Hand Gesture Recognition for Multimedia Applications

by

Moaath Al-Rajab

Submitted in accordance with the requirements for the degree of
Doctor of Philosophy

UNIVERSITY OF LEEDS

School of Computing

University of Leeds

November, 2008

Abstract

Hand gesture is potentially a very natural and useful modality for human-machine interaction. It is considered to be one of the most complicated and interesting challenges in computer vision due to its articulated structure and environmental variations. Solving such challenges requires robust hand detection, feature description, and viewpoint invariant classification.

This thesis introduces several steps to tackle these challenges and applies them in a hand-gesture-based application (a game) to demonstrate the proposed approach. Techniques on new feature description, hand gesture detection and viewpoint invariant methods are explored and evaluated. A normal webcam is used in the research as input device. Hands are segmented using pre-trained skin colour models and tracked using the CAMShift tracker. Moment invariants are used as a shape descriptor.

A new approach utilising the Zernike Velocity Moments (ZVMs, first introduced by Shutler and Nixon [1, 2]), is examined on hand gestures. Results obtained using the ZVMs as spatial-temporal descriptor are compared to an HMM with Zernike moments (ZMs). Manually isolated hand gestures are used as input to the ZVM descriptor which generates vectors of features that are classified using a regression classifier. The performance of ZVM is evaluated using isolated, user-independent and user-dependent data.

Isolating (segmenting) the gesture manually from a video stream for gesture recognition is a research proposition only and real life scenarios require an automatic hand gesture detection mechanism. Two methods for detecting gestures are examined. Firstly, hand gesture detection is performed using a sliding window which segments sequences of frames and then evaluates them against pre-trained HMMs. Secondly, the set of class-specific HMMs is combined into a single HMM and the Viterbi algorithm is then used to find the optimal sequence of gestures.

Finally, the thesis proposes a flexible application that provides the user with options to perform the gesture from different viewpoints. A usable hand gesture recognition system should be able to cope with such viewpoint variations. To solve this problem, a new approach is introduced which makes use of 3D models of hand gestures (not postures) for generating projections. A virtual arm with 3D models of real hands is created. After that, virtual movements of the hand are simulated using animation software and projected from different viewpoints. Using a multi-Gaussian HMM, the

system is trained on the projected sequences. Each set of hand gesture projections is marked with its specific class and used to train the single multi-class HMM with gestures across different viewpoints.

Acknowledgments

First of all, I would like to thank my supervisors Prof. David Hogg and Dr. Kia Ng for introducing me to the exciting field of computer vision and multimedia, for their support and stimulation, and for providing me with a firm source of knowledge whenever I needed it. My thanks go also to University of Leeds Vision Group for providing a stimulating research environment, for their inspiring enthusiasm and support.

This work would not be the same without the support from all my friends. In particular, thanks to Areej Shehab, Hazem Godda, Jill Bullock, Mohamad Zewani, Kenan khoury, Najeed Khan, Sonia Khoury and Hazem Hairy.

Special thanks to my friends Maisaa, Taha, Krishna and Tracy for their support and for the proofreading. Without you this thesis will never be done.

Many thanks to the Al'Baath University, Homs, Syria for granting me this scholarship and opportunity.

I thank my family for the support and understanding. Thanks to my wonderful angel, friend and wife, Kinda, for her endless love. I cannot remember you without smiling. At last, huge thanks and love to the only one who worried about my PhD more than anything else in the world, my father. You are my motivation which helped me to carry on with this research.

Declarations

Some parts of the work presented in this thesis have been published in the following articles.

- M. Al-Rajab, D. Hogg, and K. Ng, "A Comparative Study on Using Zernike Velocity Moments and Hidden Markov Models for Hand Gesture Recognition" in *Articulated Motion and Deformable Objects (AMDO'08)*. pp. 319-327: LNCS, Springer Berlin / Heidelberg, 2008.

- M. Al-Rajab and K. Ng, "A Novel Data Representation Method for 3D Model-Based Gesture Tracking," in *The 3rd International Conference on Information & Communication Technologies: from Theory to Applications (ICTTA'08)*. pp. 1085- 1090: IEEE Computer Society Press, 2006.

- M. Al-Rajab and K. Ng, "gHand: Pointing in an Interactive Multimedia Environment", *in: Second International Conference on Automating Production of Cross Media Content for Multi-channel Distribution Conference (AXMEDIS `06)*, pp. 31-37: Firenze University Press. 2006.

- M. Al-Rajab and K. Ng, "Framework of a Real-time Interactive Audio-video Registration & Recognition system", *International Conference on Electronic Information, the Visual Arts and Beyond (EVA2005),* London, 2005.

Contents

Table of Figures

CHAPTER 1

INTRODUCTION

Interest in hand gesture recognition has increased in recent years, motivated largely by the range of potential applications for human-machine interaction. Within the wide range of application scenarios, hand gestures can be classified into at least four categories [3]: conversational gestures, controlling gestures, manipulative gestures, and communicative gestures. Hand gestures are powerful human interface components. However, their fluency and intuitiveness have not been exploited and utilised as computer input. Recently, hand gesture applications have started to emerge but they are still not robust and are unable to recognise the gestures in a convenient and easily accessible manner by the user. Many advanced methods are still either too fragile or too coarse grained to be of any universal use for hand gesture recognition. In particular, methods for hand gesture interfaces must be improved beyond current performance in terms of speed and robustness to achieve the required interactivity and usability.

Recognizing hand gestures automatically from visual input is a complex task. It typically involves several stages including, signal processing, tracking, shape description, motion analysis, and pattern recognition.

1.1 Motivation

Hands are the most multipurpose tools to accomplish our daily tasks. They are the driving force in our user control interface and interaction application. This research is motivated largely by recent advancements in user-machine interactions. The Nintendo Wii 2005 console [5], Microsoft surface [4] and other similar applications have received a warm welcome from users. Computer games are already a big industry. Having these computer games driven by physical movements is expected to be widely encouraged by users.

For proper evaluation of hand gesture recognition systems, several datasets have been collected by different research groups. In large part, the current databases [6-9], existing for static postures or dynamic gestures, are of a limited use. Existing databases mainly consist of single handed gestures captured with limited motion, controlled lighting, consistent background, no body movements and/or coloured gloves to ease the tracking and segmentation [6-10]. There are datasets [11, 12] that do contain more expressive hand gestures relevant to a multimedia interfaces but these have not been made publicly available.

Figure 1-1: Wii console is used by the elderly for staying fit [13]

The emerging technologies and the limitations of existing datasets have contributed to our thinking of how and for what purpose new datasets should be collected and how the prototype application should be designed. Interactive user-interfaces can be used not only in computer games but also in crisis management [14], helping people with

disabilities (e.g sign language) [15-17], or for elderly people [18]. In this research, the proposed prototype is developed to be in line with applications that can be used for the elderly. Recent trends on using the Wii console to help the elderly lose weight and remain fit partly motivate this research and its direction (see Figure 1-1).

1.2 Research focus

The primary focus of this research is to create a framework for a user-interface that utilizes state of the art methods and can be used within a range of applications. To illustrate this, the developed techniques are demonstrated within a simple game. The developed techniques were not to create new segmentation, tracking, representation, or classification methods but rather to evaluate novel combinations of methods for a gesture-based multimedia interface, and to explore continuous gesture detection and viewpoint invariance.

The thesis brings together several independent techniques into one framework. Figure 1-2 illustrates the position of our research in relation to other types of study.

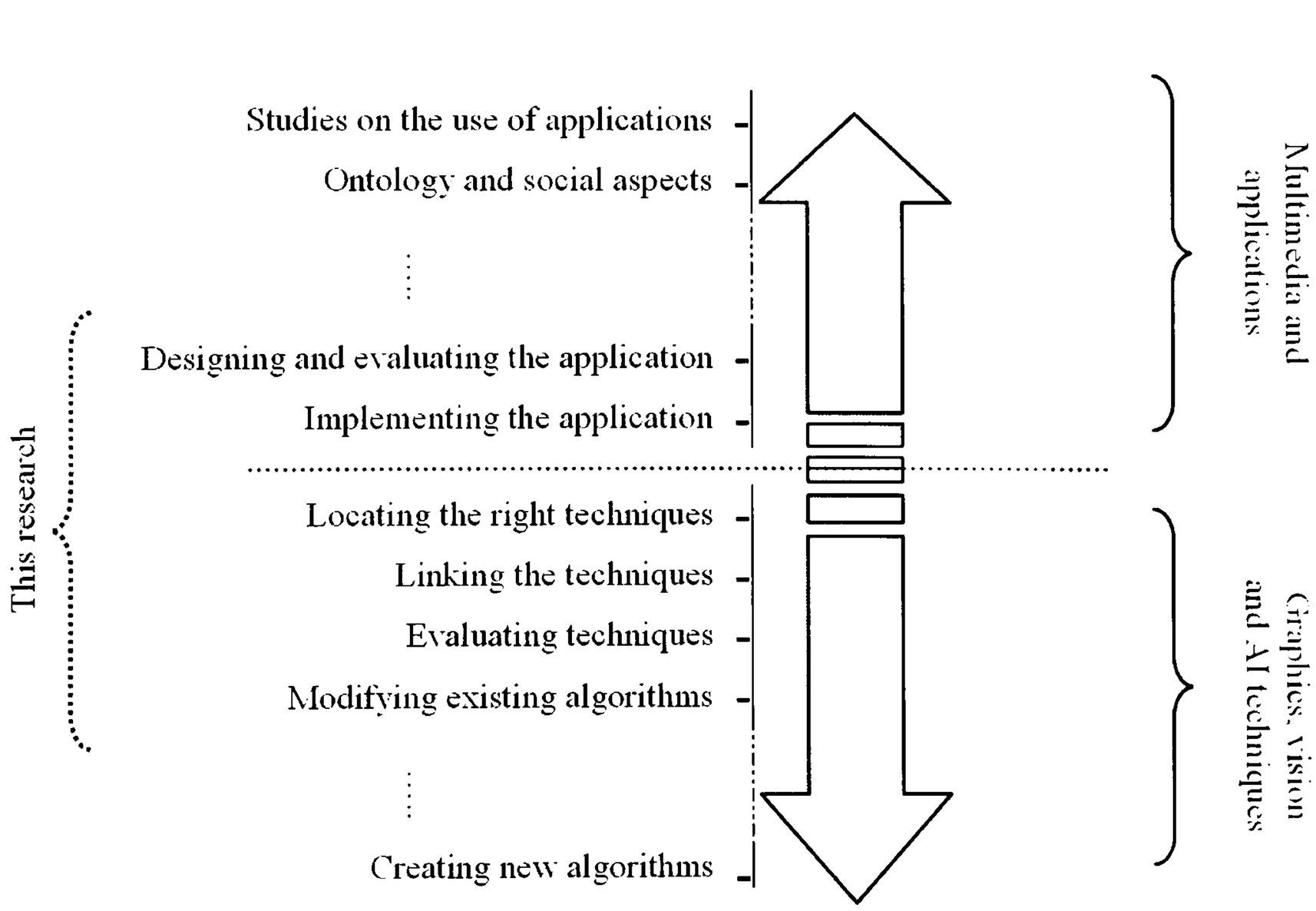


Figure 1-2: research focus diagram: this shows to where this research belongs in terms of techniques and applications

The research adopts an appearance-based approach [22-25] for processing hand gesture with special attention to feature representation and gesture classification methods. A hybrid approach (see Section 2.4.2.3) is adopted for tackling viewpoint invariance.

1.3 Research overview

The thesis evaluates a recent method for image feature description within a new context and introduces a novel method for dealing with variation in viewpoint. The thesis also presents a new discussion on hand gesture detection. To demonstrate the overall utility of the approach, a framework of a game is designed as an example of possible applications on hand gestures. The game has an entertaining chasing scenario between two characters (*Bird* and *Ant*). The player (game user) drives and controls the *Ant* while the *Bird* tries to touch the *Ant*. The 3D environment used in this game is simple as the focus is on the control more than the setting of the game. A normal webcam is used as the input device; webcams are included in most state of the art laptops and some computers by default which makes the game accessible and easy to use. From the webcam video stream, human hands are segmented using pre-trained models of the skin colour, then the hand is tracked using the CAMShift tracker [19, 20]. The foreground blob is then represented by a descriptor.

A key requirement when selecting a method for shape description is to provide sufficient discriminative information for successful classification of hand gestures. The use of moments for shape description was introduced by Hu [21] in 1962. Hu introduced a set of six functions of standard central moments which provide a description that is invariant to scaling, translation and rotation, and a seventh function invariant to scaling, translation and skew. Another form of moments is the Zernike moments (ZMs) where the kernel is a set of orthogonal Zernike polynomials defined over polar coordinates inside a unit circle. ZMs are the projection of the image function onto these orthogonal basis functions. This idea has been extended recently by Shutler and Nixon [1] who added the displacements of the center of mass to ZMs. This incorporates the velocity of the moving objects into a descriptor. They called this extension Zernike Velocity Moments (ZVMs) which they explored successfully with human gait recognition [1]. A motivation for this thesis was to use the ZVM descriptor in hand gesture systems and to evaluate its performance. The thesis compares the performance of ZVM coupled with a static classifier to ZMs coupled with a bank of HMMs. The static

classifier used is Classification via Regression (CvR) which was selected after exploring other classifiers. The performance of several experiments is presented including experiments that use isolated hand gestures, use-dependent and user-independent data.

In real life scenarios, users do not perform a gesture alone but rather the hand gesture is part of a continuous video stream. Therefore, gesture detection (spotting) is an essential part of this research. Gesture detection in a continuous video stream refers to finding the start and end of a gesture. Two methods based on the use of an HMM (Hidden Markov Models) for detecting hand gestures are presented with extended discussion and evaluation. The first method is a sliding window which takes a number of frames of a gesture and evaluates them against a set of pre-trained HMM models. The second method uses the Viterbi algorithm to find the optimal path that a certain gesture may follow using a single HMM constructed from each of the pre-trained HMM models.

Proposing a flexible system to use requires providing the user with options to perform the gesture from different viewpoints. A hand gesture recognition system should be able to cope with such variations. To solve this problem, a multi-class classification technique is trained using a set of animated gestures with their projections. These animated gestures are obtained by creating a virtual performance of the hand using the captured 3D models and animation software. The virtually created hand gestures are projected from different viewpoints. Using multi-Gaussian HMMs, the system is trained on the projected sequences. Each set of hand gesture projections is marked with its specific class for training. Results and discussion are presented.

1.4 Thesis contributions

Original contributions produced from this work emerge from several parts:

- Four new datasets have been designed for evaluating hand gesture recognition in different environments. The data is collected in the following settings: *(1)* a uniform background with controlled lighting (UBL dataset); *(2)* hands only visible in office environment (OEH dataset); *(3)* whole body visible in the office environment (OEW dataset); *(4)* virtual performance of hand gestures using 3D models. Ground truth of the data is provided using manual labelling.

- A prototype has been implemented showing how the set of controlling gestures could be used in practice (see Chapter 3). This prototype is a simple attractive chasing game involving two characters, one of which is manipulated by the user using gestures.

- An Evaluation on the newly developed datasets has been presented using a state of the art detection and tracking method (CAMShift).

- An Evaluation of ZVM introduced by [1, 2] for human gait recognition has been carried out for the hand gesture datasets.

- The ZVM is compared with a standard method used in hand gesture recognition. Until now, the ZVM descriptor has not been compared to any of the state of the art methods on identical datasets.

- Two techniques for hand gesture detection are presented with the aim to locate the start and the end of hand gestures automatically and to classify these gestures. This research also explores experiments utilizing HMMs with garbage states [26].

- Virtual hand gesture performances have been proposed for dealing with viewpoint variation. Virtual (animated) hand gestures are generated using 3D models and animation software. 3D models are captured from real hands using a laser scanner. Then a multi-class detector and classifier is trained and utilized.

1.5 Thesis structure

- **Chapter 2** presents related background and a literature review on hand gesture recognition systems. It starts with defining the meaning of hand gesture and discussing existing applications, then introduces the gesture categorization criteria. A survey on hand gesture systems is conducted in the context of multimedia. The related topics of machine vision to hand gesture detection are also reviewed.

- **Chapter 3** presents the developed datasets used in this thesis. It discusses settings and the prototype application used as a demonstration of the techniques explored.

- **Chapter 4** presents segmentation, tracking and feature representation methods. For our prototype to work, the most distinctive features of the gesture such as hand shape, trajectory, and colour need to be extracted and represented efficiently. This chapter also evaluates the use of the CAMShift tracker on the new datasets.

- **Chapter 5** presents and discusses several experiments. Firstly, hand posture recognition; Secondly, ZVMs representation + static classifier; Finally, ZMs + HMMs classification. A comparative study on both hand gesture experiments is presented.

- **Chapter 6** presents in the first part, experiments on hand gesture detection using HMMs trained using the pre-segmented gestures. The second part explores the use of 3D models for viewpoint invariant hand gesture recognition. The used 3D models have been captured from real hands and then manipulated using 3D animation software to create a viewpoint independent dataset of virtual hand gestures.

- **Chapter 7** presents summary and findings of this research. It also discusses the limitation of the system design, possible improvements and future work.

CHAPTER 2

LITERATURE REVIEW

2.1 Chapter overview

This chapter presents related background and a literature review on hand gesture recognition systems. It starts by defining the meaning of hand gestures and discusses possible applications. Next, categories of hand gesture recognition system are introduced. These categories are divided by a system's context and technology. The lion's share of the chapter deals with vision-based methods and their suitability for implementing user interfaces and multimedia applications, with special focus on appearance-based methods. Several studies have been explored in the relevant sections. Finally, results and findings are presented.

2.2 Understanding hand gesture recognition systems

2.2.1 What are hand gestures?

Biologists broadly define the term *gesture* as all kinds of instances where an individual engages in movements such that their communicative intent is manifested and openly recognized [27]. Psycholinguists define the term *gesture* as the "*critical link between our conceptualizing capacities and our linguistic abilities*" [28]. Gesture is one part of the natural aspects of human communication where humans use a range of movements from simple finger actions of pointing at objects (*Deictic Gestures*) to the more articulated motions that express feelings and allow communication with others. The communicative feature of the gesture boosts its potential role in multimedia applications [10, 28-33].

Hand gestures and speech are often associated with each other. This relation has attracted many researchers' attention. McNeill states that "*statures modify their gestures to maintain synchrony with speech, and equally, that deliberate mismatch between gesture and speech can influence a participant's recall of a narration*" [34]. Gesture and speech operate as an inseparable unit [35], reflecting different semiotic aspects of the cognitive structure that underlies both of them. This has been reflected in research on the relations between gesture and speech that are studied by Turk [36] who concluded that there is a close relation including the fact that unsmooth verbal speech disrupts gesture.

2.2.2 Hand gesture recognition

By going through early literature of hand gesture recognition systems, a few terms used within the literature as research aims can be highlighted. These terms distinguish this type of research from other research: such as "no mouse", "no keyboard", "natural interactive input methods", "effective sign language recognition", "manipulation and control gestures", "device-less remote control". None of these terms has departed from the laboratory environment to touch daily life and they were mainly a proof of concept. Computer vision and artificial intelligent challenges such as segmentation, tracking, classification and viewpoint invariance have narrowed the use of such studies on natural interactive systems.

However, the success of the *Nintendo Wii 2005 console* [5] against the well-established Sony PlayStation and MS Xbox, and more recently the Microsoft interactive table *"Microsoft surface 2007"* [4] have shown the real potential of the emerging technology. The *Nintendo Wii* system has a station which captures the motion of a handheld wireless device. The *Microsoft surface* does not require any handheld device but touching its surface is essential for the system to work. These emerging technologies help to motivate the researchers who are working on hand gesture recognition systems because such devices have extended the limited use of mouse and keyboard to explore more natural ways of interaction. The previously mentioned systems are a step closer to a flexible system which is proposed in this research.

Flexible interactive systems enable the user to interact with machines without the need to have any physical contact with the input device just like Nintendo Wii, and not to have any handheld or marker tools as with Microsoft Surface.

2.2.3 Hand gesture recognition systems' categories

Several related studies have attempted to classify gestures such as [28, 37-40]. Most of these surveys did not target a specific gesture of the body, but rather explored and classified gestures from any part of the body whether it is an eye gesture [41], hand, or head gesture [42]. Research on classifying hand gesture according to the context is limited.

This section explores hand gestures and their types and applications within a multimedia context. It aims to extend some of the related surveys such as [43]. It is well understood that hand gesture is an important example of a human gesture. However, hand gestures have a special nature due to the articulated motion and high number of degrees of freedom.

A hand gesture can be categorized according to its temporal information into:

- Static gestures (some researchers call these postures), and
- Dynamic gestures

Hand gesture recognition can be divided as follows:

- The context of the gesture (sign language, control gestures)
- The technology used in gesture recognitions (non-vision, vision)

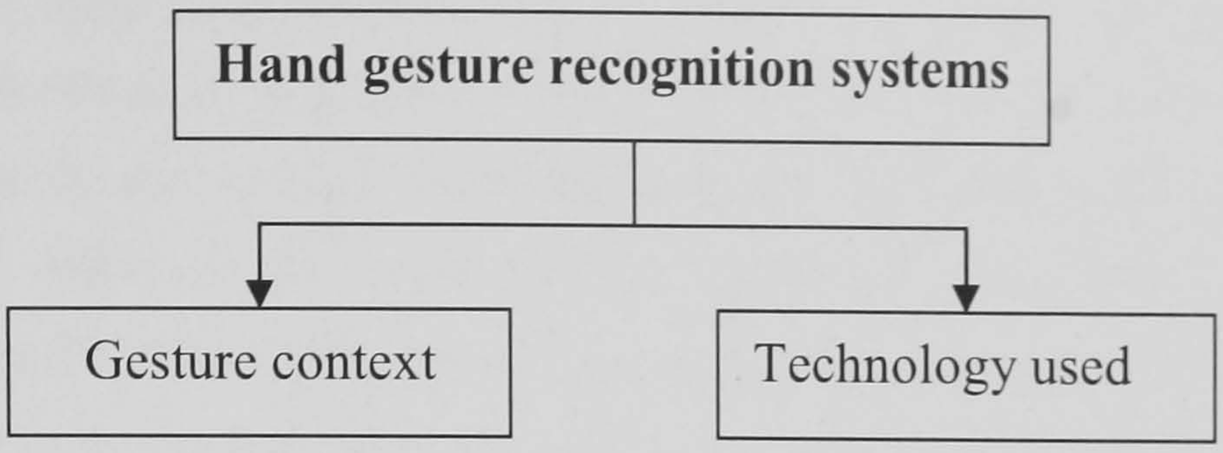


Figure 2-1: Hand gesture recognition systems can be divided and classified according to two criteria, the context of the hand gesture and the technology used for the recognition

2.3 Hand gesture recognition by context

Hand gestures recognition systems are classified according to their context and application into three types:

- Communicational gestures
- Manipulation gestures
- Interactive gestures

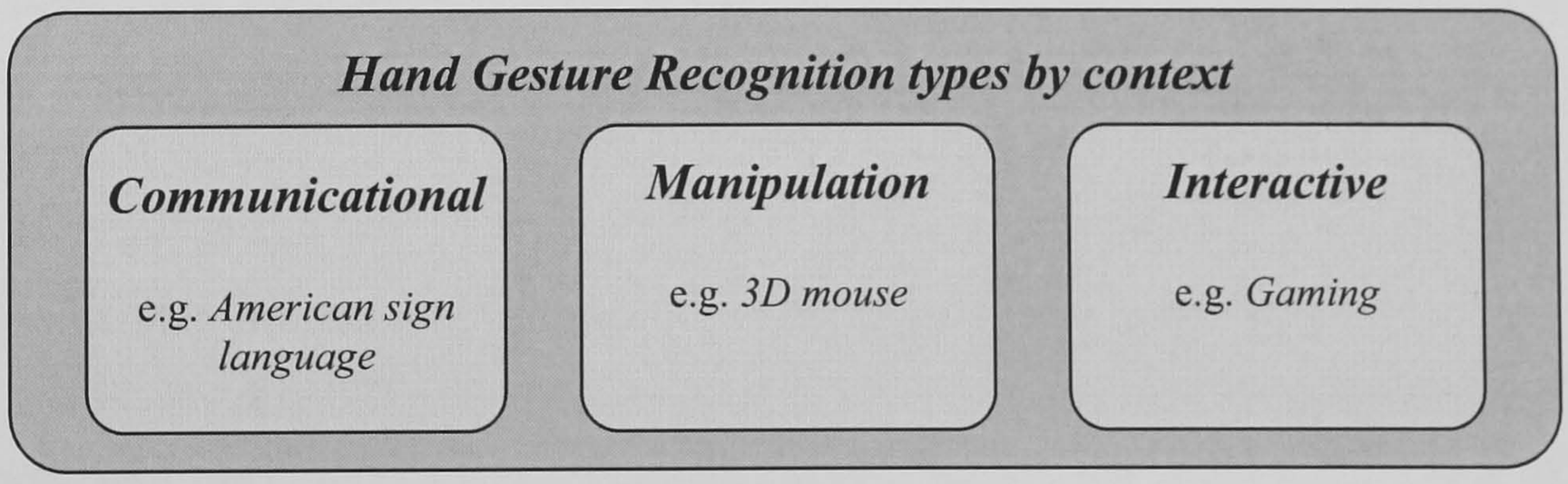


Figure 2-2: The main three classes of hand gestures according to their context and application

2.3.1 Communicational hand gestures

Communicational gestures are known as sign language gestures that are considered in some literature to be independent from other types of gesture. Sign language is linguistic-based and requires the collective interpretation of multiple, individual hand signs that are combined to form grammatical structures [22, 44-46]. Sign language

gestures are the most well-studied field in hand gesture recognition systems. Sign language extends beyond hand movements to include mouth and arm movements. Unfortunately, it is not a universal language and different countries have different vocabulary; e.g. British Sign Language (BSL) and American Sign Language (ASL). There are several research studies on forming sign language in a written way such as the Sign Writing system [47] which provides a special font (Sutton).

Sign language can be categorized into three types [25]:

a) *Finger spelling* which is used to spell a word by letters or numbers. It is usually static postures that refer to meaning [45, 48-50].
b) *Word sign,* this is the most common form of sign language. A certain motion of the hand refers to a word [17, 25, 29, 50, 51]
c) *Non-manual features* which usually involve tongue, hand and/or body position [15, 16, 39].

2.3.2 Control and manipulation hand gestures

Control and manipulation gestures are widely explored in several applications. There are creative products which aim to improve the human-computer interaction. These applications include car gadgets control as in Zobl *et al.* [52], hand drawing as in Isard and MacCormick [53, 54], large display map navigation control as in Carbini *et al.* and Wilson [31, 55], and virtual reality control as in Weissmann [56].

Zobl *et al.* [52] presented a multimodal system, which consists of a gesture-optimised user interface, a real time gesture recognition system and an adaptive help system for gesture input that is aimed to be used in a car. This system supports eleven dynamic hand gesture classes and four hand poses. They used an adaptive background removal technique to segment and threshold the images. After filtering these images with a forearm filter, Hu-moments [21] with HMM are computed. Their system was trained and tested by using data of one person and thus is highly person dependent.

Isard and MacCormick [53, 54] presented a system which uses the condensation algorithm [57] for tracking the fingertip to utilize a drawing application. The system assumes a simple controlled environment and manual initial localization of the fingertip is required. More recently, Sepehri *et al.* [58] presented a flexible approach for a hand gesture drawing system which deals with gesture in 3D space using a stereo camera to

capture a disparity map. They approximated the planer of the hand for both the disparities and motion model. They have not conducted user studies to assess the performance and fatigue that may occur in long term use.

A large display application driven by using hand gestures is explored by Carbini *et al.* [31] who also used a stereo camera as input device. A skin-colour tracker has been used for detecting the head and hands. They divided the space between the human body and the display into zones to minimize the 3D search space of the hand gesture. They used one hand for gesture and another for controlling the start and the end of their application. More studies on large display control such as the use of drag and drop objects can be found in Collomb [59], where several techniques on large display systems are compared and discussed.

Wilson [55] presented a computer vision technique to detect when the user brings their thumb and forefinger together (a pinch gesture) for close-range and relatively controlled viewing circumstances. This system used a single camera fitted on the screen against a black background (keyboard). The technique avoided complex and fragile hand tracking algorithms by detecting the hole formed when the thumb and forefinger are touching each other; this hole is found by simple analysis using the connected components analysis technique [60] of the hand segmented against the background.

Malik *et al.* [61] developed a vision-based hand tracking system over a constrained tabletop surface area to perform multi-finger and whole-hand gestural interactions with large displays from a distance. They designed a set of static and dynamic gestures as input to their system, where both hands were used for the interaction.

Tangible interfaces and gestures are another kind of control and manipulation input. Mazalek and Nitsche [62] built a physical object that can be manipulated to control corresponding digital objects. The physical object drives a game through a set of sensors on the built object.

2.3.3 Interactive hand gestures

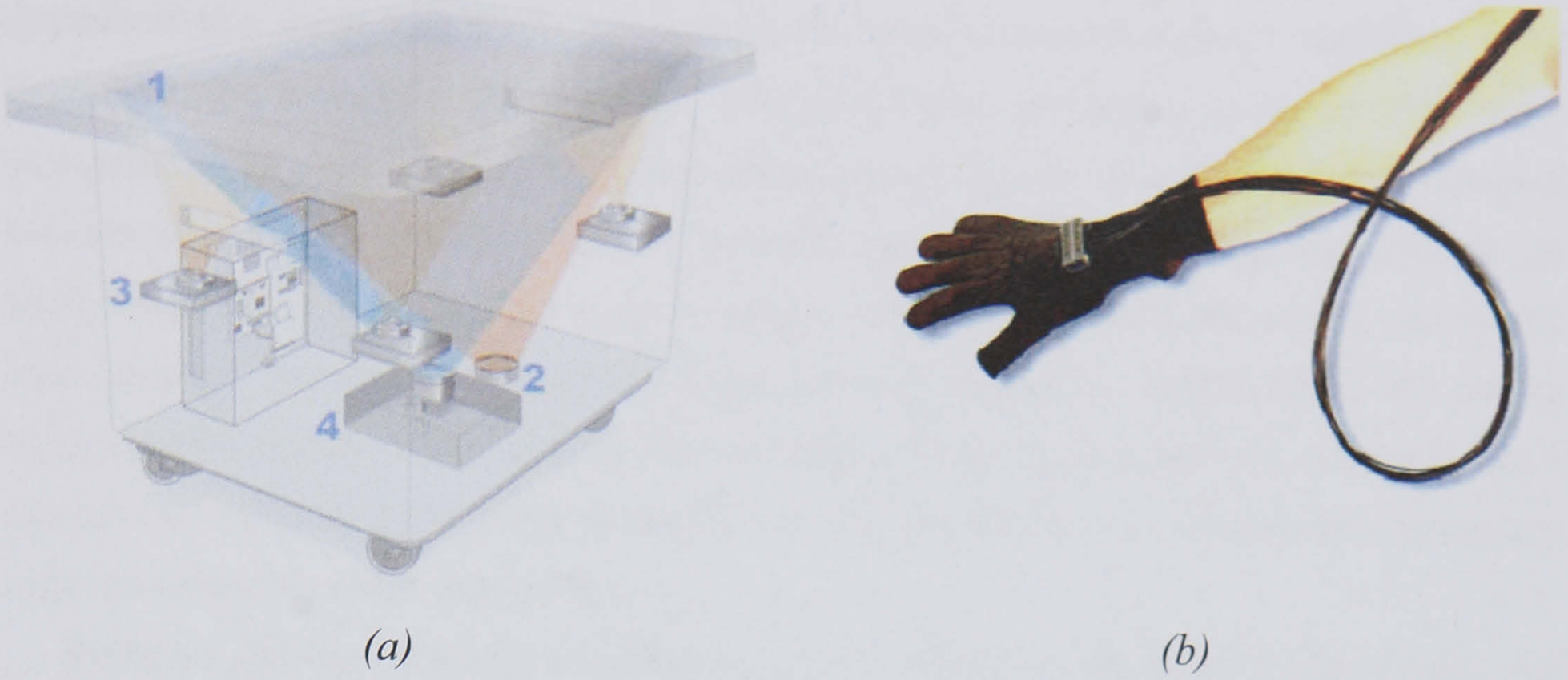
Hand gestures are considered to be interactive when users require changing the way they perform the gesture according to the application's feedback. This is mainly in real-time applications. Most games that use hand gestures are considered to be interactive ones. It is worth mentioning that interactive gestures and control gestures are similar. The feedback in the interactive gesture distinguishes this type.

There are several studies on this type of gestures including Wilson and Oliver [63]. They created a stereo-vision based system called "*GWindow*". This system utilizes not only hand gesture but also speech recognition for performing several basic *Microsoft Window* management tasks. A year later Wilson [64] presented a touch screen technology called "*TouchLight*" which uses image processing techniques to combine the output of two video cameras placed behind a semi-transparent plane in front of the user. The resulting image shows objects that are on the plane. The use of two cameras helps to filter out unwanted objects. Only objects on the surface give the same projected image. A microphone is used to detect the clicking events. A projector is used to project the resulting image on the screen.

Similarly, Ahn *et al.*[65] presented a system for interacting with a large display using a live video avatar of a user. The system enables the user to appear on a screen as a video avatar and to interact with items therein through hand gestures. The user can interact with the large display remotely by walking and touching icons through his video avatar. They developed live video composition, active infrared vision-based hand gesture recognition, 3D human body tracking system, and incorporated a voice recognition system. Surveys on interactive gesture can be found in [14, 36, 39-41, 66, 67].

2.4 Hand gesture recognition by technology

Reviewing research on hand gesture recognition and general gesture recognition has led to a categorization based on the used technology into non-vision-based methods and vision-based methods. Non-vision-based methods use all types of sensors apart from cameras to detect hand gestures. Examples of such sensors are "the magnetic" which is for detecting the 3D location of hand gesture, "touch sensors", "data-glove" and so on. Vision-based methods use cameras as the input device.



(a) (b)

Figure 2-3: (a) Microsoft Surface diagram [68]. 1) Screen 2) Infrared cameras 3) CPU 4) Projector. Number of infrared cameras depends on the design and size of the table. (b) 5DT Data-glove [69] capturing device.

2.4.1 Non-vision-based hand gesture recognition

2.4.1.1 Touch technology

One of the recent products of touch technology is the *Microsoft Surface* [4] which allowed more human computer interaction through gesture - see Figure 2-3-(a). Users can "grab" digital information with their hands and interact with content through hand movements, without the use of a mouse or keyboard. The touch technology used in the *Microsoft Surface* is based on multi-infrared cameras. Other applications and products utilized other touch technologies based on resistive media, ultrasonic waves, capacitance, and optical imaging [70]. Touch screens were in use many years ago, such as the surface notebook, mobile phones and interactive white boards [64].

2.4.1.2 Data-gloves and virtual reality

Data-gloves are physical devices attached to a computer either via wire or wireless. They are wearable gloves usually made of a fabric, see Figure 2-3 (b). These gloves contain several sensors placed in such a way that they can transfer fingers movements into signals that represent the difference between various hand shapes [37, 43, 71]. Different techniques have been used to track the changes in the hand position and orientation. These techniques depend heavily on the nature of the sensor used in the glove. LaViola's technical report [43] on hand gesture and posture recognition

described the main and common techniques used in the data-glove method. These techniques are magnetic, acoustic, and inertial tracking. Magnetic based devices have a transmitting device that emits a low-frequency magnetic field where the receiver verifies its position and orientation. Acoustic based devices or ultrasonic devices use high-frequency sound emitted from a source component that is placed on the glove. Microphones placed in the environment receive ultrasonic pings from the source components to verify their location and orientation. Finally, inertial based devices use a variety of inertial measurement sensors such as gyroscopes, servo-accelerometers, angular velocity sensor and others.

Different data-gloves have a different number of sensors installed on the fabric, and the most common type [40, 43] is the 5DT Data-Glove [72] which supports up to 5 degrees of freedom (DOF) and has 16 sensors installed. The technology of the sensors varies. Tarchanidis [73] proposed a data-glove design which captures not only the hand movements, but also the movement force through installing a force sensor. The accuracy of data-glove depends on the analogue-digital converter resolution (the higher the better). Data-glove and physical drafted hand (*e.g.* wooden model) capturing methods are used usually for virtual reality, military and medical applications [74-76].

Non-vision-based technology depends more on hardware-based solutions for capturing gesture motion. This usually results in a complexity in the system. In addition, many potential users cannot afford to buy such a technology. In contrast, researches utilize more complex computational algorithms which tend to offer a simpler interactive interface that has satisfactory acceptance from the end user.

This research aims to provide easy-to-use and affordable (low cost) systems. Therefore, this literature survey is directed towards the vision-based methods and applications.

2.4.2 Vision-based hand gesture recognition

Vision-based methods use the video camera as an input. This can be a single camera, stereo or multiple cameras depending on the application and setting. Vision-based methods have some advantages in comparison with non-vision-based methods, several surveys [43, 75, 77] have discussed these issues in detail. The advantages of vision-based methods are summarized as follows:

Accessibility and portability: Non-vision-based approaches require wearable devices (such as the data-glove), or direct contact devices (such as the touch screen),

whereas vision-based devices can be applied to some extent in any condition and anywhere. In vision-based approaches, however, the hand's size does not make any difference whereas in glove-based it does. Wearing the glove for a while can cause discomfort to some users.

Usability: It is widely known that systems with limited or free calibration usually gain more acceptance from the end user than others. Some non-vision-based applications such as the data-glove require an initial calibration while vision-based systems permit a much more flexible environment. The other aspect of the usability is how the user can cope with the system.

Affordability: Vision-based approaches provide a cost effective solution for gesture recognition while the non-vision-based systems such as the data-glove and the touch screen are much more expensive.

It can be concluded according to this comparison, that vision-based techniques have the edged over non-vision-based techniques. This comes at the expense of more complex algorithms for handling common computer vision challenges such as lighting changes, background instability, noise, collision, tracking, and recognition techniques. There has been a huge body of work on vision-based techniques to solve these challenges.

Vision-based techniques can be divided into three categories:

- Model-based
- Appearance-based
- Hybrid-based

2.4.2.1 Model-based approaches

A model-based hand gesture system generally utilizes 3D models (see Figure 2.6) of the hand. Figure 2.4 shows the general structure of the model-based approach. The general idea of this approach is to map the 3D models' parameters to correspond with an imaged hand, either for tracking proposes or for classification. Video input is composed of images that contain 2D information when a single camera is used. Therefore, the 3D model is projected and after that, a fitting function is used to compute and minimize the error mapping between the 3D-to-2D hand projections and the tracked hand. Model parameters are optimised using the output of the fitting

function and another projection is computed according to the error value. A successful hand fitting is a result of an accurate description of the tracked object and a correct classification of its actions.

The geometric hand model is usually created either synthetically, as in Figure 2.5, or can be obtained by reconstruction/scanning methods, as in Figure 2.6 which was captured using the Polhemus laser scan system [78]. The 3D model can be represented in various ways, for example deformable polygon meshes [79] or generalized cylinders [80]. Generally, the projected hand parameters (out of the 3D model) are estimated separately from the parameters of the hand in the input video stream but can be also estimated together as in [81]. This section highlights some of the research that has adopted a model-based approach.

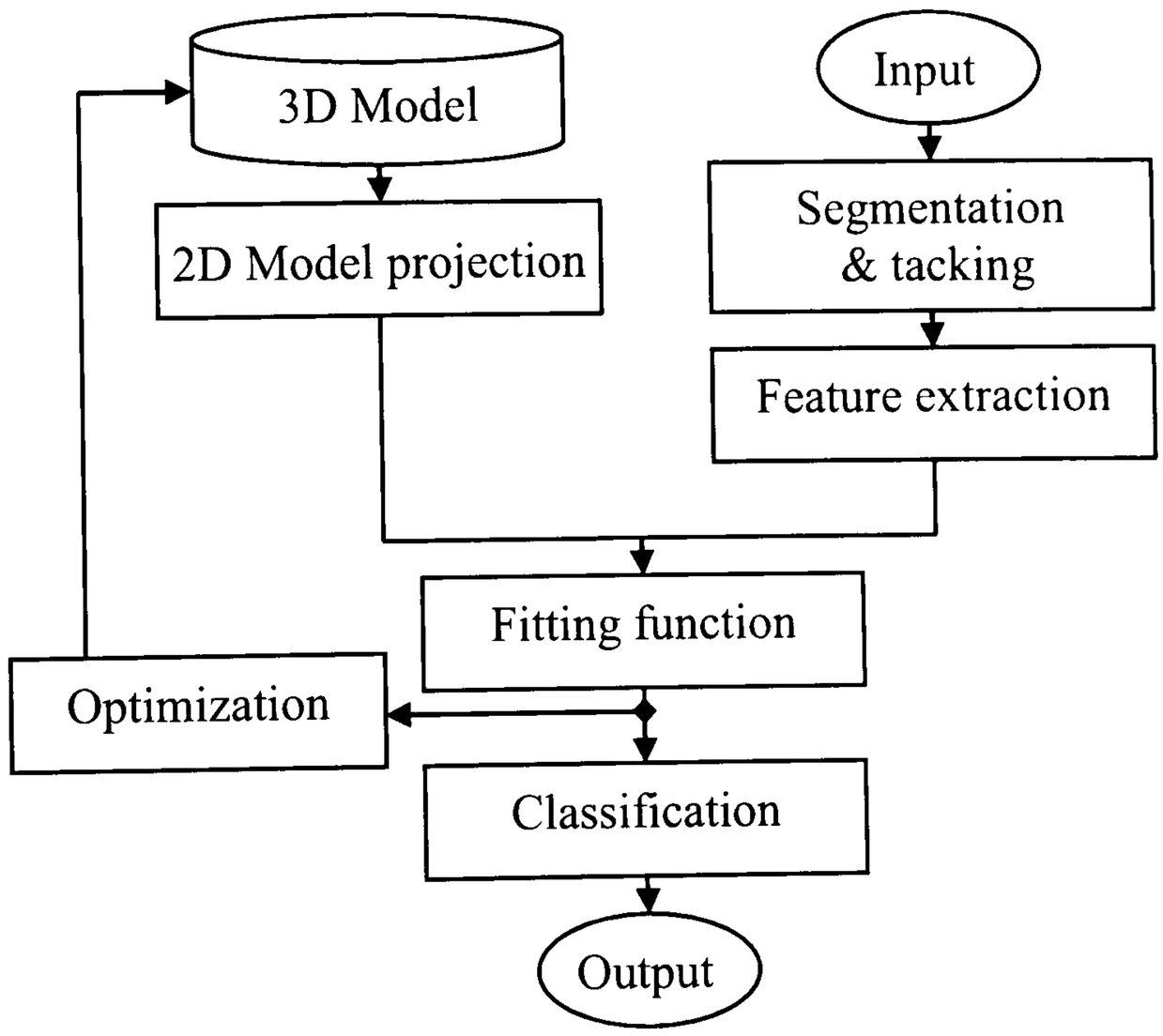


Figure 2-4: Flowchart depicts the overall structure of a model-based hand recognition system using a 3D model.

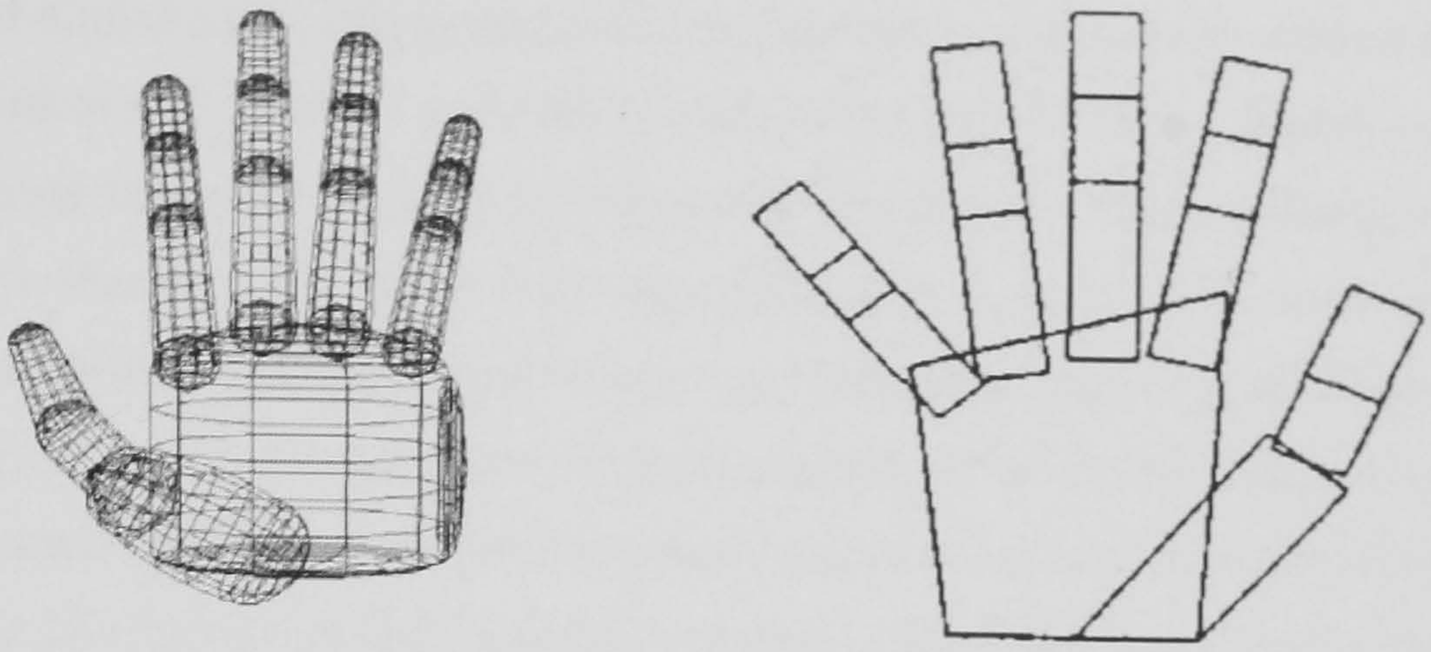


Figure 2-5: Two examples of manually created models (a) is a 3D model (b) is a 2D model [82]

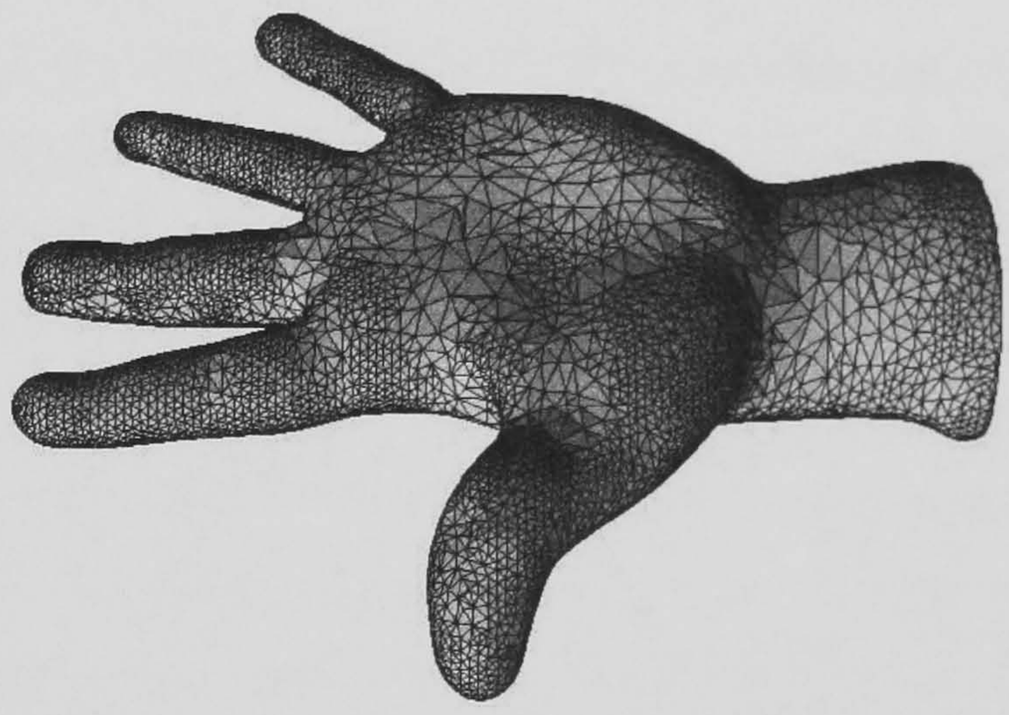


Figure 2-6: 3D Hand model of real hand scanned using a Polhemus laser scanner [83]

Model-based approaches have been used for tracking [79, 84] and dealing with viewpoint variations [85]. Using a Point Distribution Model (PDM) of the human hand, Heap and Hogg [79, 84] constructed a 3D deformable model using a statistical analysis of MRI images. They calculated the mean of these models for a specific hand, and then they calculated the variation of the models using Principal Component Analysis (PCA) (discussed further in Section 2.5.2). Using the constructed model, they tracked the hand in the scene using the projections of the 3D model. Model location parameters are used as an initial location for the next frame and so on; each time the change in the position and hand shape are calculated to deform the model accordingly. During the tracking, a uniform background is used.

Rehg and Kanade [81, 86] presented their DigitEyes framework, which utilized a 16 cylinder hand model (fingers and palm) with a total of 27 DOFs. Initial configuration was predefined using an estimation algorithm which first predicts feature parameters for the next frame, and afterwards the difference between the measured and the predicted states is minimized using inverse Jacobean mapping from the displacement in feature space into the displacement in hand configuration space (angular space). They proposed a tracking framework for articulated objects that uses explicit kinematic hand models. The kinematic models provide geometric constraints on image features. The real-time tracking systems use finger tip features and are tested to predict self-occlusion. They tried to resolve the ambiguity caused by the occlusion in [81], which occurs when the motion is directed along the viewing axis of a single camera during the hand tracking or estimation.

Lien [80] extended the work of Heap [79, 84] and Rehg [81, 86] to tackle the problem of 3D model scalability. Lien's system used a set of markers on the end of each finger and on the wrist. The 3D model is generated using articulated cylinders for the fingers; Lien tracked the markers on the hand to configure and calibrate the 3D model states. This calibration allows scaling the model using a fitting function. Lien's scalable model-based hand posture analysis system addressed marker occlusion problems in the conventional inverse kinematic algorithm. They estimated occluded markers by studying the trajectory of the hand motion. These markers are linked to the model's cylinders. To overcome the use of hand markers for constraining the movements of these cylinders, Wu and Huang [87] presented a method for capturing the articulated hand motion. The constraints of joint configurations are learned from natural hand motions with a data-glove. Dimensionality reduction techniques are used to minimize the 20 dimensions to a seven dimensional subspace by using PCA, which maintains 95% of the variation in their training data, similar to Heap and Hogg [79, 84]. Lien, Wu and Regh built their model using ridged parts representing fingers and palm as in Figure 2.5.

In a similar approach, Sudderth *et al.* [88] introduced probabilistic methods for visual tracking of a three-dimensional geometric hand model from monocular image sequences. Model components are represented by their positions and orientations. A prior model is then defined to enforce the kinematic constraints implied by the model's joints. Mapping is processed using a redundant representation that allows a colour and edge-based likelihood Chamfer distance measure [89]. Given this graphical model of hand kinematics, they tracked the hand's motion using the Non-parametric Belief

Propagation (NBP) algorithm. A KD[1]-tree based method is used to improve NBP's computational efficiency for fast Chamfer distance evaluation.

3D model projections are used in the model-based approaches to map the tracked hand to its corresponding model. This mapping requires estimating the position of the hand in a frame and mapping it to the model, then updating the states. These techniques are sometimes referred to as the "analysis-by-synthesis approach", which Shimada *et al.* [23, 90] adopted. Their system has two stages. *Firstly*, with a 23 DOFs 3D model, they generated 125 possible 2D projections from 128 viewpoints and achieved a total of 16,000 frames. To simplify the search, they built a transition network between the shape models. *Secondly*, 3D models candidates were projected to the image plane and the discrepancy of the contour of projected image and input image is evaluated. Extended Kalman Filter (EKF, in a similar way to Stenger's hand tracker [91]) is utilized to find the best match for fitting the model to the image. EKF is a recursive filter that estimates the state of dynamic system from noisy data. The system is complicated and cannot be applied to real-time applications. A Trajectory and history-based method was introduced by Morrison and McKenna [92]. They used skin colour as a common visual cue, and recognition methods based on hidden Markov models, moment features and normalised template matching. They proposed skin history images as history-based representation and reported results on a database of sixty gestures.

Athitsos and Sclaroff [85] used a hierarchical retrieval system by rejecting the vast majority of the hand shapes and then ranking the remaining candidates according to the order of similarity to the input using Chamfer distance [89, 93]. Four different similarity measures are employed based on edge location, edge orientation, finger location and geometric moments. Their dataset consists of 26 3D hand-shaped prototypes. They projected each of these prototypes from 86 viewpoints. They calculated the Hu moments of the projected images and stored these feature vectors in a database. After that, they applied PCA on their database to find the main eigenvectors. The created 3D model consists of 16 links representing the palm, and 15 segments corresponding to the fingers' parts; their models have 20 DOFs.

It should be mentioned that Erola's survey [82] on pose estimation provides extended details on model fitting and initialization methods.

---

[1] KD-trees: K-Dimensional trees

2.4.2.2 Appearance-based approaches

In appearance-based methods, there is no use of 3D models. These methods are solely based on the 2D input and analysis. Figure 2-8 depicts the overall structure of this method. Segmentation and feature extraction play a major role in the success of the overall system, and are also considered to be more suitable for real-time applications.

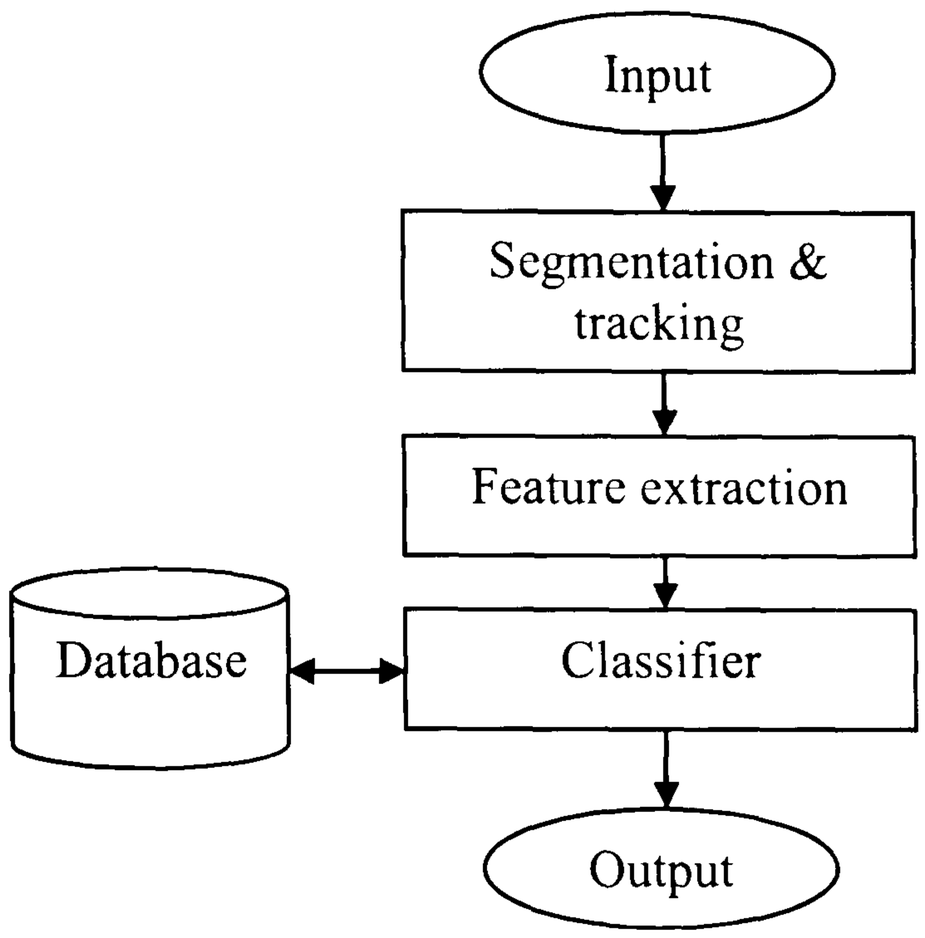


Figure 2-7: general structure of an appearance-based hand recognition system

This research deploys appearance-based techniques and algorithm. Therefore, Section 2.5 is dedicated to discuss in depth each of the stages in Figure 2.7.

2.4.2.3 Hybrid-based approaches

There is an interesting third category of approaches on vision-based system that combines appearance- and model-based methods. This approach uses the 3D models to train an appearance-based detector. The difference between hybrid- and model-based methods is that in the first one the 3D model is used only for training the system and then an appearance-based technique is deployed. In model-based methods, the 3D model is an essential part of the system during all stages of the analysis as discussed in Heap and Hogg [79, 84]. Figure 2-8 illustrates the general structure of hand gesture systems that adopt these approaches.

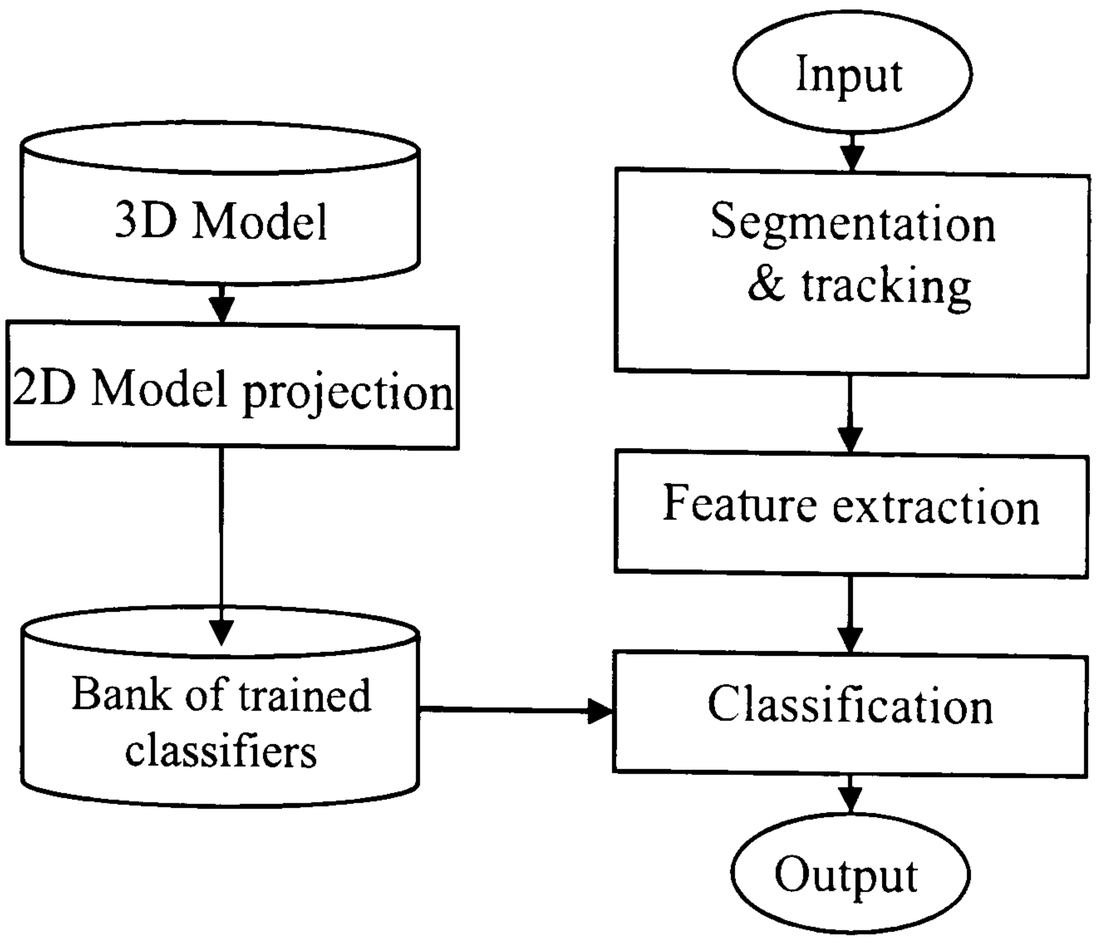


Figure 2-8: Hybrid-based gesture recognition' general structure

This approach is adopted by Zhou *et al.* [89, 93] who introduced a fast indexing system by using a single hand image. By treating local visual features as terms, training images as documents and input images as queries, they formulated the problem of object recognition into that of text mining. Such formulation opens up the opportunity to integrate some powerful data mining tools into machine vision. They trained the system using the projection of 3D model which generates different hand shapes that can be mapped to a single state. They aimed at improving the efficiency of the articulated object recognition by using an Okapi-Chamfer matching algorithm, which is based on the inverted index technique. To enable inverted indexing in an image database, Zhou *et al.* built a lexicon of local visual features by clustering the features extracted from the training images. Given a query image, visual features were extracted and quantized according to the lexicon, and then looked up in the inverted index to identify the subset of training images with non-zero matching score. The evaluation of the matching was performed by combining the modified Okapi weighting formula with the Chamfer distance. The performance of the Okapi-Chamfer matching algorithm is evaluated on hand postures recognition. As reported, the system was slow to be applied to real-time systems - 3 sec per query on an average sized dataset of training images.

Stenger *et al.* [94] proposed a model-based hand tracking system using a hierarchical Bayesian filter. They used a 3D geometric hand model which was constructed from truncated cones, cylinders and ellipsoids. The projections of the 3D model were used to map the model using the contour's edge to the real hand in the input video stream based

on skin colour. The hand tracking problem is formulated as state estimation. They used two approaches for tracking: one is an unscented Kalman filter [91]: the other is tree-based filtering which is a hierarchical Bayesian filter that allows integration of temporal information.

2.5 More on appearance-based approaches

Designing a gesture recognition system requires several important elements to be considered. Hand recognition tasks determine the mathematical model that may consider either or both spatial and temporal characteristics of the hand gestures. The approaches used for recognition play a crucial role in the nature and performance of a gesture recognition system. These approaches have their roots in computer vision. Once the gesture is segmented, tracking methods should be deployed for extracting hand gesture characteristics and parameters. Then a descriptor needs to be used with the control obtained from the tracking stage. A descriptor should produce features to represent the description of the hand pose or trajectory depending on the modelling approach that has been used. We can highlight the common stages that most systems use:

a) Hand tracking and segmentation
b) Feature extraction and representation
c) Hand gesture detection (locating the start and end)

Using the output of the descriptor, gestures can be classified and interpreted according to specific models and some grammar rules that reflect the internal syntax of gestural commands. The grammar may also encode the interaction of gestures with other communication modes such as speech, gaze, or facial expressions. Two more analysis aspects can be added to the previous three:

d) Classification method
e) Grammar roles

One of the most challenging problems of hand gesture recognition systems is detecting and classifying the gesture from different viewpoints. The literature has very limited material on viewpoint invariant hand gesture recognition, but has more material on posture viewpoint invariant systems. Using the 3D model is one of the most successful methods to deal with viewpoint in hand posture [94, 95]. From the 3D

model, several 2D projections can be generated and then a direct mapping between the real video input and the generated projection can be processed. Chapter 6 provides extended study on hand gesture and viewpoint.

2.5.1 Hand tracking and segmentation

It is an essential requirement for any hand gesture recognition system to detect the hand in the input video stream, segment the foreground from the background and track it in each frame of the input stream. Skin colour segmentation is one of the most common methods for locating the hand for several reasons:

- Hand gestures are more likely to be performed without gloves.
- Skin colour has the same colour value (in an HSV domain) and different human races' skin-colour differs in the brightness value [92].
- It is easy to implement and has effective execution time; with skin-colour segmentation a distribution function (such as Gaussian) is usually used.

A hand in motion against a static background can be segmented by simply detecting pixels that differ from the background image. There are three main methods for background removal; they are static, adaptive and probabilistic methods [96]. Shadows can be a problem in background removal algorithms; there is a huge body of work on how to remove these shadows, including studies that utilize infrared (IR) cameras [65, 97]. For example, an infrared (human's temperature) camera [98] has been used to provide a fast solution by simple thresholding operations. In a more engineering based design, Feris *et al.* [45] introduced a multi-flash camera setup to remove the shadows that occur around the finger edges. Their work targets sign language using isolated finger-spelling gestures based on the depth of the edges. They used a shift and scale-invariant descriptor to analyse the intensity of the interest point on the edge by counting the number of other edge pixels in eight neighbouring regions. They have reported improved results against the well-known Canny Edge detector [99].

Lockton and Fitzgibbon [46] put together real-time hand gesture recognition of 46 different hand postures, most of the poses are from the spellings of American Sign Language (ASL). Skin colour segmentation was used in addition to a boosting algorithm [100] for fast classification. The user was asked to wear a wristband so that the hand shape can be easily mapped to a canonical frame. A recognition rate of 99.87% is produced in controlled environments where no temporal information is considered. Another application which used human skin segmentations is Chandran and Sawa [22,

44]. They worked on sign language alphabetic recognition from a set of frames that is made with protruding fingers only. A two dimensional projection of Euler angles is the description method that is coupled with Euclidean distance to form the classification stage. They reported a recognition rate of 91%. The experiments have been conducted in controlled environments.

Tracking is commonly followed by or combined with the segmentation stage. In this research, a CAMShift tracker is adopted (see Chapter 4). Choosing the appropriate tracking and segmentation method requires testing different techniques and then evaluating their suitability for a particular task. For this propose, the use of the Leeds tracker developed by Magee [96] is explored. Magee's tracker is based on the use of GMM[1]-based adaptive colour models for both foreground and background and also incorporates object shape models. Magee's tracker proves to be effective when the tracked object is big enough and has a predictable motion. Different environments need to be learnt first by the tracker to generate its foreground and background models, which may eliminate some of the hand gestures. The tracker is designed to track blobs around the object of interest. In this research, the need is to track and segment the object not the blob. Therefore, the segmentation step is required on top of Magee's tracker for the hand skin colour. Magee's tracker suffers from changing the ID of tracked blob (the same object may have more than an ID during tracking); to solve this problem studying the trajectories is required as mentioned by Dee in [101].

The KLT (Kanade-Lucas-Tomasi) Tracker is well explained in the Shi and Tomasi [102] paper. The KLT algorithm combines a feature extraction method with tracking. It starts by selecting the best features, and keeps track of these features. The KLT algorithm makes use of the texture patches that have high intensity variation in both directions, such as a corner. Human skin-colour does not have rich texture and therefore it is harder to track. Edges can be considered as features.

Optical flow [99, 103, 104] reveals the image changes due to the motion of an object. Optical flow is the velocity field that represents motion of the object points in an image. Usually, these points relate to intensity changes in the image. Usually optical flow is associated with two assumptions: First is that the tracked feature points have constant brightness during the motion, and the second is that the points near the feature

[1] Gaussian Mixture Model

points are moving in the same direction. The algorithm yields a dense velocity field between any two images of a sequence, provided that the displacements between the two frames are not too large. Usually, the velocity vector components of direction and value are computed from the gradient in both the *x and y* direction.

A tracking method as part of integrated systems is introduced by Laptev *et al.* [105]. They track open hands and distinguish between five different hand states. Scale-space analysis was used to find intensity or colour blob features at different scales, corresponding to palm and fingers. Likelihood maps were generated for these features and were used to evaluate five hypothesized hand states that corresponded to certain fingers being bent or extended. A particle filter [106] is used for tracking, and the system operated at 10 fps. A vision-based television remote control was the target application.

Nikolay *et al.* [106] presented an approach for visual tracking of structured behaviour. An automatically acquired variable-length Markov model was used to represent the high-level structure and temporal ordering of gestures. Continuous estimation of hand posture is handled by combining the model with annealed particle filtering. The stochastic simulation updates and automatically switches between different model representations of hand posture that correspond to distinct gestures. Nikolay's implementation was executed in real time. Particle filtering has been used to approximate the posterior density with a set of particle weights. Skin-colour with a Hough transform [60] is used to extract the features of the hand and represent them.

2.5.2 Feature extraction and representation

Extracting meaningful information from a sequence of images is a crucial stage for any computer vision application. Why is this important?

Processing every part of an image with the same significance is computationally expensive. In addition, hand gestures can be accompanied with other unwanted motions like head motion where the important part is the hand motion. Feature extraction methods provide the useful information in a simpler and more meaningful form for computation. The difficulty is being able to find a method that represents the movement of the gesture in an effective way. A successful description should transform these features to computational form that can be processed and classified. Plausible features could be the finger number, position of the hand, curvature, angle and hand gesture trajectory [92].

Hand gesture feature extraction has been studied in the literature in many ways, as shall be highlighted in this section. A simple technique involves first extracting the main areas of motion in successive frames and then representing these areas. This can be calculated using frame differencing. Frame differencing requires generating a two-dimensional array by subtracting the pixel intensity values of two successive frames. From this, a binary array can be generated where *1's* and *0's* are created according to whether the difference resulting values are above or below a threshold parameter. From this, the area, centroid position, and orientation can be calculated. This provides a feature vector that can be used for further processing. The weakness of this method is that it relies on all the rest of the image within the field of view to remain stationary. Movement speed is also not included in the feature vector. It is unlikely to only capture the meaningful movements of the body.

Real-time applications contain several stages such as segmentation, feature extraction and representation, classification, detection and visualization. For these applications to work efficiently, feature extraction and representation methods should minimize their required computational time. For example, a compressed version of a sequence of images might be processed more quickly. PCA can be used as a form of compression since it is able to hold most of the important information about an image in a small amount of data [99, 107]. PCA is efficient method as it relies on similarities within an image where similarities can be ignored and focus can be focused on differences which place most of the information about an image or a sequence of images.

It is worth mentioning that it is not easy to categorise feature extraction and representation into separate modules from the tracking stage. Often, the technique used for tracking is inherently linked to extraction and representation methods.

Feature representation techniques can be classified into two types:

- Contour-based methods
- Area-based methods

The classification is based on whether the features are extracted from the contour only or are extracted from the whole shape region. Under each class, the different methods are further separated into discrete approaches and continuous approaches. This sub-class is based on whether the feature is represented as a whole or represented by segments/sections (primitives) [60, 108]. The whole hierarchy of the classification for the feature types reviewed here in Figure 2-9.

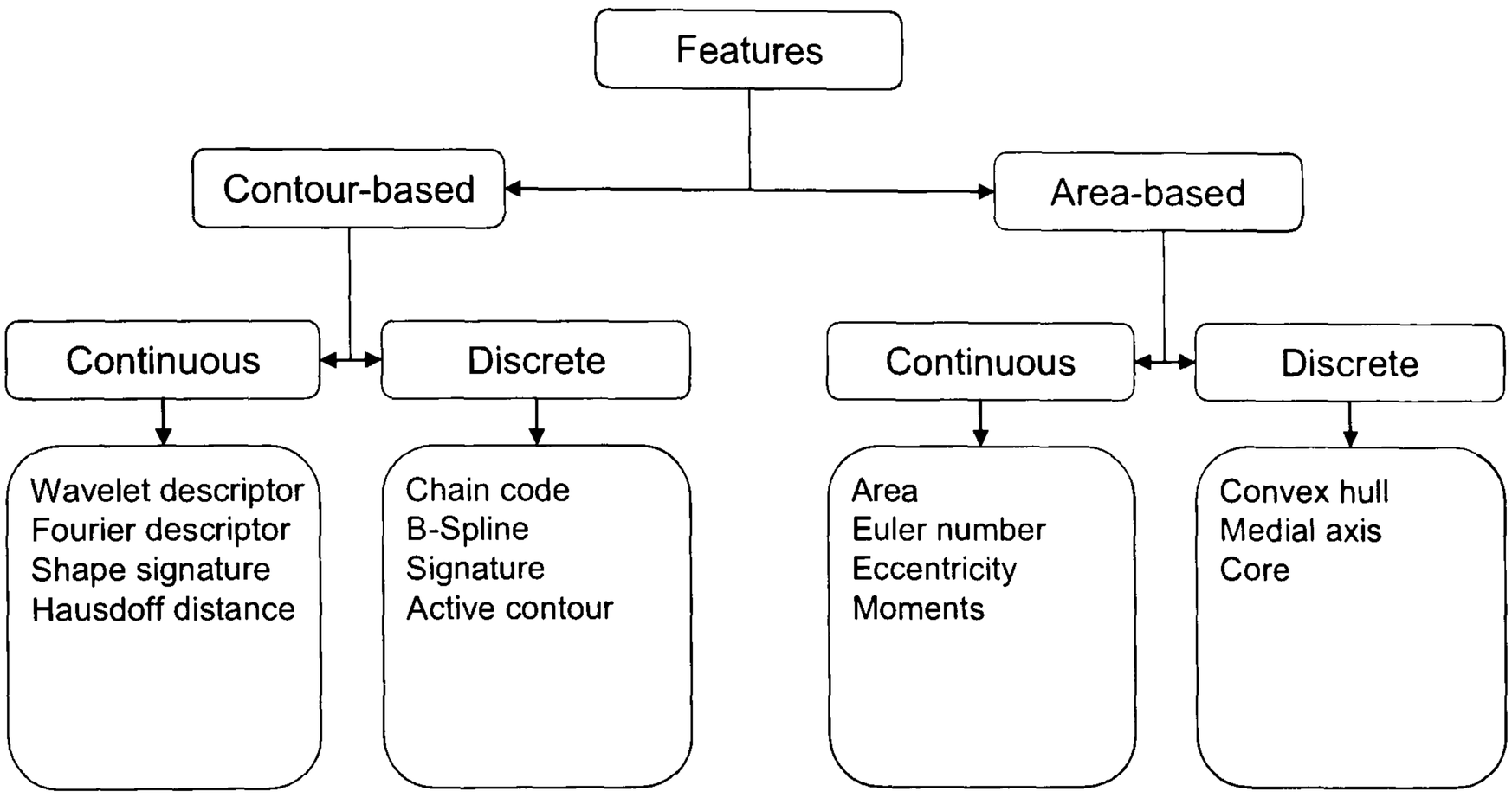


Figure 2-9: Feature representation and description techniques and their categories

There is a third type of feature representation techniques that takes into consideration the temporal information in addition to the spatial ones. This type can be called spatio-temporal description methods, see Section 2.5.2.3.

2.5.2.1 Contour-based methods

These methods use the shape boundary only for producing a feature vector to represent the object (*e.g* hand). Contour-based methods can be divided into two types: continuous approach (global) and discrete approach (structural). The measure of feature similarity is a metric distance between the acquired feature vectors. Discrete approaches sample the shape outline into primitives (segments) using a particular sampling criterion. Continuous approaches do not split the shape into sub-parts; usually a feature vector derived from the primary boundary is used to represent the feature. Fourier and wavelet

transforms are popular types of continuous descriptors. We present discrete and continuous examples of contour-based descriptors.

The discrete Fourier transform (DFT) [99, 109] estimates the Fourier transform of an image from a finite number of its sampled pixels. This to be used with contours, the edges of the target object needs to be extracted. A simple method of edge detection can be used. The sampled pixels are supposed to be typical of what the whole image continuous signal (stream) looks like at all other times. A wavelet transform is mathematical function that cuts up data into different frequency components, and then studies each component with a resolution matched to its scale. They have advantages over Fourier methods in analyzing physical situations where the signal contains discontinuities and sharp spikes [103].

A chain code [110] is one of the simplest ways to represent the boundary of an object. The chain code represents the length and direction of the boundary. Length is determined by the distance between pixels and direction, and is formed into 4 or 8 values as in Figure 2-10.

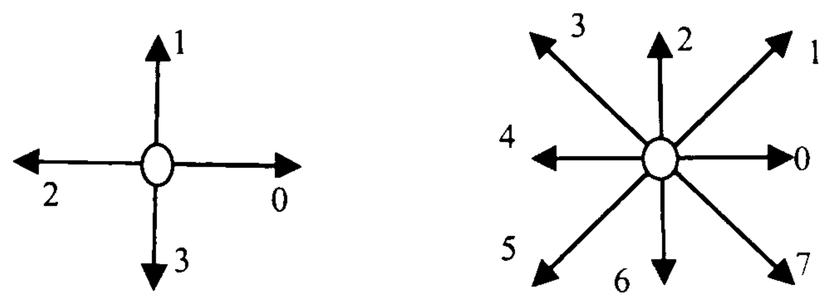
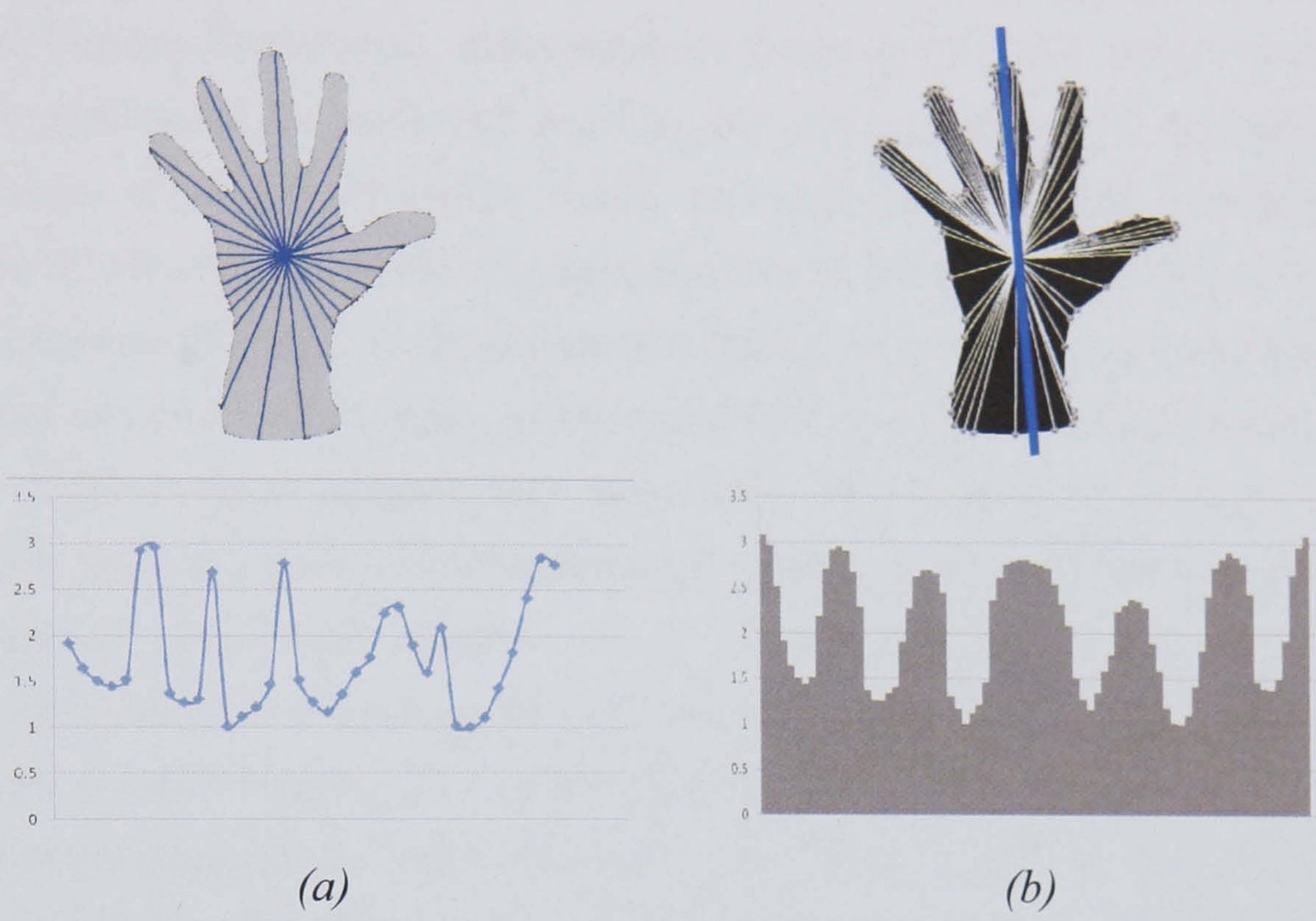


Figure 2-10: Chain code representation

Using the chain code representation has its drawbacks. This is because the resulting representation suffers from being the length of the code's chain, not tolerant to noise and not invariant to starting point and rotations. There has been lots of research on enhancing the chain code, *e.g.* [111], such as normalising the shape but that also generates a problem in the shape resolution.

Shape signature [60] is a 1D method of representing an object, it is invariant to rotation and translation as it is based on the centroid profile in the most common case. Signature is easy to compute and tolerant to noise. A number of samples are selected and located on the target boundary and then the distance to the centre of mass is calculated. Figure 2-11 depicts two examples of hand posture (see Section 3.2). In the first experiment Figure 2-11 (a), a regular step sampling on the boundary and in the second an adaptive sampling is adopted where the number of samples increases with

changes in the curvature degree. Thus a section of boundary requires fewer samples than the finger tip. The axis in Figure 2-11 (b) depicts the direction of the sampling. It is also known as centroid distance signature.



(a) *(b)*

Figure 2-11: Hand posture representation using signature method (a) with regular sampling step, (b) Adaptive sampling step

2.5.2.2 Area-based methods

Area descriptors (regional-based) are powerful when texture and the colour are the important elements and features across multiple frames. All the pixels within a shape region are taken into account to obtain the representation in contrast with the use of boundary information alone in the contour-based methods. Common region-based methods include: moments descriptors, grid method, shape matrix, convex hull and medial axis [15-17, 29, 31, 50, 52]. Similar to contour-based methods, region-based methods can also be divided into global and structural methods, depending on whether they separate shapes into sub-parts or not, see Figure 2-9. Area-based methods are utilised in this thesis and further discussion is postponed until Chapter 4 and Chapter 5.

2.5.2.3 Spatio-temporal methods

Spatio-temporal methods make use of the spatial information in each frame of a sequence as well as the temporal information. Laptev *et al.* developed a spatio-temporal descriptor [112-114] which uses features detected using a SIFT descriptor. Frame

features are tagged with its corresponding time label. They characterize local features by computing histogram descriptors of space-time volumes (cuboid) in the neighbourhood of detected interest points. For each cuboid, they created coarse histograms of oriented gradient and optic flow then they normalized both histograms. SIFT (Scale Invariant Feature Transform), introduced by Lowe [115], can match features from images regardless of the scale and rotation and provide a powerful matching across a major range of affine distortion, noise, and lighting. The fast nearest-neighbour algorithm that has been used for matching features is followed by a Hough transform to identify clusters. The SIFT method generates features by scaling searching for potential points that are invariant to scale, rotation and orientation, then the algorithm localises the key point to fit a location and scale, then one or more orientation values are assigned to each key point. The matching between features is measured by the Euclidean distance of their feature vectors.

Kim and Cipolla [116] presented a gesture recognition system that can learn 9 gestures from a small dataset. They use a pairwise feature extraction method of video volumes for classification. The method of Canonical Correlation Analysis is combined with discrimination functions and Scale-Invariant Feature-Transform (SIFT [115]) for the discriminative spatio-temporal features for robust gesture recognition. A gesture sequence is described by decomposing an input video segment into three sets of orthogonal planes, XY-, YT- and XT-planes. This allows posture information in XY-planes and joint posture/dynamic information in YT and XT-planes. Three subspaces are learnt from the three planes. The gesture recognition is done by comparing these subspaces with the corresponding subspaces from the models by classical canonical correlation analysis, which measures principal angles between subspaces. The similarity measure of any model and query spatio-temporal data is defined as the weighted sum of the normalized canonical correlations. The dataset in Kim's research is conducted in controlled environments with limited motion.

Spatio-temporal methods have received many studies recently as in [1, 22, 48, 115, 117, 184]. Most of these methods are extensions to the spatial descriptors. One of the most recent description methods is the Zernike Velocity moment (ZVM) [1, 2] which is an extension of Zernike moment. This research discusses ZM and ZVM comprehensively in Chapter 4 with experiments and evaluation in Chapter 5.

2.5.3 Hand gesture detection

Hand gesture detection or spotting refers to the mechanism that allows locating the start and the end of the gesture automatically. Several techniques have been proposed to detect hand gestures.

A common method for detecting gestures is to use continuous dynamic programming which Alon *et al.* [12, 122] utilized. They generate a sequence of feature vectors for each of a set of target gestures. After that, a test sequence of feature vectors is compared with these stored vectors using a time warping algorithm. Then they calculated the distance between both vectors. They selected a set of hand gestures which is in the form of numerical digits. They tracked the Centre of Mass (CoM) of the palm which resulted in drawing the trajectory of the gesture. Their method relies heavily on the trajectory of the CoM and does not take in consideration interpolating of different hand shapes.

There are other methods for detecting hand gestures that can be identified, as in Yoon *et al.* [120] who proposed a method for spotting hand gestures. Their method relies on the user more than the technique. Participants were asked to hold their hand in certain position for a while. Then they detected this stationary state and sequentially segment hand gestures.

Ozer and Wolf [121] described a real-time system for posture and activity recognition for uncompressed and compressed (MPEG) input video using eigenspace representation of human silhouettes obtained from a motion detection module. Their methods recognized the start of the human activities by comparing the input postures (frames) to reference database of pre-trained actions.

Early studies on spotting hand gestures utilised a model of HMM trained on isolated hand gestures and then the optimal path is selected using the Viterbi algorithm [118, 119]. Yang *et al.* [119] introduced an HMM based system for spotting human body gestures. The system is proposed to simultaneously spot and recognize the whole body key gestures. A human subject is first described by a set of features encoding the angular relations between body parts in 3D. They created what they called a garbage model for filtering out motions which are not related to the studied gestures. Their model design provided a mechanism for qualifying or disqualifying gestural temporal information candidates. They reported a reliability rate of 94.8% in the spotting task and a recognition rate of 97.4% from an isolated gesture. In Chapter 5 of this thesis,

HMM based methods are used to detect hand gesture automatically with extended evaluation and discussion.

2.5.4 Classification and grammar role

One of the simplest ways to classify data is the use of a decision tree. The decision tree is a predictive model which links observations to conclusions. Classification trees or regression trees are alternative names to decision trees. In these tree structures, leaves represent classifications and branches represent conjunctions of features that lead to those classifications. Decision trees have been used for the hand recognition tasks including the work of Mardia [123]. Other methods of classifying hand gesture utilized techniques such as the boosting algorithm which has been used for prediction and training hand gestures as in Feund [100]. Hidden Markov Models (HMM) are used in many research studies listed in this chapter to learn spatio-temporal information and classify gestures, these studies include [26, 51, 92, 124, 125, 127]. More details about HMM can be found in Bilmes tutorial [126].

According to Sonka *et al.*[99], *Grammar* can be defined as "*Mathematical model of a generator of syntactically correct words*". Grammatical approaches can be deployed with hand gesture recognition systems by representing the hand features as letters of a posture. The posture represents a word. Dynamic gesture can be represented as a sentence in this structure where each frame is a word and each frame has features of the hand postures. The key analysis is with sentence/word. Good grammar analysis leads to minimizing the description of the systems. For example, the same word can be used in different sentences (dynamic gesture).

Sharma *et al.*[14] discussed the role of semantic grammar approaches in the recognition systems. They stated "*Normally, a feature-based description is used to represent the meaning of grammatical units (words, phrases, and sentence), and unification grammar rules are used to compose meaning of an utterance from the meanings of its parts. This form of semantic analysis typically results in meaning represented in first-order predicate calculus (FOPC).*" Sharma *et al.* believe grammar role analysis can be inefficient for systems of crisis managements, but it is believe the concept of grammar role can help researches to limit the hand gesture description model which in turn helps improve system performance.

The use of grammatical approaches in classification is explored further in several surveys [28, 66, 75].

2.6 Summary and discussion

A background review of hand gesture systems has been presented and classified according to two criteria 1) *the technology used* 2) *hand gesture context*. Since the research on hand gesture recognition studies started, researchers have suggested that hand gesture may provide natural, novel and improved interaction techniques having in consideration that perceptual input is less reliable than direct input devices.

Unlike speech interactions and hand gesture recognition based on non-vision sensors which have found their way into commercial applications, vision-based hand gesture systems have yet to find their way into daily use.

The literature suggests that the interest in hand gesture and interactive systems has increased recently which yields a promising prospective for people with disability to have less constrained computerised systems. These systems have application varies from complicated ones like driving a robot by hand to the simple ones such as 2D games and controlling sliding photo albums.

Vision-based hand gestures suffer from a variety of problems that most machine vision-based systems suffer from such as lighting variation, shadows and tracking instability. From the stated surveys and literature, it seems that more time and effort are expected to be spent on these types of research before a vision-based hand gesture application that is designed for precise applications such as a surgery device, can emerge. However, interactive applications, such as games similar to Nintendo Wii games, are the most likely to emerge sooner.

This chapter forms the bases that we used to build this research on. This research extends the use of hand gesture in [55, 128] by tackling the viewpoint problem through creating 3D models. The use of one hand is more natural and convenient for people with disability and can be extended into two hands use when applications are needed.

CHAPTER 3

A GESTURE-BASED USER INTERFACE

This chapter discusses datasets, settings and a prototype system used as a demonstration of the techniques developed and explored in this thesis. This chapter starts by highlighting the motivation and assumptions that were considered during dataset collection and then goes on to discuss the prototype.

3.1 Motivation

Hands are highly multipurpose tools with which we accomplish our daily tasks. They are the driving force in our user control interface and interaction prototype. Evaluating any hand gesture recognition system requires an appropriate test dataset. All available hand gesture databases exist for static postures and/or dynamic gestures with limited motions and background variations. The next section highlights the most cited datasets.

Related datasets

FGnet's dataset [6] consists of 13 gestures, where 9 gestures are static and 4 are dynamic, with 16 video sequences for each gesture. All other hand movements and postures are included in an "unspecified gesture". This dataset is captured against a uniform background within controlled lighting. The sequences of this dataset are recorded using a high quality (low noise) 3 CCD vision camera. The scene does not contain any objects that appear to have skin colour apart from the hands, and the illumination colour ranges from indoor to outdoor.

Massey's dataset [7] is collected using a digital camera mounted on a tripod away from a hand gesture in front of a dark background and in various lighting conditions. Together with the original images, there is a clipped version of each set of images that contains only the hand image. The background in the clipped versions of the samples in normal lighting condition has been eliminated (i.e. those pixels that have RGB value (0,0,0) belong to the background). The dataset contains material gathered from 5 different individuals.

Cambridge-Hand-Gesture dataset [9] consists of 900 image sequences of 9 gesture classes, which are defined by 3 primitive hand shapes and 3 primitive motions. Hand shape primitives are flat, spread and v-shape hand. Motion primitives are move the hand leftward, rightward and contract. 100 image sequences represent each class (5 different illuminations x 10 arbitrary motions x 2 subjects). A fixed camera is used for recording each of the sequences of roughly isolated gestures in space and time. Thus, fairly large intra-class variations in spatial and temporal alignment are reflected to the dataset.

BU dataset [8]: The national center for sign language and gesture resources of Boston University published a dataset of American Sign Language (ASL) sentences. Although this dataset has not been produced primarily for image processing research, it consists of 201 annotated video streams of ASL sentences. The signing is captured simultaneously by four standard stationary cameras where three of them are black/white and one is a colour camera. Two black/white cameras, placed towards the signer's face, form a stereo pair and another camera is installed on the side of the signer. The colour camera is placed between the stereo camera pair and is zoomed to capture only the face of the signer. The movies published on the internet are at 30 frames per second and the size of the frames is 312x242 pixels.

Existing datasets were collected generally for research purposes. These datasets mainly consist of one hand gestures captured with limited motion, controlled lighting, uncluttered background, no body movements and/or coloured gloves to ease the tracking and segmentation. For instance, the Massey [7] hand gesture database lacks expressiveness of the body and its parts and lacks background variation. Therefore, it cannot be used for analysing realistic human interactive real hand gesture. It should be mentioned here that some researchers presented studies that included a more realistic hand gesture datasets but we did not have access to their datasets to evaluate them, such as the work of [11, 12].

Proposing a more realistic scenario for hand gesture recognition systems requires datasets that represent the use of hand gestures in everyday environments. For instance, an office environment is the natural location for applications in which hand gestures are used to control a slide presentation. Any hand gesture recognition application utilizes several intermediate stages and techniques - see Section 3.3. These stages and techniques can be evaluated more efficiently if the same hand gesture is captured in different conditions. For example, segmentation would be easier and more reliable if the hand gesture is captured with consistent background and lighting (using threshold-ing, see Section 4.1.1). This consequently increases the recognition rates. The trade of this increase in recognition rate is at the cost of a less natural dataset. Having a dataset that represents a more natural system tends to generate more errors from segmentation, tracking and other stages. When collecting the same hand gesture data in different settings, the comparative performance of each stage would be more visible. This helps to study hand gesture errors that occur at each stage and then to improve and enhance the less performing stages of the recognition system.

Games are already a big industry. It is well-accepted nowadays that children and even adults like to spend some time on computer games. Having these computer games driven by physical movements will be widely welcomed by users. It is important to choose a prototype application which tests the proposed recognition system and at the same time provides an enjoyable experience for the user.

The previous issues have all contributed to our thinking of how and for what purpose the dataset should be collected and the application designed. In Section 3.2, datasets, assumptions, settings, examples and evaluation of data collection are presented. In Section 3.3, the prototype which provides a simple computer game is presented. Initially, the design of the game is drafted, and then game scenarios, characters and environment are modified according to the participants' feedback.

3.2 Dataset and settings

3.2.1 Data collection

3.2.1.1 Instructions to participants

In these datasets, 10 participants (4 females and 6 males) were told that they should try to reproduce the same performance of a gesture each time the experiment is repeated. Participants had been given the chance to practice the performance of the gesture before the recording session. To simplify the pre-processing of the gesture recognition system, users have been asked to perform the gestures at almost a stationary body state where only their hands are moving. It has been observed that in the first 200 seconds, users usually perform gestures almost identically. The design of the tracker is a skin-colour based system. Therefore, participants had been asked not to wear clothes with colours similar to the human skin.

3.2.1.2 Assumptions and collection context

We took into consideration, when capturing gestures, the need to have a hand gesture dataset which can simulate the real environment variations. The data collection focused on experiments using one hand. For the target prototype application design, the following assumptions during recording are maintained:

a) Hand gestures should be as natural as possible to control a task; if the task resembles an action that is performed in other situations, the gesture should be also similar.

b) The gesture should not involve uncomfortable, painful postures or even motions which would attract one's attention and distraction from the task.

c) The relationship between learning curve and usefulness of the gesture should be considered carefully. Ideally, gestures should take less time to be learnt by experienced users of human-computer interfaces. For example, keyboard shortcuts.

d) The gesture should be performed in reasonable environments. It is desirable to consider the variations in environment and the distance that the gesture

will be performed at. For example, being too far from the camera can cause loss of detail and being too close would not be considered a natural behaviour.

e) The speed of gesture performance and the variation of performers should be controlled.

The previous assumptions are important for building a vision-based interface. Camera-user distance and environment conditions play a major role in the complexity of the system. In the following experiments, data collection has been performed in three different settings as follows:

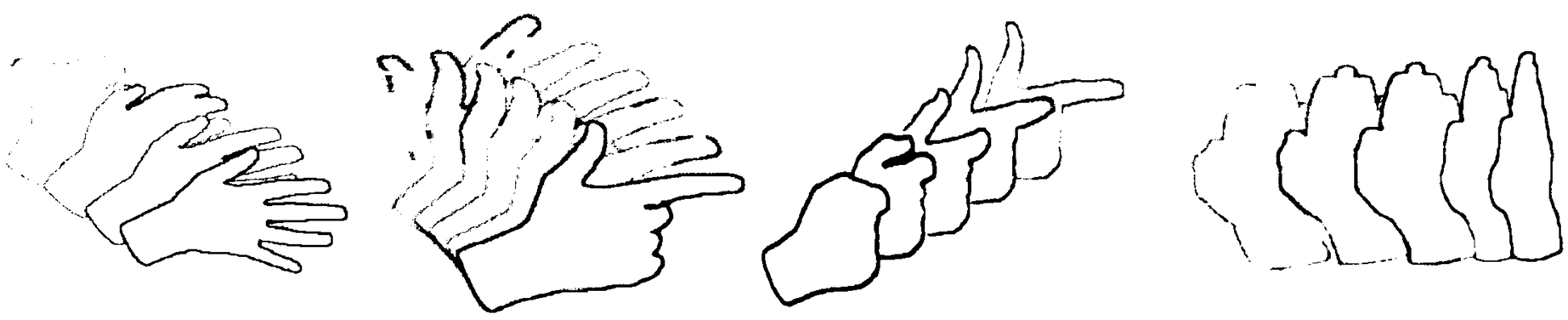
- Controlled lighting with uniform background
- Office environment
  - Hands only in the scene
  - Whole body is present in the scene
- 3D model construction using laser scan system

3.2.2 Gesture set

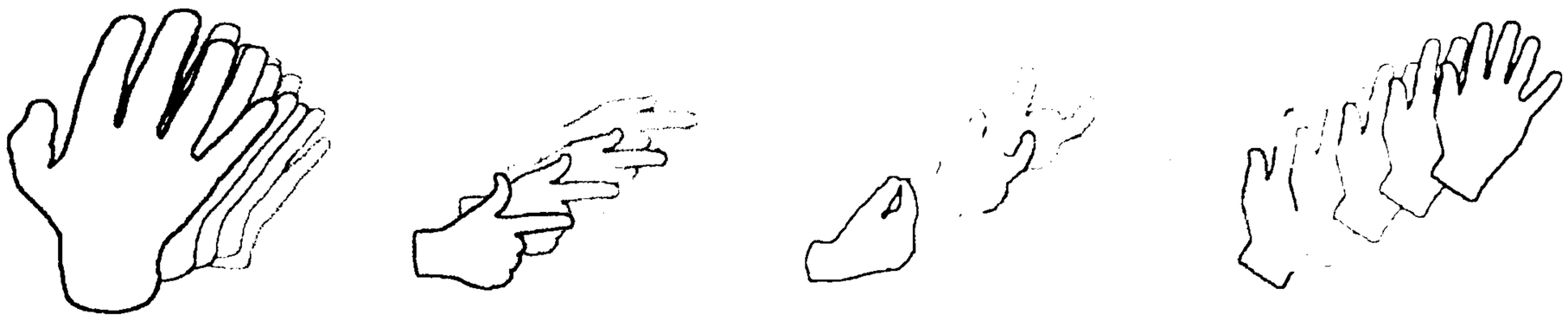
The aim was to select a set of gestures that would be suitable for controlling a typical multimedia application. For example, in controlling a slide presentation, at least five gestures would be required to control main actions of the presentation; *e.g*: 1) start the presentation, 2) end, 3) move forward, 4) move backward and 5) play a video. Another example is controlling a photo-album where eight gestures may be needed. Taking account of examples such as these, the aim was to select eight gestures for the purposes of testing. In informal experiments with four subjects, around twenty initial gestures were reduced to eight.

In these experiments, the subjects were asked to comment on how easy it is to perform these gestures for a specific task. And then, the subjects were asked to comment on the environments where the gesture is performed and especially to focus on hand-camera distance, camera location, background representation and suitability. After their feedback, the most appropriate settings were selected. Having chosen suitable settings for the gesture interface with the aid of informal experiments with users, datasets were collected.

Eight hand gestures were captured by taking into consideration the obtained feedback and assumptions explained in Section 3.2.1. Figure 3-1 depicts the eight hand gestures. A normal webcam has been used for the collection (Logitech Pro5000 [129] with 320x240 frame resolution at 30 frames per second). Examples from the datasets are available online [83]. These selected eight gestures could be used for example in a photo album application as follows: Gesture "A" for changing the album; Gesture "B" for browsing the next photo; Gesture "C" for switching the application on/off; Gesture "D" for rotating a photo; Gesture "E" for zooming a photo by moving the hand towards/away from the camera; Gesture "F" for running the album view on automatic play mode; Gesture "G" for holding a photo; Gesture "H" for navigating and viewing different parts of a photo.



Gesture "A" *Gesture "B"* *Gesture "C"* *Gesture "D"*



Gesture "E" *Gesture "F"* *Gesture "G"* *Gesture "H"*

Figure 3-1: chosen dataset of 8 hand gestures.

3.2.3 Capturing data in uniform background and lighting (UBL)

Segmentation is one of the main challenges in machine vision research especially with a noisy background and inconsistent lighting. Avoiding such a challenge provides a powerful way to evaluate further stages of hand gesture recognition systems such as representation, detection and classification easily and effectively. Clean segmentation which can be achieved by simple image processing marginally isolates the pre-

processing error from other errors (as mentioned before) such as those resulting in using a particular descriptor or classification method. Therefore, this dataset is designed to be easily segmented.

Figure 3-2 depicts the setup that was used to capture the eight hand gestures. The distance between the webcam and the hand is set to simulate the actual distance where users mount their webcam on the screen. The camera is in the range of 60-80 cm from the table surface. Figure 3-4 shows examples of the dataset sequences according to the setting of this recording session. Participants were asked to repeat the performance of each gesture 5 times. Therefore, the dataset has 50 instances for each of the eight hand gestures. The start and end of these instances are manually marked. Two video sequences which include natural continuous performance from one participant are also collected. In these sequences, there are non-gestural motions as well as the eight pre-defined hand gestures. The length of each sequence is 3500 frames and each gesture has been randomly repeated five times in the sequence. For simplicity, this dataset is named ***UBL*** (Uniform Background and Lighting dataset).

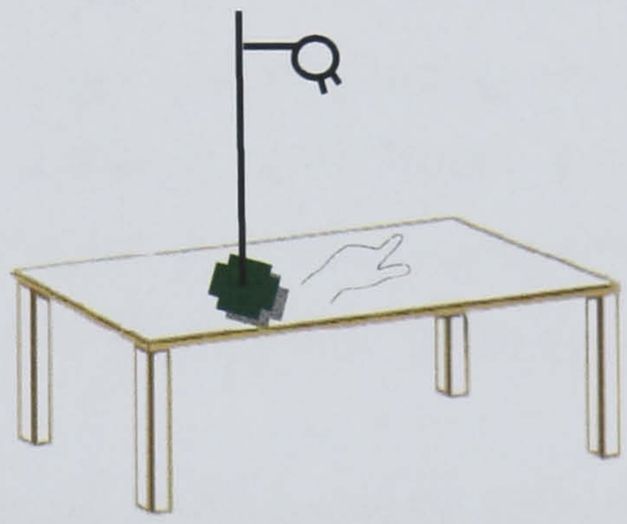


Figure 3-2: the camera setup for collection of UBL dataset.

3.2.4 Capturing data in office environment

For real applications, it is unlikely for a gesture to be performed against a uniform background under consistent lighting. Therefore, capturing a dataset under conditions more natural was required for training and evaluating our system. In office environments, the hand gestures were captured in two scenarios: 1) the hand is the only moving object and 2) the whole body is in the scene. In addition, two video sequences of gestural and non-gestural movements similar to the UBL dataset are collected from one participant.

3.2.4.1 Hand only in the scene (OEH)

Skin colour segmentation is probably the most commonly used method in vision-based literature on hand gesture analysis. For that reason and from experiments presented in this thesis, this dataset is captured while the hand only is in the scene given that the previous instructions of the participants are followed (see Section 3.1.2). Some part of the arm appears during the capturing of data (see Figure 3-5) but these parts do not include any skin colour and hands are the target to be segmented in later stages. Five sequences per gesture from 5 participants and 12 sequences per gesture from the other 5 participants were collected. These sequences have a length ranging between 30-130 frames. They are collected from a camera at 30 frames per second with 240x320 pixel resolution. Figure 3-5 depicts examples of the gestures in these settings. This dataset is named ***OEH*** (Office Environment Hand only dataset).

3.2.4.2 Whole body in the scene (OEW)

A more realistic environment is achieved when the whole body of the participant is present in the scene. For this purpose, the same gestures were captured where the camera was about two meters away from the participants and the lighting is under normal office conditions. Five sequences per participant per gesture were captured of a length ranging between 30-130 frames at 30 frames per second and 240x320 pixel resolution. Figure 3-6 depicts examples of the gestures in these settings. This dataset is named ***OEW*** (Office Environment Whole body dataset).

3.2.5 3D models and virtual performance of hand gestures

Using a Polhemus FastSCAN system [78], various 3D hand models were captured. This handheld system uses a laser scanner which consists of a non-contact range finder, based on projection and simultaneous detection of laser light, coupled with a means of tracking (magnetic tracker) the position and orientation of the range finder as it is scanned over the object's surface. The resulting 3D model is in the format of a multi-layered 3D point cloud that needs further processing to reduce its size and to smooth it. The number of layers in the model depends on the number of scans performed. This is discussed further in Chapter 6.

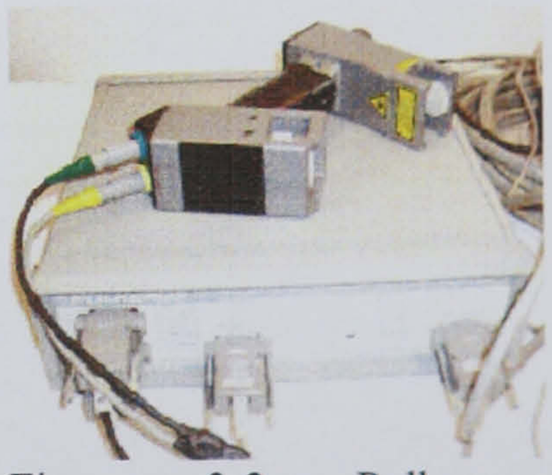


Figure 3-3: Polhemus FastSCAN system used for capturing 3D models of the hand.

To convert the multi-layer 3D point cloud data into one layer, an interpolation based method of Radial Basis Functions is used. This operation reduces the size of the model and smoothes its surface.

To generate 3D animation of gestures, the Blender [130] software was used. The use of Blender helps to create 3D models of the intermediate hand shapes by deforming the captured 3D models. Intermediate models refer to the number of hand shapes the gesture takes during the performance. In reality, this can be a huge number but only 10 models for each gesture are used in this dataset to represent the whole sequence. To simulate the performance of the hand gesture, we created virtual hands as shown in Figure 3-7 and rendered the hand from different viewpoints. This will be discussed in detail in Chapter 6. Examples of the gestures are shown in Figure 3-8.

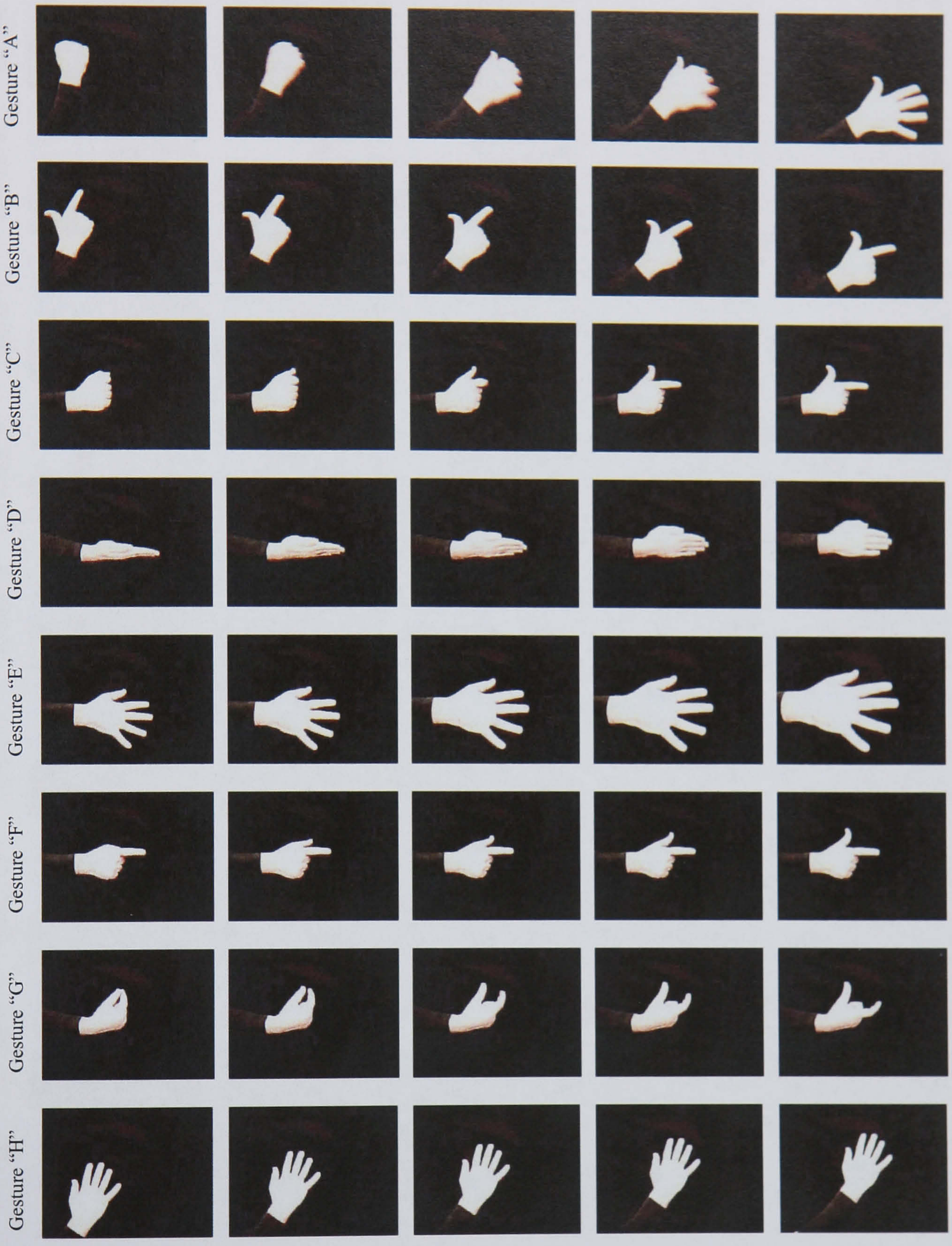


Figure 3-4: Sampled frames from an example of each of the eight gestures in the UBL dataset.

Figure 3-5: Sampled frames from an example of each of the eight gestures in the OEH dataset.

Figure 3-6: Sampled frames from an example of each of the eight gestures in the OEW dataset.

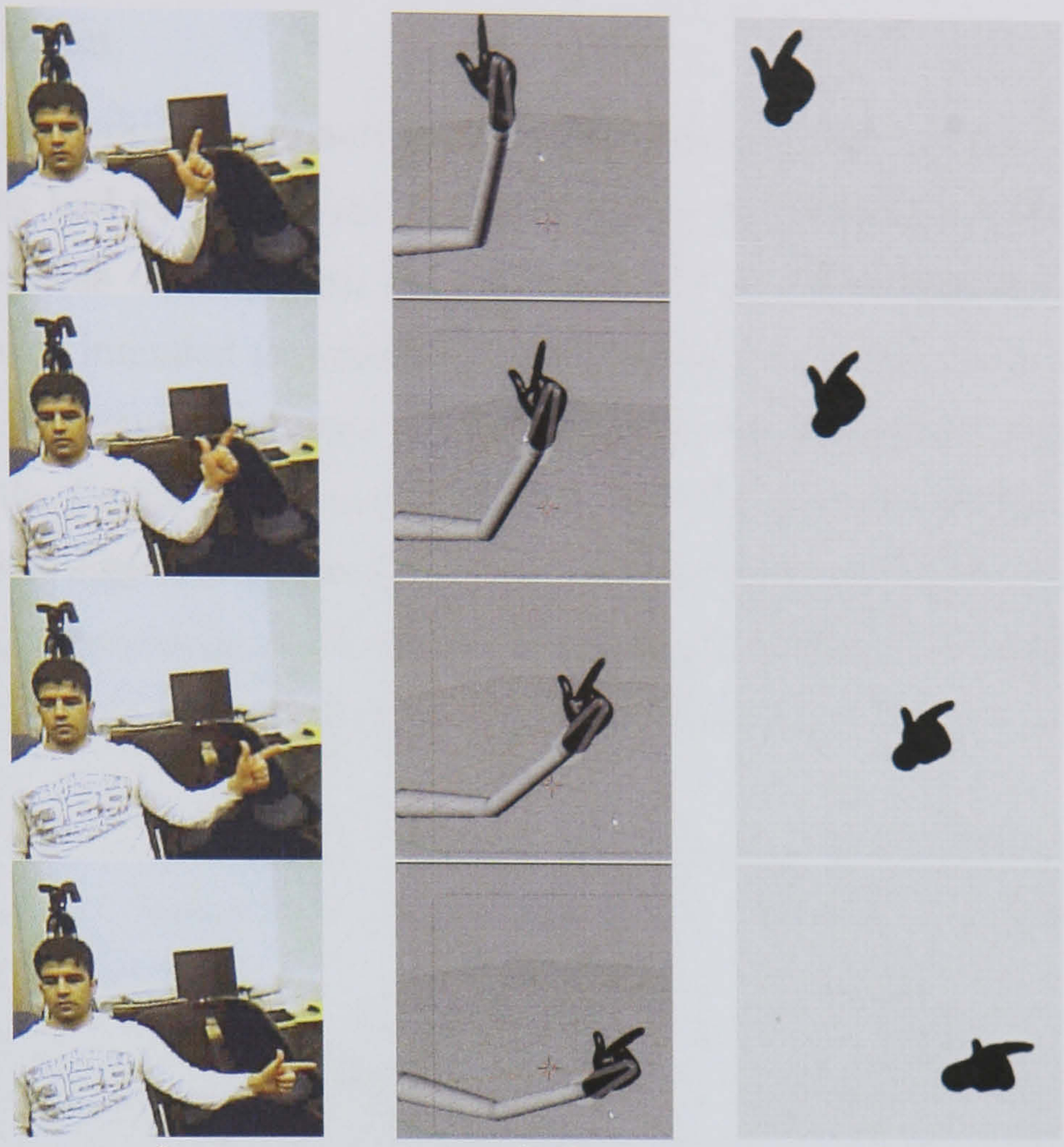


Figure 3-7: Depicts the use of the 3D model for generating gesture clips from different viewpoints. Seq 1 is the corresponding real gesture. Seq 2 is the 3D model fitted onto an artificial arm. Seq 3 is a rendering of one instant during the performance of the hand gesture from different viewpoints.

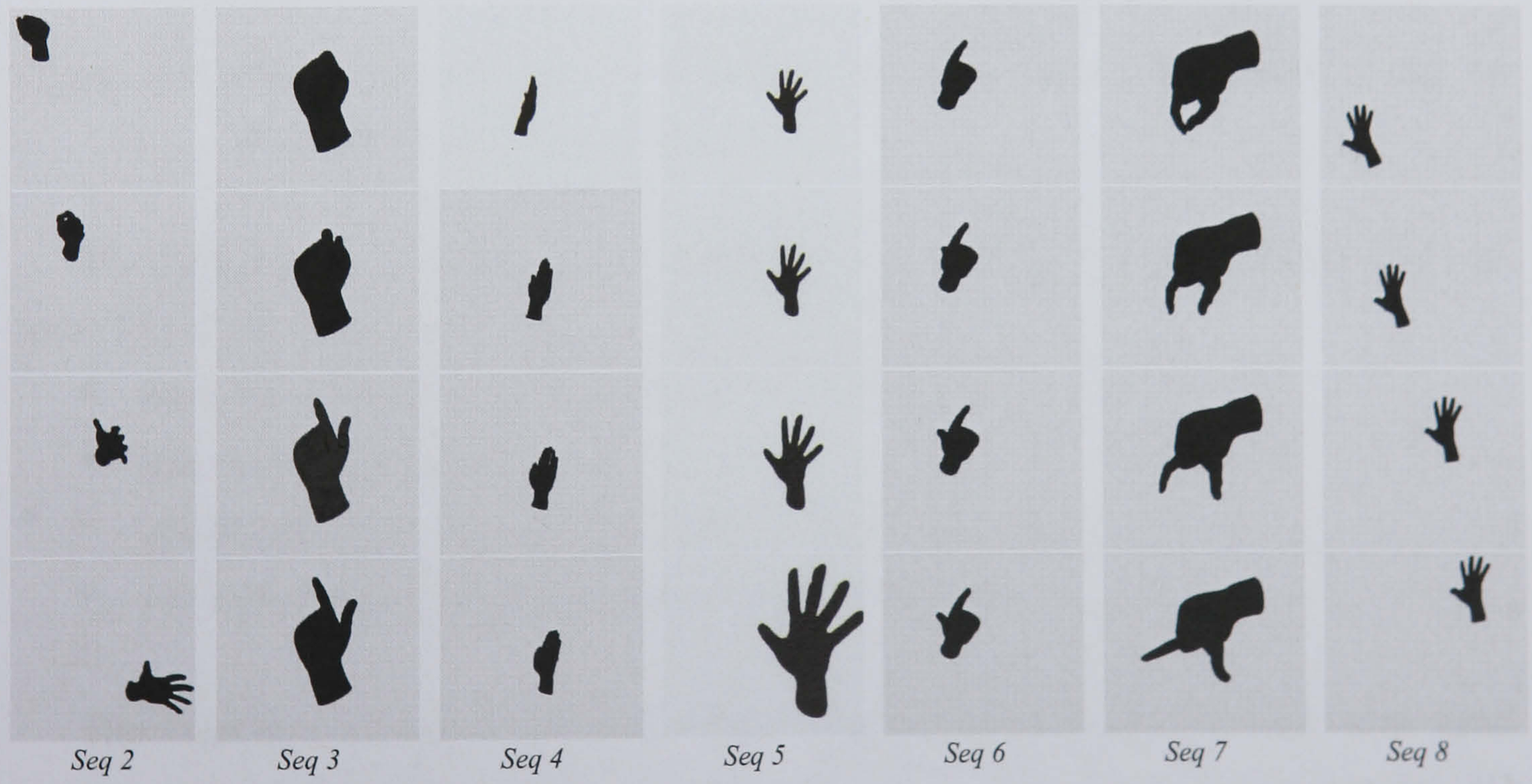


Figure 3-8: Sample frames for each gesture from a single viewpoint.

3.2.6 Discussion

Fairly consistently, the participants' body positions remained constant shortly (200ms) after their hands had performed the gestures. Thereafter, very little unwanted motion of hand and body was observed till the end of the trial. Collecting the data in various environments was intended to simulate the real application when testing the system. To illustrate the differences between gestures performed by different participants, the trajectories of the Centre of Masses (CoMs) of the hand gestures are plotted as shown in Figure 3-9 which depicts that gestures of the same type from different participants (as in gesture "A") have produced similar paths. This suggests it should be in principle possible to train a classifier on gestures from one group of individuals and test on the gestures of another group.

Figure 3-9: The trajectories of the CoMs in the image plane from each of the 10 participants performing gesture "A".

The four presented datasets have an advantage over existing datasets for several reasons. The existing datasets either:

- have limited motion of the hands as in [7, 9];
- use markers as in [6];
- are designed for a specific task as in [7-9];
- are captured in controlled and typical environments.

The combined dataset is novel for two reasons. Firstly, it contains the same hand gestures captured in different environments. Secondly, it contains a corresponding 3D

animation for each gesture. This is used to study viewpoint invariant hand gesture recognition, see Chapter 6.

3.3 Prototype application

3.3.1 Overview

In this section, a representative prototype application is presented based on the developed hand gesture system proposed in this thesis. This prototype is a simple attractive chasing game of two characters (*Ant* and *Bird* see Figure 3-11). Users drive the *Ant* using the movements of one hand, and the computer controls the *Bird* which is trying to touch the *Ant*. Users keep moving their hands around for a certain time aiming to keep the *Ant* away from the *Bird* and when they succeed, they move to the next level where the computer provides the *Bird* with extra speed and reduces its weight. If the *Bird* manages to touch the *Ant,* then the game is over and the player needs to start over again, otherwise they move up one level and so on. Figure 3-10 depicts the setup and environments.

Figure 3-10: the proposed prototype setup and environment. Gestures are performed under the camera against any background.

3.3.2 Scenario

The aim is to design a modular system that utilises hand gestures as an input method regardless of the application. Currently, standard input devices such as the mouse and keyboard are widely used to drive several applications and games. In a similar way, the aim is to explore ways that enable computers to use hand gesture for driving applications that do not require a high degree of precision.

This prototype is a simple game which demonstrates the potential of using hand gesture for multimedia applications. To simplify the implementation, it is assumed that the hand gestures are performed in a 2D plane where the viewpoint is fixed. The hand is the only moving object in the scene, lighting is constant, and the background is cluttered.

Figure 3-11 depicts the two characters in this game, the *Bird* and the *Ant*. Using single hand gestures and by moving them around in the camera view area, players control the *Ant* in order not to be touched by the *Bird* which is controlled by the computer. The prototype assigns a weight to the *Bird* to control its manoeuvre speed.

This game is divided into 5 levels. In the first level, the application gives the *Bird* the highest possible weight. In the successive levels, less weight is assigned to speed up the movements of the *Bird* and make the game harder. Users keep moving their hands around for a certain time; when they move successfully to the next stage, the computer provides the *Bird* with more speed and less weight.

Figure 3-11: Game characters, the *Ant* and the *Bird*.

This game utilizes 5 gestures from the eight chosen gestures: "C", "D", "E", "F" and "H" (see Figure 3-1). Gesture "C" is utilized to switch the game on, "D" to switch the game off, "E" to resize the view of the game, "F" to restart a new game, and finally "H" is used to manoeuvre and move the *Ant* in the 2D plane.

In this application, gesture "H" is not treated as normal gesture with start and end, but rather the hand posture of "H" needs to be recognized on a frame by frame basis. The CoMs of the hand in gesture "H" are used to position the *Ant*.

3.3.3 Software design

This game is designed to be a module-based application. Therefore, a block for capturing the video from the camera is created. The capturing block is responsible for acquiring the video in RGB colour mode. A second block which processes a video signal and tracks the hand is created. The third block is the representation block which converts the tracked blobs into a feature description. The classification block is responsible for determining the identity of the gesture and provides input to drive the visualization block.

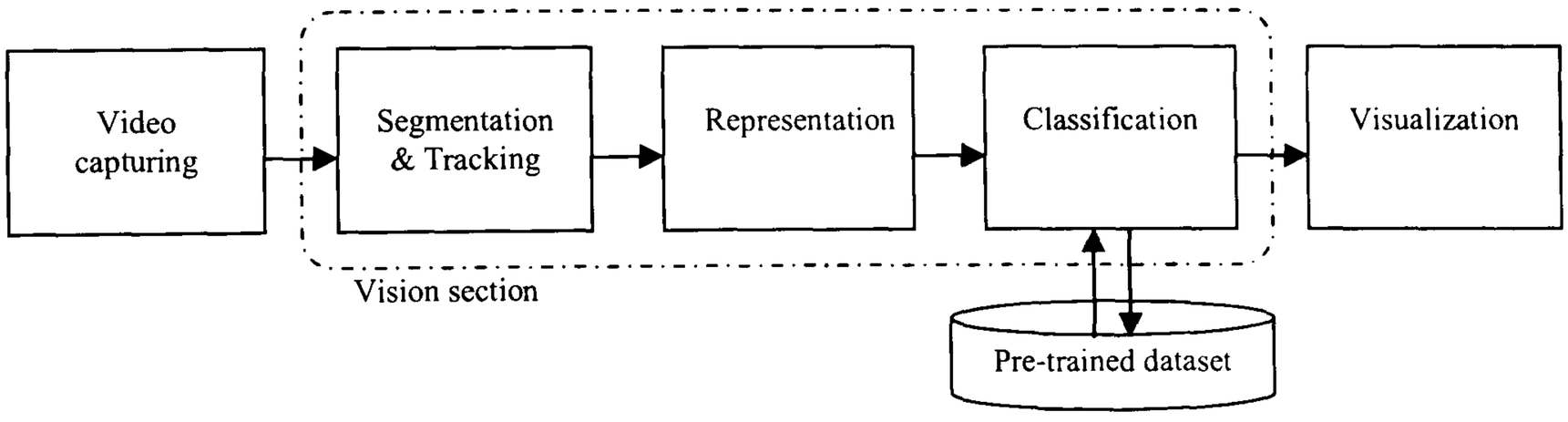


Figure 3-12: prototype internal block structure.

Matlab software has been used to implement this game. A Simulink program which is a part of Matlab [131] is utilized. Simulink provides a graphical user interface (GUI) that is used in building block diagrams, performing simulations, as well as analyzing results. In Simulink, models are hierarchical so that a system can be viewed as a high level design, and then by double-clicking on blocks viewed down through the design levels. Figure 3-13 depicts the design of the game in Matlab. Simulink provides built-in functions for capturing and processing the video. The input video from the camera is

connected to the capturing block and later the colour format of frames is converted into grey-level. After that, thresholding is performed (see Section 4.1.1), the filtering to remove noise, and the blob analysis which is connected to a CAMShift tracker (see Section 4.1.2). In the next stage, Zernike moments are computed (see Section 4.2.3.3). Then, feature vectors are passed to the classifier (see Chapter 5). Simulink provides blocks that host the script of Matlab "*m*-files", *S*-function or *C*. In this game, *m*-file and *C* coding are used. The classification block is based on Murphy's code [132] HMM classifier code (see Chapter 5). Figure 3-13 shows the structure of the visualization block. A Virtual Reality (VR) block links to the VR Matlab engine and editor where the game characters are created and where the rules of the movements are defined.

This game can be extended to work in various environments and to utilize both hands. In some applications, a second hand is required to work as a clutch when performing certain actions. A viewpoint invariant capability could be added to enhance the experience in a target application, although this has not been implemented. Personalized use of the application increases the gesture recognition and the detection rate as discussed in Chapter 5 and 6.

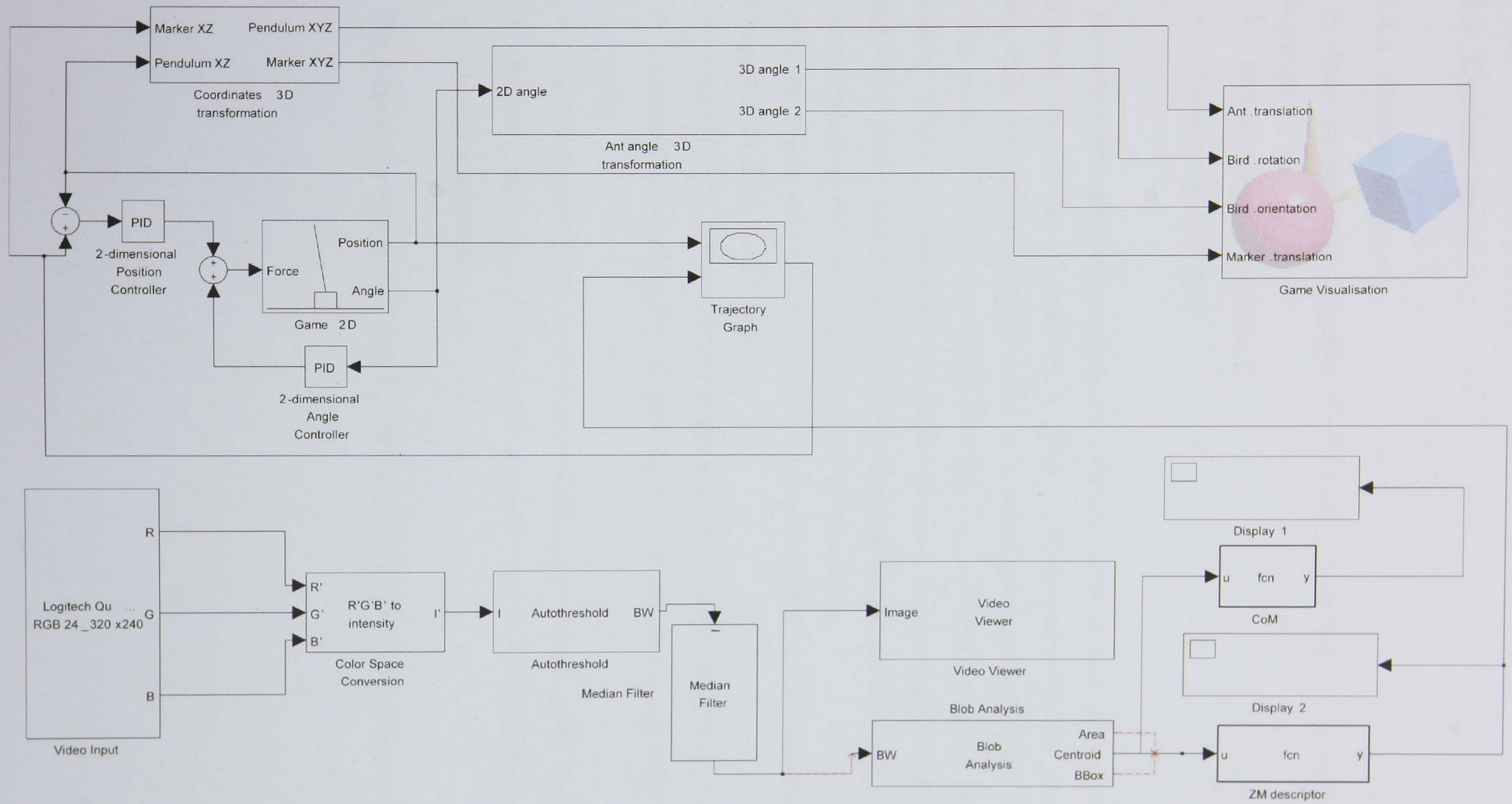


Figure 3-13: Simulink blocks diagram of the application. The game uses built-in blocks for capturing the video stream. Processing blocks are connected to the visualisation part which is based on existing examples in Simulink [131].

3.3.4 Evaluation

3.3.4.1 Methodology

To evaluate the prototype, a survey of users was conducted. Users were divided into two groups:

- 10 non-computer specialists
- 10 computer specialists

The evaluation started by explaining to the test subjects how the application works, and asking them to try the application. After the participants finished playing with the game, they were given a survey to fill in. Afterwards, they were asked if they would use the game in case it was made available. An optional comment field was also provided. While the participant tested the interface, a test observer was taking notes about which of the tasks were successfully completed. The test observer also took notes about the causes of the difficulties encountered by the user.

All participants were asked to comments on the application and rate on a scale from 1-10 against the following criteria:

- *Usability*: this refers to whether the application can be applied to everyday use.
- *Learning curve*: this refers to how easy it was to master the game and get it going.
- *Usefulness*: this refers to the benefit of this game, like keeping the elderly fit.
- *Stability*: this refers to how steady the performance of the application is.
- *Overall experience*

The following Figures 3-14 and 3-15 show the average feedback from each group.

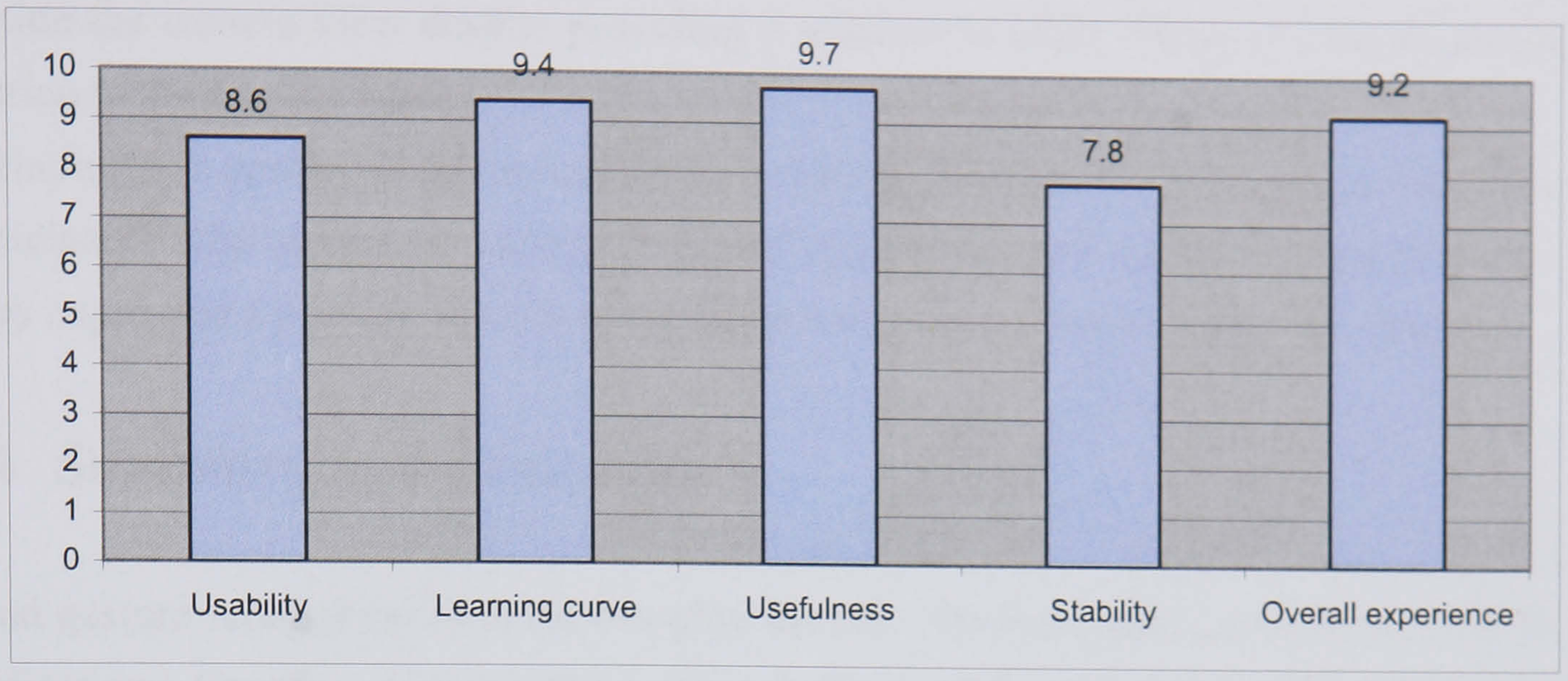


Figure 3-14: A chart shows the average response of our survey from non-computer specialist group.

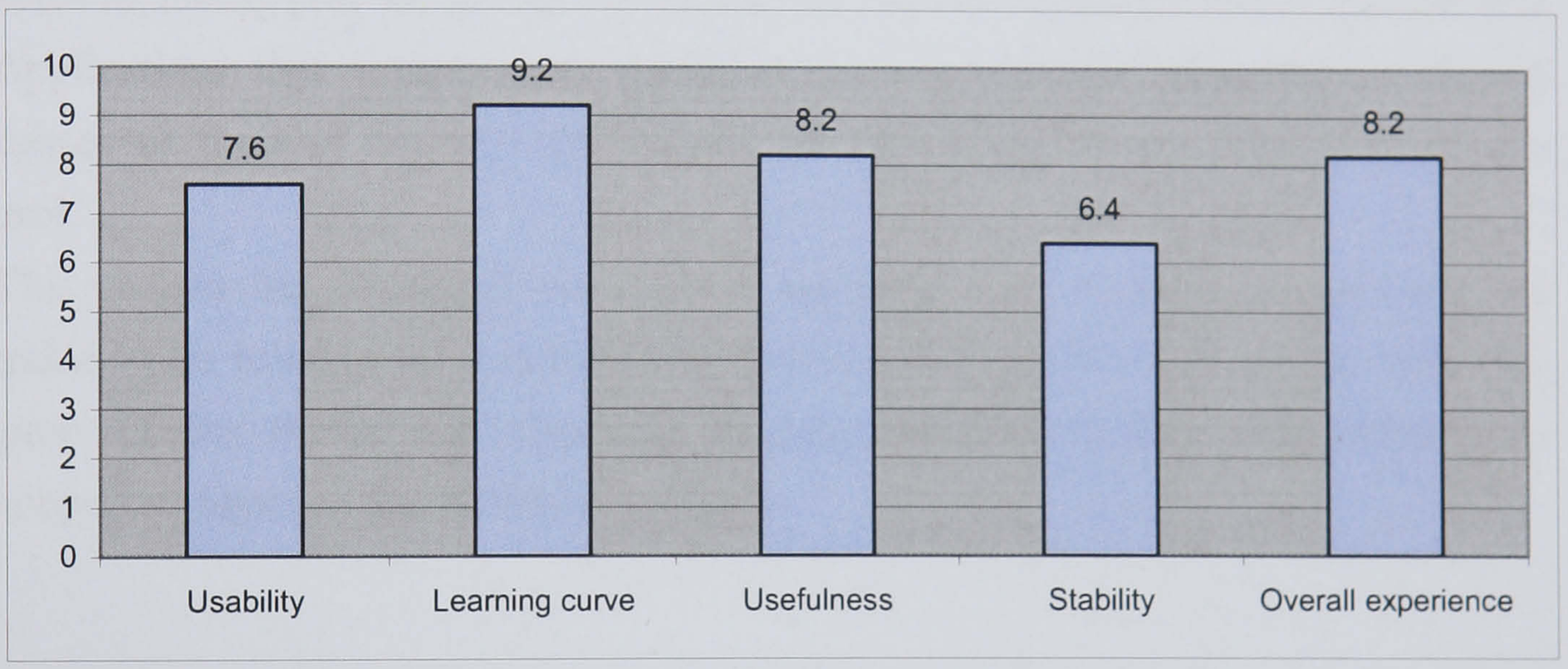


Figure 3-15: A chart shows the average response of our survey from computer specialist group.

Figures 3-14 and 3-15 show that stability is the most unsatisfactory issue in the game for both groups. Computer specialists were more critical of the game's stability than the non-computer specialists group. The game had a high positive response from both groups for its learning curve. Users picked up the game's scenario quickly and then performed the game. On the usability issues, all participants were happy with the minimal requirements needed to run the game (webcam and game software). Further, a repeated comment from several participants on the usability was that the game requires marking the field of play. It has been noticed that some participants moved their hands

outside the camera view despite providing a window to show where the hands are in relation to the camera view.

Having this prototype driven and controlled by the hand seems to inspire most of the participants who suggested excitedly several alternatives and supporting applications. They expressed a positive attitude towards the usefulness of such a game for children.

3.4 Summary and discussion

Hand gesture recognition systems can play the role of mouse and keyboard in various applications. Creating a mouse and keyboard driver replacement opens the possibility for hand gesture systems to be used with most of the current games. It can be used with word processor applications where users can write on air and from anywhere in a room and convert that to the word processor editor. There are several studies on hand gestures including [14, 33, 51, 55, 56, 63, 64, 74, 76] that explored several applications.

Applications that require more precision such as sensitive industrial production processes or medical surgery applications are less likely to use hand gestures for control.

This chapter has described our datasets depicting a set of hand gestures that are intended to be suitable for a multimedia interface (see Section 3.3). In the following chapters of this thesis, the introduced datasets are used to train and evaluate the developed recognition and detection methods.

CHAPTER 4

SEGMENTATION, TRACKING AND FEATURE REPRESENTATION

Segmentation, tracking and feature representation are core elements of most hand gesture recognition systems, and so must be fast, reliable, consistent, and produce discriminative outputs. For our prototype to work, the most distinctive features of the gesture such as hand shape, trajectory, and colour need to be extracted and represented efficiently. In common with appearance-based methods (see Section 2.4.2.2), we focus on segmenting and extracting the contour in each video frame and we make use of the skin colour for hand segmentation. After segmenting the hand, a reliable tracking technique is required to trace the hand in successive frames. Previous studies [10, 20, 31, 133] on skin colour show that in the HSV (Hue, Saturation, Value) domain, human skin colours have almost the same hue value and only differ in the brightness value. Therefore, skin colour segmentation and tracking are the preferred techniques to be used.

4.1 Segmentation and tracking

This research aims to segment the hand assuming there is no collision with any other objects. Segmentation and tracking of the hand are not the focus of this research and therefore standard methods are adopted as described in this section.

4.1.1 Segmentation

Segmentation is a process of grouping regions and features of the image that have similar characteristics together, with the aim of detecting semantically important objects. Two methods have been used:

For UBL and virtually created gestures datasets:

Optimal thresholding [134] for segmentation and CAMShift[1] [19] for tracking are used.

For OEH and OEW datasets:

Where the background has a wide range of variation (see Section 3.2.4), a more complex and flexible approach to segment and track hand gestures is used including modelling the skin colour distribution, similar to [133, 135]. Skin colour was modelled by collecting skin colour samples from training video clips and then fitting the probability distribution with a Gaussian model. Skin colours typically occupy a relatively compact area of HSV-space [134]. Skin distribution model is used to segment the hand with a CAMShift tracker.

4.1.1.1 Optimal thresholding segmentation

In thresholding, the value of each pixel in the image is classified as foreground and background according to whether or not its value exceeds a given value. There are several forms of thresholding including global, adaptive and optimal [99, 103]. The general form of thresholding can be described using Equation 4.1 where $I(i,j)$ is the intensity function, $B(i,j)$ is the output bitmap deciding the foreground/background class assignment, and T is the threshold value:

$$B(i,j)=\begin{cases}1 & for \quad I(i,j)\geq T\\ 0 & for \quad I(i,j)<T\end{cases} \qquad 4\text{-}1$$

[1] Continuously Adaptive Mean Shift

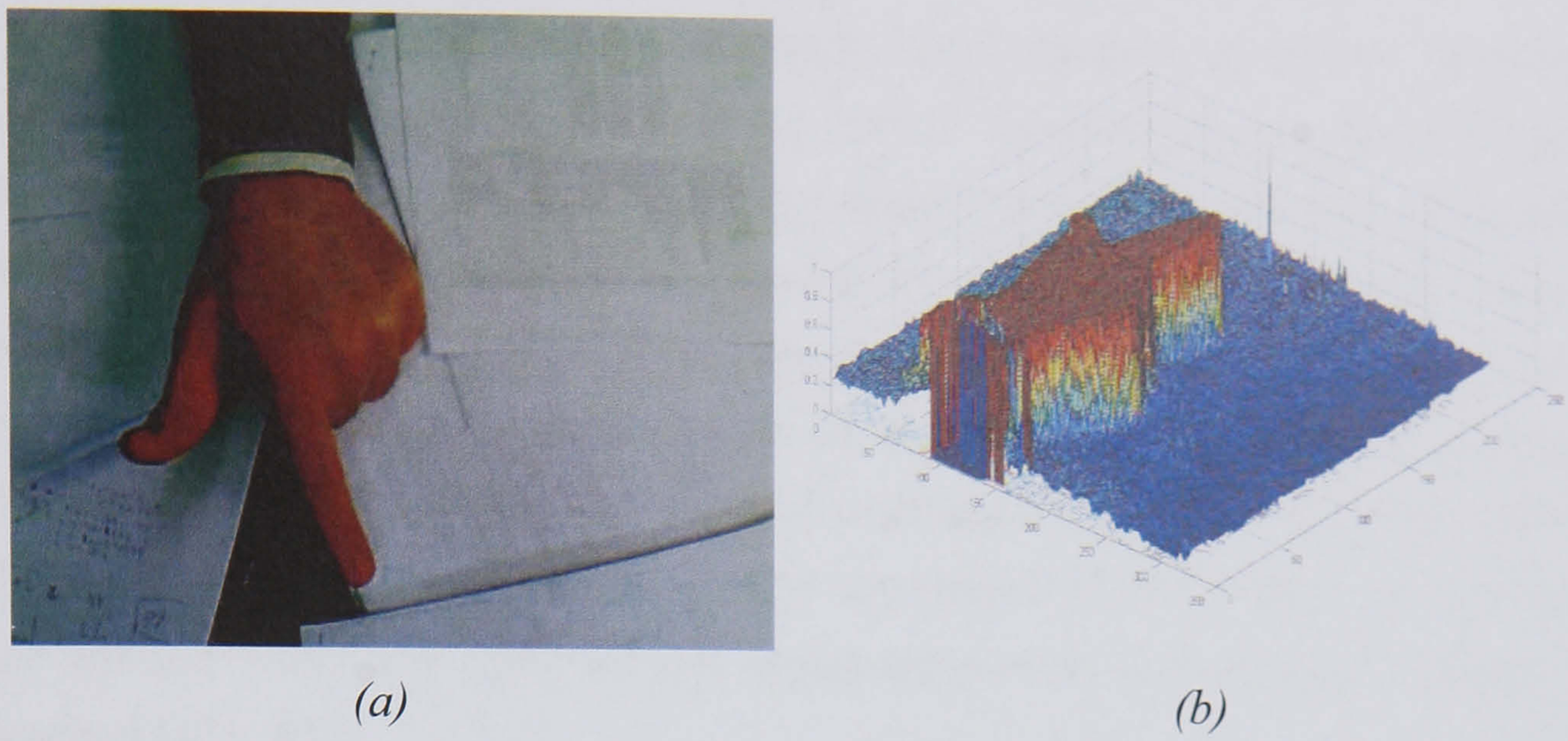
In this research, skin colour is the desired colour to be segmented; therefore, the chosen colour domain plays a major role. Figure 4-1 depicts an example of a hand segmented using thresholding in gray-level and HSV (Hue channel only). The gray level histograms of the input image shown in Figure 4-2 (a) is shown in Figure 4-3 (a), and the Hue channel histogram is shown in Figure 4-3 (b).

Thresholding techniques are very useful when the background does not have skin colour information. In contrast with gray image (mean of RGB) segmentation, segmenting the hand in HSV domain yields significant improvements, see Figure 4-1.

Histograms peaks do not always give a clear indication of where an appropriate threshold is located, especially when there is more than one peak. Optimal thresholding requires estimation of a parametric density function of the gray level distribution for finding the optimal threshold between the foreground and the background (more details is in [99, 103]). Usually in the optimal thresholding, estimate assumes that the gray level distributions are Gaussian. This estimation is not always ideal as the Gaussian distribution is not a generic representation. This estimation requires determining the parameters of the probability density function. In contrast, mean shift estimates the mode of the distribution function.

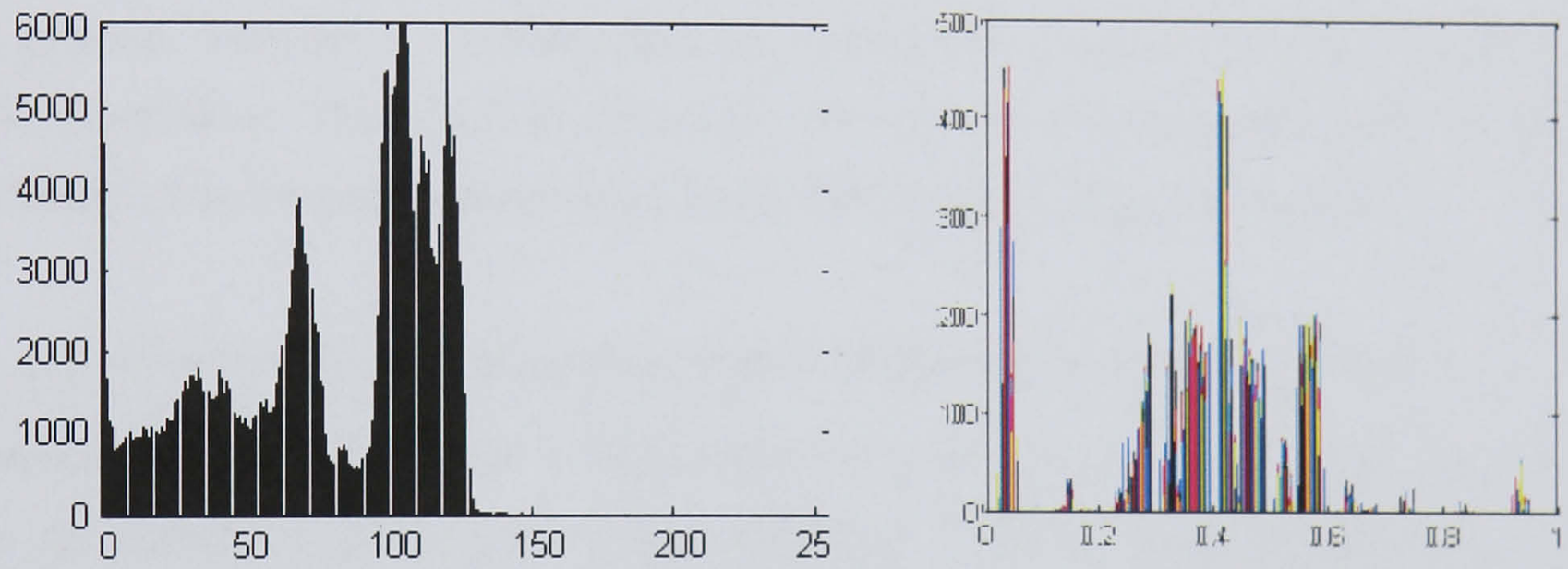
(a) *(b)* *(c)*

Figure 4-1: depicts in (a) an input image, (b) is the thresholded gray level image, while (c) is the thresholded image using hue channel in HSV colour domain which shows an improved result.



(a) (b)

Figure 4-2: depicts in (a) an input image of a hand, (b) is the HSV colour representation of image (hue channel).



(a) (b)

Figure 4-3: (a) the gray level histogram of input image Figure 4-2 (a), (b) is the histogram of the Hue channel of the same input image.

4.1.1.2 Mean shift segmentation

Pixel values can be seen as points in a multi-dimensional feature space (*e.g* RGB, HSV). Prominent structures such as hand in an image often result dense clusters in this feature space. These clusters can be used to segment the foreground from the background. By mapping all image points, the feature space gets much denser in locations corresponding to significant image features, namely the colour features. In our case, the dense regions form clusters of foreground or background.

The Mean Shift algorithm is a non-parametric technique that follows the gradient of a probability distribution up to find the nearest dominant mode (peak) of an image distribution. The Mean Shift algorithm clusters image features by associating each point with a peak of the data set's probability density. For each point, Mean Shift computes its associated peak by first defining a window at the data point of radius r and computing the mean of the points that lie within the window. The algorithm then moves the window to the mean and iterates until convergence, *i.e.* until the shift is less than a threshold which is usually determined by the spread of data points. At each iteration, the window shifts to a more densely populated portion of the data set until a peak is reached [136, 137].

4.1.2 Tracking

Tracking aims to recover a stable stream of hand segments during the performance of the gesture. Section 4.1.1 discussed the segmentation stage and the use of the Mean Shift algorithm. This section discusses the use of tracking methods by presenting CAMShift tracking technique steps and then evaluating its performance.

4.1.2.1 Continuously Adaptive Mean Shift tracking (CAMShift)

Bradski's CAMShift tracker [20] has proved to be one of the most successful trackers due to effective performance and minimal training requirements. For example, CAMShift performs efficient tracking of head and face in a perceptual user interface [20]. The algorithm is a generalization of the Mean Shift algorithm [136-138] which can only deal with static distributions. The Mean Shift algorithm provides a way to find the density modes without estimating the density. On the contrary, the CAMShift tracker is designed for dynamically changing distributions. The size and location of the probability distributions change during tracking due to object movement, changing illumination conditions, viewing angle, shadows and so on.

CAMShift uses colour information to generate a probability distribution which is utilized to locate and then to subsequently track an object in a video sequence. It locates the peak of a distribution by iterating in the direction of maximum increase in probability density. The probability density is recomputed in each frame on the basis of the histogram back-projection [20]. The density associated with each pixel represents the likelihood of this pixel belongs to ROI. Spatial moments are used during iterations

towards the mode of the distribution. The CAMShift algorithm differs from Mean Shift in that the target and the candidate distributions are used to iterate towards the peak.

Parametric and non-parametric statistical methods can be utilized to represent colour distributions. The histogram is the most widely applied non-parametric density estimator. A histogram is usually computed by counting the number of pixels in a region of interest that has a given colour, in this research the skin colour. The colours are quantized into bins. This operation allows similar colour values to be clustered as a single bin.

CAMShift can be summarized in the following steps [20, 139, 140] see Figure 4-4:

I. Allocate the region of interest (ROI) of the probability distribution function to the entire image.
II. Change colour space from RGB to HSV
III. Detect the skin colour in the initial frame using the pre-trained skin colour distribution model.
IV. Find the boundaries of the detected blob and use it as an initial location of the Mean Shift search window. The selected location is the target distribution to be tracked.
V. Calculate a non-parametric colour probability distribution of the region centred at the Mean Shift search window.
VI. Repeat Mean Shift algorithm to find the centroid of the probability distribution. Then, store the zero moment (area) and centroid location.
VII. In the successive frame, move the search window centre to the mean location found in Step IV and set the size of the window to a function of the zero moment. Go to Step III.

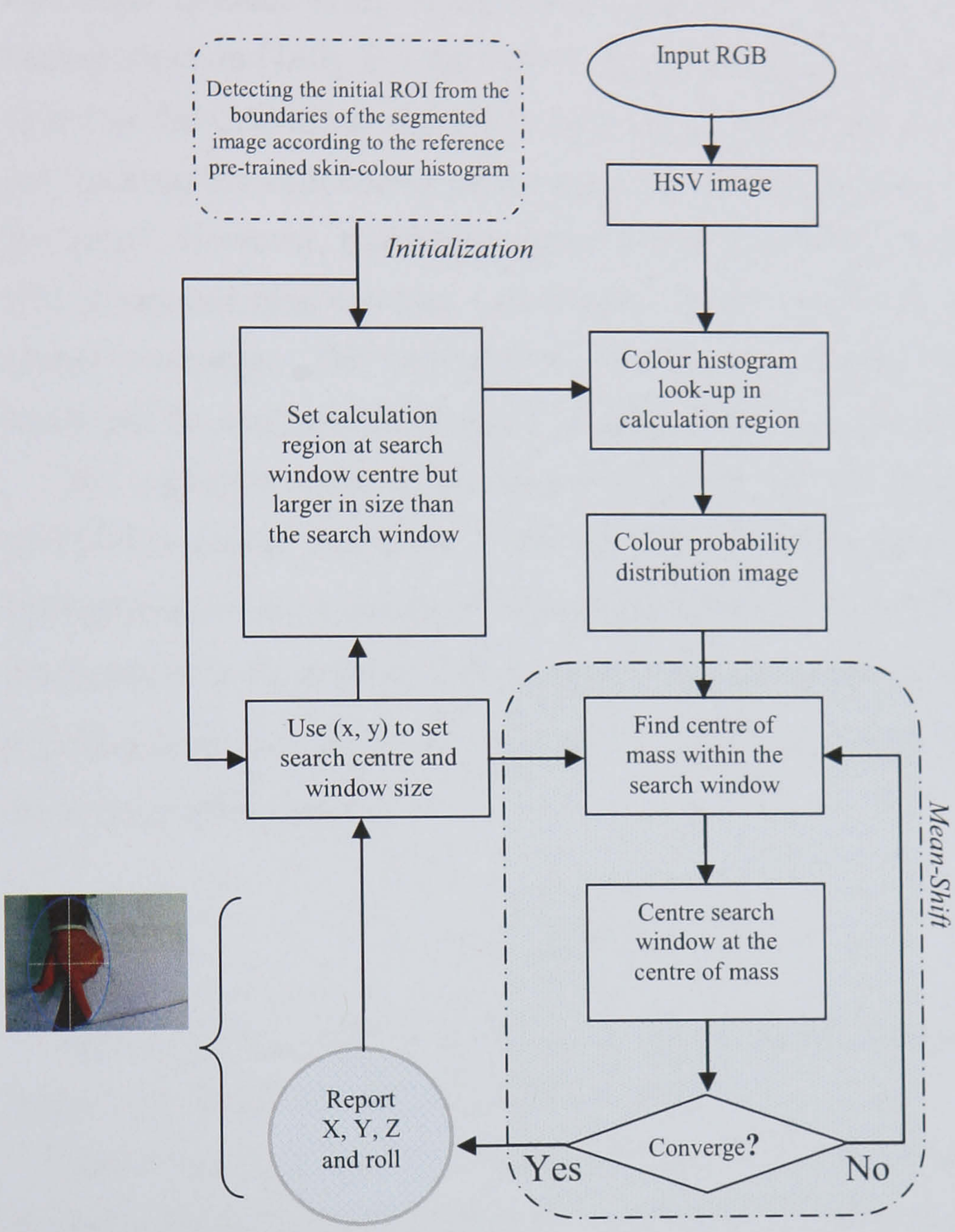


Figure 4-4: Block diagram of CAMShift tracker. This chart is based on OpenCv design in [139].

Computing the probability distribution function of an image (PDF) requires a method that associates a pixel value with a probability that the given pixel belongs to the target (human skin). Associating image pixel values to histogram bins is called back-projection [140]. For the PDF to be generated, an initial histogram is computed at *Step I* of the CAMShift algorithm from the initial ROI of the filtered image. Since we are tracking the skin colour of the hand, the hue channel of the HSV colour domain is the target. However, multidimensional histograms from any colour space can be used. The histogram bins are then scaled between the minimum and maximum probability image intensities. The values of the bins in the resulting histogram reflect the likelihood that the corresponding colour is classified to the skin colour.

The back-projection of the target histogram for any successive frame produces a probability image where the value of each pixel describes probability. Using m-bin histograms and an n image of the normalised pixel at locations $\{x_i\}_{i=1\ldots n}$ a $\{b\}_{u=1\ldots m}$ histogram can be defined. The histogram can be computed as in Equation 4-2: where function b maps from pixel at location x_i to the histogram bin index $b(x_i)$, $\delta[.]$ is the Kronecker delta function:

$$q_u = \sum_{i=1}^{n} \delta[b(x_i) - u] \qquad 4\text{-}2$$

Histogram bin values are scaled to be within the discrete pixel range of the 2D probability distribution function of an image.

The colour distribution is to some extent stable under object rotation and scaling. It is also sufficiently stable to partial occlusion while edge-based methods are not. The major drawback with modelling the colour distribution with histograms is the lack of convergence of the true density if the data set is small. A shadow does not significantly affect the hue colour component of the CAMShift tracker when using the HSV domain. Shadow reduces mainly the illumination component and affects the saturation component. Since we are using this tracker as part of application, the algorithm is intended to spend the lowest possible number of CPU cycles. The colour model is created by taking only a 1-D histogram of the hue component. This algorithm may fail to track the object when hue alone cannot be sufficient to distinguish the targets from the background.

Search window parameters

The centroid of the PDF over the search window computed in Step III is found using moments. Given that $P(x, y)$ is the intensity function of the discrete probability image at (x, y) within the search window. Equation 4-3 is the general formula for calculating 2D moments of any order:

$$M_{nm} = \sum_x \sum_y P(x,y).x^n.y^m \qquad 4\text{-}3$$

Moment M_{00} is the total summation of pixel values in the probability image:

$$M_{00} = \sum_x \sum_y P(x,y) \qquad 4\text{-}4$$

Moments M_{10} and M_{01} are the distribution of pixel values on both axes:

$$M_{10} = \sum_x \sum_y P(x,y)x \quad \textit{and} \quad M_{01} = \sum_x \sum_y P(x,y)y \qquad 4\text{-}5$$

Coordinates of the mean of the search window are given by:

$$\bar{x} = \frac{M_{10}}{M_{00}} \quad \textit{and} \quad \bar{y} = \frac{M_{01}}{M_{00}} \qquad 4\text{-}6$$

The algorithm iterates the computation of the centroid and relocates the search window until the location difference is beneath a predefined value. Using the zero moment M_{00}, CAMShift modifies the search window size in the course of its operation providing M_{00} is not equal to zero. The initial location and size of the search window is set manually in the early experiments that were performed using the CAMShift tracker. Then we select the location and the size of the search window automatically using a predefined skin colour distribution (Hue channel only). This is done by evaluating each pixel in the initial frame against the predefined skin colour distribution. If the tested pixel value is within one standard deviation of the distribution mean then it is considered as skin pixel and labelled and so on. Consequently a segmented image is produced. Using the produced image, a rectangle around the detected object is fitted. This rectangle is the initialization window for the tracker. If no skin colour is detected in the frame then the next frame is considered to be the initial one and so on.

Bradski's CAMShift tracker [20] computes the orientation and scaling of the tracked object. The orientation θ of major axis and scale of the detect window is computed by finding the equivalent blob which shares the same moments measure from the used probability distribution image. Moments M_{20} and M_{02} are the eigenvalues of the pixels distribution which represent the length and width of the probability distribution for the tracked window.

(a) (b) (c)

Figure 4-5: (a) input image. (b) the output of CAMShift tracker without weighting the window. (c) the output of CAMShift after weighting the skin colour using the Euclidian distance function, pixels closer to the centre have been given more weight.

$$M_{20} = \sum_x \sum_y P(x,y)x^2 \quad and \quad M_{02} = \sum_x \sum_y P(x,y)y^2 \qquad 4\text{-}7$$

Moment M_{11} is the variance of pixels:

$$M_{11} = \sum_x \sum_y P(x,y)yx \qquad 4\text{-}8$$

The following parameters are used for finding the orientation and equivalent search window dimensions:

$$\alpha = \frac{M_{20}}{M_{00}} - \bar{x}^2 \quad \& \quad \beta = 2\left(\frac{M_{11}}{M_{00}} - \bar{x} \cdot \bar{y}\right) \quad \& \quad \eta = \frac{M_{02}}{M_{00}} - \bar{y}^2 \qquad 4\text{-}9$$

θ is the orientation of the tracked window:

$$\theta = 0.5 \tan^{-1}\left(\frac{\beta}{\alpha - \eta}\right) \qquad 4\text{-}10$$

λ_1 and λ_2 are the dimension of the equivalent rectangle from the centroid distribution:

$$\lambda_1 = \sqrt{\frac{(\alpha+\eta) + \sqrt{\beta^2 + (\alpha-\eta)^2}}{2}} \qquad \lambda_2 = \sqrt{\frac{(\alpha+\eta) - \sqrt{\beta^2 + (\alpha-\eta)^2}}{2}} \qquad 4\text{-}11$$

The Intel OpenCV implementation of CAMShift tracker has been used in this research [19]. As mentioned earlier in this section, the manual ROI initialization which requires drawing a rectangle is removed by building the skin colour distribution that was used to detect and to initialize the tracker.

Calculating the target model for localization has been discussed in Comaniciu *et al.* [136, 138] and Allen *et al.* [140].

4.1.2.2 Evaluation

Tracking performance can be evaluated in several ways. For evaluating the performance of the CAMShift tracker on the captured datasets, positional tracking accuracy is utilized by computing the output of the tracker with ground truth obtained by manual labelling. Measuring how well the CAMShift tracker is able to determine the position of a target hand gesture is the aim. Several metrics exist for positional tracker evaluation as in Needham and Boyle [141]. These metrics provide measurements of the mean, standard deviation, median, min and max of two trajectories. These two trajectories are captured: the first by the CAMShift tracker and the second by manually labelled video streams (ground truth).

Given a trajectory T of a hand gesture, we can describe the trajectory as a series of points in the image plane, as in Equation 4-12:

$$T = \{(x_i, y_1), (x_2, y_2), ...(x_n, y_n)\} \qquad 4\text{-}12$$

Consider two trajectories (T, T') with the same number of points:

$$T = \{(x_i, y_i)\} \text{ and } T' = \{(x_i', y_i')\} \qquad 4\text{-}13$$

The Euclidian distances between corresponding points in T and T' are given by Equation 4-14:

$$d_i = \sqrt{(x_i' - x_i)^2 + (y_i' - y_i)^2} \text{ and } d = (d_1, d_2 .... d_n) \qquad 4\text{-}14$$

Generally, the mean is used to summarise these distances in addition to median, standard deviation, min, and max. Given d is an ordered series, Equations 4-15 shows how to compute these values:

Mean $$\bar{d} = 1/n \sum_{i=1}^{n} d_i$$

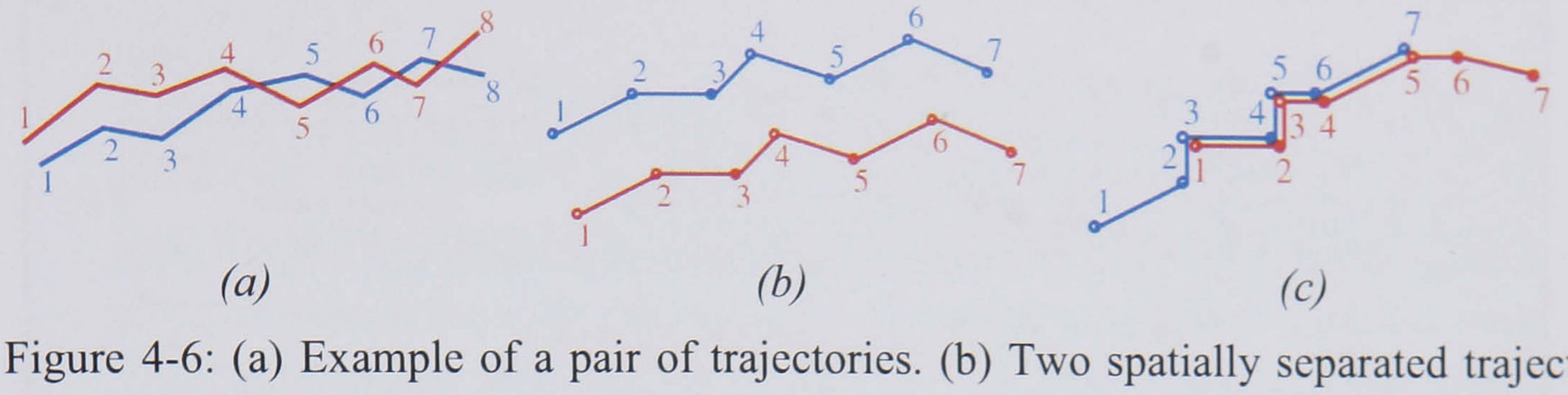
Median $$med = \begin{cases} d_{\frac{n+1}{2}} & if\ n\ \ odd \\ 1/2\left(d_{\frac{n}{2}} + d_{\frac{n}{2}+1}\right) & if\ n\ \ even \end{cases} \qquad 4\text{-}15$$

Std deviation $$\sigma = \sqrt{1/n \sum_i^n \left(d_i - \bar{d}\right)^2}$$

Max max= the largest d_i

Min min= the smallest d_i

To evaluate the similarity between two trajectories, different types of trajectory matching can be studied. The pairs of trajectories in Figure 4-6 illustrate the matching procedure. More sophisticated methods can be used to allow for consistent spatial or temporal displacements. Figure 4-6 contrasts such trajectories with fully aligned trajectories that are utilized here.



(a) (b) (c)

Figure 4-6: (a) Example of a pair of trajectories. (b) Two spatially separated trajectories. (c) Two temporally separated trajectories [141].

A ground truth of 15 hand gestures (sequences) is generated by marking the Centre of Mass (CoM) of 5 gestures from each setting: UBL, OEH and OEW datasets.

These hand gesture sequences have a length that varies from 63-97 frames per gesture with an average of 74 frames. Simple annotation software using Matlab is developed to mark the CoM of each frame of the gesture. The developed annotation software displays each hand gesture as a sequence of frames. The operator (user) checks how clean the segmentation of the hand is in each frame. If the segmentation is not sufficient enough (visual inspection), a replacement manual segmentation is performed. On the other hand, if the segmentation is good enough, the operator should expect the predicted CoMs to be close to the real one of the hand (what the operator subjectively believes this to be). Figure 4-7 shows a snapshot of developed annotation software. The red-cross in the middle of the processed frame (on the left) is the predicted CoM and the top of the yellow pointer is the actual one. The Operator can perform a manual segmentation on the frame in order to get a more precise prediction of the CoMs. Clicking on the correct CoMs adds the (x, y) to the output file (annotation result) in the form of a spreadsheet with gesture and frame IDs.

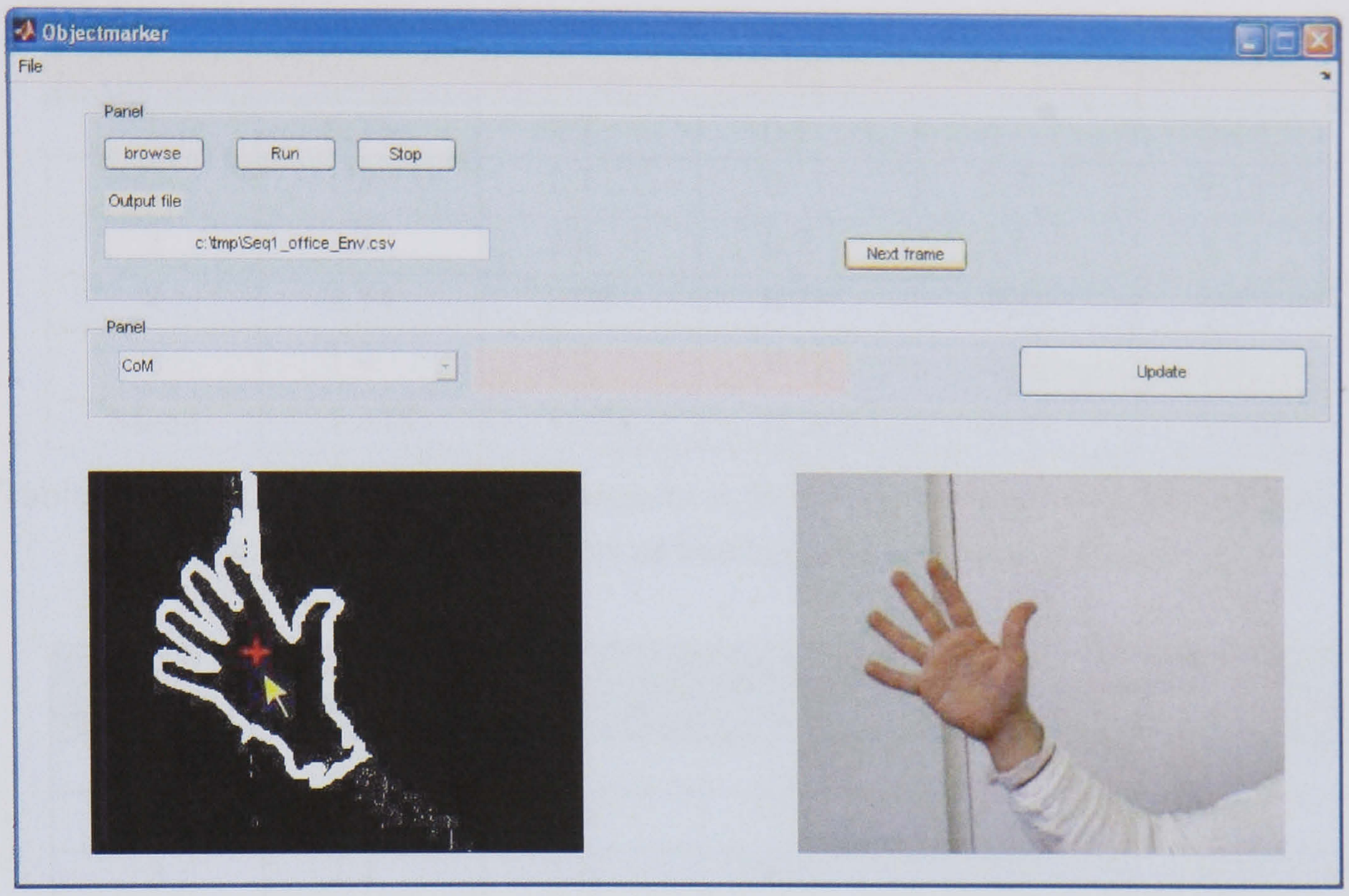


Figure 4-7: The developed annotation software: the yellow pointer refers to where the CoM should be and the red-cross is the predicted CoM. Notice that the shift is due to the error in the segmentation. For each frame of the hand gesture, the CoMs are marked to produce our ground truth. The output is a spreadsheet file which contains gesture ID, frame ID and CoMs information.

In these evaluation experiments, no spatial or temporal shift of hand gesture trajectories is made. Trajectories which correspond to Figure 4-6 (a) are the only ones studied here. Table 4-1, 4-2 and 4-3 show the results obtained. The notation T_i refers to the sequence and G marks the ground truth corresponding to the same sequence. Five sequences $i = \{1, 2, 3, 4, 5\}$ are studied. $\overline{T}$ is the average for each of the measures.

Sequence	Measure of distance				
	Mean	Median	Min	Max	S.d
1	0.25	0.7	0.1	4	1.15
2	1.5	1.7	0	3.9	1.11
3	2.3	2.4	0.2	4.6	1
4	2.14	2.24	0.3	3.84	1.5
5	1.2	2.1	0.11	3.68	1.3
Mean	1.478	1.828	0.142	4	1.142

Table 4-1: Results of comparing 5 sequences from UBL dataset with ground truth, for various distance measuers of two trajectories, measured in pixels.

Sequence	Measure of distance				
	Mean	Median	Min	Max	S.d
1	2.6	2.3	1.08	7.9	1.48
2	2.9	2.9	2.1	9.1	2.4
3	1.1	1.6	0.6	4.3	1.45
4	2.1	2.6	1.89	6.2	1.79
5	3.07	3.5	2.6	8.74	2.65
Mean	2.354	2.58	1.654	7.248	1.954

Table 4-2: Results of comparing 5 sequences from OEH dataset with ground truth, for various distance measuers of two trajectories, measured in pixels.

Sequence	Measure of distance				
	Mean	Median	Min	Max	S.d
1	10.3	11	1	20.6	6.16
2	11.69	12.6	6.3	34.6	9.6
3	9.8	10.6	8.6	18.97	4.66
4	14.1	15.3	3.5	42.1	12.47
5	18.2	18.9	7.1	39.4	9.2
Mean	12.818	13.62	5.3	31.134	8.418

Table 4-3: Results of comparing 5 sequences from OEW dataset with ground truth, for various distance measuers of two trajectories, measured in pixels.

Tables 4-1, 4-2 and 4-3 present the evaluation results of the CAMShift tracker on hand gesture in different settings. The tracker seems to deliver much better results when the gesture is performed in controlled environments (Table 4-1). When the skin-like colour patches in the scene increase, the performance of CAMShift seems to decrease (Table 4-3).

Quantitative positional evaluation provides a useful tool that can indicate how well a specific tracker is performing. CAMShift has performed reasonably well under all settings on a small sample of the developed dataset as shown in the previous tables. In this framework, the overall recognition rate is influenced not only by the tracker accuracy, but also by the classification accuracy (Chapter 5).

4.2 Feature representation

In Chapter 2, Section 2.5.2 methods of feature representation are reviewed. In this research, an objective is to investigate the use of Zernike Velocity Moments (ZVM) in hand gesture recognition. The use of ZVM are evaluated by comparing the results to the standard use of Zernike moments with an HMM. The next section introduces moments, and the use of moments within classification is discussed in the next chapter.

4.2.1 Moments

Moment invariants were first presented many years before the appearance of the first computer by David Hilbert [142] in algebraic invariants theory of the 19th century. Moment invariants were presented to the world of shape description and pattern recognition in 1962 by Hu [21]. Hu utilized the results of the theory of algebraic invariants and derived seven algebraic expressions. These expressions were invariant to rotation, scaling and translation of 2-D objects after normalization. Since that time, numerous contributions have been devoted to improvements and generalizations of Hu's invariants.

4.2.1.1 Basics of moments

Let f denotes the pixel values in an image as a function of x and y. The discrete centralized moment μ_{pq} of order (p, q) is given in Equations 4-16 and 4-17. η_{pq} is the normalized central moments.

$$\mu_{pq} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q f(x, y) \qquad 4\text{-}16$$

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}} \quad \& \quad \gamma = \frac{p+q}{2} + 1 \qquad 4\text{-}17$$

$(\bar{x}, \bar{y})$ denote the CoM of the targeted object.

Different degrees of moments represent different features of the shape. For example: μ_{00} is the summation of pixel values in the whole image (if image is binary, the number of white pixels), μ_{10} and μ_{01} represent the pixel distribution along X and Y axes, moments of degree two represent the variance; skewness is the moments of degree three, kurtosis is the moments of degree four and so on. Combining moment features with higher degrees of moments provides a discriminative description of an image.

4.2.1.2 Hu moments

Invariant description is useful in the domain of pattern recognition. Most description methods can be modified to give a scaling and translation invariant description by firstly normalizing the size and centralizing the object of interest in the image. Rotation invariance is much harder to obtain. Hu [21] combined the algebraic invariant with moments and introduced the following seven expressions that are invariant to scaling, translation and rotation.

$$H_1 = \eta_{20} + \eta_{02} \qquad 4\text{-}18$$

$$H_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \qquad 4\text{-}19$$

$$H_3 = (\eta_{30} - 3\eta_{12})^2 + 3(3\eta_{21} - \eta_{03})^2 \qquad 4\text{-}20$$

$$H_4 = (\eta_{30} + 3\eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \qquad 4\text{-}21$$

$$\begin{aligned} H_5 = {} & (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12}) \\ & \left[(\eta_{30} - 3\eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\right] \\ & + (3\eta_{21} - \eta_{03})(\eta_{12} + \eta_{03})\left[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right] \end{aligned} \qquad 4\text{-}22$$

$$H_6 = (\eta_{20} - \eta_{02})\left[(\eta_{30}+\eta_{12})^2-(\eta_{21}+\eta_{03})^2\right] + 4\eta_{11}(\eta_{30}+\eta_{12})(\eta_{21}+\eta_{03}) \qquad 4\text{-}23$$

$$H_7 = (3\eta_{12} - \eta_{30})(\eta_{30} + \eta_{12})\left[(\eta_{30}+\eta_{12})^2-3(\eta_{21}+\eta_{03})^2\right] + (3\eta_{12}-\eta_{30})(\eta_{21}+\eta_{03})\left[3(\eta_{30}+\eta_{12})^2-(\eta_{21}+\eta_{03})^2\right] \qquad 4\text{-}24$$

Equations 4-[18 to 24] show that Hu moments are based on basic moments of lower degree; therefore, some features such as kurtosis are not counted.

4.2.1.3 Zernike moments

Zernike moments are a class of orthogonal moments that have been effective for image representation. The magnitudes of Zernike moments are rotation and reflection invariant [143] and can be easily constructed to an arbitrary order [144]. Although higher order moments carry more fine details of an image, they are also more susceptible to noise. The Zernike polynomials are a set of complex, orthogonal polynomials defined over the interior of a unit circle [144]:

$$V_{pq} = V_{pq}(\rho,\theta) = R_{pq}(\rho)e^{jp\theta} \qquad 4\text{-}25$$

$$R_{pq}(\rho) = \sum_{s=0}^{(p-|q|)/2}(-1)^s\left(\frac{(p-s)!}{s!\left[\frac{p+|q|}{2}-s\right]!\left[\frac{p-|q|}{2}-s\right]!}\right)\rho^{p-2s} \qquad 4\text{-}26$$

$$\textit{where} \quad \theta = \tan^{-1}\frac{y}{x} \qquad \rho = \sqrt{x^2+y^2} \qquad |q| \le p \qquad 4\text{-}27$$

By projecting the image function onto the basis set, the Zernike moment of order p with repetition q [144] can be calculated by:

$$A_{pq} = \frac{p+1}{q}\sum_x\sum_y f(x,y)V_{pq}(x,y) \qquad 4\text{-}28$$

$$\textit{where} \quad x^2+y^2 \le 1$$

4.2.1.4 Zernike velocity moments

Zernike velocity moments (ZVM) [1, 2] are essentially a weighted sum of Zernike moments over a sequence of frames (length T), weighted by a real-valued scalar function of the displacement of the CoM between consecutive frames:

$$A_{mn\beta\lambda} = \frac{m+1}{\pi} \sum_{i=2}^{T} \sum_{x} \sum_{y} f_i(x,y) U(i,\beta,\lambda) S(m,n) \qquad 4\text{-}29$$

$$S(m,n) = \left[V_{mn}(\rho,\theta) \right]^* \qquad 4\text{-}30$$

$$\bar{A}_{mn\beta\lambda} = \frac{A_{mn\beta\lambda}}{A.I} \qquad 4\text{-}31$$

$$U(i,\beta,\lambda) = (\bar{x}_i - \bar{x}_{i-1})^{\beta} (\bar{y}_i - \bar{y}_{i-1})^{\lambda} \qquad 4\text{-}32$$

Where A is the average area (in number of pixels) of the moving object, i is the number of images in the sequence, $\bar{A}_{mn\beta\lambda}$ is the normalized ZVM and * denotes the complex conjugate. Equation (4-31) is restricted to the condition $x^2 + y^2 \leq 1$, while the shape structure contributes through the orthogonal polynomials [1].

ZVMs showed encouraging results when used for human gait recognition [1, 2]. This research introduces and evaluates ZVM in hand gesture recognition systems. ZVMs provide spatio-temporal descriptions that were classified using standard classification methods as will be discussed in Chapter 5. The results of ZVM are compared to the well established HMM coupled with spatial moments.

4.3 Summary and discussion

Segmentation and tracking are the foundations of any hand gesture recognition system that needs to be working effectively with minimized errors. This chapter has discussed the segmentation techniques that were explored in the presented experiments of Chapter 5 and 6.

The CAMShift tracker is an algorithm based on Mean Shift segmentation. It has been utilized in this research because it:

- requires minimal training
- can track irregular shapes and objects

- tolerates noise
- does not require specific hardware or specialised cameras
- is computationally efficient

One of the main problems of CAMShift is the need to select an initial blob for tracking. This problem is addressed by creating a Gaussian distribution model of the human hand skin colour, similar to [135], which was used to segment the hand in the initial frame and to create a blob that consequently tracked using CAMShift. It is important to consider how accurate the target application needs tracking to be. The hand gesture application can tolerate some error in the trajectory detection as long as this error does not exceed a certain limit which might change the nature of the gesture or cause misinterpretation with other similar gestures [94, 145].

In the evaluation stage of the CAMShift tracker, human mark-up of ground truth data is used which is a subjective process, and there are typically differences between ground truth sets marked up by different people. A simple way to deal with this issue is to take the mean of the mark-up of several markers. This will hopefully produce more accurate labelling. When a system is required to be at least as good as a human, the tracked trajectories should be compared to how well humans can mark-up the trajectories, and a statistical test needs to be performed to identify if they are significantly different.

It is hard to separate tracking, feature extraction and representation methods, in most cases they are all integrated with each other. It is important to note that descriptors need to produce a unique representation for an object and it is not necessary for the process to be reversible. This chapter has presented the segmentation and tracking techniques used followed by moments as a feature descriptor. The following chapters (Chapter 5 and 6) present experiments utilizing moments coupled with two classification methods. Evaluation of the overall performance and results are then presented.

CHAPTER 5

POSTURE AND GESTURE RECOGNITION

This chapter continues presenting each stage of our prototype framework (see Figure 3-12). Chapter 4 focused on segmentation and tracking, and introduced moments as the intended description methods. In this chapter, experimental results which use moments with classification methods are presented. These experiments are presented as follows:

Firstly, key-frames recognition; hand gesture is a sequence of poses as depicted in the chosen 8 dynamic hand gestures in this research (see Section 3.2). Nevertheless, users can also perform static gestures (postures) only to convey a specific message such as holding a thumb up for a while as a gesture of things going ok. Therefore, it would be beneficial to explore the key-frame recognition capability. Experiments on key-frame recognition are introduced using Hu and Zernike moments descriptors coupled with several classification methods.

Secondly, ZVM representation; a novel use of ZVM on hand gesture applications is presented. The ZVM descriptor produces a spatio-temporal representation of each gesture. The output of ZVM is classified using a Classification-via-Regression (CvR) method.

Finally, HMM classification; an experiment on the use of HMM as dynamic classifier is presented. A left-to-right HMM topology [146] is utilised. Each frame of the hand gesture is represented using the ZM descriptor and then HMMs are deployed to represent the gestures temporally. HMM parameters are learned from the training data. The potential of using HMM for gesture recognition was demonstrated by the experiments on both segmented (isolated) and continuous gesture recognition and detection.

The main focus of this chapter is to evaluate the suitability of ZVM coupled with CvR classification for hand gesture recognition and to introduce a new design of HMM classification using ZM features. The theory of both classification methods is presented briefly.

The implementation of ZM and ZVM is based on the LANS Matlab package [147]. All experiments are performed on a machine of 2.2 GHZ dual core 2 processor with 2 GB Ram.

5.1 Key-frame recognition

As a prelude to gesture recognition, preliminary experiments were conducted on the recognition of posture instances from the eight chosen gestures. If successful, this could be used to 'spot' gestures and justified further processing, as in Laptev [105, 112-114].

This section presents several experiments. Initially, unique key-frames are manually identified from the selected 8 gestures (see Figure 3-1). These key-frames are shown in Figure 5-1. These experiments use the three datasets (UBL, OEH and OEW) each of which corresponds to different environment settings as discussed in Chapter 3. As mentioned in Chapter 4, the thresholding technique is applied on the UBL dataset. Skin-colour CAMShift is applied on OEH and OEW datasets and then the stationary skin colour objects (head) is removed using frame differencing. The aim of these experiments is to measure how successfully key-frames can be classified into 8 categories. Each one of these categories represents a key-frame as in Figure 5-1.

To avoid biased results, an equal number of instances for each key-frame is selected. Each posture has 150 frames per dataset. These frames are collected from different participants. Hu and Zernike moments descriptors are computed from each foreground

segment. Table 5-1 shows the recognition rates when using Hu moments (see Equations 4-18 to 4-24) combined with three classifiers. Table 5-2 shows the recognition rates when using ZM with different degrees of moments. ZM degrees have been chosen by trial and error and through literature guidance [117, 144, 149, 150].

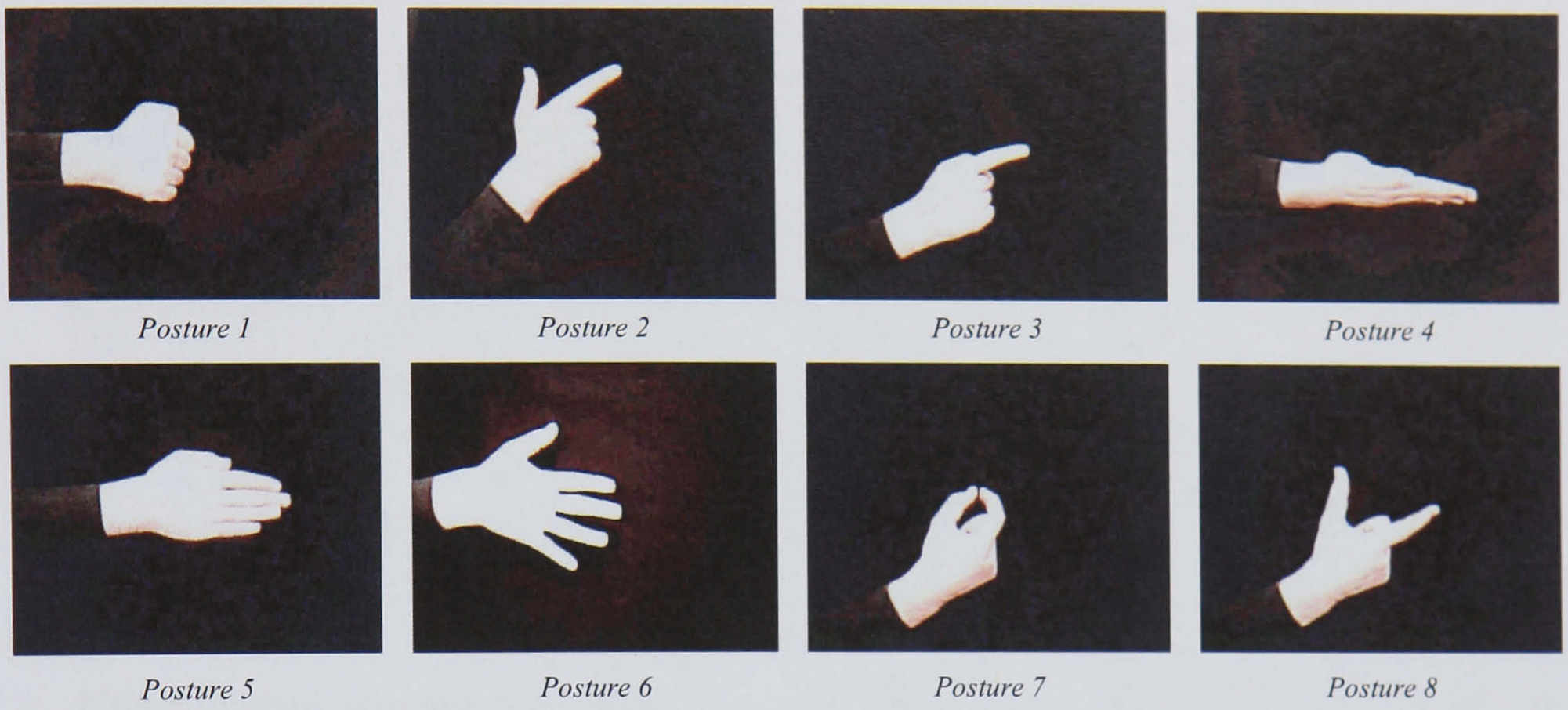


Figure 5-1: 8 unique frames are from our dataset testing gestures (in controlled background). These key-frames (postures) may appear more than once in the same gesture as in gesture "*A*" see Figure 3-1, where two key postures (1 and 6) mark the start and the end of this gesture "*A*".

The three classification methods used are K-means based classifier, Sequential Minimal Optimization (SMO), and Classification via Regression (CvR), see *Appendix A*. The Weka [148] implementations are used for these experiments. 10 fold cross-validation is used.

Hu moment with:	UBL dataset	OEH dataset	OEW dataset
K-means	100%	98.6%	83.5%
CvR	100%	98%	85.3%
SMO	100%	95.2%	77.75%

Table 5-1: shows the recognition rates on using Hu moments with K-means, SMO and CvR on three datasets: UBL, OEH and OEW.

	Zernike moments				
	(4,0)(6,0)	(6,0)(6,2)(8,0)	(6,0)(6,2)(8,2)(8,4)	(8,0)(8,2)(10,2)(10,4)	(10,2)(10,4)(12,2)(12,4)
K-means clustering					
UBL dataset	56%	91.1%	100%	100%	100%
OEH dataset	52.5%	68%	86.3%	97.1%	96.4%
OEW dataset	39.2%	46.4%	61.9%	83.2%	83.3%
CvR classifier					
UBL dataset	61.4%	93.3%	100%	100%	100%
OEH dataset	54.5%	69.4%	89.2%	96.1%	97.8%
OEW dataset	28.3%	45.6%	66.1%	84.5%	85.7%
SMO classifier					
UBL dataset	52.5%	75%	98.7%	100%	100%
OEH dataset	48%	65.7%	83.6%	92.7%	92.5%
OEW dataset	31.9%	43.4%	57.8%	78.1%	81.2%

Table 5-2: shows the recognition rates on using Zernike moments with K-means, SMO and CvR on three datasets: UBL, OEH and OEW.

Noise filtering was not performed in any of these experiments after the segmentation process. Furthermore, the effect of adding noise to postures on the recognition rate has not been explored. The effect of noise on posture, with the use of moments as descriptor, has been studied in Paschalakis and Lee's research [149] on aircrafts dataset. They reported a 100% recognition rates using moments of 5^{th} and 6^{th} degree with no noise. However, when 5% of the image is noise the recognition rate drops by 10%. Unfortunately, the comparison between the obtained recognition rates in this thesis and others is infeasible because of dataset mismatching.

The processing times of these experiments were 55 ms for each feature vector to be generated using the 7 Hu moments and 15-65 ms for ZMs, depending on the degree of ZM used. Hu and Zernike moment descriptors are implemented in Matlab [147] under Windows Vista. These processing times do not include the time required by the classifiers.

5.1.1 Discussion

Posture recognition can play a major part in enhancing the performance of hand gesture recognition by detecting some predefined frames. This is supported by the high recognition rate illustrated in Tables 5-1 and 5-2. In particular, it could help the hand gesture recognition system to locate the start and the end of the gesture automatically similarly to Laptev [105, 112-114]. Training the system on postures requires picking up

(isolating) key-postures from the video stream manually which is a subjective process and can result in error.

ZM of high degree gives a much improved performance as shown in Table 5-2. This finding is taken forward when applying ZM and ZVM for dynamic hand gesture recognition.

Tables 5-1 and 5-2 show that the recognition rates have dropped significantly when using the OEW dataset, around 20% less than the UBL dataset. This is due to segmentation error and loss of resolution. In the OEW dataset, hand gestures are about 2 meters away from the camera; therefore, it provides less detail and even a small segmentation error can significantly change the description vector. Figure 5-2 shows two examples from the OEH and OEW datasets, before and after segmentation. Figure 5-1 depicts examples from the UBL dataset. Clean segmentation, as in Figure 5-1, clearly give a higher recognition rates than noisier segmentation, as in Figure 5-2.

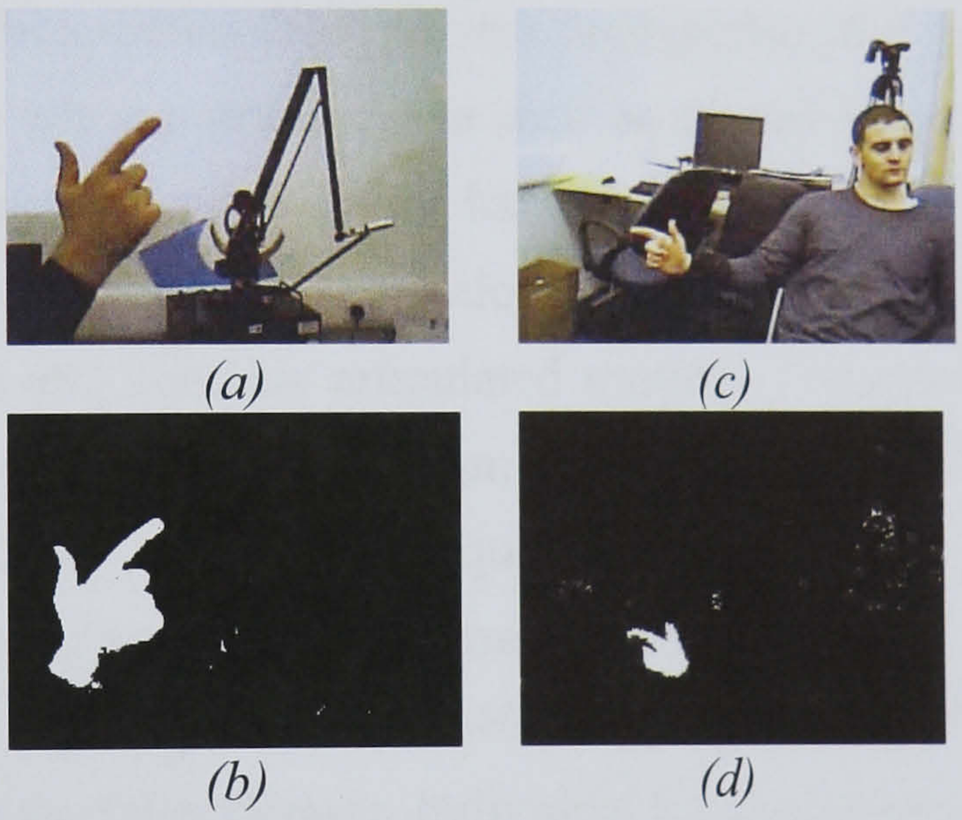


Figure 5-2: (*a)* is the input posture from OEH dataset, and (*b)* is the output after segmentation based on the skin colour. (*c)* is the input posture from OEW dataset, and (*d)* is the output after a segmentation based on the skin colour and removing the less mobile objects (such as the head).

Using Weka software [147], it is straightforward to experiment with different classification and clustering methods. The classification has been explored using several methods and we present only the best three from the experiments.

Recognition postures are correct in 85.7% of the cases when selecting the CvR classifier in the worst possible environment where the whole body is present in the scene (OEW dataset). In general, this is not good enough for a robust high-end user interface. However, considering that this result was achieved by analysing data

collected from different people rather than from what the recognizer was trained on, a system of user-dependent data (trained and tested) would yield an improved correct classification rates. On the other hand, the target system can be designed to allow the user to get immediate feedback if they performed a mis-classified gesture. This allows the participants to do small adjustment to improve the recognition. This can be done, for example, by showing the user the mis-classified postures and how they should be.

This experiment can be further validated through independent research such as Hse and Newton [144] on sketched symbols, Paschalakis and Lee [149] on aircraft recognition, and Erola *et al.* [82] comprehensive recent survey on pose recognition and estimation. Previous studies showed similar recognition rates on key-frames recognition.

5.2 Hand gesture recognition

Hand gesture recognition differs from posture recognition as it requires considering the temporal information when processing, as well as spatial features. In Section 5.1, the potential of using posture recognition for hand gesture applications was illustrated, but in general postures are ambiguous in relation to gestures. In fact, the majority of hand gestures are dynamic and contains articulated motion. For example, Gesture "*A*", see Figure 5-3, has different hand shapes from the start to the end and has continuously changing locations in each frame of the sequence.

In this section, manually isolated gestures from our datasets UBL, OEH and OEW are used; (Section 3.2) giving a total of 240 instances for the UBL and OEW datasets and 520 instances for the OEH dataset. Individual hand gestures are delineated by hand via marking the start and the end of each gesture. The UBL dataset examples are processed using thresholding, as discussed in Chapter 4, for generating the hand segments. OEH and OEW datasets are segmented using a skin-colour model. 10 fold cross-validation is performed on all of following experiments in this section. For classification, CvR and HMM methods are used.

12 instances per gesture from one participant in office environment settings (OEH dataset) are collected for testing hand gesture in personalised fashion (user-dependant), where the system was trained and tested on gesture examples collected from the same user. While in the rest of the experiments we tested and trained data of user-independent.

5.2.1 Using ZVM+CvR

5.2.1.1 Overview

In the first experiment, we used a ZVM descriptor to characterise the video segment corresponding to each gesture instance. The speed at which a gesture is performed results in video segments of different lengths within the dataset (between 30 and 120 frames and an average of 90). There was no attempt to interpolate gestures to a fixed number of frames since the weighted mean computed by the ZVM is invariant to the overall speed of a gesture, although not to non-linear variations in the temporal rate of execution. Computing the ZVM on a typical video segment takes up to 1.5 sec (in Matlab).

5.2.1.2 Experiments

Evaluating 8 gestures:

A feature vector was obtained from three Zernike Velocity Moments, defined by setting the four parameters (p,q,β,λ) to (12,4,0,0), (12,4,0,1), and (12,4,1,0) - see Equation 4-29.

After experimenting with the following standard methods: K-means clustering, CvR and SMO classifiers, it has been observed that the best overall performance was obtained from the CvR classification method. The previously reported experiments on posture recognition support this finding. The confusion matrices using CvR are laid out in Tables 5-3, 5-4, 5-5 and 5-6.

		Predicted Gesture							
		"A"	"B"	"C"	"D"	"E"	"F"	"G"	"H"
Actual Gesture	"A"	**83.3%**	0%	3.3%	0%	3.3%	3.3%	0%	6.7%
	"B"	0%	**86.6%**	6.7%	0%	0%	3.3%	3.3%	0%
	"C"	0%	6.7%	**86.6%**	0%	0%	0%	6.7%	0%
	"D"	0%	0%	0%	**100%**	0%	0%	0%	0%
	"E"	3.3%	0%	0%	0%	**73.3%**	3.3%	0%	20%
	"F"	0%	0%	6.7%	0%	3.3%	**83.3%**	6.7%	0%
	"G"	0%	0%	3.3%	0%	0%	6.7%	**90%**	0%
	"H"	0%	3.3%	0%	0%	16.7%	3.3%	0%	**76.6%**

Table 5-3: The confusion matrix using CvR classifier and ZVM descriptor for UBL dataset (see Figure 3-1) with 240 instances and 10 fold cross-validation. The mean recognition rate is 84.9%.

		Predicted Gesture							
		"A"	"B"	"C"	"D"	"E"	"F"	"G"	"H"
Actual Gesture	"A"	**63.3%**	0%	6.7%	0%	10%	3.3%	0%	16.7%
	"B"	0%	**66.7%**	13.3%	0%	0%	13.3%	6.7%	0%
	"C"	6.67%	16.7%	**60%**	0%	0%	6.7%	10%	0%
	"D"	6.67%	0%	6.67%	**83.3%**	0%	0%	3.3%	0%
	"E"	13.3%	0%	0%	0%	**46.7%**	0%	0%	40%
	"F"	0%	16.7%	13.3%	0%	0%	**60%**	10%	0%
	"G"	0%	6.7%	13.3%	0%	0%	10%	**70%**	0%
	"H"	10%	0%	0%	0%	36.6%	0%	0%	**53.3%**

Table 5-4: The confusion matrix using CvR classifier and ZVM descriptor for OEH dataset (see Figure 3-1) with 240 instances and 10 fold cross-validation. The mean recognition rate is 62.3%.

		Predicted Gesture							
		"A"	"B"	"C"	"D"	"E"	"F"	"G"	"H"
Actual Gesture	"A"	**26.7%**	6.7%	13.3%	10%	16.6%	3.3%	3.3%	20%
	"B"	16.6%	**20%**	30%	6.7%	3.3%	10%	6.7%	6.7%
	"C"	13.3%	3.3%	**23.3%**	6.7%	6.7%	23.3%	13.3%	10%
	"D"	10%	6.7%	6.7%	**36.7%**	10%	13.3%	10%	6.7%
	"E"	13.3%	13.3%	6.7%	3.3%	**26.7%**	6.7%	10%	20%
	"F"	10%	6.7%	13.3%	3.3%	13.3%	**40%**	6.7%	6.7%
	"G"	6.7%	6.7%	13.3%	6.7%	6.7%	16.7%	**33.3%**	10%
	"H"	13.3%	6.7%	6.7%	10%	26.7%	13.3%	6.7%	**16.7%**

Table 5-5: The confusion matrix using CvR classifier and ZVM descriptor for OEW dataset (see Figure 3-1) with 240 instances and 10 fold cross-validation. The mean recognition rate is 28.9%.

Evaluating 5 distinctive gestures:
Out of the 8 gestures (see Figure 3-1), there are 5 gestures that are highly distinctive from one another, as shown in Figure 5-3. ZVM is tested on these 5 gestures using 80 instances for each of the 5 gestures. These instances are collected from 10 participants in an office environments with hands only in the scene (OEH dataset) [150].

Gesture "A" *Gesture "B"* *Gesture "C"* *Gesture "D"* *Gesture "E"*

Figure 5-3: Five distinctive gestures used to train and test ZVM and CvR.

		Predicted Gesture				
		"A"	*"B"*	*"C"*	*"D"*	*"E"*
Actual Gesture	*"A"*	**82.5%**	2.5%	3.75%	1.25%	10%
	"B"	6.25%	**80%**	3.75%	6.25%	3.75%
	"C"	6.25%	2.5%	**87.5%**	1.25%	2.5%
	"D"	0	1.25%	2.5%	**90%**	6.25%
	"E"	7.5%	2.5%	2.5%	5%	**82.5%**

Table 5-6: The confusion matrix for CvR using ZVM descriptor on five gestures each of 80 instances (OEH dataset). The mean recognition rate is 84.5% [150].

5.2.1.3 Discussion

ZVM is a weighted sum of Zernike moments (see Equation 4-29), and normalized centralized Zernike moments are rotation, translation, scale and reflection invariant [1, 2, 150] which explains why in gestures that have similar intermediate postures, as in gestures "*E*" and "*H*", the recognition rate has dropped sharply regardless of the environment (Tables 5-3, 5-4 and 5-6). It is also clear that in the office environment, where the gesture suffers from segmentation error, the effect of intermediate postures similarly becomes problematic, although still well above the baseline (chance) recognition rate of 100/8=12.5% (Table 5-5).

The ZVM descriptor given in Equation 4-29 and the motion term in Equation 4-32 illustrate that ZVM parameters should be chosen in a way that they do not cancel the motion and spatial information.

A potential problem with the ZVM is that low displacement components of the centre of mass will result in clustering of feature vectors around the origin in feature space. For some classifiers, this may have an impact on the ability to discriminate between gestures that involve small horizontal or vertical components of displacement. If either or both of (β, λ) are not equal to zero and the gesture has no vertical or horizontal displacements, ZVM output would be zero. For example, gesture "*D*" has very small vertical displacements $(\overline{y}_i - \overline{y}_{i-1} \simeq 0)$ so when $\lambda = 0$ the term $(\overline{y}_i - \overline{y}_{i-1})^{\lambda} = 0^0 = 1$ and ZVM does not cancel the displacements happening horizontally. If $\lambda \neq 0$ then ZVM=0 for the whole gesture. This seems an important limitation of the ZVM descriptor. To avoid the elimination of the motion details in the gesture, ZVM parameters (p, q, β, λ) have been chosen to be (12,4,0,0), (12,4,0,1), and (12,4,1,0).

The ZVM's discriminative abilities have been shown to degrade when analysing sequences of gesture that share similarities with some other gestures. This has been highlighted by the comparative study on the full set of 8 gestures (see Table 5-4) vs the 5 most distinctive gestures (see Table 5-6). Gestures in both experiments are collected in OEH settings. The results show that the overall recognition rate improves from 62.3% (when 8) to 84.5% (when 5).

It is expected that reducing the number of gestures would increase the recognition rate, but the improvements are beyond the reduction of number only. This can be justified as follows. The base recognition rate of 5 gestures is 20% and for 8 is 12.5%, so the reduction with no other interference might be expected to improve the rate by only 7.5%. The correct classification rate in the last experiment of 84.5% can be effective especially if combined with other features as will be discussed in Section 5.2.2.

The previous experiments did not explore user-dependent analysis, which would reflect the real scenario of using this interface by the end user, as in commercial speech recognition systems. To explore this further, the ZVM and CvR in the user-dependent experiment is performed, where 12 instances taken from one participant per gesture is used. This yields improved results with a recognition rate of 88%. The achieved recognition rate shows the potential of using these methods for running a gesture recognition system. The only concern here is that these results are obtained when the whole body is not present in the scene.

By comparing the classification rates in Table 5-4 and 5-5, the results seem to drop significantly when the hand loses its shape details due to distance. In principal, it would

recognition system. The only concern here is that these results are obtained when the whole body is not present in the scene.

By comparing the classification rates in Table 5-4 and 5-5, the results seem to drop significantly when the hand loses its shape details due to distance. In principal, it would be possible to design a system that takes into consideration this problem and advises the user to optimise the best operating distance and when the hardware becomes more sophisticated the camera can zoom in/out to the optimal operating distance automatically.

In the presented experiments, ZVM processing time for each gesture takes up to 1.5 seconds which makes it unsuitable for real time use. However, it is possible to speed the processing time by sub-sampling of the sequence (*e.g* taking one in *n* frames). Hand postures under the normal speed of the gesture do not encounter big changes between neighbouring frames. It has been noticed that recognition rate did not decrease when frame rate reduced to 10 frames per second. Reducing the sampling frame rate further did decrease the recognition rate however.

The ZVM descriptor is the first spatio-temporal descriptor based on traditional Zernike moments. Its simplicity allows the descriptor to be developed further. It should be mentioned that in this research, the collision of two hands or the presence of obstructions that block part of the hand have not been explored. The ZVM descriptor has been examined under these conditions by Shutler and Nixon [1, 2].

This research is the first to introduce ZVM to hand gesture recognition. More generally, it is important to compare the performance of ZVM against the dominant approaches of event/action recognition. HMM as a temporal classification method that has been well explored in many researches contexts [11, 25, 26, 51, 119, 120, 125, 146, 150-157]. Therefore, Evaluating ZVM+CvR is opted to be against ZM+HMM. This evaluation would allow determining accurately how well ZVM is performing.

5.2.2 Using ZM+HMM

5.2.2.1 Overview

Since hand gestures are movements represented by temporal sequences of frames (postures), it is natural to model each hand gesture using an HMM. In this thesis, eight gestures are used. Therefore, there will be eight HMMs, one for each gesture (see Figure 3-1).

There are several choices of HMM model structure that can be used. The two main models are the left-to-right model and the ergodic model. In the left-to-right model, the probability of going back to the previous state is set to zero; therefore, the model will always start from a certain state and end up in an 'exiting' state. The model can jump one or more states in the left-to-right structure, but only in one direction. In the ergodic model, every state can be reached from any other state in a finite number of time steps.

In this research the left-to-right model is adopted, since this is the natural way to model a movement that has a fixed sequence of intermediate configurations given there is no jump of states. It also requires fewer parameters and therefore is easier to train [158]. For unlimited training data, the ergodic model would end up as the left-to-right model, if that is indeed the right model.

5.2.2.2 Zernike moments

Zernike moments are used as descriptors to produce the feature vectors which are consequently used by the HMM. The values of (p, q) parameters were set to be (10, 2), (10, 4), (12, 2) and (12, 4) see Equation 4-28. A plot of the feature vectors of gesture "A" is shown in Figure 5-4. Figure 5-5 depicts ZM (12, 2) features of gesture "A". Figures 5-4 and 5-5 show that the representation of the first 20 frames of gesture "A" did not have a big change.

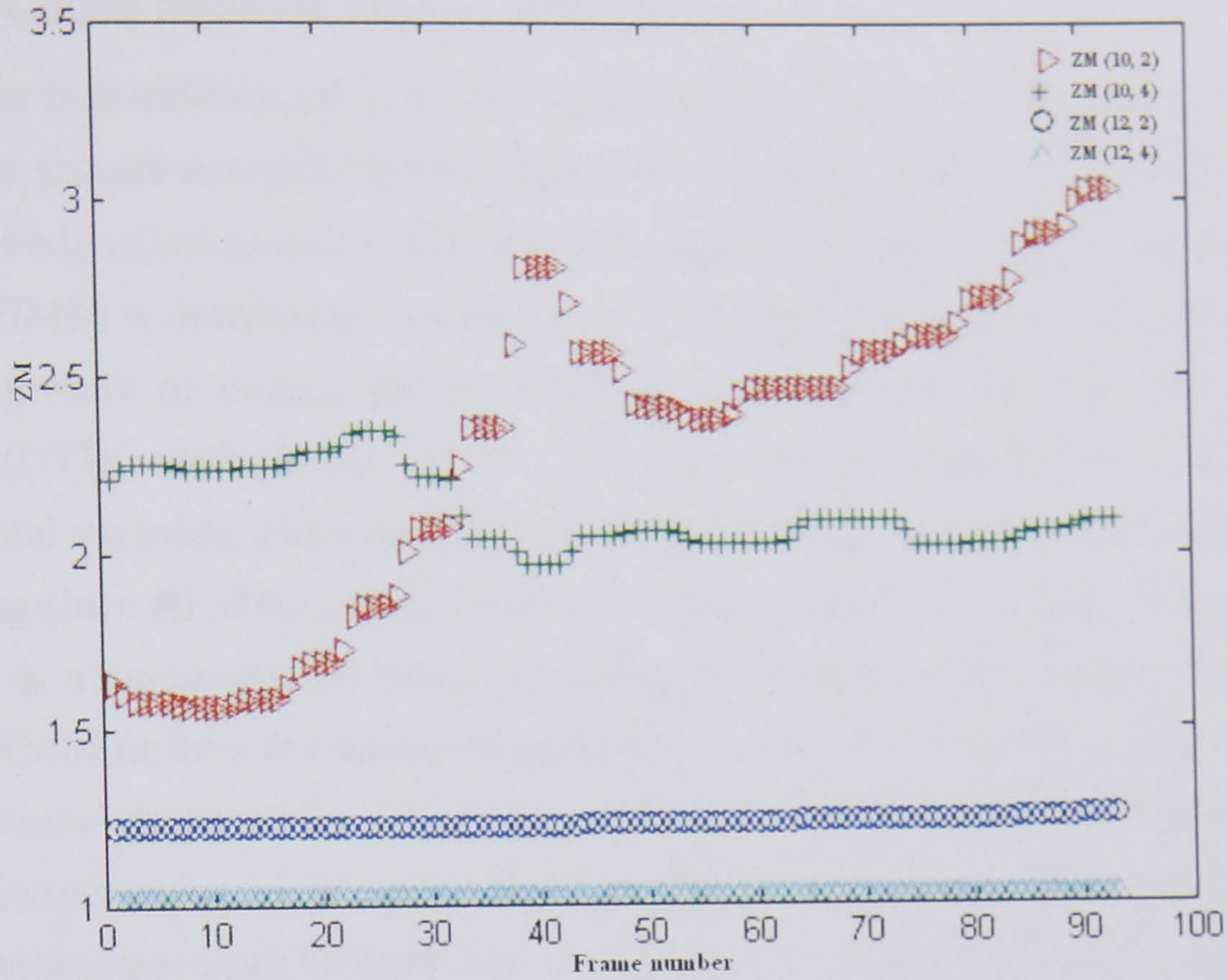


Figure 5-4: plot of gesture "A" representations using ZM descriptor. Higher degrees of moments carry more details as depicted in Figure 5-5.

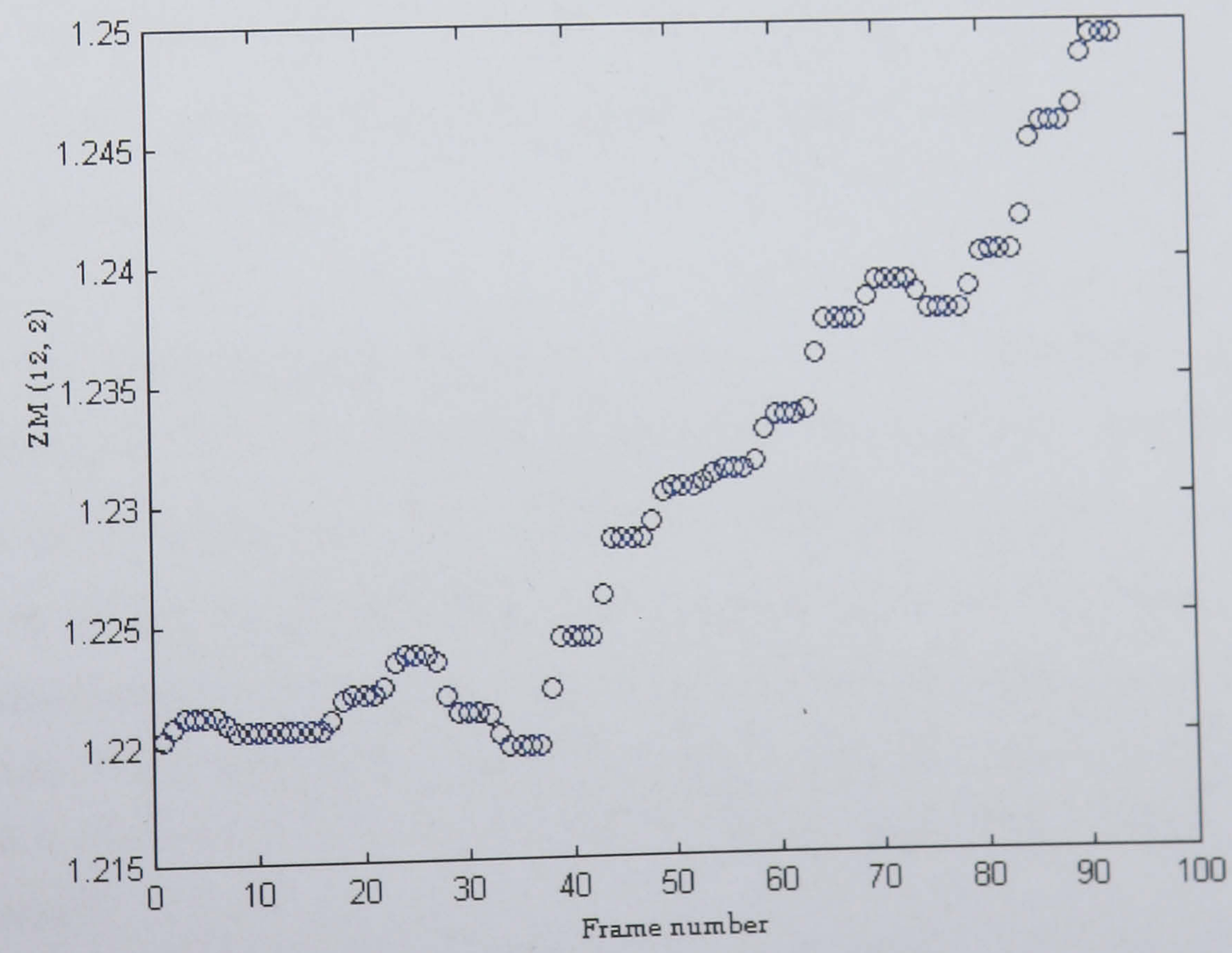


Figure 5-5: plot of ZM (12, 2) representations of gesture "A".

5.2.2.3 Hidden Markov Models (HMMs)

HMMs are well-established and have been used for many classification problems, as well as for gesture recognition problems. Speech recognition is a field that received an intensive body of research on HMMs – for example, the early work on speech recognition and HMM is dominated by Rabiner *et al.* [159]. HMMs have the ability to model dynamic gesture or events, but it is not the only method to do so. Dynamic Time Warping (DTW) methods [12, 25, 50, 122] allow one to align signals so as to account for temporal variation. However, the main disadvantage of DTW is that it is very time-consuming since all of the stored (training) sequences are used to find the best match.

HMM is a parametric probabilistic technique for extracting a hidden time warping pattern. HMM utilizes the transition probabilities between the hidden states and learns the conditional probabilities of the observations given the state of the model. The hand gesture is represented by the measurements of its motions which are dynamic in nature, since a hand gesture can be displayed at varying rates and with varying intensities even for the same individual.

An HMM is defined by the following set of parameters:

$$\lambda = (A,\ B,\ \pi) \qquad 5\text{-}1$$

$$a_{ij} = P(q_{t+1} = S_j \,|\, q_t = S_i) \quad \text{where} \quad (1 \le i, j \le N) \qquad 5\text{-}2$$

$$B = \{b_j(O_t)\} = P(O_t \,|\, q_t = S_j) \quad \text{where} \quad (1 \le j \le N) \qquad 5\text{-}3$$

$$\pi_j = P(q_1 = S_j) \qquad 5\text{-}4$$

A is the state transition probability matrix, B is the observation probability distribution, $\pi = \{\pi_j\}$ is the initial state distribution. The number of states of the HMM is given by N, $(S_1, S_2, \ldots S_N)$. The observation symbol (O_t) at time t can be either discrete or continuous, and can be a vector. q_t denotes the state of the system at time t. In the discrete case, B becomes a matrix of probabilities (Conditional Probability Table), and in the continuous case, B will be given by the parameters of the probability distribution function of the observations (normally Gaussian distribution or a mixture of Gaussians).

Given an HMM model, there are three issues which need to be addressed. The first is how to efficiently find the probability of the observation sequence for a given set of

model parameters. This is a classification problem which gives a measure of how well a certain model describes an observation sequence $P(O_1, O_2, ...O_T | \lambda)$. The second is infering the corresponding state sequence in some optimal fashion (optimal path) given a set of observations and the model parameters. This will become an important part of the algorithm to recognize the hand gesture from live input and will be described later in Chapter 6. The third is how to learn the parameters of the model λ given the set of observations, so as to maximize the probability of the observations given the chosen model. This problem relates to the learning phase of the HMMs which describes each hand gesture sequence.

With the regard to the first issue, the forward procedure which involves an induction approach is used to compute the probability $P(O | \lambda)$. The forward variable $\alpha_i(i) = P(O_i, O_2, ...O_t, q_t = S_i | \lambda)$ is computed at every time step in an inductive way and the probability is computed as $P(O | \lambda) = \sum_{i=1}^{N} \alpha_T(i) = sum$. The second issue essentially involves maximizing the probability $P(Q | O, \lambda)$. The state propagation across time is modelled as a lattice structure and at every time instant *t* the quantity:

$$\delta_t(i) = \max_{q_1, q_2 .... q_{t-1}} P(q_1, q_2 .. q_t = S_i,\ O_i, O_2, ...O_t | \lambda) \qquad 5\text{-}5$$

This is called the best path taken to reach state S_i at time *t*. Finally, the optimal state sequence is decided by the Viterbi algorithm. The third issue which deals with adjusting the HMM parameters $\lambda = (A, B, \pi)$ is solved using the Beum-Welch algorithm [160]. A broad discussion on the theory and use of HMMs is given by Rabiner [161].

5.2.2.4 Training and testing of HMMs

Matlab HMM code implemented by Murphy [132] is used. Prior to training and testing, each gesture sequence is linearly interpolated to a fixed number of frames equals to the length of the longest hand gesture for each HMM. This was necessary for compatibility with Murphy's package.

The experiments using HMMs involved two steps, training and testing. The system is trained using isolated hand gestures labelled for each class. The testing procedure is to select the model with maximum likelihood as the chosen gesture:

$$\lambda = \arg\max_{k}(P(O \mid \lambda_k)) \qquad 5\text{-}6$$

Figure 5-6 illustrates the structure of the HMMs used in these experiments. It consists of 8 parallel HMMs

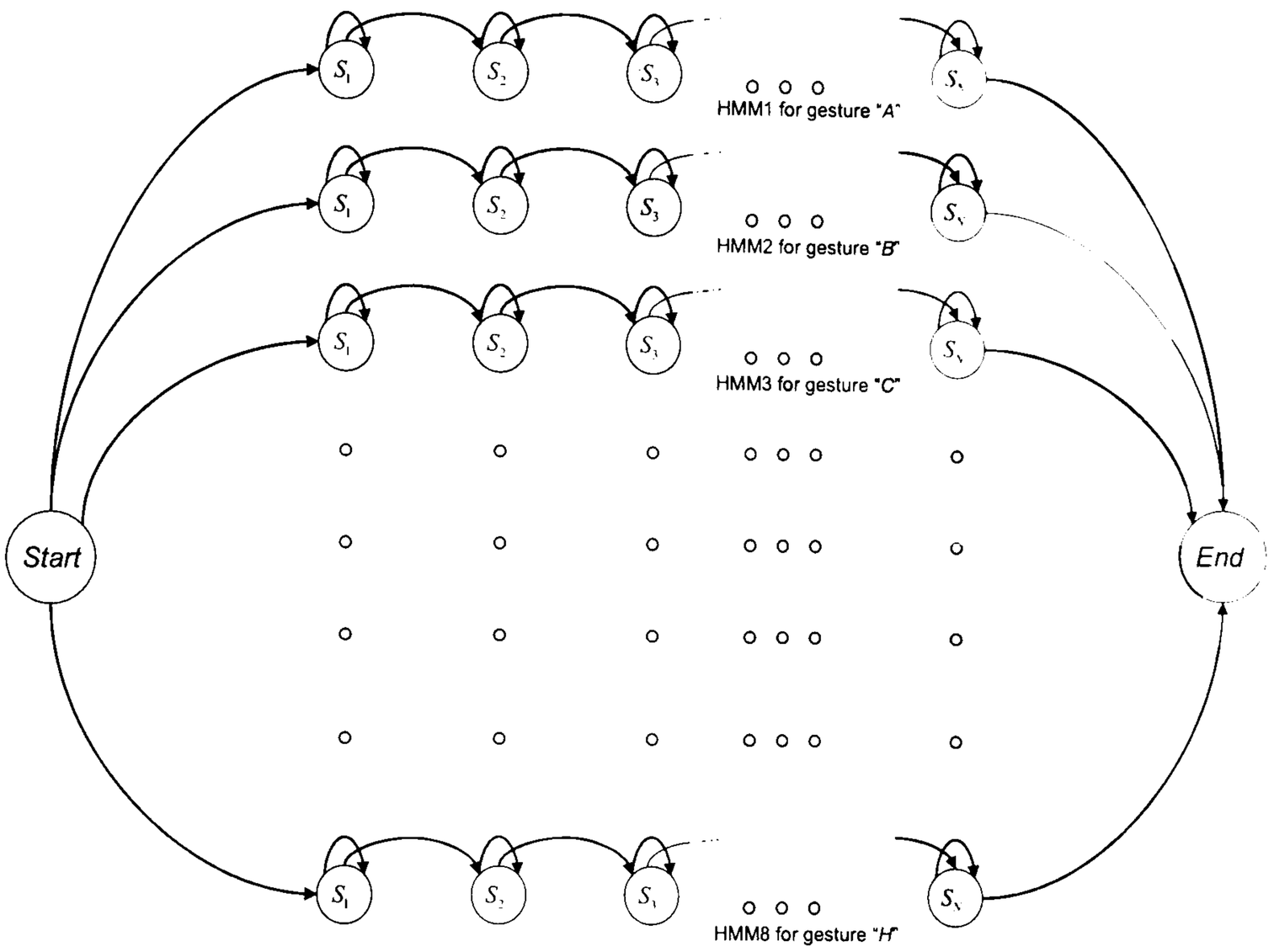


Figure 5-6: shows the structure of HMM models, each one of the 8 gestures is represented by one HMM model. The optimal number of states for each HMM is calculated.

5.2.2.5 Experiments

For the HMM models, different number of states were tried (N=2, 3, 4…8) and the best configuration was selected. Initially, the number of iterations that EM (Expectation Maximization) uses to estimate the HMM parameter values was fixed to 10. The selection of this number was from trials. In later stages and to avoid over-training HMM models, a threshold is set. This compares the likelihood from the previous iteration to the current one. If the difference is within an accepted limit then the

estimation of HMM parameter values is considered to be satisfactory. Figure 5-7 shows the learning curve of HMM for iteration (*itr* = 1-20).

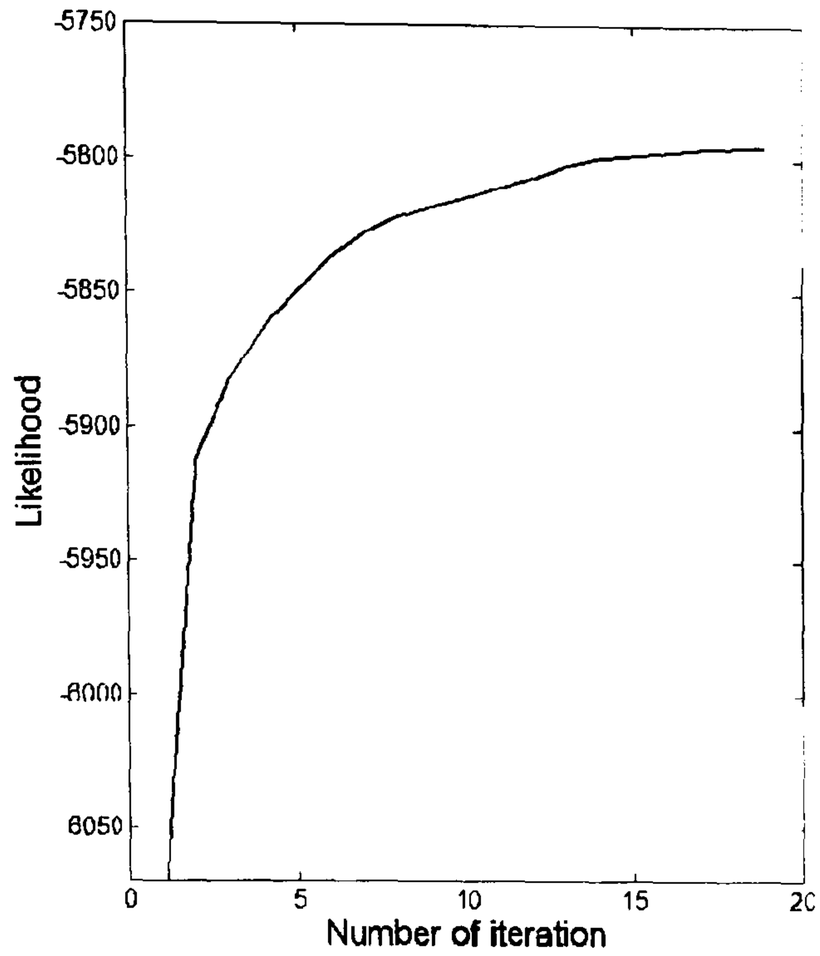


Figure 5-7: learning curve shows the increasing likelihood of gesture "*A*" against the number of iterations during training.

Evaluating 8 gestures:

The number of Gaussian components, associated with the observation density for each HMM state, is set to be one. The size of the feature vector is 4 the length of the gesture. Length of gesture varies between 30-120 frames. Using 30 examples for gesture "A" per setting as mentioned earlier (user-independent samples), the recognition rate for classification using 10 fold cross-validation is plotted. The optimal number of states is 4 for the HMM of gesture "A" (Figure 5-8). For compatability with Murphy's code, HMM models of all gestures have been given the same number of states 4.

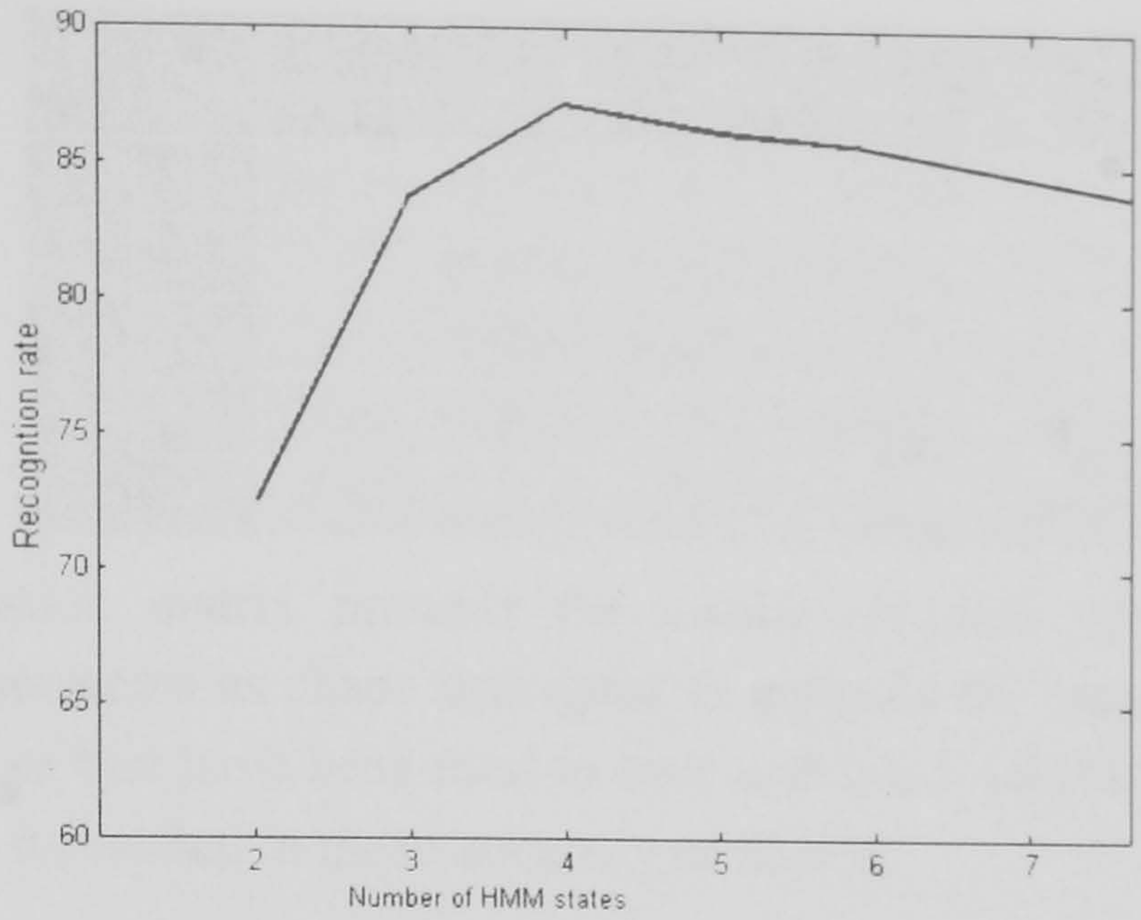


Figure 5-8: shows that 4 is the optimal number of states for HMM model represents gesture "A". This is obtained using samples of the UBL dataset.

Table 5-7 shows the recognition rate achieved using 8 gestures, 30 instances for each dataset.

Dataset	Recognition rate
UBL	87.1%
OEH	71.7%
OEW	45.8%

Table 5-7: the mean recognition rates of using ZM+HMM with 8 hand gestures.

Evaluating 5 distinctive gestures:

When comparing results achieved in the previous experiment by using ZM+HMM to those achieved by ZVM+CvR, an improvement can be noticed (see Table 5-7 and 5-9). ZVM is a weighted sum of ZMs, where the weighting takes into consideration the movement of CoM (see Equations 4-29 and 4-32). Motivated by this observation, the experiment was performed with the addition of CoM displacements to the feature vector. Adding the displacement of CoMs to the feature vector makes the comparison more reasonable.

An experiment is performed using five gestures (see Figure 5-3) on 80 instances per gesture on the OEH dataset with feature vector of ZMs+displacements of the CoMs.

		Predicted Gesture				
		"A"	"B"	"C"	"D"	"E"
Actual Gesture	"A"	**96.125%**	2.5%	0	1.125%	0
	"B"	0	**98.875%**	0	1.125%	0
	"C"	0	1.125%	**98.875%**	0	0
	"D"	2.5%	0	0	**97.5%**	0
	"E"	0	0	0	0	**100%**

Table 5-8: Confusion matrix presents the results obtained using the ZMs+CoM displacements as shape descriptor to generate the training and testing sequences that have been used to train and test 5 HMM models for the gestures. It resulted in mean accuracy of 98.3%.

When performing the experiment on five gestures without the CoM displacements, it resulted in mean accuracy rates of 94.5% [150]. However, when performing this experiment for user-dependent with CoM displacements, the mean accuracy rates was 99.1%.

5.2.2.6 Discussion

In this research, a hand gesture recognition prototype is proposed which allows flexibility in performing hand gestures. The flexibility means that the system is invariant to rotation, scaling and reflection of the hand which is what the Zernike moment descriptor provides. The selection of ZM parameters is assessed by experiments on posture recognition which confirmed that a higher degree of moments produces a higher accuracy rate. Therefore, moments of 10 and 12 degrees are used. ZM descriptor has been studied in [1, 2, 144, 154, 162]. Plotting the feature vector of ZM for gesture "A" shows the discriminative ability of the descriptor.

Using the obtained feature representation of each gesture, a parallel left-right HMM structure is used as a classifier. HMM yields much improved results in all the experiments for both 8 and 5 classes with and without CoM displacements. When using the five distinctive gestures, HMM produces a higher recognition rate. Further discussion is in Section 5.3.

5.3 Summary and discussion

Hand gesture recognition with ZVM has been introduced and investigated. ZVM has been previously used successfully for human gait analysis. These results indicated a potential use of ZVM due to their simplicity.

8 gestures Datasets	Recognition rate	
	ZM+HMM	ZVM+CvR
UBL	87.1%	84.9%
OEH	71.7%	62.3%
OEW	45.8%	28.9%

Table 5-9: the mean recognition rates of using ZM+HMM and ZVM+CvR with 8 hand gestures. 30 instances for each gesture are used.

From the confusion matrices and the relative accuracies obtained in the two main experiments (ZVM+CvR and ZM+HMM), it is clear that the ZM+HMM combination used in the second experiment (see Section 5.2.2) has given substantially better results than the ZVM+CvR combination used in the first (see Section 5.2.1). Additionally, it seems intuitively plausible that the displacement of the centre of mass between frames carries discriminative information on the set of gestures. To explore this further, an experiment was carried out in which this displacement is added to the feature vector of ZM used in the HMM. This increased the mean accuracy obtained from 94.5% to 98.3% (see Table 5-10), although this only represents a small number of additional examples being correctly classified.

Experiment	Recognition rate
ZVM+CvR	84.5
ZVM+CvR + user-dependent	88%
ZM+HMM	94.5%
ZM+HMM with CoM	98.3%
ZM+HMM + CoM+ user-dependent	99.1%

Table 5-10: the mean recognition rates using 5 gestures. Each gesture has 80 instances. In user-dependent experiment only 12 instances are used to train the system. Dataset used is OEH.

Unfortunately, it is infeasible to compare the processing time for ZM+HMM with ZVM+CvR as different packages are used. CvR is from Weka (Java) while HMM is in Matlab. However, it is possible to report that both classification methods require a significant processing time for training and a fraction of a second for testing.

During the design of both experiments (ZVM+CvR and ZM+HMM), a mechanism for updating the trained model parameters with each testing gesture was not considered. The focus of the previous experiments was to test and evaluate the newly introduced ZVM descriptor against the well-established ZM+HMM only.

Ideally, it would be preferable to compare the obtained results to those of other researchers, but the lack of testing standards and datasets made this infeasible. Furthermore, the task of distinguishing exactly these postures/gestures is very specific and comparison with results from smaller, larger, or different sets of data and their recognition rates would not be feasible.

The evaluation of the presented experimental module thus has to rely on the recognition rates as reported in Section 5.1 and 5.2. The ZVM descriptor has not been evaluated before against any other techniques. The previous experiments provide this evaluation which concluded that ZM+HMM performs better than ZVM coupled with CvR. As mentioned in the previous section, CvR was chosen after exploring several methods.

The reported accuracy rates are not good enough for high-end user interfaces. However, when analysing user-dependant data, the recognition rates have improved from 84.5 to 88% (see Section 5.2.1). Mis-classifications can be due to participants performing gestures wrongly or the system recognizing a wrong gesture. The intended hand gesture interface can be designed in a way that users get feedback if their gesture is not recognized at all and participants can make small changes to the way in which they execute a gesture in order to improve recognition. It is likely therefore that over time the user would adapt to the system.

CHAPTER 6

GESTURE DETECTION AND VIEWPOINTS

The main limitation of the approaches adopted in Chapter 5 is that they work on pre-segmented sequences of hand gestures. This pre-segmentation is widely done manually. Therefore, there is a need to explore techniques for segmenting and detecting hand gestures automatically.

HMMs were first used for representing phonemes in conjunction with the use of grammar in many systems for continuous speech recognition [161, 163]. Dynamic time warping for continuous action detection has also been proposed before [122]. Key-frame spotting of human actions has been utilized to locate the start and the end of pre-defined actions automatically [112, 113]. Applying any of the previous systems to the hand gesture detection problem is not straightforward since there is no clear standard for performing hand gestures (with the exception of sign language and similar systems).

Additionally, the previous chapters did not discuss recognition across multiple viewpoints. All pre-segmented gestures were captured from a single fixed viewpoint. In reality, this would limit the flexibility of the system that this research is aiming for. There have been several studies that tried to address viewpoint variation including the use of multiple cameras with marked gloves [164, 165], on a single camera assisted with a mirror to provide another view [166]. These approaches [164-166] require complex hardware which this research is trying to avoid. Appearance-based methods that utilise a large number of 3D hand models in the database for posture recognition have been also explored [85]. This research aims to apply the 3D models for dynamic hand gesture recognition not only postures.

This chapter addresses two developments of the HMM based hand recognition system presented in Chapter 5. Firstly, automatic segmentation of hand gesture from a continuous video stream is explored. Secondly, a novel use of 3D models for viewpoint invariant hand gesture recognition is introduced. The 3D models have been captured from a real hand and then manipulated using 3D software to create a view-independent dataset of virtual hand gestures. The dataset obtained is used to train an HMM based system like that described in Chapter 5.

6.1 Hand gesture detection

In this section, two methods for locating the start and the end of hand gestures automatically are presented. The trained HMMs of 5 gestures (the most distinguished gestures) from Chapter 5 are utilized in experiments with two techniques.

In the first, a sliding window: a buffer is created which stores a certain number of frames. The stored sequence is then scanned between the minimum and the maximum possible length. Each scan is evaluated against the known HMMs and the highest likelihood is selected as a correspondent to the correct gesture.

In the second, a single HMM with Viterbi: the previously trained HMMs are used to create one HMM by linking the exit states of left-to-right HMMs to the start with equal probability. Then, the Viterbi algorithm is deployed to find the best path cycling through the HMM over multiple gestures.

The experimental results presented for both methods are performed offline. Two sequences of hand gestures each containing 3500 frames from UBL and OEH datasets (see Section 3.2) are used. In each sequence, there are 25 performed gestures of five classes. A ground truth for these sequences is labelled manually. The minimum length

of a gesture in these datasets is 65 frames and the maximum is 125, with mean of 93 frames.

6.1.1 Detecting using sliding window

6.1.1.1 Direct evaluation

In this approach, a gesture is detected by sliding a window over the continuous video stream, where the window is of varying lengths. Although the experiments are performed offline, it is useful to think of this as an online process using a buffer to store previous frames leading to a gesture that ends at the current time step.

A buffer is created which has a maximum length of 125 frames ($T_{\max} = 125$). This number has been selected according to the knowledge that was gathered when labelling the ground truth. Similarly, the minimum length of the gesture is 65 frames ($T_{\min} = 65$). This buffer is required to store the last stack of performed frames. The used buffer is always updated with new frames and the oldest ones are lost. This works as a delay circuit. After that, these stored frames are segmented for evaluation.

The segmentation of these frames is performed by selecting a point at time t where the evaluation is due to begin. The end of each segment is at the frame t and starts at $(t - T_j)$ where T_j is the length of a sequence $T_{\min} \leq T_j \leq T_{\max}$. Then, a number of sequences is generated from this buffer. This number depends on the sampling rate S.

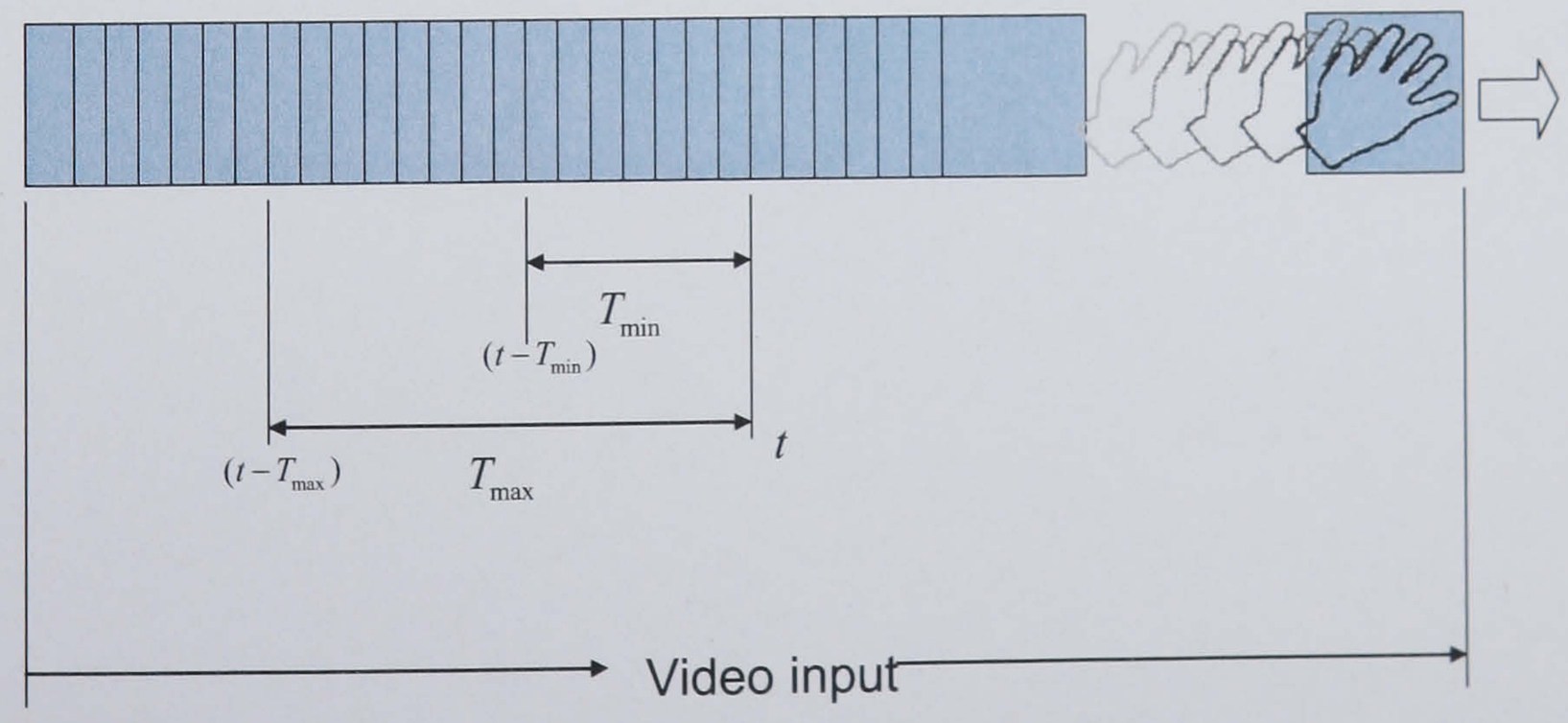


Figure 6-1: hand gesture sequence structure with its sliding window parameters

The range of length between T_{max} and T_{min} is sampled uniformly into S samples. The resulting sequences are evaluated against the pre-trained HMMs (λ_i) where i is the number of classes. For example, for $S = 10$, $T_{max} = 125$ and $T_{min} = 65$ then the sampled sequences would be of length $(65, 71, 77 ... 125)$.

After evaluating each sequence against the pre-trained HMMs, the obtained likelihoods $\log(P(O_t \mid \lambda_i))$ are recorded and tagged with its corresponding class. Then, the next starting point is processed. The next starting point can be the next frame or after a certain step and then the same steps as explained before are performed. In this experiment, the next frame is the starting point. The maximum likelihood at each end point is the detected hand gesture.

Figure 6-2 illustrates the output of the detected hand gestures. The coloured rows denote the sequence ground truth and the vertical lines mark the number of recognized scans (sequences) at each starting point. The sampling resolution is 10, and the starting point of the sliding window has moved frame by frame.

This experiment does not show exactly where the end of the gesture is but rather the range between the possible minimum and maximum length of a scan. Section 6.1.2 addresses this issue and introduces a method for locating the end of a detected gesture.

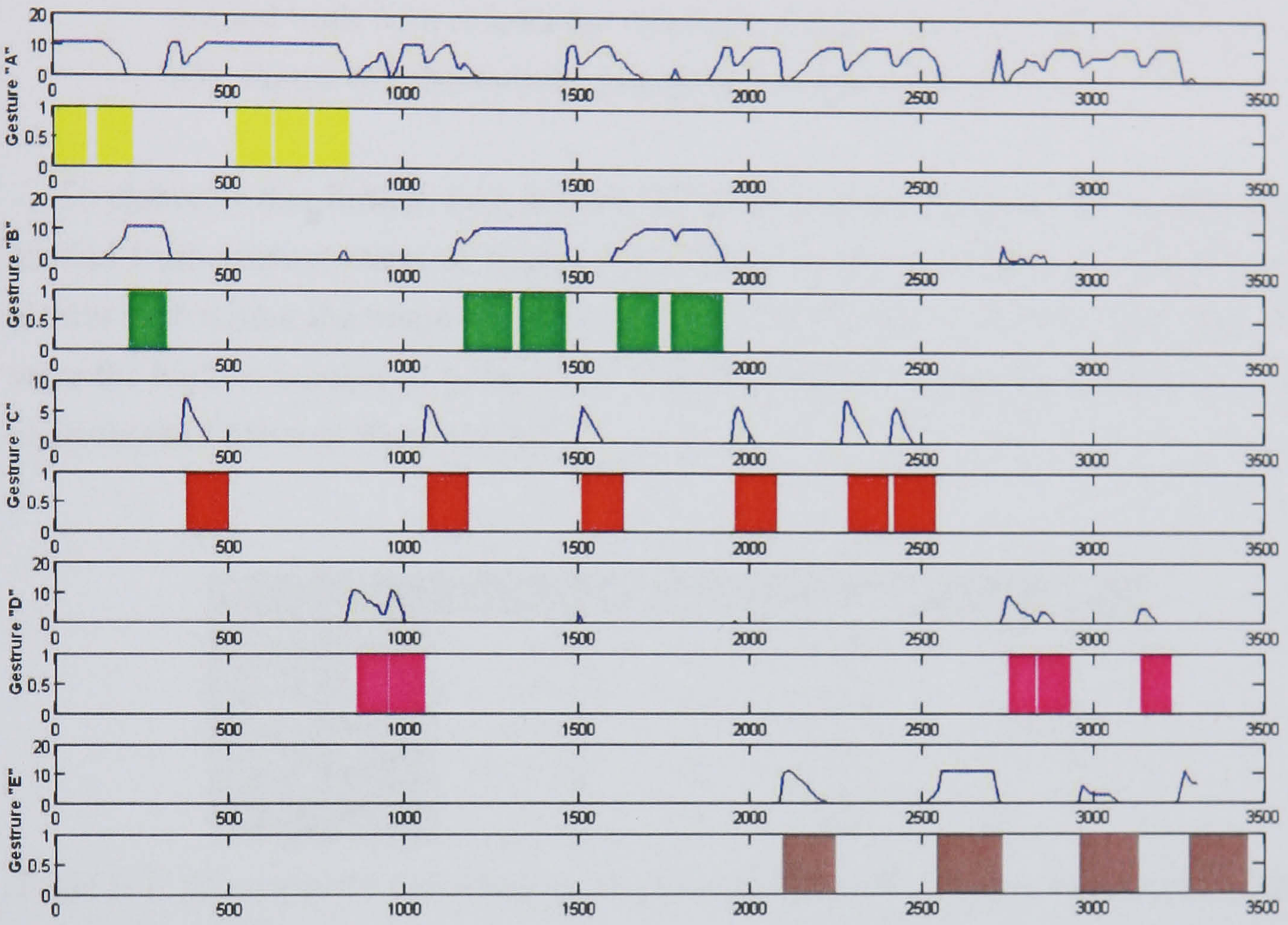


Figure 6-2: hand gesture detection using the sliding window technique. The coloured bars mark the ground truth and the lines mark the number of recognized sequences at each start point of scanning. The sequence is from UBL data-set.

When calculating the detection rate (accuracy rate), a certain scan (sequence) is considered to be recognized successfully if the following conditions are met:

- A Scan should start in a frame belonging to the range $[t-0.25*T \text{ to } t+0.25*T]$.
- The length of a scan should overlap with the corresponding gesture in the ground truth with at least the minimum length a gesture might take (here it is 65). Figure 6-2 shows only scans that met this condition.

To elaborate this further, take gesture "C" as an example; gesture "C" corresponding ground truth gesture starts at frame 1100; therefore the prediction of possible match should start within the range of [1074 to 1126]. Only scans at t=[1100, 1101 and 1102] were the highest number in comparison to other gestures , especially gesture "A". This is discussed further in Section 6.1.3.

	Correct classified	Incorrect classified	width
Gesture "A"	180	0	74
Gesture "B"	124	86	85
Gesture "C"	98	214	105
Gesture "D"	64	96	65
Gesture "E"	56	192	125
	522	588	

Table 6-1: the correctly classified and mis-classified scans which correspond to Figure 6-2.

The resulted accuracy rates are $\frac{522}{522+588} = 47\%$.

Figure 6-2 shows that gesture "A" is acting like a default and to reduce this effect, the likelihood of the other appearing gestures is multiplied by 2. This increased the accuracy rate from 47% to 60%. The reason why gesture "A" is acting like a dummy model is: it contains the widest variation of frames between all other gestures. This indicates the necessity to have a dedicated dummy state (garbage) which has been introduced next.

6.1.1.2 HMM with garbage model

It is unlikely for a person to go from one gesture to another without passing through a neutral posture with no or useless hand movements. Handling such cases is done by slightly modifying the structure of the HMM models. A new ergodic HMM structure has been added to represent the garbage gestures. So what garbage gesture/structure means?

Unlike the pre-defined hand gestures, there are no constraints on the remaining non-gesture patterns. A non-gesture is any motion other than the pre-defined gestures. There are no specific constraints on the follow of the motion and to reflect that in the design of the HMM model, a type of ergodic or a fully-connected model is chosen. In the ergodic HMM, each state of the model can be reached from all other states. The use of the garbage state has been explored in several studies including [119, 120, 167, 168]. Literature suggested other methods for addressing the non-gestural actions as the use of both hands; where one hand performs the gesture and the other one works as a clutch [128, 169], others suggested performing certain postures after each hand gesture to indicate the end of the gesture [48].

Using the same sequences explored in the previous experiment, Figure 6-3 shows the number of recognized sequences for each gesture. The garbage model has been trained on a selection of random sequences that contains 800 frames. The accuracy rate has risen from 47% to 66%.

So far, the number of sequences that corresponds to the highest likelihood at each starting is calculated and the length of the sequence that achieves the maximum likelihood for each gesture is not addressed. Finding the length of the sequence will locate accurately the start and the end of the gesture. The next section presents methods for locating the start and the end.

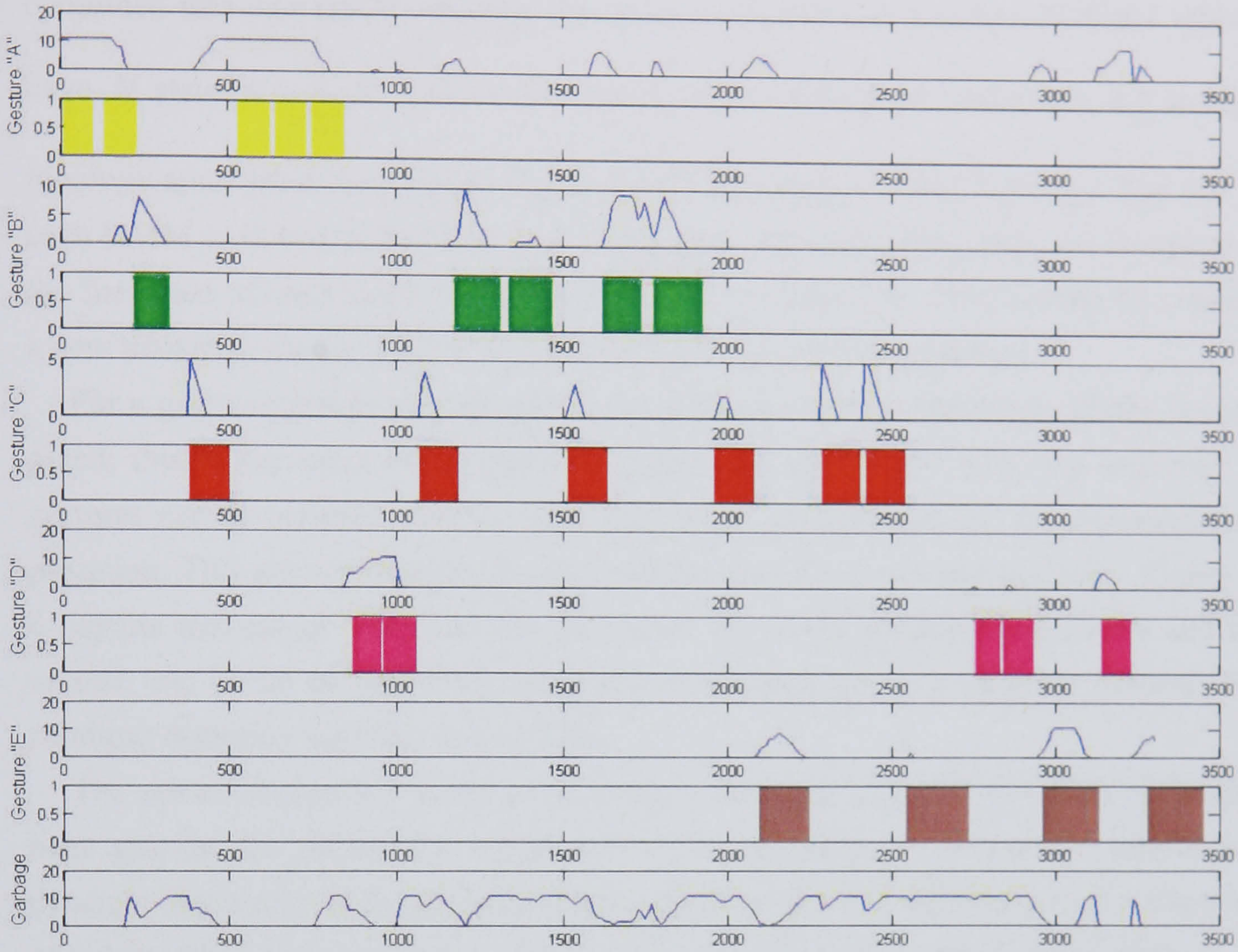


Figure 6-3: hand gesture detection using the sliding window technique. The coloured bars mark the ground truth and the lines mark the number of recognized sequences at each start point of scanning. Last bar represents the sequences recognized as garbage. The sequence is from UBL dataset.

6.1.2 Detecting using a single HMM

In the second technique, the trained five HMMs models and the garbage model are combined into one HMM structure as shown in Figure 6-4. Let the combined HMMs have N states $N = \sum_{i=1}^{6} N_i$ (23 in this case). Each individual HMM has left-to-right topology apart from the garbage model which has ergodic HMM topology. The exit of each HMM is linked to the start of the structure. The start of the structure is linked to the first state of each model with equal assigned probabilities. This is done by creating a new transition matrix of (N x N) where N = *no. states* of all gestures.

For a given sequence of gestures, the Viterbi path through the single HMM is computed; this is the most likely path. By examining the Viterbi path, the sequence of gestures can be inferred, and the start and end of each gesture can be estimated with precision. This allows locating the start and the end of the gestures precisely. Figure 6-6 depicts the output. The first bar of Figure 6-6 is the recognized gestures and the second line is the ground truth. Each colour corresponds to a different gesture. The resulting detection accuracy rate is 52%.

The achieved accuracy is not good enough. This is because of the effect of the garbage gesture. To reduce this effect the Viterbi algorithm is examined. Viterbi is a dynamic alignment function which checks each sequence of observation to a sequence of hidden states and according to the likelihood it defines the corresponding states.

In an attempt to increase the accuracy, the likelihood at each state is examined and compared to the second highest likelihood. If the highest likelihood was belonged to the garbage gesture and the second highest is within a small range (10%) then the second likelihood is promoted to be higher than the garbage. Figure 6-7 illustrates the recognized sequences after promoting the second highest likelihood. This produced a significant improvement with 76% accuracy rate.

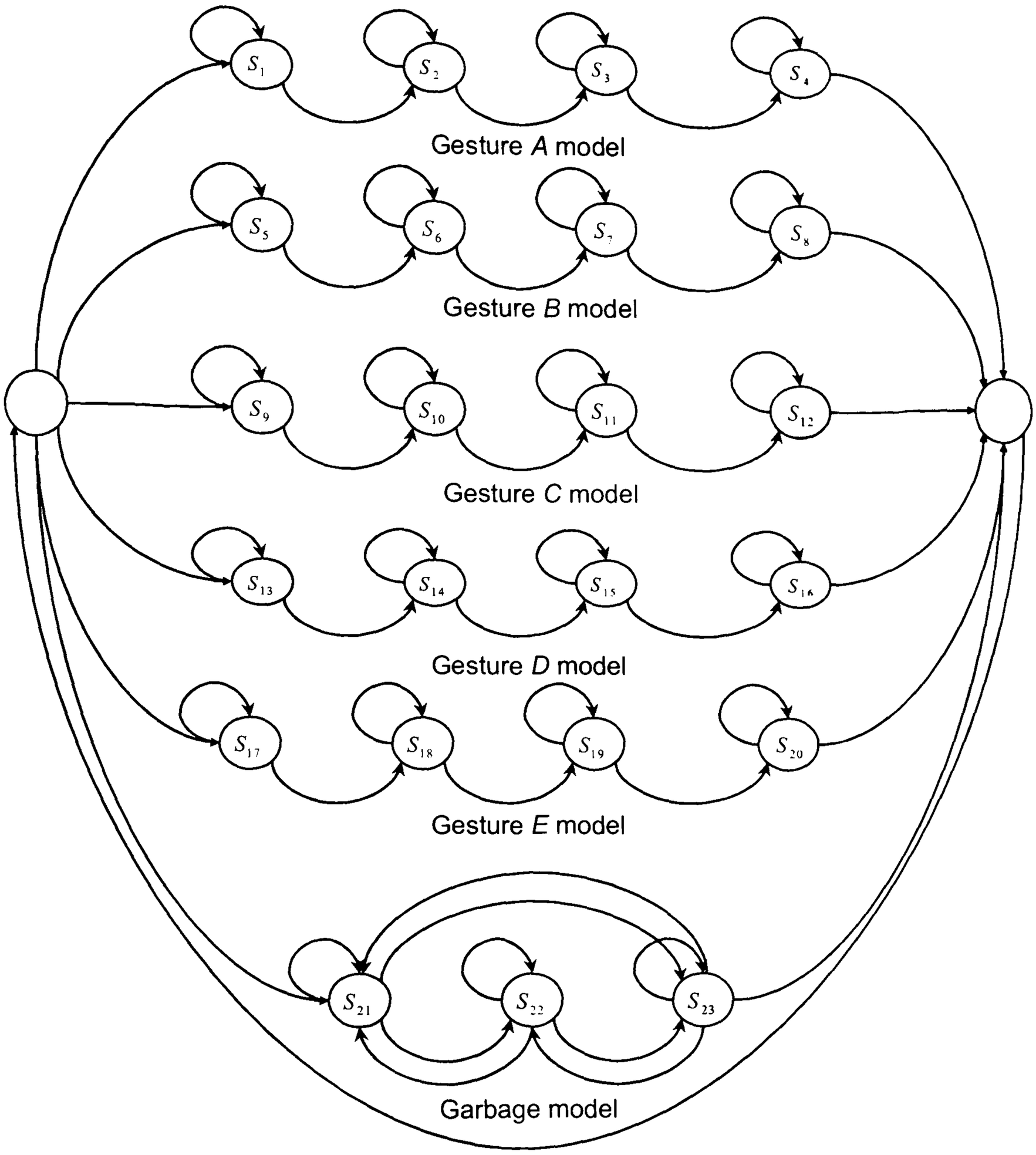


Figure 6-4: connected HMMs that consist of 5 pre-trained HMMs on isolated hand gestures and a garbage model. Garbage model is ergodic HMM while others are left-to-right. The link from the exit to the start of the HMMs gives equal probability to all other HMMs.

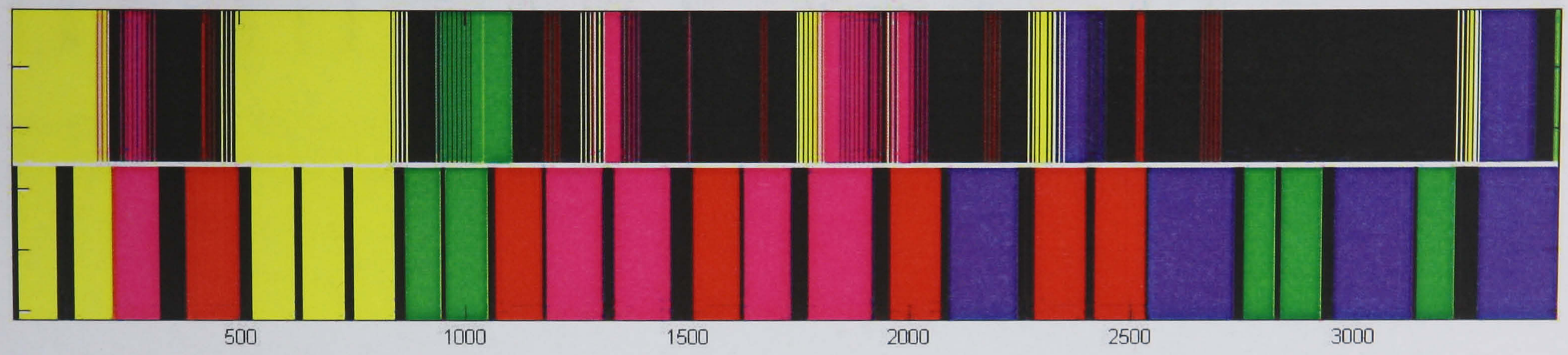


Figure 6-5: the output of the sliding window experiment. The first row shows the recognized sequences and the second row is the ground truth. Each colour corresponds to a gesture yellow, pink, red, green, blue and black are respectively gestures “A”, “B”, “C”, “D”, “E” and garbage.

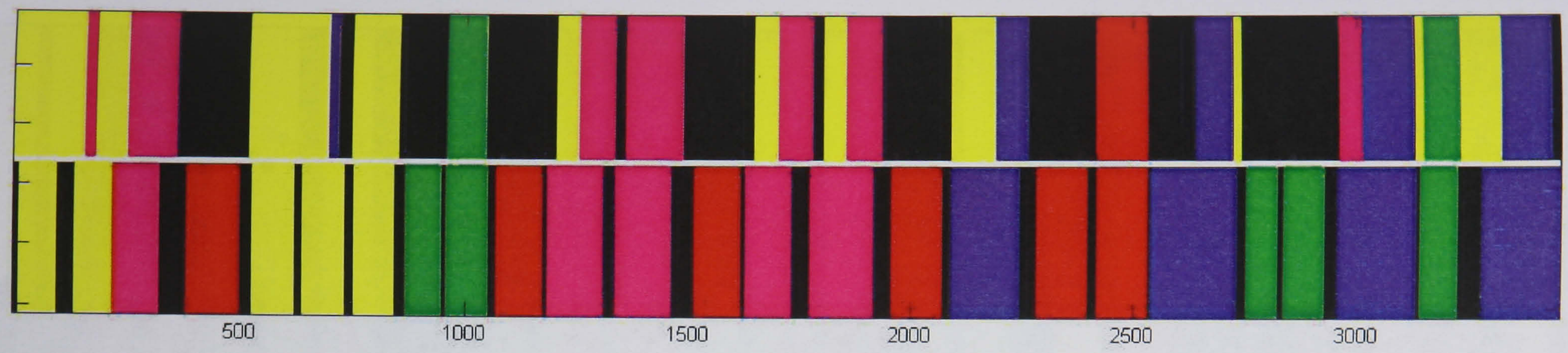


Figure 6-6: the output of single HMM and Viterbi on the OEH dataset. The first row shows the recognized sequences and the second row is the ground truth. Each colour corresponds to a gesture yellow, pink, red, green, blue and black are respectively gestures "A", "B", "C", "D", "E" and garbage.

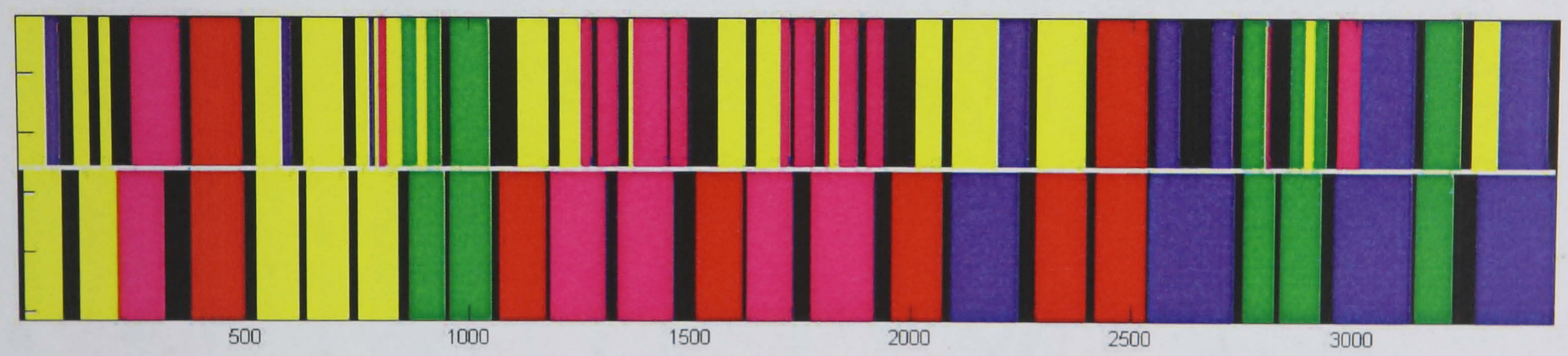


Figure 6-7: the output of single HMM and Viterbi on the OEH dataset. The first row shows the recognized sequences and the second row is the ground truth, each colour corresponds to a gesture yellow, pink, red, green, blue and black are respectively gestures "A", "B", "C", "D", "E" and garbage. The second highest likelihood is promoted when the garbage is the highest.

Figure 6-8: this figure shows a direct comparison between the different approaches described in this section. The first row shows the recognized sequences out of experiment 6.1.2 and the second row is the output after promoting the second highest likelihood. The third row is the output using sliding window in experiment 6.1.1.2. Last bar is the ground truth each colour corresponds to a gesture yellow, pink, red, green, blue and black are respectively gestures “A”, “B”, “C”, “D”, “E” and garbage.

6.1.3 Discussion

In this section, several methods for hand gesture detection from continuous sequences, that contain hand gestures, have been presented. The intention was to perform and explore an extensive evaluation of different methods using HMMs.

In the first experiment, hand gestures are detected using direct evaluation of HMMs that were trained using the pre-segmented gestures in Chapter 5. A method to capture sequences of different length and evaluate them was introduced. This method is time consuming and not suitable for real-time application when using fine sampling steps. As depicted in Figure 6-2, one of the gestures "A" which contains the widest variety of frames is acting like a default gesture, in that all non-gesture movements are recognized as gesture "A". Therefore it was necessary to negate this effect by adding a new HMM structure for garbage motions to the set of HMMs.

Adding the new garbage model that was trained on a collection of random non-gesture sequences and frames increased the overall performance by 10%.

In the second experiment, a single HMM model with the Viterbi algorithm were used. The Viterbi algorithm aligns an observation sequence to the relevant states by evaluating the likelihood at each state. This to some extent is similar to the sliding window approach where each resulted sequence is evaluated against all HMM models. The difference is that Viterbi algorithm takes the whole sequence and outputs the relevant states in one operation, while the sliding window approach segments sequences and evaluates each of them. This result in iterating the evaluation of the same sequence several times which makes the sliding window approach slower to process.

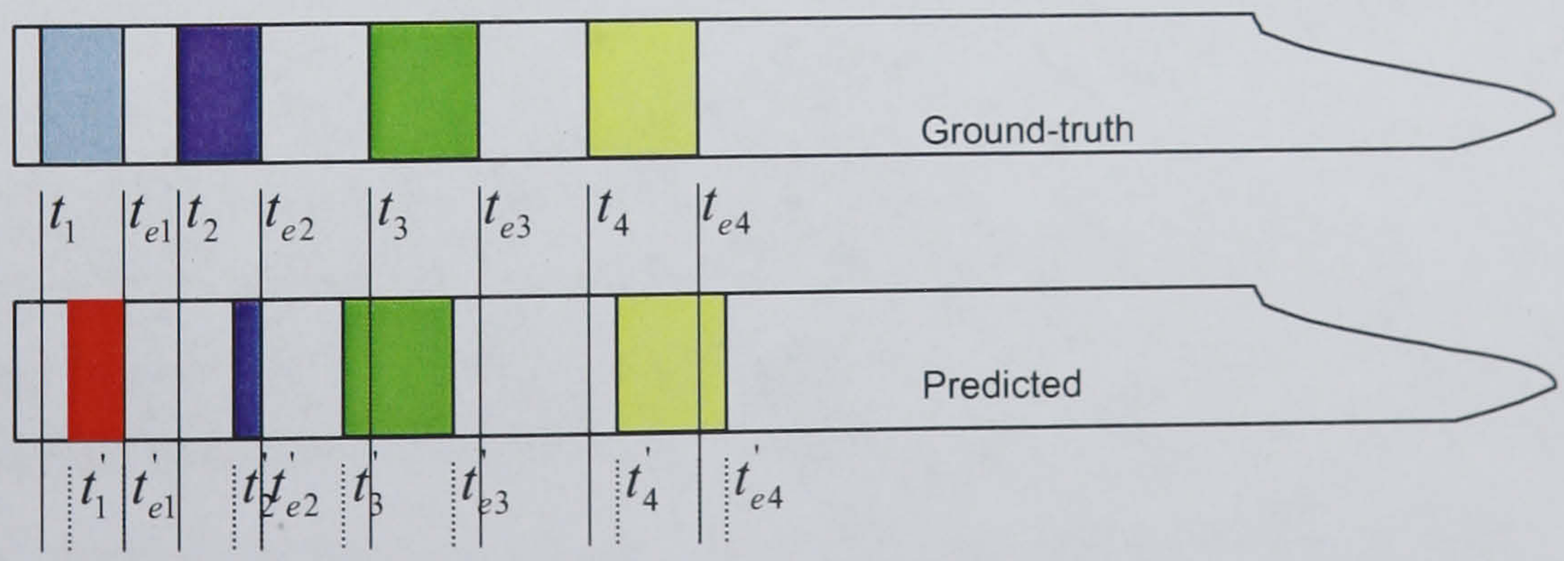


Figure 6-9: the overlap of predicted hand gesture to the ground truth. The match takes place when the overlap is between the predicted and ground truth is 75% and both of same class.

Each predicted gesture is considered to be correctly detected if there is an overlap with a ground truth of at least 75%. The degree of overlap is defined as follows:

$$0.75 \leq \left(\frac{\left| T_i \cap T_i^{'} \right|}{\left| T_i \right|} \right)_{i=1,2,3...n} \quad \text{where } T_i = [t_i, t_{ei}] \text{ and } T_i^{'} = [t_i^{'}, t_{ei}^{'}] \qquad 6\text{-}1$$

Figure 6-9 shows examples of correctly detected gestures. The gesture starting at t_1 is incorrect because it has a different class (colour) to ground truth it overlaps. The gesture at t_2 is also incorrect because it does not meet the 75% condition even though it overlaps with ground truth's gesture of the same class. Gestures at t_3 and t_4 are correct.

The single HMM approach produced an improved results (accuracy and processing time) when compared to the sliding window approach. The accuracy rates of 52% and 76% were obtained for UBL sequences, and 43% and 68% for the OEH sequences (see Table 6-2).

The achieved accuracy rates are not good enough to perform automated hand gesture detection. However, they can probably be improved by redesigning the system to adapt and learn from previous mistakes. It is widely accepted that designing the system to learn and modify the parameters of the learnt HMMs would improve over time the detection performance. This will benefit from the fact that this system is intended to be used interactively therefore users can get feedback about their performance instantly.

Experiment	UBL sequence	OEH sequence
Sliding window no garbage, no weighting	47%	41%
Sliding window no garbage, with weighting	60%	57%
Sliding window with garbage, with weighting	66%	53%
Viterbi path	52%	43%
Viterbi path with second likelihood.	76%	68%

Table 6-2: accuracy rates achieved in experiments on hand gesture detection.

The current design of the system did not consider optimising different numbers of states for each gesture which could improve the performance. However, the ergodic HMM has been explored during the development of experiments and it has been

observed that the transition matrix in the ergodic HMM for each of the gestures is almost a left-to-right one.

In this section on hand gesture detection, data from UBL and OEH datasets are only used with five gestures. It is expected that the hand gesture detection rate would drop when using the OEW dataset, but if the right hardware is used this problem will be rectified. This can be achieved by zooming to the target object to increase the resolution of frames.

As shown in Figure 6-4, and described above, the link from the exit of HMMs to the start has been given equal probabilities. In other words, the design did not consider whether a certain gesture is more likely to appear after another. This can be extended in further research beyond the aim of this thesis.

6.2 Viewpoint invariance using 3D models

During the performance of hand gestures, viewpoint plays a major role in the success of any recognition system. This proves to be a difficult issue to be solved. It is widely assumed when a gesture recognition system is studied; a certain limitation is imposed on the hand gesture to avoid this problem. Recently, several researchers have attempted to solve these viewpoint variation issues. Some of them suggested hardware solutions such as data acquisition systems (stereo camera and multiple cameras) as in the research of Holte and Moeslund [171]. Other researchers suggested using the variation of key-frames (postures) as in Laptev [112, 113]. Posture recognition systems based on 3D models has been explored by several researches including [23, 107, 170], where the 3D models of the hand are not deformable. Generally, the 3D models were produced using artificial moulds of the hand or geometrical objects.

This research explores a new method that utilizes 3D models of real hands, and then by using 3D animation software a virtual performance of the hand gesture is created. After that the virtual gesture is projected from several arbitrary points that cover the whole sphere. The projected sequences are used to train our HMM based system. This is widely known as analysis-by-synthesis [79, 172]. This technique is developed with the intention to be used with a user interface having one camera.

6.2.1 Capturing models

Six 3D hand models were captured using the Polhemus FastSCAN [173] system. These 3D models represent the distinctive shapes of the hand gesture. These models are the base that was used to derive all intermediate models. The handheld system uses a laser scanner which consists of a non-contact range finder based on projection and simultaneous detection of laser light, coupled with a means of tracking (magnetic tracker) the position and orientation of the range finder as it is scanned over the object's surface. The resulting 3D model is in multi-layers 3D point cloud format that needs processing. This processing stage is essential for reducing the size of the model. Each layer corresponds to a scan of the hand (see Figure 6-10).

To reduce the number of points in the resulting model and to align these multi-layers to one layer (see Figure 6-11), the RBF (Radial Basis Functions) interpolation function is used. RBF function is embedded in the FastScan software [173]. The number of facets in each of the models is 5000 (see Figure 6-12)

Figure 6-10: 3D model captured using laser scan system; this model contains multiple layers of point cloud pixels.

Figure 6-11: 3D model captured using laser scan system; this model contains one layer after aligning the multiple layers in Figure 6-10.

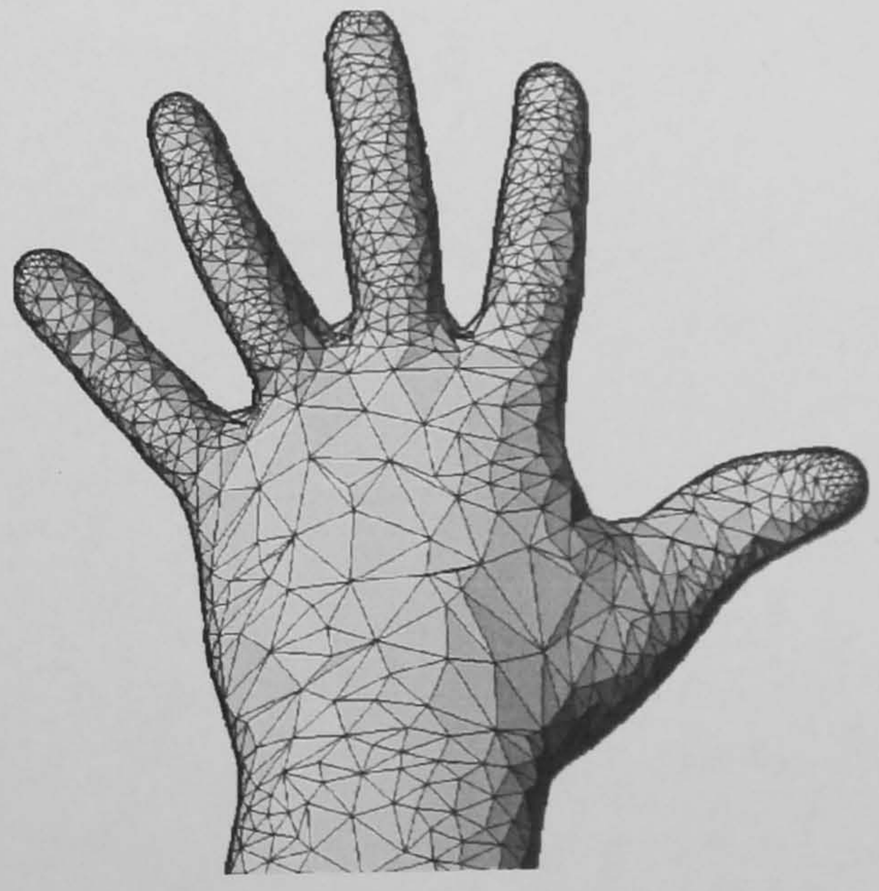


Figure 6-12: 3D model with a reduced number of points, this model contains 5000 facets.

6.2.2 Experiment and results

In this experiment, the Lewis *et al.* pose deformation technique [174] is used to create the intermediate stages. This deformation is carried out using Blender software [130], after creating 10 intermediate poses for each one of the eight hand gestures (see Figure 5-3). Figure 3-8 depicts examples of these poses.

Each 3D hand model has been linked to 17 bones. These bones are a skeleton that guided us to generate the intermediate models. It should be emphasised here that this experiment did not create the actual skeleton but rather used the essence of it to deform the 3D models. This is done by deforming the 3D model's finger parts at points that corresponds to joints. Each finger is bent to match the desired intermediate 3D model. For example, the index finger has two joints and a joint links it with the palm. The intermediate stages between having the finger fully closed and fully opened are divided into 10 stages. At each stage finger segments are bent according to its corresponding skeleton (the operator estimates that manually). Each finger is processed separately.

To simulate the performance of the hand in the virtual world, virtual gestures are created as shown in Figure 3-7 and rendered from 30 different viewpoints. This is enough to cover all possible viewpoints of the sphere. Unfortunately, a proof of how this number (30) is enough to cover the sphere cannot be presented but rather by studying the characteristics of the used descriptor. The ZM descriptor is invariant to scaling, translation, rotation, and reflection and ZM can measure skewness and kurtosis. This provides some variation margin in the viewpoint. For example, all gestures performed in -/+25 degrees of viewpoint variation can be considered as a single sector in the sphere. This finding is concluded from the posture recognition experiment (see Section 5.2). Users had a freedom of -/+25 degrees (hand-camera distance is 85 cm) of variation and the system was able to deliver 100% recognition rates.

The eight virtually created hand gestures are projected from 30 different viewpoints. These generated sequences are used to train 8 GMM HMM structures of left-to-right topology. The length of each gesture is set to be 90 frames. This number has been selected from prior knowledge of the mean length of the training gestures of the obtained datasets. Blender allows rendering a video with any number of frames. The software does the approximation between frames. To evaluate the performance of the projected sequences (animated gestures), the same steps that were explained in Section 5.3.2 are followed. The number the GMM for each HMM is set to be 3: this number is selected after several trials. 10 fold cross-validation methods are used. Feature vectors of size are 5 (4 ZMs and the CoM displacements).

Table 6-2 demonstrates the confusion matrix on using animated hand gestures. This resulted in mean accuracy rates of 43.75%.

		Predicted Gesture							
		"A"	"B"	"C"	"D"	"E"	"F"	"G"	"H"
Actual Gesture	"A"	**12**	0	3	3	5	1	1	5
	"B"	2	**11**	4	2	1	2	5	3
	"C"	3	0	**13**	2	2	4	3	3
	"D"	3	1	1	**15**	3	2	3	2
	"E"	4	1	3	1	**13**	2	1	5
	"F"	2	1	2	0	3	**17**	3	2
	"G"	2	0	4	3	1	6	**10**	4
	"H"	3	1	3	2	1	4	2	**14**

Table 6-2: The confusion matrix using 8 virtually created hand gestures. ZM descriptor is used with 8 HMMs. The number of training and testing instances is 240 instances, each gesture of 30 with 10 folds cross-validation. The mean accuracy rate is 43.75% where the base recognition rate is 100/8= 12.5%.

6.2.3 Discussion

This section introduced experiments on a new method that uses 3D models. These models have been captured from real hands by laser scanner. After that, animated hand gestures are created using 3D animation software. Then, the analysis-by-synthesis approach is adopted to tackle the viewpoints problem by projecting the virtually created hand gestures from several points in the sphere. The projected sequences are used to train the target HMM system.

The presented approach of tackling the viewpoint problem by 3D models and then deforming these models is validated by similar research on tracking hands recently presented in CVPR08 by Gorce *et al.* [172]. They used a 3D model for tracking 3D hand tracking using monocular video system. They estimated the pose, texture, and illumination dynamically by minimizing an objective function.

This section presents sequences that have been created from one user's hand. Consequently, the performed experiment is a user-dependant one. It is expected that when testing the performance on user-independent data, the overall accuracy would drop in the same measure as shown in the previous experiment in Chapter 5. This drop was about 5%.

The testing methods that were deployed can be extended in further research beyond the scope of the current research to explore how good a structure of HMMs trained on virtually created data is performing when tested on real data from different settings (UBL, OEH and OEW).

For a personal flexible user interface system, there are certain poses that the user takes when performing gestures. Therefore, when designing the viewpoint invariant system a dynamic learning algorithm should be included to update the system parameters with every new sequence. This will create over time a prior probability over the viewing directions that should improve detection performance.

The accuracy rate achieved in this experiment of 43% gives a good indication that this approach can be extended and could provide a solution to the afore-mentioned problem. However, it is clear that the currently achieved rate is not good enough for a multimedia application.

6.3 Summary and discussion

In this chapter, several experiments for hand gesture detection and one on viewpoint invariant hand gesture recognition are presented. The intention was to perform an extensive evaluation of different methods using HMMs. These experiments form the base that has been used to have a better understanding of how the hand gesture based user interface should be designed.

Two methods for hand gesture detection are discussed. First, detection using a sliding window is presented to form sequences that were evaluated with our pre-trained HMMs. The pre-trained HMMs were obtained from the isolated hand gestures (see Chapter 5). Second, a method on hand detection using the single HMM with the Viterbi algorithm is presented, with discussion on how to improve the overall accuracy rates.

In the second section of this chapter, an experiment on viewpoint invariant hand gesture recognition using a 3D model of a real hand is presented. The 3D model is captured using a laser scanner. Then, animated (virtual) hand gestures using the animation software are created. 8 HMMs are trained and tested on the animated hand gestures. The results obtained have been reported and discussed.

CHAPTER 7

CONCLUSION AND FUTURE DIRECTIONS

A game as an example of a computer user interface has been presented using vision techniques. One hand is used to control the game where hand gestures are defined as commands. Several related experiments have been explored for evaluating the proposed techniques.

New datasets of hand gestures are collected, with ground truth of manually segmented gestures and labelled sequences. Skin-colour has been tracked and segmented to provide the control for the descriptor. ZVM as a new spatio-temporal descriptor is introduced for hand gesture recognition system, with extensive evaluation and discussion. A comparative study of using ZVM+CvR and ZM+HMMs is introduced.

The research also tackled the problem of hand gesture detection. Two methods are presented for locating the start and the end of hand gestures automatically with a new discussion. Viewpoint invariant gestures are explored using 3D models captured from real hands. These techniques and experiments are utilized in a prototype framework. This framework is a simple interactive game.

7.1 Summary of achievements

For the hand gesture recognition system to be developed, training and testing datasets are required. This research introduces new datasets designed and used for this purpose. The datasets have been collected in three different settings, controlled lighting with consistent background (UBL dataset), hands only in an office environment (OEH dataset), whole body in the same office environment, and virtual performance (animated) of hand gestures using 3D models. Ground truth of the data has been provided using manual mark-up.

The CAMShift tracker [19] has been utilized in this research as it requires minimal training and can track irregular shapes and objects. Additionally, CAMShift can tolerate noise, does not require specific hardware or cameras, and is computationally fast to run. On the other hand, the main problem of CAMShift is the initialization where a blob needs to be selected for tracking. This problem has been tackled by creating a distribution model of the human hand skin colour. This model is used as a reference for tracking; checking if the tracked pixels belong to skin values.

It is hard to separate tracking, feature extraction and representation methods; in most cases they are all integrated with one another. It is important to note that descriptors need to produce a unique representation for an object and it is not necessary for the process to be reversible. Chapter 4 presented the segmentation and tracking techniques followed by moments as a feature descriptor.

The Zernike Velocity Moments descriptor has been introduced and evaluated with hand gesture recognition systems; previously the ZVM used successfully for human gait analysis [1, 2]. The results yield a potential use of ZVMs for their simplicity. The ZVM descriptor has not been evaluated before against any other techniques. The presented experiments (Chapter 5) provide this evaluation which concluded that ZM+HMM perform better than ZVM coupled with CvR [175, 176]. As mentioned in the previous section, CvR was chosen after exploring several methods.

From the confusion matrices and relative accuracies obtained in the two main experiments reported in Chapter 5 (ZVM+CvR and ZM+HMM), it is clear that the

ZM+HMM combination used in the second experiment (see Section 5.3) has given substantially better results than the ZVM+CvR combination used in the first experiment (see Section 5.2). Additionally, it seems intuitively plausible that the displacement of the centre of mass between frames carries discriminative information on the set of gestures. To explore this further, we carried out an experiment in which this displacement is added to the feature vector of Zernike moments used in the HMM [5-10, 45, 47-50, 58, 150]. This has produced an improved performance and the recognition rates increased from 94.5% to 98.3% (OEH dataset).

The accuracy rates of most experiments reported in Chapters 4, 5 and 6 are not reliable enough for high-end user interfaces. However, user-dependant experiments show an improved recognition rates that have increased from 84.5 to 88% (see Section 5.3.1.3). Mis-classifications can be due to participants performing gestures wrongly or the system recognizing a wrong gesture. The intended hand gesture interface could be designed in a way that users get feedback if their gesture is not recognized and thereby enable them to make small changes in order to improve recognition. In this way, the user would over time adapt to the system.

3D models have been captured using a laser scan system. These models were deformed to generate the intermediate models of hand shapes that gestures may take. A virtual arm was created using Blender animation software. Then the 3D models are fitted on the arm to produce the animated virtual hand gestures. These animated gestures are projected from different viewpoints. The projected sequences have been used to train and test a HMM based system. The obtained accuracy rates showed the potential of extending these methods. This accuracy is well above the baseline.

A prototype system using the developed hand gesture techniques has been described in Chapter 3. The prototype demonstrates the efficiency of the gesture sets (UBL). It uses thresholding for segmentation, ZM as a descriptor and HMM for classification and detection. The user-based evaluation shows the potential of such multimedia application.

7.2 Research limitations

- This research did not perform any experiments using data collected in outdoor environments. Moreover, it has been assumed that home environments are close to office environments. This assumption is almost correct for lighting conditions as both are indoor environments, but they vary in the nature of the background.

- It has been assumed that there is no collision and/or obstruction disturbing the viewpoint of the hand gesture. This is not always the case in the real word. Furthermore, this research did not consider the use of both hands together in any of the experiments. Speech intervention on hand gesture [34-36] has not been explored.
- The effect of changing the resolution on hand gesture recognition accuracy has not been explored fully. Only in the studied experiments take two resolutions are explored. First, when the camera is about 60-80 cm away from the hand. Second, when the camera is about 2 meters away from the hand. These two settings give a good indication of the effect of distance; where the accuracy rate in the second resolution has dropped sharply. Further research is required.
- Manual marking of the datasets is a subjective process. Different markers typically label the same data slightly differently. Therefore, to obtain an objective measure, a means of combining different markers' labellings is required. This is not performed in this research, rather only the labelling of one marker is used.
- Capturing the 3D models using a Polhemus laser scanner [78] is a difficult process, since users may move their hands during the acquisition of the data. Each model requires about 45 minutes to be captured. The resulting model is a multilayer one which needs further processing. This is widely done by approximating methods that interpolate different scan layers and generate one smooth layer (see Chapter 6). Approximation methods cause a loss in the real hand's features. Therefore, alternative scanning tools should be explored.
- Deforming the 3D model to generate the intermediate models involves approximating the new shape according to the skeleton of the hand [174]. Ideally, more intermediate 3D models should be captured from the real hand rather than the artificially creation. This could not be performed due to the difficulty of using the Polhemus scanning system.
- The skin colour distribution used for tracking (in Chapter 4) is a static one. This distribution does not update its parameters with new samples of the video input. Having a dynamic skin colour distribution is believed to enhance the performance of the segmentation. Further research is required.
- Applications that require precision, such as sensitive industrial productions or medical surgery applications, are less likely to use hand gestures for control.

- ZVM is a weighted sum of ZMs. This requires processing time that makes the use of this descriptor currently infeasible for real time applications. Future developments of the processing hardware may overcome this current limitation and promote the use of such descriptor.
- Ideally, it would be preferable to compare our results to those of other researchers, but the lack of testing standards and datasets made this infeasible. Furthermore, the task of distinguishing exactly these postures/gestures is very specific and comparison with results from smaller, larger, or different sets of data and their recognition rates would not be accurate.

7.3 Avenues for future studies

As with any research project, many improvements and extensions appear during the course of developing the work. This usually fits within the developed framework. Nevertheless, during the development new ideas and directions might become apparent even if they contradict the initial outline of the framework. This work is not an exception. We divided possible future work into two types as follows:

7.3.1 Framework enhancements

It is not possible to overcome all the limitations of the proposed methods but solving the possible ones is the obvious start for enhancing any framework. In addition, the segmentation method that we used in this research is based solely on skin-colour values. The skin-colour based algorithms suffer from mis-classified objects when the back-ground has colour values close to the skin-colour, or the lighting conditions are poor. Enhancing the segmentation requires the use of additional features such as edges [177].

ZVM suffers from low displacement components of the centre of mass. This will result in clustering the feature vectors around the origin in feature space. For some classifiers, this may have an impact on the ability to discriminate between gestures that involve little horizontal or vertical components of displacement. It is important to modify the notation of Equation 5-6 to avoid this effect. A potential solution to this problem is to use a conditional measure with two equations: one dealing with low horizontal displacement, and the other dealing with low vertical displacements.

The presented HMM design does not consider a different number of states for each individual structure, rather the presented experiments optimised the number of states

for all gestures. It would be beneficial to explore the performance of HMMs when each gesture has been optimized alone.

The current design of HMMs does not provide a mechanism for the learnt HMMs to update parameters when a new gesture arrives, as in [106]. Designing a system able to re-train the HMMs will enhance the accuracy rates.

It is more likely for users to perform certain hand gestures after other ones. In other words, the probability of a gesture to be performed after another is not equal. This is very similar to speech recognition system formulation [161]. Deploying this would improve the recognition rates. This can be done by modifying the transition matrix of the HMMs in Section 6.1.2, instead of using uniform probability for the transition between gestures.

7.3.2 Framework modifications

It would be much to the benefit of all research in the hand gesture recognition field to design standard datasets for training and testing. These datasets include most possible hand gestures with generic labelling. This would allow researchers to use these datasets for specific tasks. These datasets can benefit from recent researches on hand gesture ontology such the recent work of Karam [40] and Town [178].

This research has adopted a modular approach (see Figure 3-12) where each block of the system is dedicated for a certain task. Future work may extend this concept to have a hardware independent system. Currently, most of the research depends on particular hardware (including this research) and when the system uses different hardware (e.g. cameras) the performance changes dramatically. This may be due to calibration values. For example, creating a driver for the target machines that can calibrate the hardware to a certain standard would resolve this issue. This driver can interface the hardware and pipe the output of the designed system to any application, just like keyboards and mice.

Stereo cameras provide video input and disparity maps of the objects in the scene. Disparity maps provide 3D information about the performance of the hand gesture. The disparity map can be connected to the zooming mechanism so that cameras can readjust the resolution of its input to maintain a certain quality and level of hand details. Cameras with pan and tilt capability are also a potential use for hand recognition systems.

BIBLIOGRAPHY

[1] J. D. Shutler and M. S. Nixon, "Zernike Velocity Moments for Sequence-Based Description of Moving Features," *Image and Vision Computing*, vol. 24, pp. 343-356, 2006.

[2] J. D. Shutler and M. S. Nixon, "Zernike Velocity Moments for the Description and Recognition of Moving Shapes," in *Proceeding of the British Machine Vision Conference BMVC*, vol. 2. Manchester, 2001.

[3] Y. Wu and T. S. Huang, "Human Hand Modeling, Analysis and Animation in the Context of HCI," in *IEEE International Conference Image Processing*. Kobe, Japan, 1999.

[4] "Launches New Product Category: Surface Computing Comes to Life in Restaurants, Hotels, Retail Locations and Casino Resorts," Microsoft PressPass, 2007.

[5] B. Sinclair and R. Torres, "TGS 2005: Iwata speaks," in *http://uk.gamespot.com/*, 2005.

[6] J. L. Crowley, "FGnet datasets," INRIA Rhône-Alpes, http://www-prima.inrialpes.fr/FGnet/data/03-Pointing/index.html, Last access October 2008.

[7] F. Dadgostar, "Massey Hand Gesture Database," Massey University, http://tur-www1.massey.ac.nz/~fdadgost/xview.php?page=hand_image_database/default. Last access October 2008.

[8] "ASL BOSTON Database," Boston University, http://www.bu.edu/asllrp/ncslgr.html, Last access October 2008.

[9] T.-K. Kim, "Cambridge-Hand-Gesture Database," University of Cambridge, http://mi.eng.cam.ac.uk/~tkk22 Last accessed June 2008.

[10] Q. Yuan, S. Sclaroff, and V. Athitsos, "Automatic 2D Hand Tracking in Video Sequences," in *IEEE Workshop on Applications of Computer Vision*. pp. 250 - 256 2005.

[11] N. Liu, B. C. Lovell, and P. J. Kootsookos. "Evaluation of HMM Training Algorithms for Letter Hand Gesture Recognition," in *International Symposium on Signal Processing and Information Technology*: IEEE Computer Society press, pp. 648- 651, 2003.

[12] J. Alon, V. Athitsos, and S. Sclaroff, "Accurate and Efficient Gesture Spotting via Pruning and Subgesture Reasoning," in *International Conference on Computer Vision - Human-Computer Interaction ICCV-HCI*: pp. 189-198 Springer 2005.

[13] "Wii are getting fitter: Retirement home installs computer game to keep residents trim," in *Mailonline, http://www.dailymail.co.uk/sciencetech/article-481658/Wii-getting-fitter-Retirement-home-installs-game-residents-trim.html*, 13 September 2007

[14] R. Sharma, M. Yeasin, N. Krahnstoever, I. Rauschert, G. Cai, I. Brewer, A. M. Maceachren, and K. Sengupta, "Speech–Gesture Driven Multimodal Interfaces for Crisis Management," in *Proceedings of the IEEE 91*. pp. 1327-1354: IEEE Computer Society Press, 2003.

[15] R. Bowden, D. Windridge, T. Kadir, A. Zisserman, and M. Brady, "A Linguistic Feature Vector for the Visual Interpretation of Sign Language," in *European Conference on Computer Vision ECCV'04*. pp. 391-401: Springer-Verlag, 2004.

[16] R. Bowden, A. Zisserman, T. Kadir, and M. Brady, "Vision based Interpretation of Natural Sign Languages," in *International Conference on Computer Vision Systems ICVS'03*. pp. 391-401: Springer-Verlag, 2003.

[17] K. Fujimura and X. Liu, "Sign Recognition using Depth Image Streams," in *International Conference on Face and Gesture Recognition*. pp. 381-386: IEEE Computer Society Press, 2006.

[18] S. Borland, "Elderly 'addicted' to Nintendo Wii at care home," in *TELEGRAPH*. 17 Sept 2007.

[19] A. R. J. François, "CAMSHIFT Tracker Design Experiments with Intel OpenCV and SAI," Institute for Robotics and Intelligent Systems IRIS-04-423. 2004.

[20] G. R. Bradski, "Computer Vision Face Tracking For Use in a Perceptual User Interface," *Intel Technology Journal*, 1998.

[21] M. Hu, "Visual Pattern Recognition by Moment Invariants," *IEEE Transactions On Information Theory*, vol. 8, pp. 179-187, 1962.

[22] S. Chandran and A. Sawa, "Appearance-Based Real-Time Understanding of Gesture Using Projected Euler Angles," in *Real-Time Vision for Human-Computer Interaction*, B. Kisačanin, V. Pavlović, and T. S. Huang, Eds. pp. 57-66: Springer US, 2005.

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

[24] H. Guan, J. S. Chang, L. Chen, R. S. Feris, and M. Turk, "Multi-view Appearance-based 3D Hand Pose Estimation," in *Conference on Computer Vision and Pattern Recognition Workshop*. pp. 154-160: IEEE Computer Society Press, 2006.

[25] P. Dreuw, T. Deselaers, D. Rybach, D. Keysers, and H. Ney, "Tracking using Dynamic Programming for Appearance-based Sign Language Recognition," in *International Conference on Automatic Face and Gesture Recognition*

Automatic Face and Gesture Recognition. pp. 293 -298: IEEE Computer Society Press, 2006.

[26] H. Park, E. Kim, S. Jang, and H. Kim, "An HMM Based Gesture Recognition for Perceptual User Interface,"*Advances in Multimedia Information Processing*, pp. 1027-1034, 2004.

[27] J. Nespoulous, P. Perron, and A. R. Lecours, *The Biological Foundations of Gestures: Motor and Semiotic Aspects*. New Jersey London: Lawrence Erlbaum associates, 1986.

[28] A. Jaimes and N. Sebe, "Multimodal Human Computer Interaction: A Survey," *Computer Vision and Image Understanding*, vol. 108, pp. 116-134, 2007.

[29] K. Nickel, E. Seemann, and R. Stiefelhagan, "3D-Tracking of Head and Hands for Pointing Gesture Recognition in a Human-Robot Interaction Scenario," in *Proceeding on International Conference on Automatic Face and Gesture Recognition* Seoul, Korea: IEEE Computer Society Press, pp. 565- 570, 2004.

[30] T. Kirishima, K. Sato, and K. Chihara, "Real-Time Gesture Recognition by Learning and Selective Control of Visual Interest Points," *IEEE Transactions on Pattern Analysis and Machine Intelligence,*, vol. 27, pp. 351-365, 2005.

[31] S. Carbini, J. E. Viallet, and O. Bernier, "Pointing Gesture Visual Recognition for Large Display," in *International Conference of Pattern Recognition*. pp. 203-208: IEEE Computer Society Press, 2004.

[32] S. Marukatat, T. Artières, and P. Gallinari, "A Generic Approach for on-line Handwriting Recognition," in *IEEE International Workshop on Frontiers in Handwriting Recognition IWFHR-9*, vol. 9, pp. 401- 406, 2004.

[33] B. Signer, U. Kurmann, and M. C. Norrie, "iGesture: A General Gesture Recognition Framework," in *Ninth International Conference on Document Analysis and Recognition; pp.* 954-958, 2007.

[34] D. McNeill, "Gesture," in *Cambridge Encyclopaedia of the Language Sciences; URL: http://mcneilllab.uchicago.edu/pdfs/cambridge_encyclopedia.pdf*, 2006.

[35] D. McNeill, *Gesture and Thought*: published by the University of Chicago Press, 2005.

[36] M. Turk, "Gesture Recognition," in *The human-computer interaction handbook*: Lawrence Erlbaum Associates, 2003.

[37] R. Watson, "A Survey of Gesture Recognition Techniques," Department of Computer Science. Trinity College, Dublin 1993.

[38] M. Hasanuzzamana, T. Zhanga, V. Ampornaramvetha, H. Gotodaa, Y. Shiraib, and H. Uenoa, "Adaptive Visual Gesture Recognition for Human–Robot Interaction using A Knowledge-Based Software Platform," *Robotics and Autonomous Systems*, vol. 55, pp. 643-657, 2007.

[39] S. Mitra and T. Acharya, "Gesture Recognition: A Survey," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 37, pp. 311-325, 2007.

[40] M. Karam, "A Framework for Research and Design of Gesture-based Human Computer Interactions," in *Faculty of Engineering, Science and Mathematics; School of Electronics and Computer Science*, PhD: University of Southampton, 2006.

[41] H. Drewes and A. Schmidt, "Interacting with the Computer Using Gaze Gestures," *International Federation for Information Processing*, vol. 4663, pp. 475-488, 2007.

[42] L. Lee and W. E. L. Grimson, "Gait Analysis for Recognition and Classification," in *Fifth IEEE International Conference on Automatic Face and Gesture Recognition,* pp. 155-163, IEEE Computer Society Press, 2002.

[43] J. LaViola, "A Survey of Hand Posture and Gesture Recognition Techniques and Technology," Brown University 1999.

[44] S. Chandran and A. Sawa, "Real-Time Detection and Understanding of Isolated Protruded Fingers," in *Computer Vision and Pattern Recognition Workshops*: IEEE Computer Society Press, 2004.

[45] R. Feris, M. Turk, R. Raskar, K. Tan, and G. Ohashi. "Recognition of Isolated Fingerspelling Gestures Using Depth Edges," in *Real-Time Vision for Human-Computer Interaction*, B. Kisačanin, V. Pavlović, and T. S. Huang, Eds.: Springer, pp. 43-56, 2005.

[46] R. Lockton and A. W. Fitzgibbon, "Real-time Gesture Recognition using Deterministic Boosting," in *British Machine Vision Conference*. pp. 817-826, 2002.

[47] M. Everson, "SignWriting The Sutton Fonts, http://www.signwriting.org/," Last accessed October 2008.

[48] O. Al-Jarrah and A. Halawani, "Recognition of gestures in Arabic sign language using neuro-fuzzy systems," *Journal on Artificial Intelligence* vol. 133, pp. 117-138 2001.

[49] S. K. Liddell, *Grammar, Gesture, and Meaning in American Sign Language*: Cambridge University Press, 2003.

[50] C. Wu, Y. Chiu, and K. Cheng, "Error-Tolerant Sign Retrieval Using Visual Features and Maximum a Posteriori Estimation," *Pattern Analysis and Machine Intelligence*, vol. 26, pp. 495- 508, 2004.

[51] T. Starner and A. Pentland, "Visual Recognition of American Sign Language Using Hidden Markov Models," in *International Workshop on Automatic Face and Gesture Recognition*. pp. 189-194: IEEE Computer Society Press, 1995.

[52] M. Zobl, R. Nieschulz, M. Geiger, M. Lang, and G. Rigoll, "Gesture Components for Natural Interaction with In-Car Devices." in *Gesture-Based Communication in Human-Computer Interaction*, vol. 2915, pp. 367-368, Ed.: Springer Berlin / Heidelberg, 2004.

[53] J. MacCormick and M. Isard, "Partitioned Sampling, Articulated Objects, and interface-quality hand tracking," in *European Conference on Computer Vision*, 2000.

[54] M. Isard and J. MacCormick, "Hand Tracking for Vision-Based Drawing," Tech Report, University of Oxford, 2000.

[55] *A. D. Wilson*, "Robust Computer Vision-Based Detection of Pinching for One and Two-Handed Gesture Input," in *User Interface Software and Technology*, pp. 255 - 258 ACM Press, 2006.

[56] J. Weissmann and F. Salomon, "Gesture Recognition for Virtual Reality Applications Using Data Gloves and Neural Networks," in *International Joint Conference on Neural Networks*: IEEE Computer Society Press, 1999.

[57] M. Isard and A. Blake, "Condensation Conditional Density Propagation for Visual Tracking," *International Journal of Computer Vision*, vol. 29, pp. 5-28, 1998.

[58] A. Sepehri, Y. Yacoob, and L. S. Davis, "Parametric Hand Tracking for Recognition of Virtual Drawings," in *Proceeding of the fourth IEEE International Conference on Computer Vision Systems*, pp. 6-15: IEEE Computer Society, 2006.

[59] M. Collomb, M. Hascoët, P. Baudisch, and B. Lee, "Improving drag-and-drop on wall-size displays," in *Proceedings of the conference on Graphics interface*: Canadian Human-Computer Communications Society, 2005.

[60] L. d. F. Costa and R. M. Cesar, *Shape Analysis and Classification. Theory and Practice* vol. 25, CRC Press LLC, 2001.

[61] S. Malik, A. Ranjan, and R. Balakrishnan, "Interacting with Large Displays from a Distance with Vision-Tracked Multi-Finger Gestural Input," in *User interface software and technology*: ACM Press, pp. 43-52, 2005.

[62] A. Mazalek and M. Nitsche, "Tangible Interfaces for Real-time 3D Virtual Environments," in *Proceedings of the international conference on Advances in computer entertainment technology*. pp. 155 - 162: ACM, 2007.

[63] A. Wilson and N. Oliver, "GWindows: Towards Robust Perception-Based UI," in *Workshop on Computer Vision and Pattern Recognition for Human Computer Interaction*: IEEE Computer Society Press, pp. 46-54, 2003.

[64] A. D. Wilson, "TouchLight: An Imaging Touch Screen and Display for Gesture-Based Interaction," in *International Conference on Multimodal Interfaces (ICMI'04)*. pp. 69 - 76: ACM, 2004.

[65] S. C. Ahn, T. Lee, I. Kim, Y. Kwon, and H. Kim, "Large Display Interaction Using Video Avatar and Hand Gesture Recognition," in *ICIAR'04*: Springer-Verlag Berlin Heidelberg, pp. 261-268, 2004.

[66] Z. Zeng, M. Pantic, G. I. Roisman, and T. S. Huang, "A Survey of Affect Recognition Methods: Audio, Visual and Spontaneous Expressions," in *International Conference on Multimodal Interfaces*. pp. 126-133, ACM, 2007.

[67] Z. Mo, J. P. Lewis, and U. Neumann, "Smartcanvas: a Gesture-Driven Intelligent Drawing Desk System," in *10th International Conference on Intelligent User Interfaces*: ACM Press, pp. 239-243, 2005.

[68] J. Mullinax, "What is Surface?, http://blogs.msdn.com/johnmullinax/archive/2007/06/04/what-is-surface.aspx," Last accessed October 2008.

[69] "5DT Data Glove, www.motionwerx.com," Last accessed October 2008.

[70] A. Holzinger, *Finger Instead of Mouse: Touch Screens as a Means of Enhancing Universal Access*: Springer Berlin / Heidelberg, 2003.

[71] D. J. Sturman and D. Zeltzer, "A Survey of Glove-based Input," *IEEE Computer Graphics and Applications*, vol. 14, pp. 30-39, 1994.

[72] Fifth_Dimension_Technologies, "VR Training Simulators and VR Peripherals." http://www.5dt.com/products.html, 2007.

[73] K. N. Tarchanidis and J. N. Lygouras, "Data Glove with a Force Sensor," *IEEE Instrumentation and Measurement Technology Conference*, vol. 52, pp. 984-989, 2001.

[74] V. I. Pavlovic, R. Sharma, and T. S. Huang, "Gestural Interface to a Visual Computing Environment for Molecular Biologists," in *Proceedings of the 2nd International Conference on Automatic Face and Gesture Recognition*: IEEE Computer Society Press, pp. 964-968, 1996.

[75] V. I. Pavlovic, R. Sharma, and T. S. Huang, "Visual Interpretation of Hand Gestures for Human-Computer Interaction: A Review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, pp. 677-695, 1997.

[76] T. B. Moeslund and L. Nørgaard, "A Brief Overview of Hand Gestures used in Wearable Human Computer Interfaces," Laboratory of Computer Vision and Media Technology, Aalborg University, Denmark, Technical report 2003.

[77] Ying Wu and T. S. Huang, "Vision-Based Gesture Recognition: A Review," in *International Gesture Workshop*. pp. 103 - 115 Computer Lecture Notes, 1999.

[78] "Polhemus Laser handheld scanner," http://www.polhemus.com/, Last accessed October 2008.

[79] A. Heap, "Learning Deformable Shapes Models for Object Tracking," PhD thesis, *Computing School*, University of Leeds, 1998.

[80] C. C. Lien, "A Scalable Model-based Hand Posture Analysis System," *Machine Vision and Applications*, vol. 16, pp. 157-169, 2005.

[81] J. Rehg, D. D. Morris, and T. Kanade, "Ambiguities in Visual Tracking of Articulated Objects Using Two- and Three-Dimensional Models," *International Journal of Robotics Research*, vol. 22, pp. 393 - 418, 2003.

[82] A. Erola, G. Bebisa, M. Nicolescua, R. Boyleb, and X. Twombly, "Vision-based Hand Pose Estimation: A review," *Computer Vision and Image Understanding* vol. 108, pp. 52-73, 2007.

[83] M. Al-Rajab, "Leeds Hand Gesture Datasets." www.comp.leeds.ac.uk/moaath/gHand/DS/, University of Leeds. Last access October 2008, 2008.

[84] A. Heap and D. Hogg, "Towards 3D Hand Tracking using a Deformable Model," in *2nd International Conference on Automatic Face and Gesture Recognition*: IEEE Computer Society Press, pp. 140-145, 1996.

[85] V. Athitsos and S. Sclaroff, "An Appearance-based Framework for 3D Hand Shape Classification and Camera Viewpoint Estimation," in *Fifth IEEE International Conference Proceedings on Automatic Face and Gesture Recognition*, vol. 1: IEEE Computer Society Press, pp. 45-50, 2002.

[86] J. Rehg and T. Kanade, "Digiteyes: Vision-based Hand Tracking for Human-Computer Interaction," in *Proceedings of the workshop on Motion of Non-Rigid and Articulated Bodies*. pp. 16-22: IEEE Computer Society Press, 1994.

[87] Y. Wu, J. Y. Lin, and T. S. Huang, "Capturing Natural Hand Articulation," *Computer Graphics and Applications*, vol. 2, pp. 426-432, 2002.

[88] E. B. Sudderth, M. I. Mandel, W. T. Freeman, and A. S. Willsky, "Visual Hand Tracking Using Nonparametric Belief Propagation," in *Conference on Computer Vision and Pattern Recognition Workshop*: IEEE Computer Society Press, pp. 189-197, 2004.

[89] H. Zhou, D. J. Lin, and T. Huang, "Okapi-Chamfer Matching For Articulate Object Recognition," in *Tenth IEEE International Conference on Computer Vision ICCV'05*, vol. 2: IEEE Computer Society, 2005.

[90] N. Shimada, Y. Shirai, Y. Kuno, and J. Miura, "Hand Gesture Estimation and Model Refinement using Monocular Camera - Ambiguity Limitation by

Inequality Constraints," in *Face and Gesture Recognition Conference*. pp. 268-275: IEEE Computer Society Press. 1998.

[91] B. Stenger, P. Mendonca, and R. Cipolla, "Model-Based Hand Tracking Using an Unscented Kalman Filter," in *Proceedings of the British Machine Vision Conference BMVC*. London, pp, 63-72, 2001.

[92] K. Morrison and S. J. McKenna, "An Experimental Comparison of Trajectory-Based and History-Based Representation for Gesture Recognition," in *Gesture-Based Communication in Human-Computer Interaction*: Springer Berlin / Heidelberg, pp. 152-163, 2004.

[93] H. Zhou, D. J. Lin, and T. S. Huang, "Static Hand Posture Recognition Based on Okapi-Chamfer Matching," in *Real-Time Vision for Human-Computer Interaction*, B. Kisačanin, V. Pavlović, and T. S. Huang, Eds.: Springer, pp. 85-101, 2005.

[94] B. Stenger, A. Thayananthan, P. H. S. Torr, and R. Cipolla, "Model-Based Hand Tracking Using a Hierarchical Bayesian Filter," *Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, pp. 1372- 1384, 2006.

[95] H. I. Christensen and H.-H. Nagel, *Cognitive Vision Systems: Sampling the Spectrum of Approaches*. Berlin Springer, 2006.

[96] D. R. Magee, "Tracking Multiple Vehicles using Forground, Background and Motion Models," *Image and vision computing, Elsevier*, vol. 22, pp. 143-155, 2004.

[97] Q. Pham, L. Gond, J. Begard, N. Allezard, and P. Sayd, "Real-Time Posture Analysis in a Crowd using Thermal Imaging," in *IEEE Conference on Computer Vision and Pattern Recognition CVPR '07*: IEEE Computer Society Press, pp. 1-8, 2007.

[98] K. Oka and Y. Sato, "Real-Time Fingertip Tracking and Gesture Recognition," in *International conference on Computer Graphics and Applications*. pp. 64-71: IEEE Computer Society Press, 2002.

[99] M. Sonka, V. Hlavac, and R. Boyle, *Image Processing, Analysis, and Machine Vision*, 3rd ed: Thomson, 2008.

[100] Y. Freund and R. E. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," in *European Conference on Computational Learning Theory*. pp. 119-139: IEEE Computer Society Press, 1995.

[101] H. M. Dee, "Explaining visible behaviour," in *School of Computing*. PhD. Leeds, University of Leeds, 2005.

[102] J. Shi and C. Tomasi, "Good Features to Track," in *IEEE Conference on Computer Vision and Pattern Recognition*: IEEE Computer Society Press, pp. 593-600, 1994.

[103] W. Snyder and H. Qi, *Machine Vision*: Cambridge University Press, 2004.

[104] J. Ohya, A. Utsumi, and J. Yamato, *Analyzing Video Sequences of Multiple Humans : Tracking, Posture Estimation and Behavior Recognition*. London: Kluwer Academic, 2002.

[105] I. Laptev and T. Lindeberg, "Tracking of Multi-state Hand Models Using Particle Filtering and a Hierarchy of Multi-scale Image Features," presented at Proc. Scale-Space and Morphology in Computer Vision, pp. 63-74, 2001.

[106] S. Nikolay, G. Aphrodite, and H. Roger, "A Real-time Hand Tracker using Variable-Length Markov Models of Behaviour," *Comput. Vis. Image Underst.*, vol. 108, pp. 98-115, 2007.

[107] Y. Wu and T. S. Huang, "View-independent Recognition of Hand Postures," in *Computer Vision and Pattern Recogntion Conference*: IEEE Computer Society Press, pp. 88-94, 2000.

[108] D. Zhang and G. Lu, "Review of Shape Representation and Description Techniques," *Pattern Recognition, Elsevier*, vol. 37, pp. 1-19, 2004.

[109] M. Sonka, V. Hlavac, and R. Boyle, *Image Processing, Analysis, and Machine Vision*, vol. Third Edition: PWS, Brooks and Thomson Publishing, 2008.

[110] G. Ritter and J. Wilson, *Handbook of Computer Vision Algorithms in Image Algebra*. London Boca Raton 1996.

[111] W. Xiaoling and X. Kanglin, "A Novel Direction Chain Code-based Image Retrieval," in *Computer and Information Technology*. pp.190-193: IEEE Computer Society press, 2005.

[112] I. Laptev and P. Perez, "Retrieving Actions in Movies," in *International Conference on Computer Vision*: IEEE Computer Society Press, pp. 1-8, 2007.

[113] I. Laptev, M. Marszaek, C. Schmid, and B. Rozenfeld", "Learning Realistic Human Actions from Movies," in *IEEE Conference on Computer Vision Pattern Recognition*. pp. 1-8: IEEE Computer Society press, 2008.

[114] I. Laptev and T. Lindeberg, "Local Descriptors for Spatio-temporal Recognition," in *Spatial Coherence for Visual Motion Analysis*. pp. 91-103: Springer Berlin / Heidelberg, LNCS, 2006.

[115] D. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints." *International Journal of Computer Vision*, vol. 60, pp. 91-110, 2004.

[116] T. Kim and R. Cipolla, "Gesture Recognition Under Small Sample Size," in *International Asian Conference on Computer Vision (ACCV'07)*: Springer-Verlag Berlin Heidelberg, pp. 969-976, 2007.

[117] Q. Ying-Dong, C. Cheng-Song, C. San-Ben, and L. Jin-Quan. "A Fast Subpixel Edge Detection Method Using Sobel-Zernike Moments Operator." *Image Vision Computing*, vol. 23, pp. 11 - 17, 2005.

[118] P. Morguet and M. Lang, "Spotting Dynamic Hand Gestures in Video Image Sequences Using Hidden Markov Models," in *International Conference Image Processing (ICIP98)* pp. 193-197: IEEE Computer Society Press, 1998.

[119] H. D. Yang, A. Y. Park, and S. W. Lee, "Robust Spotting of Key Gestures from Whole Body Motion Sequence," in *International Conference on Automatic Face and Gesture Recognition Automatic Face and Gesture Recognition*: IEEE Computer Society Press, pp. 231-236, 2006.

[120] H.-S. Yoon, J. Soha, Y. J. Baea, and H. S. Yang, "Hand Gesture Recognition Using Combined Features of Location, Angle and Velocity," *Pattern Recogntion*, vol. 34, pp. 1491-1501, 2001.

[121] B. Ozer and W. Wolf, "Real-time Posture and Activity Recognition," in *Workshop on Motion and Video Computing*: IEEE Computer Society Press, pp. 133- 138, 2002.

[122] J. Alon, V. Athitsos, Q. Yuan, and S. Sclaroff, "Simultaneous Localization and Recognition of Dynamic Hand Gestures," in *IEEE Workshop on Motion and Video Computing*: IEEE Computer Society Press, pp. 254-260, 2005.

[123] K. V. Mardia, N. M. Ghali, T. J. Hainsworth, M. Howes, and N. Sheehy, "Techniques for Online Gesture Recognition on Workstations." *Image and Vision Computing*, vol. 11, pp. 283-294, 1993.

[124] M. Piccardi and O. Perez, "Hidden Markov Models with Kernel Density Estimation of Emission Probabilities and their Use in Activity Recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'07)*: IEEE Computer Society Press, pp. 1-8, 2007.

[125] R. Huang, V. Pavlovic, and D. N. Metaxas, "Embedded Profile Hidden Markov Models for Shape Analysis," in *11 IEEE International Conference on Computer Vision*: IEEE computer society press, pp. 1-8, 2007.

[126] J. Bilmes, "What HMMs Can Do," Tech Report, University of Washington 2002.

[127] A. F. Bobick and Y. A. Ivanov, "Action Recognition using Probabilistic Parsing," in *Computer Vision and Pattern Recognition*, pp. 196-204: IEEE Computer Society Press, 1998.

[128] A. Just and S. Marcel, "Two-Handed Gesture Recognition," Tech Report, IDIAP 2005.

[129] Webcams, "http://www.logitech.com," Last accessed October 2008.

[130] Blender, "www.blender.org ", Last accessed October 2008.

[131] M. Inc., "Matlab Simulink Examples, http://www.mathworks.com/products/simulink/ ", Last accessed October 2008.

[132] K. Murphy, "Hidden Markov Model (HMM) Toolbox for Matlab," in *http://www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html*, Last accessed October 2008.

[133] M. J. Jones and J. M. Rehg, "Statistical Color Models with Application to Skin Detection," *International Journal of Computer Vision*, pp. 81- 96, 2002.

[134] F. T. Marti, "Optimal thresholding, http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=132," Last accessed October 2008.

[135] N. Johnson and D. Hogg, "Representation and Synthesis of Behaviour using Gaussian Mixtures " *Image and Vision Computing*, vol. 20, pp. 889-894, 2002.

[136] D. Comaniciu and P. Meer, "Mean shift: a Robust Approach Toward Feature Space Analysis," *IEEE Transactions Pattern Analysis and Machine Intelligence*, vol. 24, pp. 603-619, 2002.

[137] D. Comaniciu and P. Meer, "Robust Analysis of Feature Spaces: Color Image Segmentation," in *IEEE Conf. on Comp. Vis. and Pattern Recognition*: IEEE Computer Society, pp. 750-755, 1997.

[138] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-Based Object Tracking," *Pattern Analysis Machine*, vol. 25, pp. 564-575, 2003.

[139] "Open Source Computer Vision Library Reference Manual," Intel Corporation, 2001.

[140] J. G. Allen, R. Y. D. Xu, and J. S. Jin, "Object Tracking Using CamShift Algorithm and Multiple Quantized Feature Spaces," in *Proceedings of the Pan-Sydney area workshop on Visual information processing* Darlinghurst, Australia: Australian Computer Society, 2004.

[141] C. J. Needham, R D. Boyle, "Performance evaluation metrics and statistics for positional tracker evaluation" *in: Crowley, J L, Paiter, J H, Vincze, M & Paletta, L (editors) Computer Vision Systems Third International Conference*, ICVS 2003, pp. 278-289 Springer-Verlag. 2003.

[142] J. Flusser, "On the Independence of Rotation Moment Invariants," *Elsevier Journal on Pattern Recognition*, vol. 33, pp. 1405-1410, 2000.

[143] R. Bailey and M. Srinath, "Orthogonal Moment Features for Use with Parametric and Non-Parametric Classifiers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, pp. 389-399, 1996.

[144] H. Hse and A. R. Newton, "Sketched Symbol Recognition using Zernike Moments," in *17th International Conference on Pattern Recognition (ICPR'04)*, vol. 1. pp.367-370: IEEE Computer Society Press, 2004.

[145] B. D. R. Stenger, "Model-Based Hand Tracking Using A Hierarchical Bayesian Filter," in *School of Computing*. PhD. University of Cambridge, 2004, pp. 127.

[146] J. Yang and Y. Xu, "Hidden Markov Model for Gesture Recognition," Robotics Institue, Carnegie Mellon University, Pennsylvania 1994.

[147] K. CHANG, "LANS Pattern Recognition Toolbox ": http://www.lans.ece.utexas.edu/~lans/, Last access January 2008.

[148] "University of Waikato, Weka 3: Data Mining Software in Java," http://www.cs.waikato.ac.nz/~ml/weka/, Last access October 2008.

[149] S. Paschalakis and P. Lee, "Pattern Recognition in Grey Level Images using Moment Based Invariant Features," in *Image Processing And Its Applications*. pp. 245-249: IEEE Computer Society press, 1999.

[150] M. Al-Rajab, D. Hogg, and K. Ng, "A Comparative Study on Using Zernike Velocity Moments and Hidden Markov Models for Hand Gesture Recognition " in *Articulated Motion and Deformable Objects*. pp. 319-327: LNCS, Springer Berlin / Heidelberg, 2008.

[151] M. T. Pengyu Hong, Thomas Huang, "Gesture Modeling and Recognition Using Finite State Machines," in *IEEE Conference on Face and Gesture Recognition*, pp. 410-415, 2000.

[152] M. H. Jeong, Y. Kuno, N. Shimada, and Y. Shirai, "Recognition of Shape-changing Hand Gestures based on Switching Linear Model," in *11th International Conference on Image Analysis and Processing*. pp. 14-21: IEEE Computer Society Press, 2001.

[153] P. R. Raamana, D. Grest, and V. Krueger, "Human Action Recognition in Table-Top Scenarios : An HMM-Based Analysis to Optimize the Performance," in t*he 12th International Conference on Computer Analysis of Images and Patterns*: Springer, LNCS, pp. 101-108, 2007.

[154] Y. Wang, K. Huang, and T. Tan. "Human Activity Recognition Based on R Transform," in *Computer Vision and Pattern Recognition*. pp. 1-8: IEEE Computer Society Press, 2007.

[155] I. Cohen, N. Sebe, A. Garg, and L. S. Chen, "Facial Expression Recognition from Video Sequences: Temporal and Static Modeling," *Computer Vision and Image Understanding*, vol. 91, pp. 160–187, 2003.

[156] J. Hannuksela, C. Information, M. Barnard, P. Sangi, and J. Heikkilä, "Adaptive Motion-Based Gesture Recognition Interface for Mobile Phones," in *6th International Conference on Computer Vision Systems (ICVS08)*. Greece: LNCS, Springer, pp. 271-280, 2008.

[157] P. Buehler, M. Everingham, D. P. Huttenlocher, and A. Zisserman, "Long Term Arm and Hand Tracking for Continuous Sign Language TV Broadcasts," in *BMVC*. Leeds, 2008.

[158] N. Sebe, M. S. Lew, and T. S. Huang, "Computer Vision in Human-Computer Interaction," presented at International Conference on Machine Vision ICCV, Beijing, China, pp. 121-128, 2006.

[159] L. R. Rabiner, S. E. Levinson, and M. M. Sondhi, "On the Application of Vector Quantization and Hidden Markov Models to Speaker-Independent Isolated Word Recognition," *Bell System Technical Journal*, vol. 62, pp. 1075-1105, 1983.

[160] R. Boyle, "HMM tutorial," *URL* http://www.comp.leeds.ac.uk/roger/HiddenMarkovModels/html_dev/main.html, Last access October 2008.

[161] L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *IEEE Transactions on Information Theory*, vol. 77, pp. 257-286, 1989.

[162] A. Chalechale, F. Safaei, G. Naghdy. and P. Premaratne. "Hand gesture selection and recognition for visual-based human-machine interface." in *International Conference on Electro Information Technology*: IEEE Computer Society press, pp. 279-293, 2005.

[163] F. Jelinek, *Statistical Methods for Speech Recogntion*, 1998.

[164] Q. Wang, X. Chen, L.-G. Zhang, C. Wang, and W. Gao, "Computer Vision and Image Understanding," *Computer Vision and Image Understanding*, vol. 108, pp. 87-97, 2007.

[165] V. Parameswaran and R. Chellappa, "View Invariance for Human Action Recognition," *International Journal of Computer Vision*, vol. 66, pp. 83-101, 2006.

[166] Y. Chen, J. Liu, and X. Tang, "Sketching in the Air: A Vision-Based System for 3D Object Design," in *Computer Vision and Pattern Recognition CVPR08*: IEEE Computer Society press, pp. 1-6, 2008.

[167] T. Sowa, "The Recognition and Comprehension of Hand Gestures - A Review and Research Agenda," in *Modeling Communication with Robots and Virtual Humans*. pp. 38-56: LNCS, Springer Berlin / Heidelberg, 2008.

[168] H.-K. Lee and J.-H. Kim, "Gesture Spotting from Continuous Hand Motion," *Pattern Recognition Letters*, vol. 19, pp. 513–520, 1998.

[169] M. Kolsch, "Vision Based Hand Gesture Interfaces for Wearable Computing and Virtual Environments," in *Computer Science*, PhD: University of California, 2004.

[170] I. Cohen, A. Garg, and T. S. Huang, "Emotion Recognition from Facial Expressions using Multilevel HMM," in *NIPS Workshop on Affective Computing*: IEEE Computer Society Press, 2000.

[171] M. B. Holte and T. B. Moeslund, "View Invariant Gesture Recognition Using 3d Motion Primitives," in *IEEE International Conference on Acoustics, Speech and Signal Processing, 2008. ICASSP 2008.* pp. 797-803: IEEE Computer Society press, 2008.

[172] M. d. L. Gorce, N. Paragios, and D. J. Fleet, "Model-Based Hand Tracking with Texture, Shading and Self-occlusions," in *Computer Vision and Pattern Recogntion*. USA: IEEE Computer Society press, pp. 1-8, 2008.

[173] FastScan, "www.polhemus.com," 2008.

[174] J. P. Lewis, M. Cordner, and N. Fong, "Pose Space Deformation: a unified Approach to Shape Interpolation and Skeleton-driven Deformation," in *International Conference on Computer Graphics and Interactive Techniques*: ACM Press, pp. 165-172, 2000.

[175] M. Ceci, A. Appice, and D. Malerba, "Comparing Simplification Methods for Model Trees with Regression and Splitting Nodes," in *Fourteenth international symposium on methodologies for intelligent systems*. pp. 49-56: Springer-Verlag in LNCS/LNAI, 2008.

[176] D. Malerba, F. Esposito, Michelangelo Ceci, and A. Appice, "Top-Down Induction of Model Trees with Regression and Splitting Nodes," *Pattern analysis and machine intelligence*, vol. 26, pp. 612 - 625, 2007.

[177] S. L. Phung, A. Bouzerdoum, and D. Chai, "Skin Segmentation Using Color And Edge Information," in *Symposium on Signal Processing and its Applications*. pp. 525-528: IEEE Computer Society Press, 2003.

[178] C. P. Town, "Ontology based Visual Information Processing," in *Computer Laboratory*, PhD, University of Cambridge, pp. 268, 2004.

[179] E. Frank, Y. Wang, S. Inglis, G. Holmes, and I. H. Witten, "Using Model Trees for Classification," Department of Computer Science, University of Waikato, Hamilton, New Zealand 1998.

[180] F. Jurie and B. Triggs, "Creating Efficient Codebooks for Visual Recognition." in *Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05)* pp. 604 - 610: IEEE Computer Society Press, 2005.

[181] J. B. MacQueen, "Some Methods for classification and Analysis of Multivariate Observations," in *5-th Berkeley Symposium on Mathematical Statistics and Probability*. Berkeley: University of California Press, 1967.

[182] N. Sebe and M. S. Lew, *Robust Computer Vision : Theory and Applications*. Dordrecht ; London: Kluwer Academic, 2003.

[183] D. Forsyth and J. Ponce, *Computer Vision A Modern Approach*: Prentice Hall, *2003*.

[184] A. F. Bobick and J. W. Davis, *"The Recognition of Human Movement Using Temporal Templates,"* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 23, pp. 257-267, 2001.

APPENDIX A

Classification methods

I. Classification Via Regression (CvR)

The CvR method is a classifier based on binary tree-structured regression models that associate leaves with linear regression functions [148]. CvR is constructed of two steps. Firstly, an ordinary decision tree is created, using as splitting criterion the maximization of the intra-subset value variation of the target. Secondly, prunes back this tree by replacing sub-trees with linear regression functions wherever this seems suitable [179].

For prediction, CvR effectively smoothes the sharp discontinuities that will inevitably occur between adjacent linear models at the leaves of the pruned tree. The idea of smoothing is first to use the leaf model to calculate the predicted value, and then to filter that value along the path back to its root, smoothing it at each node by combining it with the value predicted by the linear model for that node. More details on CvR can be found in [148 , 175, 176, 179].

The Weka [148] implementation is used in this research. We have explored 3 classification methods and CvR produced the best results as shown in Table 5-1. The effective performance of CvR can be explained by its simplicity and being the non-parametric method where the final results of using tree methods for classification or regression can be summarized in a series of intersected logical if-then conditions.

II. Sequential Minimal Optimization (SMO)

Sequential minimal optimization was first proposed by John C. Platt [180] and can be considered a special case of the decomposing algorithm. Training a support vector machine requires solving a very large quadratic programming (QP) optimization problem. SMO tackles this problem by breaking it down in small QP problems that can be solved analytically. Unlike other approaches, SMO always solves the smallest possible optimization problem in each step.

SMO scales linearly in memory for the amount of training data and somewhere between linear and quadratic for calculation time. SMO's computation time is dominated by SVM evaluation and is thus fastest for linear SVMs.

There are three main components in the SMO algorithm:

- Solving two Lagrange multipliers analytically
- Heuristic to choose which multipliers to optimize
- Calculating the SVM threshold

III. K-means

K-means is introduced by MacQueen [181] in 1967. K-means is a simple unsupervised learning algorithm that addresses the clustering problem. The k-means method classifies a given data set through a certain number of clusters (assume k clusters) fixed a priori. The main idea is to define k centroids, one for each cluster. These centroids should be placed in a predefined way because of different location causes different result. So, the better choice is to place them as much as possible far away from each other. The next step is to take each point belonging to a given data set and associate it to the nearest centroid. When associating all points, the first step is completed and an early clustering is done. At this point we need to re-calculate k new centroids of the clusters resulting from the previous step. After we have these k new centroids, a new iteration has to be done between the same data set points and the nearest new centroid. A loop has been generated. As a result of this loop we may notice that the k centroids change their location step by step until no more changes are done [180]. In other words centroids do not move any more. More information on classification methods are in [60, 99, 182, 183].