77	73	60	49	51	51	47	53	46	52	57	58	149	892
Subject 12	107	144	56	133	94	110	88	53	44	54	56	44	60	56	1099
Subject 13	94	159	63	86	83	111	53	67	53	60	74	57	62	57	1078
Subject 14	152	102	69	97	75	89	101	99	44	84	75	44	50	134	1216
Subject 15	93	117	76	90	72	70	67	51	91	63	66	73	62	85	1077
Subject 16	138	59	61	78	97	138	76	69	56	64	61	54	70	82	1102
Subject 17	137	59	89	58	59	75	92	93	63	54	51	57	76	99	1062
Subject 18	134	79	72	61	76	65	93	51	94	54	63	75	84	80	1080
Subject 19	93	70	67	55	56	79	95	96	96	73	51	74	73	52	1030
Subject 20	47	89	62	63	61	60	96	57	96	58	59	70	64	90	971
Subject 21	140	82	66	105	71	73	63	71	68	53	91	67	51	69	1070
Subject 22	129	86	53	69	69	74	91	59	63	63	45	71	85	66	1023
Subject 23	85	74	62	72	73	70	54	52	61	63	49	50	81	92	938
Subject 24	74	96	47	62	69	88	58	77	51	95	52	61	63	52	947
Subject 25	65	67	107	63	56	44	75	51	62	71	38	52	76	95	922
Subject 26	61	64	93	53	42	141	56	59	67	47	66	57	52	93	952
Subject 27	76	61	57	60	62	59	62	64	50	74	55	67	49	54	850
Subject 28	60	72	60	68	67	54	52	67	71	60	58	43	57	98	888
Subject 29	60	89	77	56	55	108	94	69	66	52	36	42	49	63	915
Subject 30	47	56	82	116	75	106	99	78	51	160	62	52	56	63	1104
Total	2982	2646	2101	2480	2193	2582	2237	2193	1875	2048	1879	1772	1930	2368	31267

Table A.7: Matches per sign video, per subject for Mod.

Bibliography

[1] I. Achmed, "Upper body pose recognition and estimation towards the translation of South African Sign Language," Master's thesis, University of the Western Cape, Computer Science, 2010.

[2] A. Agarwal and B. Triggs, "Recovering 3D human pose from monocular images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, pp. 44–58, 2005.

[3] U. Ahlvers, R. Rajagopalan, and U. Zölzer, "Model-free face detection and head tracking with morphological hole mapping," in *Proceedings of the European Signal Processing Conference*, 2005.

[4] J. Allard and B. Raffin, "A shader-based parallel rendering framework," in *Visualisation, VIS, International Conference*, 2005.

[5] H. K. Almohair, A. R. Ramli, A. M. Elsadig, and S. J. Hashim, "Skin detection in luminance images using threshold technique," *International Journal of The Computer, the Internet and Management*, vol. 15, pp. 25–32, 2007.

[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*, 2004, pp. 268–275.

[7] P. Bakkum and K. Skadron, "Accelerating SQL database operations on a GPU with CUDA," in *Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units*, 2010, pp. 94–103.

[8] A. Baksheev, "OpenCV Wiki," October 2012, [Online] Available at http://opencv.willowgarage.com/wiki/OpenCV_GPU.

[9] M. Bayazit, A. Couture-Beil, and G. Mori, "Real-time motion-based gesture recognition using the GPU," in *Proceedings of the IAPR conference on Machine Vision Applications*, 2009, pp. 9–12.

[10] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002, pp. 509–522.

[11] I. Borovikov, "GPU-acceleration for surgical eye imaging," in *Proceedings of the Fourth SIAM Conference on Mathematics for Industry*, 2009, pp. 37–442.

[12] G. Bradski and A. Kaehler, *Learning OpenCV*. O'Reilly Media, 2008.

[13] J. Brand and J. Mason, "A comparative assessment of three approaches to pixel level human skin-detection," in *Proceedings of the 15th International Conference on Pattern Recognition*, 2000, pp. 1056–1059.

[14] H. Cao, "Example-based methods for estimating 3D human pose from silhouette image using approximate chamfer distance and kernel subspace," Ph.D. dissertation, Nagoya University, Computer Science, 2007.

[15] H. Cao, N. Ohnishi, Y. Takeuchi, T. Matsumoto, and H. Kudo, "Fast human pose retrieval using approximate chamfer distance," *Transactions of the Institute of Electrical Engineers of Japan*, vol. 126, no. 12, pp. 1490–1496, 2006.

[16] D. Chen, P.-C. Chou, C. B. Fookes, and S. Sridharan, "Multi-view human pose estimation using modified five-point skeleton model," in *International Conference on Signal Processing and Communication Systems*, 2008.

[17] Q. Chen, E. Zheng, and Y. Liu, "Pose estimation based on human detection and segmentation," *Science in China Series F: Information Sciences*, vol. 52, no. 2, pp. 244–251, 2009.

[18] A. Crauser, "A parallelization of Dijkstra's shortest path algorithm," in *Proceedings of the 23rd International Symposium on Mathematical Foundations of Computer Science*, 1998, pp. 722–731.

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

[20] F. Dadgostar and A. Sarrafzadeh, "A fast real-time skin detector for video sequences," in *Image Analysis and Recognition*, 2005, pp. 804–811.

[21] F. Dadgostar and A. Sarrafzadeh, "An adaptive real-time skin detector based on hue thresholding: A comparison on two motion tracking methods," *Pattern Recognition Letters*, vol. 27, no. 12, pp. 1342–1352, 2006.

[22] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005, pp. 886–893.

[23] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," in *Proceedings of the 6th Conference on Symposium on Operating Systems Design and Implementation*, 2004.

[24] S. Deane, "A comparison of background subtraction techniques," *IRP Report*, 2007.

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

[26] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient matching of pictorial structures," in *Proc. IEEE Computer Vision and Pattern Recognition Conf.*, 2000, pp. 66–73.

[27] R. B. Fisher, S. Perkins, A. Walker, and E. Wolfart, *HIPR—The Hypermedia Image Processing Reference.* J. Wiley & Sons, 1996.

[28] M. Fleck, D. Forsyth, and C. Bregler, "Finding naked people," in *Proceedings of the 4th European Conference on Computer Vision*, 1996, pp. 593–602.

[29] M. Ghaziasgar, "The use of mobile phones as service-delivery devices in a sign language machine translation system," Master's thesis, University of the Western Cape, Computer Science, 2010.

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

[31] K. Grauman, G. Shakhnarovich, and T. Darrell, "Inferring 3D structure with a statistical image-based shape model," in *Proceedings of the Ninth IEEE International Conference on Computer Vision*, 2003, pp. 641–647.

[32] K. Grochow, S. Martin, A. Hertzmann, and Z. Popović, "Style-based inverse kinematics," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 522–531, 2004.

[33] A. Gupta, A. Mittal, and L. S. Davis, "Constraint integration for efficient multiview pose estimation with self-occlusions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 3, pp. 493–506, 2008.

[34] S. Howard, *Finger Talk: South African Sign Language dictionary.* South Africa: Mondi, 2008.

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

[36] 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.

[37] R. Hsu, M. Abdel-Mottaleb, and A. Jain, "Face detection in color images," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002, pp. 696–706.

[38] G. Hua, M. Yang, and Y. Wu, "Learning to estimate human pose with data driven belief propagation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2005, pp. 747–754.

[39] J. Huang, S. P. Ponce, S. I. Park, Y. Cao, and F. Quek, "GPU-accelerated computation for robust motion tracking using the CUDA framework," in *Visual Information Engineering 5th International Conference*, 2008, pp. 437–442.

[40] M. Huenerfauth, "Generating American Sign Language classifier predicates for english-to-asl machine translation," Ph.D. dissertation, University of Pennsylvania, Computer and Information Science, 2008.

[41] S. Ioffe and D. A. Forsyth, "Probabilistic methods for finding people," *International Journal of Computer Vision*, vol. 43, pp. 45–68, 2001.

[42] T. Jaeggli, E. Koller-Meier, and L. van Gool, "Learning generative models for monocular body pose estimation," in *Proceedings of the 8th Asian conference on Computer vision*, 2007, pp. 608–617.

[43] S. Ju, M. Black, and Y. Yacoob, "Cardboard people: A parametrized model of articulated image motion," in *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*, 1996, pp. 38–44.

[44] P. Kakumanu, S. Makrogiannis, and N. Bourbakis, "A survey of skin-color modeling and detection methods," *Pattern Recognition*, vol. 40, no. 3, pp. 1106–1122, Aug. 2007.

[45] W. Kelly, A. Donnellan, and D. Molloy, "Screening for objectionable images: a review of skin detection techniques," in *Proceedings of the 2008 International Machine Vision and Image Processing Conference*, 2008, pp. 151–158.

[46] D. Kirk and W. Hwu, *Programming Massively Parallel Processors.* Morgan Kaufmann, 2010.

[47] N. Kyriazis, I. Oikonomidis, and A. Argyros, "A GPU-powered computational framework for efficient 3D model-based vision," ICS-FORTH, Tech. Rep., July 2011.

[48] M. W. Lee and I. Cohen, "Human upper body pose estimation in static images," in *Proceedings of the European Conference on Computer Vision*, 2004, pp. 126–138.

[49] B. Leibe, E. Seemann, and B. Schiele, "Pedestrian detection in crowded scenes," in *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005, pp. 878–885.

[50] L. Li, W. Huang, I. Y. H. Gu, and Q. Tian, "Foreground object detection from videos containing complex background," in *Proceedings of the Eleventh ACM international conference on Multimedia*, 2003, p. 10.

[51] P. Li, "Hand shape estimation for South African Sign Language," Master's thesis, University of the Western Cape, Computer Science, 2010.

[52] D. Lim, "Design of a vision system for an autonomous underwater vehicle," Master's thesis, University of Western Australia, Electronic and Computer Engineering, 2004.

[53] J. MacCormick and A. Blake, "A probabilistic exclusion principle for tracking multiple objects," *International Journal of Computer Vision*, vol. 39, no. 1, pp. 57–71, 2000.

[54] M. Martínez-Zarzuela, F. J. Díaz-Pernas, M. Antón-Rodríguez, F. Perozo-Rondón, and D. González-Ortega, "AdaBoost face detection on the GPU using Haar-like features," in *Proceedings of the 4th international conference on Interplay between natural and artificial computation: new challenges on bio-inspired applications - Volume Part II*, 2011, pp. 333–342.

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

[56] A. Micilotta, E. Ong, and R. Bowden, "Real-time upper body 3D pose estimation from a single uncalibrated camera," in *Proceedings of Eurographics, Short Presentations*, 2005, pp. 41–44.

[57] T. Moeslund and E. Granum, "A survey of computer vision based human motion capture," *Computer Vision and Image Understanding*, vol. 81, pp. 231–268, 2001.

[58] G. Mori and J. Malik, "Estimating human body configurations using shape context matching," in *Proceedings of the 7th European Conference on Computer Vision Part III*, 2002, pp. 666–680.

[59] G. Mori and J. Malik, "Recovering 3D human body configurations using shape contexts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 7, pp. 1052–1062, 2005.

[60] G. Mori, X. Ren, A. A. Efros, and J. Malik, "Recovering human body configurations: combining segmentation and recognition," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2004, pp. 326–333.

[61] N. Naidoo, "South African Sign Language recognition using feature vectors and hidden markov models," Master's thesis, University of the Western Cape, Computer Science, 2009.

[62] K. Nallaperumal, S. Ravi, C. N. K. Babu, R. K. Selvakumar, A. L. Fred, S. Christopher, and S. S. Vinsley, "Skin detection using color pixel classification with application to face detection: A comparative study," in *Proceedings of the International Conference on Computational Intelligence and Multimedia Applications*, 2007, pp. 436–441.

[63] W. Noble, "What is a support vector machine?" *Nature Biotechnology*, vol. 24, no. 12, pp. 1565–1567, 2006.

[64] NVIDIA, "CUDA C best practices guide," May 2011, [Online] Available at http://docs.nvidia.com/cuda/cuda-c-best-practices-guide/index.html.

[65] NVIDIA, "What is CUDA?" October 2012, [Online] Available at https://developer.nvidia.com/what-cuda.

[66] V. Oliveira and A. Conci, "Skin detection using HSV color space," in *H. Pedrini and J. Marques de Carvalho, Workshops of Sibgrapi*, 2009.

[67] V. Parameswaran and R. Chellapa, "View independent human body pose estimation from a single perspective image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2004, pp. 16–22.

[68] I. K. Park, M. Germann, M. D. Breitenstein, and H. Pfister, "Fast and automatic object pose estimation for range images on the GPU," *Springer-Verlag, Machine Vision Applications*, vol. 21, no. 5, pp. 749–766, Aug. 2010.

[69] P. Peer and F. Solina, "An automatic human face detection method," in *Proceedings of the Computer Vision Winter Workshop*, 1999, pp. 122–130.

[70] M. Piccardi, "Background subtraction techniques: a review," in *Proceedings of the 2004 IEEE International Conference on Systems*, 2004, pp. 3099–3104.

[71] J. C. Platt, N. Cristianini, and J. Shawe-taylor, "Large margin dags for multiclass classification," in *Advances in Neural Information Processing Systems 12*, 2000, pp. 547–553.

[72] C. Rajah, "Chereme-based recognition of isolated, dynamic gestures from South African Sign Language with Hidden Markov Models," Master's thesis, University of the Western Cape, Computer Science, 2006.

[73] J. Relethford, "Human skin color diversity is highest in sub-Saharan African populations," *Human biology; An International Record of Research*, vol. 72, no. 7, pp. 773–780, 2000.

[74] L. Ren, G. Shakhnarovich, J. Hodgins, H. Pfister, and P. Viola, "Learning silhouette features for control of human motion," in *ACM SIGGRAPH 2004 Sketches*, 2004, p. 129.

[75] X. Ren, E. C. Berg, and J. Malik, "Recovering human body configurations using pairwise constraints between parts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 1, pp. 824–831, 2005.

[76] G. Rizk and D. Lavenier, "GPU-accelerated RNA folding algorithm," in *Proceedings of the 9th International Conference on Computational Science: Part I*, 2009, pp. 1004–1013.

[77] T. Roberts, S. McKenna, and I. Ricketts, "Human pose estimation using learnt probabilistic region similarities and partial configurations," *European Conference on Computer Vision*, vol. 29, no. 9, pp. 291–303, 2004.

[78] R. Ronfard, C. Schmid, and B. Triggs, "Learning to parse pictures of people," in *Proceedings of the 7th European Conference on Computer Vision-Part IV*, 2002, pp. 700–714.

[79] R. Rosales and S. Sclaroff, "Specialized mappings and the estimation of human body pose from a single image," in *Proceedings of the Workshop on Human Motion. IEEE Computer Society*, 2000, pp. 19–25.

[80] J. Sanders and E. Kandrot, *CUDA by Example: An Introduction to General-Purpose GPU Programming.* Addison-Wesley Professional, 2010.

[81] A. Sen-ching and C. Kamath, "Robust techniques for background subtraction in urban traffic video," in *Proceedings of the SPIE*, 2004, p. 881.

[82] G. Shakhnarovich, P. Viola, and T. Darrell, "Estimating human body configurations using shape context matching," in *Proceedings of the Ninth IEEE International Conference on Computer Vision*, 2003, pp. 750–757.

[83] S. Shelly and J. Schneck, *The complete idiots guide to learning sign language.* Alpha Books, 1998.

[84] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[85] M. Shin, K. Chang, and L. Tsap, "Does colorspace transformation make any difference on skin detection?" in *Proceedings of the 6th IEEE Workshop on Applications of Computer Vision*, 2002, pp. 275–279.

[86] L. S. Sigal, M. Isard, B. H. Sigelman, and M. J. Black, "Attractive people: assembling loose-limbed models using non-parametric belief propagation," in *Advances in Neural Information Processing System*, 2004, pp. 1539–1546.

[87] L. Sigaland and S. Sclaroff, "Estimation and prediction of evolving color distributions for skin segmentation under varying illumination," Boston University, Tech. Rep., 1999.

[88] W. Skarbek and A. Koschan, "Colour image segmentation: A survey," University of Berlin, Tech. Rep., 1994.

[89] C. Sminchisescu and B. Triggs, "Estimating articulated human motion with covariance scaled sampling," *International Journal of Robotics Research*, vol. 22, no. 6, p. 371, 2003.

[90] P. Srinivasan and J. Shi, "Bottom-up recognition and parsing of the human body," in *Proceedings of the 6th International Conference on Energy Minimization Methods in Computer Vision and Pattern Recognition*, 2007, pp. 824–831.

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

[92] J. Sullivan and S. Carlsson, "Recognizing and tracking human action," in *European Conference on Computer Vision*, 2002, pp. 629–644.

[93] M. R. Tabassum, A. U. Gias, M. M. Kamal, H. M. Muctadir, M. Ibrahim, A. K. Shakir, A. Imran, S. Islamm, M. G. Rabbani, S. M. Khaled, M. S. Islam, and Z. Begum, "Comparative study of statistical skin detection algorithms for sub-continental human images," University of Dhaka, Tech. Rep., 2010.

[94] C. Taylor, "Reconstruction of articulated objects from point correspondences in a single uncalibrated image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2000, pp. 677–684.

[95] A. Thayananthan, R. Navaratnam, B. Stenger, P. H. S. Torr, and R. Cipolla, "Pose estimation and tracking using multivariate regression," *Pattern Recognition Letters*, vol. 29, no. 9, pp. 1302–1310, 2008.

[96] S. Theodoridis and K. Koutroumbas, *Pattern Recognition.* Academic Press-Elsevier, 2003.

[97] M. Tipping and A. Faul, "Fast marginal likelihood maximisation for sparse bayesian models," in *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, 2003, pp. 3–6.

[98] N. Tsumura, R. Usuba, T. Nakaguchi, M. Shiraishi, N. Ojima, N. Okiyama, K. Takase, K. Hori, S. Okaguchi, and Y. Miyake, "Real-time image-based control of skin melanin texture," in *ACM SIGGRAPH 2005 Sketches*, 2005.

[99] A. Tzotsos, "A support vector machine approach for object based image analysis," in *Proceedings of the 1st International Conference on Object-based Image Analysis*, 2006, pp. 3099–3104.

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

[101] G. Venter, "Particle swarm optimization," in *American Institute of Aeronautics and Astronautics Journal*, 2002, pp. 1583–1589.

[102] V. Vezhnevets, V. Sazonov, and A. Andreeva, "A survey on pixel-based skin color detection techniques," in *Proceedings of the Graphicon*, 2003, pp. 85–92.

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

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

[105] B. D. Zarit, B. J. Super, and F. K. H. Quek, "Comparison of five color models in skin pixel classification," in *Proceedings of the International Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems*, 1999, pp. 58–66.

[106] J. Zhang, R. Collins, and Y. Liu, "Body localization in still images using hierarchical models and hybrid search," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006, pp. 1536–1543.

[107] Y. Zhuge, Y. Cao, J. K. Udupa, and R. W. Miller, "GPU-acceleration for surgical eye imaging," in *Annual Internation Conference of the IEEE Engineering in Medicine and Biology Society*, 2009, pp. 6341–6344.