Gesture Recognition Using Principal Component Analysis, Multi-Scale Theory, and Hidden Markov Models

Wu Hai

Submitted in fulfilment of the requirements for a Ph.D Degree

School of Computer Applications

Dublin City University

Ireland

Supervisor: Dr. Alistair Sutherland

May, 2002

Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Doctor for Philosophy is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed: Ha: wu (Candidate)

ID No.: 98970224

Date: Aug. 23, 2002

Acknowledgements

First and foremost, I am deeply grateful to Dr. Alistair Sutherland, who gave me an opportunity and guided me throughout my Ph.D studies. I am also thankful for the friendship and guidance of Dr. Martin Crane. I acknowledge Professor Heather Ruskin's help in writing this thesis.

I would also like to thank Hyowon Lee, Paul Browne, Kieran Mc Donald, Jiamin Ye, Yongle Liu, Cathal Gurrin, Tom Södring, Jer Hayes, Aidan Haran and all the other lads and lasses in the postgraduate lab.

Finally, my great appreciation to my parents for their love and support.

Table of Contents

Abstract

In this thesis, a dynamic gesture recognition system is presented which requires no special hardware other than a Webcam. The system is based on a novel method combining Principal Component Analysis (PCA) with hierarchical multi-scale theory and Discrete Hidden Markov Models (DHMMs). We use a hierarchical decision tree based on multi-scale theory. Firstly we convolve all members of the training data with a Gaussian kernel, which blurs differences between images and reduces their separation in feature space. This reduces the number of eigenvectors needed to describe the data. A principal component space is computed from the convolved data. We divide the data in this space into several clusters using the *k*-means algorithm. Then the level of blurring is reduced and PCA is applied to each of the clusters separately. A new principal component space is formed from each cluster. Each of these spaces is then divided into clusters and the process is repeated. We thus produce a tree of principal component spaces where each level of the tree represents a different degree of blurring. The search time is then proportional to the depth of the tree, which makes it possible to search hundreds of gestures with very little computational cost. The output of the decision tree is then input into the DHMM recogniser to recognise temporal information.

Chapter 1 Introduction

In the past decade, the computational power of computers has doubled every eighteen months, while the human computer interface (HCI) has not changed too much. When we work with a computer, we are constrained by intermediary devices: keyboards and mice. However, these are inconvenient and have become a bottleneck in human-computer interaction especially in a three-dimensional (3D) world. For example, in virtual environments, keyboards, mice, wands and joysticks are still the most popular devices. They work in a two-dimensional world and are not suitable for 3D navigation. In our daily life, we use speech to communicate with each other, and use gestures to point, emphasise and navigate. They are the more natural and preferable means to interact with computers for human beings. To make computers understand this, however, is not an easy job. Apart from HCI, speech and gesture recognition can also be applied to many other fields, such as Sign language translation, industrial robot control and teleconferencing *etc.* Speech recognition has been thoroughly studied by researchers and many commercial products are available on the market.

This thesis is a study of recognition of human hand gestures. It is worth noting that although we only concentrate on hand gesture recognition, other parts of human bodies are also very valuable for correctly understanding gestures. In fact, many gestures involve co-operation of different parts of the whole body.

1.1 Problem Description

We are interested in recognising human hand gestures using computer vision principles. One might ask what is meant by gesture? Gestures are usually understood as hand and body movement that can pass information from one person to another. In *Webster's* dictionary, we can find:

> "A gesture is a movement usually of the body or limbs that expresses or emphasises an idea, sentiment, or attitude."

Since we only consider hand gestures, the movement of the hand that expresses or emphasises an idea, sentiment, or attitude belongs to a gesture. This is a rather broad and general definition and it will make the system performance test very hard. We need to divide the whole set of gestures into more detailed subgroups. Wu and Huang [1,2] state that hand gestures can be classified into four categories according to the different application scenarios: conversational gestures, controlling gestures, manipulative gestures and communicative gestures. We use conversational gestures to help express ourselves more clearly in our daily life. They are very subtle and need careful psychological studies [1,2]. Controlling gestures can be designed to navigate in a virtual environment. For example, we can ask the computer to drive a car to the south by pointing in that direction. Manipulative gestures serves as a natural way to interact with virtual objects [3]. A famous example is the seminal "put-that-there" work by Bolt [4,5]. Sign language is an important case of communicative gesture. Deaf people rely on it to talk to each other. It is objective and well defined and rarely causes ambiguity which makes it suitable as a test-bed for gesture recognition systems. We will discuss it in more detail in the next section.

Many recognition systems are based on the dataglove, an expensive wired electronic device. Various sensors are placed on the glove to detect the global position and relative configurations of the hand. However, even without considering the price of such a glove (hundreds of dollars in general), this is unnatural and uncomfortable for users since it needs a physical link between the user and the computer. Because of this shortcoming, more and more research groups show interest in vision-based systems, which are wire-less and the only thing needed is one or multiple cameras. The user then has more freedom and feels more comfortable. In this thesis, we present a vision-based system using one camera.

Speech recognition has been thoroughly studied by researchers and many commercial products are available on the market [16, 59]. In comparison to speech recognition, gesture recognition is a more sophisticated task mainly because of the immense variation of gestures. In general, to extract a voice from its background noise is not a difficult task. While for a computer vision system, things are much harder. Different illumination conditions and different background can make the same object look

extremely different. Even if we ignore the variability of lighting, shading and complex scenes containing cluttered background, human beings still produce considerably differing views depending on what they are performing and their position.

1.2 Irish Sign Language (ISL)

Sign language is a formal language employing a system of hand gestures for communication by the Deaf. As the Deaf are the ultimate recipients of these signs, great care was taken to devise appropriate signs for specific words in the first place. This explains why Sign language is highly structural and barely causes ambiguity.

There are different Sign languages for Deaf people from different countries, such as American Sign Language (ASL), Chinese Sign Language (CSL), British Sign Language (BSL), Irish Sign Language (ISL), and so on. We use ISL as a performance test-bed for our system. While there is nearly no problem for Irish people communicating with American people in English, ISL is totally different from ASL so that the communication between Irish Deaf people and American Deaf people is just as hard as the communication between a non-native English speaker and a native speaker. Helping Deaf people from all over the world talk to each other is one of the motivations of our research.

ISL is composed of two types of gestures: static gestures and dynamic gestures. Static gestures are characterised by their different hand shapes. The performer usually puts his/her hand in front of the chest. Each specific hand shape would represent a special meaning. ISL contains 26 gestures corresponding to the 26 English letters of the alphabet. 23 of them are static gestures, while the other 3 have to be expressed by dynamic gestures. Figure 1.1 shows some static gestures:

Figure 1.1 Three static gesture examples of ISL. From left to right: A, C, F.

Dynamic gestures are more sophisticated. Traditionally, there are three major points of an ISL dynamic sign:

- The shape or configuration of the hand.
- The relative position of the hand against other parts of body.
- Motion of the hand.

A typical description of a dynamic gesture in ISL is like this: right hand in "A" position, palm to self, move it down from lip to chin. Comparing to static gestures, which only contains spatial characteristic, dynamic gestures include both spatial and temporal characteristics. Intuitively, the latter makes it harder to recognise dynamic gestures than static ones. Figure 1.2 shows two examples that represent "J" and "Z".

Figure 1.2: Two dynamic gesture examples of ISL. From left to right: J and Z respectively.

1.3 Outline of the Thesis

In this thesis, we concentrate our study on human hands and ignore other parts of the body although these are also very important for understanding gestures. We show how Principal Component Analysis, Multi-scale theory and Hidden Markov Models are integrated into a whole system to recognise both static gestures and dynamic gestures. Gestures from ISL will be used to evaluate the system.

The remainder of the thesis is divided into seven parts. Chapter 2 gives the theoretical background knowledge used in the thesis. Chapter 3 discusses the application of gesture recognition systems and reviews previous work. Chapter 4 introduces the aims of the system. A few assumptions will be given to restrict our research to a reasonable range. The system framework is given as well. Chapter 5 deals with the pre-processing procedure and introduces a standard PCA-based system. Various classifiers will be given. 23 static gestures in ISL corresponding to 23 English letters of the alphabet will

be used for the purpose of evaluation. These gestures are referred to as *finger-spelling* gestures. Chapter 6 extends this work and introduces a novel method that combines a hierarchical decision tree with multi-scale theory to speed up the recognition procedure. In chapter 7, we discuss how to recognise dynamic gestures by combining the hierarchical decision tree and DHMMs. A set of 35 dynamic gestures taken from ISL are used to test the system performance. Finally we draw some conclusions and discuss possible future work in chapter 8.

Chapter 2 Definitions and Descriptions

In this chapter, we give the definitions and notations which will be used throughout the rest of the thesis. Meanwhile we describe the theoretical backgrounds of the major approaches that are employed in our system.

2.1 Basic Definitions in Pattern Recognition

Like face recognition, which is inherently a classification problem in a high dimensional feature space [7], we also treat the recognition of hand gestures as a problem in the field of pattern recognition, and indeed, many techniques we employ in the thesis fall into the class of statistical pattern recognition. Let's first define some basic concepts that are commonly used in this area.

Given an object to be classified [8],

- *features* are the measurements or properties that are used to classify the object,
- the types or categories into which they are classified are called *classes*,
- The individual objects to be classified will be referred as *samples*.

Often there is a set of samples used to design the system. The set of samples is called the *training set*. In addition, a data set called the *test set* must also be provided for testing the system [8].

In most cases l features x_i, $i=1,2,\ldots,l$, are used to form the *feature vector* [6]

$$\mathbf{x} = [x_1, x_2, \ldots x_l]^T$$

where T denotes transposition. If we treat each feature as a dimension, l features constitute an l-dimensional space, called the *feature space*. Each feature vector corresponds to a single point in such a space. Since each of the feature vectors identifies uniquely a single object, each point in the feature space also represents a unique object.

In our research, an object is a gesture to be recognised. Given a training set, we attempt to define a set of features that are able to separate the gestures into several

clusters in the feature space, with each cluster representing a class. Ideally, there should be no overlapping between the clusters.

2.2 Principal Component Analysis (PCA)

Normally the dimensionality of the feature space is high, for example, in image processing, the dimensionality can be as high as 320*240, or even higher. However, the clusters within the space will often lie on a low-dimensional subspace because of correlations between the features. We need a technique which will find the dimensionality of the subspace.

PCA is a rather general statistical technique that can be used to reduce the dimensionality of the feature space. The idea behind PCA is quite old. Pearson [84] first introduced it in 1901 as linear regression. Hotelling [84] then proposed it for the purpose of revealing the correlation structures behind many random variables. During the 1940s, Karhunen and Loeve independently extended it into a continuous version by using single-parameter functions replacing vectors now called the K-L Transform (KLT). In this section, we explain the theoretical details of PCA.

Given a set of training objects represented by their feature vectors, $\mathbf{x}_i$ ($1<i<M$), where M is the number of samples, the training set can be written in the format of $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2..., \mathbf{x}_M\}$ whose mean vector is $\mathbf{x}_m$ and covariance matrix is $\mathbf{R}_x$, defined by the following equations:

$$\mathbf{x}_m = \frac{1}{M}\sum_{i=1}^{M}\mathbf{x}_i$$
$$\mathbf{R}_x = \frac{1}{M}\sum_{i=1}^{M}(\mathbf{x}_i - \mathbf{x}_m)(\mathbf{x}_i - \mathbf{x}_m)^T \qquad ...(2.1)$$

The mean vector is a column vector and the covariance matrix is a real symmetric square matrix of size N by N, where N is the length of the feature vector. T defines the transpose of a matrix.

The training set $\mathbf{X}$ corresponds to a cluster of data points in an N dimensional feature space. There exists redundancy in the feature space since the features, i.e. the

dimensions, are not independent of each other. By PCA we can eliminate the redundancy by transforming the original feature space into a so-called *PC space* in terms of *Principal Components* (PCs). The transformation is an orthogonal, linear procedure, formalised in the following way:

$$\mathbf{y} = \mathbf{W}^T\mathbf{x} \qquad \ldots(2.2)$$

where $\mathbf{y}$ is the new feature vector in the PC space, and $\mathbf{W}$ is defined as follows:

$$\mathbf{W} = [\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_N]$$

$\mathbf{e}_i$ is the *i*th *Principal Component* (PC), or the orthonormal basis, of the feature space. All PCs have to be normalised, that is, $\mathbf{e}_i{}^T\mathbf{e}_i = \mathbf{I}$. The dimensionality of the PC space is still *N*, the same as that of the original feature space. However, the correlation between the features disappears. Thus by the transformation we map a feature vector from the original feature space into the PC space as a single isolated point where the basis are independent from each other. See Figure 2.1.

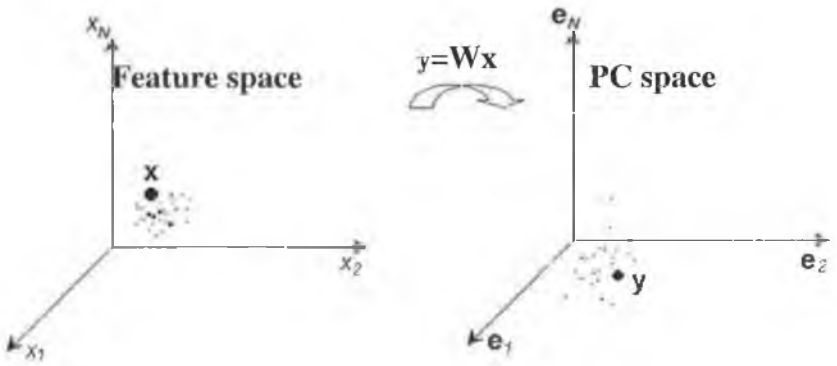


Figure 2.1: The transformation from the feature space into the PC space. Only 3 dimensions are shown here for the sake of visualisation convenience.

No information is lost during this transformation. The original feature vector can be fully reconstructed using the transpose of the PCs:

$$\mathbf{x}' = \mathbf{W}\mathbf{y} = \mathbf{W}\mathbf{W}^T\mathbf{x} \qquad \ldots(2.3)$$

where $\mathbf{x}'$ is the reconstructed image. Since there is no loss of information, $\mathbf{x}' = \mathbf{x}$.

The question is now how to find the PCs? Various methods have been developed. For example simple neural network architectures have been suggested for recursively estimating the subsets of the PCs [83]. However, the most widely used method is to

take advantage of the covariance matrix of the feature values of the training set and solve the *eigenvalue decomposition problem*.

Given the mean vector $\mathbf{x_m}$ and the covariance matrix $\mathbf{R_x}$ of a set of training samples, the eigenvalue decomposition problem is to find the solution for the following equation:

$$\mathbf{R}_x\mathbf{e}_i = \lambda_i\mathbf{e}_i \qquad \ldots(2.4)$$

where λ_i is the *eigenvalue* corresponding to the *i*th PC. This solution can be found by solving its characteristic equation; details are shown in [84]. Since $\mathbf{R_x}$ is positive semidefinite, all eigenvalues would be non-negative real numbers.

For an *N* by *N* covariance matrix, *N* eigenvalues and *N* PCs exist. In the PC space, the data lies on a low dimensional subspace because of the correlation between the features, i.e. the data variation focuses only on some of the dimensions while the variation along the remaining dimensions is almost zero. Given the fact that each eigenvalue represents the data variation along its PC, those eigenvalues corresponding to the small data variation are close to zero. See Figure 2.2.

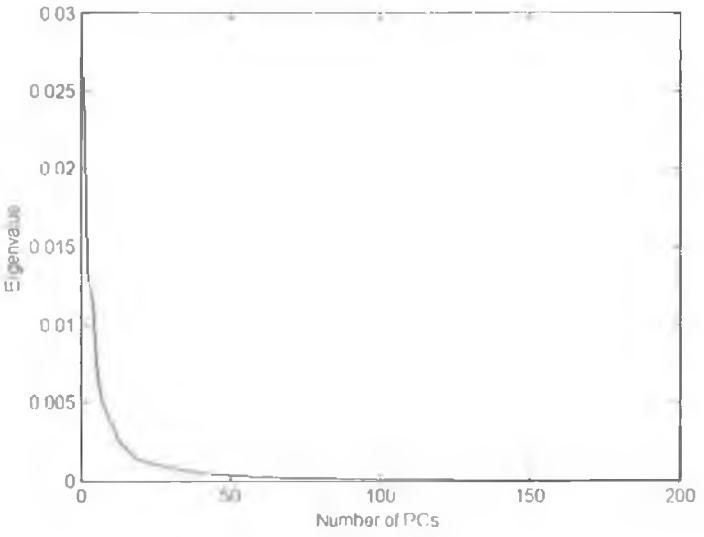


Figure 2.2: An example of a plot of eigenvalues, extracted from a data set of 640 images of the human hand. For ease of visualisation, only the first 200 out of a total of 1024 eigenvalues are shown.

If we arrange the eigenvalues in decreasing order, **y** can be represented in a compact way by maintaining only the PCs corresponding to the first few largest eigenvalues while ignoring the rest. That is, the feature vector **y** is computed in the following reduced way:

$$\mathbf{y} = \mathbf{W_C}^T \mathbf{x} = [\mathbf{e}_1, ..., \mathbf{e}_K]^T \mathbf{x} \qquad ...(2.5)$$

where $\mathbf{W_C}$ is the matrix that contains only the first K PCs. K is the number of PCs that are retained and $\mathbf{e}_1$ corresponds to the largest eigenvalue, $\mathbf{e}_2$ the second largest eigenvalue, and so on. Usually $K << N$, and the dimensionality is reduced from N to K. The reconstruction of the original feature vector is given by:

$$\mathbf{x}' = \mathbf{W_C}\mathbf{y} = [\mathbf{e}_1, ..., \mathbf{e}_K][\mathbf{e}_1, ..., \mathbf{e}_K]^T \mathbf{x} \qquad ...(2.6)$$

However, at this stage, the reconstructed vector is different from the original one: $\mathbf{x}' \neq \mathbf{x}$. To measure the difference, we define the energy of the training set in Equation 2.7:

$$E = \sum_{i=1}^{N} \lambda_i \qquad ...(2.7)$$

The error between the reconstructed vector $\mathbf{x}'$ and the original feature vector $\mathbf{x}$ can be measured using the criterion of percentage of total energy remaining. We can write the percentage of the remaining energy as a function of the number of the retained PCs:

$$E_{remain}(K) = \sum_{i=1}^{K} \lambda_i \Big/ E \qquad ...(2.8)$$

PCA minimises the mean square reconstruction error of the training set [84]:

$$\begin{aligned} \varepsilon &= \min\left[\sum_{i=1}^{M} \| \mathbf{x}_i - \mathbf{x}_i' \|\right] = \sum_{i=K+1}^{N} \lambda_i \\ &= E - \sum_{i=1}^{K} \lambda_i \qquad ...(2.9) \\ &= E(1 - E_{remain}(K)) \end{aligned}$$

The Equation 2.9 tells that the greater K, the smaller the reconstruction error, and the more PCs retained the more energy remaining.

In essence, PCA assumes the projection of the training samples in the PC space is bounded by a hyperellipsoid. λ_i is the variance of the data along the ith PC – more

than 98% of distribution of the data is covered in the range $[-3\sqrt{\lambda_i}, +3\sqrt{\lambda_i}]$ along this PC, as depicted in Figure 2.3.

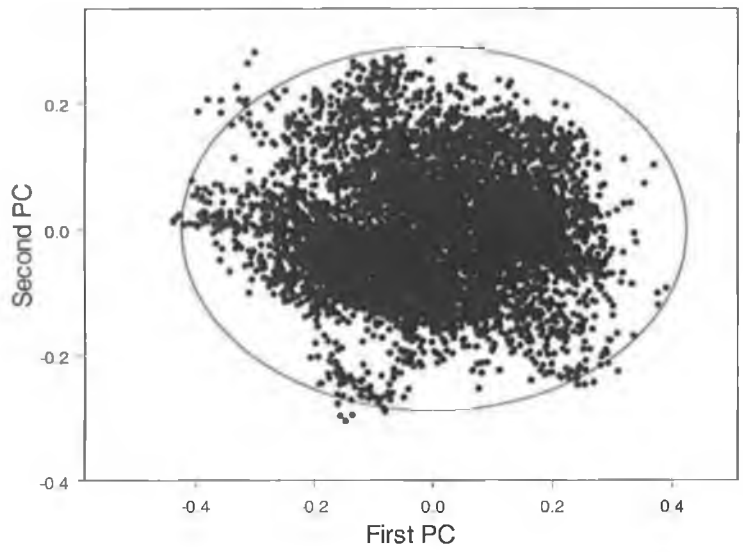


Figure 2.3: Bounding ellipse of data cloud along the directions of the first two PCs. The long axis and short axis of the ellipse are $3\sqrt{\lambda_1}$ and $3\sqrt{\lambda_2}$ respectively.

2.3 Multi-scale Theory

Multi-scale theory is a complementary technique for dimensionality reduction [79]. Let us consider a two-dimensional image I. In scale space theory, I is represented by a family of smoother versions of I: $I(\sigma)$, where $I(0)$ corresponds to the original image, and as the value of σ increases, $I(\sigma)$ becomes smoother and smoother, i.e., the greater the scale parameter σ, the more details in the original image are eroded. This procedure can be formalised by:

$$I(x, y, \sigma) = I_0 * G(x, y, \sigma) \qquad \ldots (2.10)$$

where * denotes a convolution, I_0 represents the original image, σ is the scale parameter which is always non-negative, and $G(x,y,\sigma)$ stands for a two-dimensional Gaussian kernel defined as:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \qquad \ldots (2.11)$$

The use of different smoothing kernels other than Gaussian is possible. However, the Gaussian kernel's remarkable mathematical properties make it the first choice for any multi-scale analysis. The most important one is that the Gaussian kernel will not create extra structures when convolved with an image [79], which is crucial for the system's stability since a fake structure could attract the system's attention from the real useful structures.

Figure 2.4 shows a family of smoother version of an image under different scale parameters. Image (1) is the original image that represents the hand shape of "d" in ISL. From image (1) to (6), the scale parameter σ increases gradually and as can be seen, the image details are eroded and fewer details can be recognised. In the extreme situation, when σ becomes very large (see the image (6)), the image has been totally washed out and the gesture is not recognisable any more.

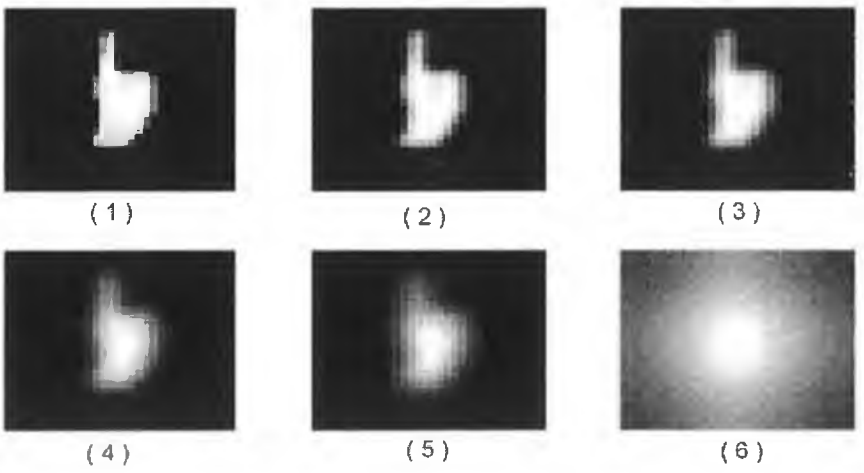


Figure 2.4: A family of smoother versions of the image that represents the shape "d" in ISL. (1) is the original shape. From (1) to (6) the scale parameters are 0, 0.5, 1.0, 1.5, 2.0, 10 respectively.

Convolution of an image with a Gaussian kernel is a time-consuming operation in the spatial domain. Convolution theory tells us that a convolution operation in the space domain can be computed as a multiplication in the frequency domain. Thus we have

$$I(x, y, \sigma) = I_0 * G(x, y, \sigma) = F^{-1}(F(I_0) \bullet F(G(x, y, \sigma)) \qquad \ldots(2.12)$$

where $\bullet$ denotes a multiplication operation. F denotes a two-dimensional Fourier transform:

$$F(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I_0(x,y)e^{-i2\pi(ux+vy)}dxdy \quad \ldots(2.13)$$

and F^{-1} a two-dimensional inverse Fourier transform:

$$I(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(u,v)e^{i2\pi(ux+vy)}dudv \quad \ldots(2.14)$$

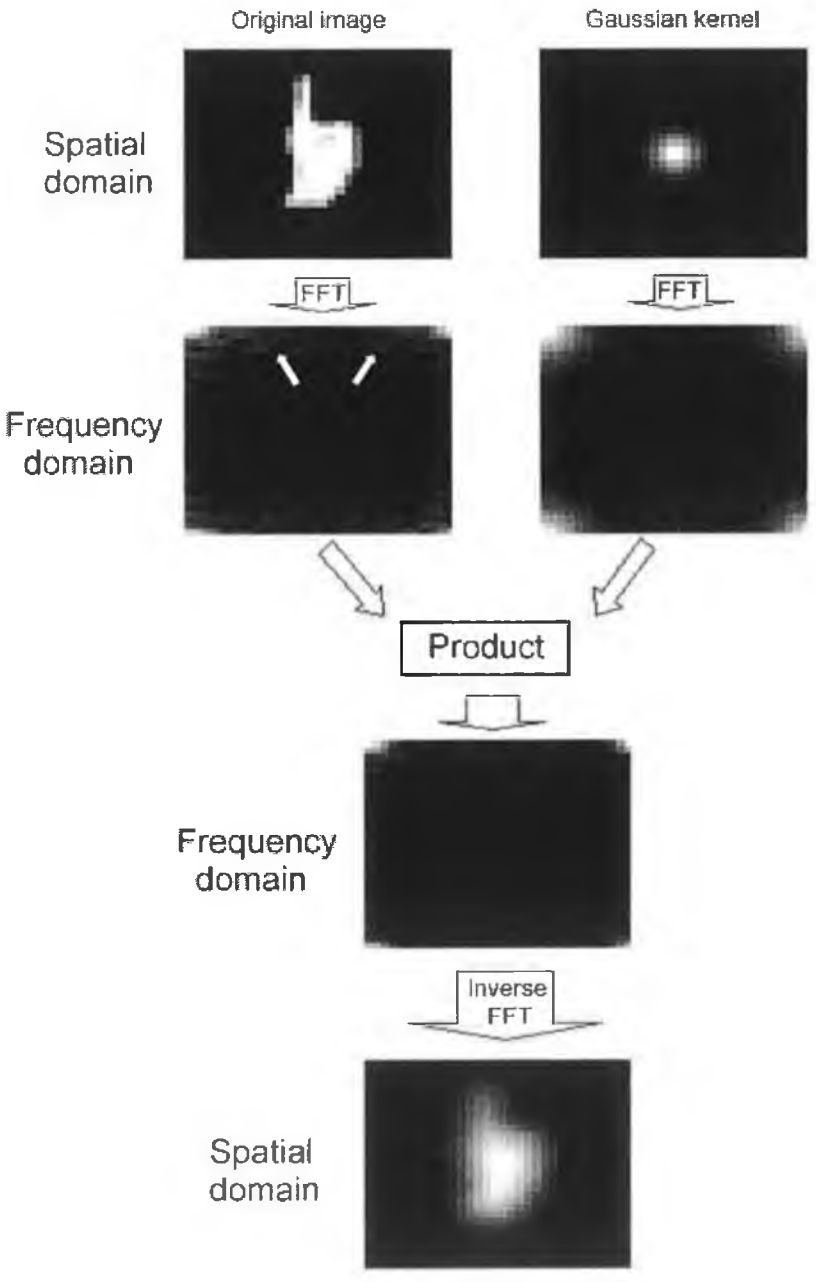


Figure 2.5: Blurring an image in frequency domain. The scale parameter of the Gaussian kernel is 2.0.

When the Fast Fourier Transform (FFT) is used to speed up the transform from the time domain to the frequency domain and its inverse, the most computationally expensive operation in Equation 2.12 is the two-dimensional FFT. Consequently, the complexity of Equation 2.12 is bounded by this operation into $O(n^2(log_2\ n)^2)$ when applying the convolution to an $n \times n$ image with an $n \times n$ kernel, while the complexity would be $O(n^4)$ without using FFT.

A diagram of the above algorithm is given in Figure 2.5, where the original image and a Gaussian kernel are first transformed from the spatial domain into the frequency domain. In the pictures which visualise the 2-D frequency domain, the frequency varies from low to high from the corners to the centre. The original image thus contains some high frequencies (pointed to by the arrows in the frequency domain picture of the original image), which correspond to the fine structures as well as the noise in the original image. After the product between the image and the Gaussian kernel is computed in the frequency domain, the high frequencies disappear, hence some fine structures and the noise are eliminated. This effect can be seen in the spatial domain: the image is the blurred version of the original one.

2.4 Hidden Markov Models (HMMs)

In this section, we will briefly introduce the basic ideas of HMMs. We only discuss the discrete HMMs (DHMMs) here since it is the technique used in our system. A large number of tutorials about the topic have been published. Good ones include [64], [65] and [74].

HMMs separate a sequence of observations into several segments, the so-called "states". In each state, the observations are relatively stable, and thus can be simulated using probabilistic models: Gaussian, Mixed Gaussian, and so on. Their power mainly lies in two aspects: first, they attempt to simulate what state the system is in "now" according to the jth most recent states – the Markov property. If the status of the system purely depends on the most recent state, then the HMM is first-order. Second, an HMM makes transitions from one state to another stochastically, and generates observations from the current state stochastically – the doubly stochastic property. The

transition depends on the so-called *state-transition probability*, and the generation of the observation is controlled by the *observation symbol probability*.

It is the Markov property that simulates but simplifies the events in reality, especially when first-order HMMs are used. Higher order systems are possible, but their training and decoding can be a huge problem and certainly will be much slower than the first-order systems. Our system uses a first-order HMM to encode the dynamic gestures. On the other hand, it is the doubly stochastic property that enables the application of HMMs in many real problems because traditional Markov models can only model stationary processes [64].

Before proceeding to more details, we first give some notation:

- s is a state of the HMM. State i at time t is denoted as $s_t = i$.
- o is an observation symbol. o_t stands for the symbol at time t.
- v is the alphabet, i.e. the set of observation symbols, which is a non-empty finite set $\{ v_1, v_2, \ldots v_k\}$, where k is the size of the alphabet.
- a is the state-transition probability. Specifically, a_{ij} is the probability of the transition from state i to j.
- b is the observation symbol probability. Specifically, $b_j(k)$ represents the probability of generating a certain observation symbol v_k in state j.

A DHMM is characterised by a quintuple: $\Lambda=(N, M, A, B, \pi)$ where

- N is the number of states in the model. In practice, there will always be two special states being used as *start* and *end* states. Thus $N \geq 2$.
- M is the number of distinctive observations in the model, i.e. the discrete alphabet size. $M > 0$.
- A is the state-transition probability distribution. $A=\{a_{ij}\}$ where

$$a_{ij} = P[s_{t+1} = j \mid s_t = i], \qquad 1 \leq i,j \leq N \qquad \ldots(2.15)$$

  More specifically, when the DHMM is first-order, a_{ij} is larger than 0 only when $j=i$ or $j=i+1$.
- B is the observation symbol probability distribution. $B = \{b_j(k)\}$ where $b_j(k)$ is the observation symbol probability and

$$b_j(k) = P[o_t = v_k \mid s_t = j], \qquad 1 \le k \le M,\ 1 \le j \le N \qquad \ldots(2.16)$$

- π is the initial state distribution. $\pi = \{\pi_i\}$ in which

$$\pi_i = P[s_1 = i] \qquad \ldots(2.17)$$

In practice, although the above 5 items are required to specify a complete DHMM, people tend to use the following compact notation, ignoring the two parameters N and M in the representation since they are only two constants and will always be fixed beforehand:

$$\Lambda = (A, B, \pi) \qquad \ldots(2.18)$$

A very simple example of a first-order DHMM with seven states including the start and end states is given in Figure 2.6. Note that the more states the model has, the more powerful the model is. However, as the complexity of the model increases exponentially with respect to the number of sates, larger training set is required for reliable model estimation [4].

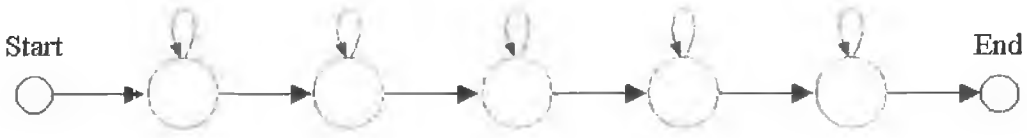


Figure 2.6: A sample DHMM topology with seven states including the *start* and *end* states.

Given the above form of DHMMs, three basic problems have to be answered [65]. These are:

Question 1:

Given the model $\Lambda = (A, B, \pi)$, and the observation sequence $O = (o_1, o_2, \ldots, o_T)$, where T is the length of the observation sequence, how do we compute the probability of occurrence of O, $P(O|\Lambda)$?

Question 2:

How do we adjust the model parameters $\Lambda = (A, B, \pi)$ so that the likelihood function $P(O|\Lambda)$ is maximised?

Question 3:

Given the model $\Lambda = (A, B, \pi)$, how do we choose a state sequence $q = (q_1, q_2, \ldots q_T)$ so that $P(O, q|\Lambda)$, the joint probability of the observation sequence $O = (o_1, o_2, \ldots, o_T)$ and the state sequence is maximised?

These three questions are called the *evaluation* problem, the *estimation* problem, and the *decoding* problem [92]. We introduce the solutions for them briefly.

2.4.1 Solution to Question 1 – the Evaluation Problem

First, let's compute the likelihood $P(O|\Lambda)$ given an observation sequence $O = (o_1, o_2, \ldots, o_T)$ according to the model Λ with the parameters A, B, and π. A number of methods have been developed to solve the problem. Two of them are important, namely the forward algorithm and the backward algorithm, and are described below.

The forward algorithm

Given the observation sequence $O = (o_1, o_2, \ldots, o_T)$, we define the forward variable $\alpha_t(i)$ as the probability of the joint event that $o_1\ o_2 \ldots\ o_t$ are observed, and the state at time t is i. It can be formalised as follows:

$$\alpha_t(i) = P(o_1 o_2 \ldots o_t, q_t = i \mid \Lambda) \qquad \ldots(2.19)$$

When $t = 0$, the system begins from the start state, and $\alpha_0(start) = 1$ by default. When $t = T$, the system stops at the end state, and the forward variable is $\alpha_T(end)$, which is exactly the likelihood $P(O|\Lambda)$ we are trying to work out. In practice, we first initialise the forward variable with $\alpha_0(start) = 1$, and then $\alpha_t(i)$ can be computed recursively by simple induction [65]:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(o_{t+1}), \qquad 1 \le t \le T\text{-}1, \quad 1 \le j \le N \qquad \ldots(2.20)$$

The time complexity of the above procedure is $O(N^2T)$.

The backward algorithm

The backward algorithm is the reverse of the forward algorithm. Given the observation sequence $O = (o_1, o_2,\ldots, o_T)$, we define the backward variable $\beta_t(i)$ as the conditional probability of the partial observation sequence from $t+1$ to the end $(o_{t+1}\ o_{t+2}\ldots\ o_T)$ given state i at time t and the model Λ. It can be formalised as follows:

$$\beta_t(i) = P(o_{t+1}o_{t+2}...o_T \mid q_t = i, \Lambda) \qquad \ldots(2.21)$$

Similarly, the backward variable can be calculated inductively: we initialise it with $\beta_T(end) = 1$, and the recursive step is given by [65]

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1})\beta_{t+1}(j), \quad t = T\text{-}1, T\text{-}2,\ldots,1 \qquad \ldots(2.22)$$

$$1 \le i \le N$$

This time the likelihood $P(O|\Lambda)$ is given by $\beta_0(start)$. Again, the time complexity of the above procedure is $O(N^2T)$.

2.4.2 Solution to Question 2 – the Estimation Problem

To solve the estimation problem requires an adjustment of the parameters of the model according to a set of training data. The result of the adjustment should maximise the likelihood $P(O|\Lambda)$. An iterative procedure, well known as the Baum-Welch method (essentially an expectation-maximisation method [65]), has been developed for this task. In this section, we describe the estimation procedure.

Firstly, with the aid of the forward and backward variables, we can define a new variable $\gamma_t(i)$, the probability of being in state i at time t, given the entire observation sequence $O = (o_1, o_2,\ldots, o_T)$:

$$\gamma_t(i) = P(q_t = i \mid O, \Lambda) = \frac{\alpha_t(i)\beta_t(i)}{P(O \mid \Lambda)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)} \qquad \ldots(2.23)$$

Since both the forward and backward variables are computable, $\gamma_t(i)$ is easy to calculate as well.

Secondly, to fully describe the estimation procedure, we need to define another variable $\xi_t(i,j)$, given by the following equation [65]:

$$\xi_t(i,j) = P(q_t = i, q_{t+1} = j \mid O, \Lambda) = \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(i)}{P(O \mid \Lambda)}$$

$$= \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(i)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)} \quad \ldots(2.24)$$

Since $\gamma_t(i)$ is the probability of being in state i at time t, the sum of $\gamma_t(i)$ over t would be the number of times state i is visited. Since $\xi_t(i,j)$ represents the probability of being in state i at time t and being in state j at time $t+1$, we can estimate the initial state distribution π_j and the state-transition probability between state i and j by:

$$\bar{\pi}_i = \gamma_1(i) \quad \ldots(2.25)$$

$$\bar{a}_{ij} = \sum_{t=1}^{T-1} \xi_t(i,j) \Big/ \sum_{t=1}^{T-1} \gamma_t(i) \quad \ldots(2.26)$$

and the observation symbol probability $b_i(k)$ is estimated by:

$$\bar{b}_i(k) = \sum_{\substack{t=1 \\ q_t, t, o_t = v_k}}^{T} \gamma_t(i) \Big/ \sum_{t=1}^{T} \gamma_t(i) \quad \ldots(2.27)$$

Obviously, based on the above procedure, the parameters of the model can be estimated inductively. It will maximise the likelihood $P(O|\Lambda)$. Methods other than Baum-Welch method are also possible, such as Gradient Descent and Viterbi Learning [65, 74].

2.4.3 Solution to Question 3 – the Decoding Problem

The decoding problem is to determine the best state sequence to describe the given observation sequence. The well-known Viterbi algorithm can be used for this purpose. For the Viterbi algorithm, we introduce the new variable $\delta_t(i)$, the highest probability along a path given the first t observation symbols ($o_1\ o_2 \ldots\ o_t$) and path ends in state i.

$$\delta_t(i) = \max_{q_1, q_2, \ldots, q_{t-1}} P[q_1 q_2 \ldots q_{t-1}, q_t = i, o_1 o_2 \ldots o_t \mid \Lambda] \quad \ldots(2.28)$$

The Viterbi algorithm calculates $\delta_t(i)$ inductively to find out the most probable path given the observation sequence O and the model Λ:

Initialisation: $\delta_1(i) = \pi_i b_i(o_1), \quad 1 \le i \le N$

Induction: $\delta_{t+1}(j) = [\max_{1 \le i \le N} \delta_t(i) a_{ij}] b_j(o_{t+1}), \quad 1 \le t \le T\text{-}1, 1 \le j \le N$

Termination: $P(O \mid \Lambda) \approx \max_{1 \le i \le N} [\delta_T(i)]$

If we keep recording the result of $\delta_t(i)$ for every step during the induction, we would be able to trace back the state having the highest probability at each step, and this state chain would be the best state sequence. The time complexity of the algorithm is $O(N^2T)$. Note, that in practice, to speed up the procedure, researchers tend to use the probability of the best state sequence to approximate the likelihood $P(O|\Lambda)$, instead of using the sum of the probabilities of all possible state sequences, hence the "≈" in the termination.

2.5 Summary

In this chapter, we introduced some basic concepts commonly used in the area of pattern recognition. We described the theoretical background of PCA, including how to transform the original feature space into the PC space, and how to measure the loss of information during the procedure. We introduced the multi-scale theory, where an image is embedded in a family of smoother versions of the original image. The greater the scale parameter, the more details in the original image are eroded. In the section of DHMM, we gave its definition, and discussed the solutions to three important problems.

Chapter 3 Literature Review

"Good bye keyboard, so long mouse. Hello smart rooms and clothes that recognise acquaintances, understand speech, and communicate by gesture. And that's just the beginning... "

---*Pentland* [31]

A lot of effort has been put into hand gesture recognition [1, 2, 3, 4, 30]. This work shows the great potential of gesture recognition in many areas [1, 9, 10], such as virtual environments, advanced user interfaces, Sign language recognisers, and so on. Hand gesture recognition falls into the general field of *object recognition*, which includes the following applications, such as hand-written character recognition [43], vehicle detection [44], military target recognition [45], and so on. An important sub-field is human motion recognition, which includes face [80, 81, 82], lips [38], eyes [58], and whole body motion recognition [24, 54]. A number of promising applications for human motion recognition have been developed; smart surveillance for example [32, 33]. Many techniques developed for these applications also can be used for hand gesture recognition. In this thesis, we are specifically interested in hand gesture recognition. Hence we only discuss the research in which hand gestures play a major role.

In hand gesture recognition, an important concept is the degree of freedom (DOF), which measures the flexibility of the human hand. When the hand configuration is not so important, for example in human motion recognition, the movements of the hand fingers are ignored (only the location of the whole hand is considered). The hand shows in total 3 degree of freedoms (DOFs) in a 2D image. That is, the hand can change in x and y translation, rotate in the 2D image plane. However, researchers are usually interested in the details of human hands when recognising hand configurations. In this case, the hand is treated as a non-rigid object, and there are 27 DOFs corresponding to the 27 bones in a hand, see Figure 3.3.

This chapter will discuss the previous work and the state of the art in hand gesture recognition. We will review the work according to the hand models used. The reasons are, first, most of the systems are model-based systems, and second, feature extraction and analysis in hand gesture recognition are tightly related to the specific model. In the remainder of the chapter, we first introduce possible applications of hand gesture recognition. Then we discuss different systems according to the spatial and temporal hand modelling techniques they use. Finally, we have the summary.

3.1 Application Systems

The potential applications of human gesture recognition are the driving force of the research. We consider two main application areas: advanced user interfaces and Sign language translation.

An important application domain is the advanced user interface or Multimodal Human Computer Interface [18, 19, 20]. The development of User Interfaces (UI) accompanied the evolution of computing technologies. From the original Text-based User Interface to the current Graphical User Interface (GUI), people have always been limited to a two-dimensional plane: either keyboards or mice. People are now seeking for a more natural and convenient way to interact with computers, for instance using speech and gestures. Due to the advance of speech recognition, a number of commercial products now allow humans to talk to computers directly [16, 59]. Hand gesture recognition is useful to complement speech recognition in a natural, especially high-noise, environment. On the one hand, hand gestures can replace the traditional mechanical devices, i.e. keyboards, mice, and so on, so that users can copy, delete or execute a program by their hand gestures instead of the normal click; (for example, a vision-based system, "FingerMouse", has been presented by Mysliwiec [23] that allows users to use their fingers to take the place of mice.) On the other hand, hand gesture recognition makes possible 3D human computer interaction, especially in a virtual environment. This used to be very hard when controlled by 2D input devices, or even by speech. Utsumi and Ohya [11] have designed a system where a user can create virtual 3D objects having predefined primitive shapes and change the objects' positions, sizes, colours, etc. with hand gestures. Lee[13] uses the user's hand as a 3D cursor in a virtual environment which makes the interaction between users and

computers easier. Lee also presents another gesture recognition system [14] that uses two sensors attached to the arm and allows users to interact with the computer by the movement of their arms. "DigitEyes" [15] is a model-based hand tracking system that can track the 27 DOFs of the configuration of the user's hand using video cameras, which has been applied to a 3D-mouse system.

Thanks to the boosting of computer power, people can integrate these state-of-the-art technologies into their daily life. "Smart Room" is one example [24]. A "Smart Room" is a room (or a building) with lots of sensors and cameras that respond to the words, movements and gestures of inhabitants. Residents can turn off the TV or lights by simply waving their hands, or tell the room they feel cold by rubbing both hands; the room will turn on the heater automatically. In the Artificial Intelligence Lab of MIT, Moy has a pet robot [25] that responds to the owner's gestures. The gesture lexicon includes linear (vertical, horizontal and diagonal) as well as circular (clockwise and counterclockwise) gestures. Roth *et al.* [54] in Mitsubishi Electric Research Laboratories apply gesture recognition theory to various systems: vision-based computer games, a hand signal recognition system, and a television set controlled by hand gestures. Some of these applications employ a special artificial retina chip for image detection or pre-processing.

Another important application for hand gesture recognition is in the field of Sign language translation. For many Deaf people, Sign language is the principal means of communication. However, due to the differences between Sign languages from different countries, it is very hard for the Deaf from different countries to understand each other. They may be confined in many of their interactions to communicate only with other Deaf people from their own country. Surely hand gesture recognition has a role to play here. One of the most famous systems is designed by Starner and Pentland [26], which can recognise sentence-level continuous American Sign Language (ASL) using a single camera. An Irish Sign language Recognition system [27, 28] has also been reported by the author, where both static and dynamic gestures can be recognised fast on an entry-level PC.

3.2 Previous Work

In this section, we review previous work from the technical point of view. Many systems have been presented previously. To describe them structurally, we need to first classify them into separate groups. A number of criteria exist that can be applied to categorise different systems, for example, the type of hand models used (2D-hand model vs. 3D-hand model), the number of cameras involved (multiple vs. single-view), dimensionality of the tracking space, and so on. Our review is based on two criteria; it distinguishes

- single-camera from multiple-camera (Section 3.2.1)
- 2D models from 3D models (Section 3.2.2).

3.2.1 Stereo vs. Single-view

When designing a vision-based system, the first question to ask might be how many cameras should be employed? One choice is to use two or more cameras so that the computer can "see" the hand from more than one angle simultaneously. This will provide extra robustness against some factors: (self-) occlusion and accuracy of hand models etc., but the price is the expense of computing resources. In general, stereo systems are expensive and slow and need many resources, which makes them not suitable for real-time purposes. Furthermore, even with one camera, in the case of occlusion, we can still recover 3D-hand structures given image sequences that contain multiple views of the same object. Many researchers have developed their systems based on a single camera [50, 51, 52, 61, 62, 63].

3.2.2 2D-hand model vs. 3D-hand model

In this section, we discuss the vision-based systems according to the hand models they use. Unless clearly stated, all systems discussed from now on are based on a single camera.

The models used up to now can be categorised into two classes: 2D-hand models, also known as appearance-based models, and 3D-hand models. These two classes of model are intrinsically different in the following way: given a training set, the parameters or

features of the 2D-hand model are extracted by analysing the raw image data. In other words, we do not know what the model looks like before. In contrast, when a 3D-hand model is used, we start with the model, and search for evidence from the raw data to support it. We can adapt the model to fit the data. This is essentially a top-down procedure and *a priori* knowledge can be embedded into the model beforehand that makes it often outperform the 2-D appearance-based method. 3D-hand models are computationally expensive because of the high computational complexity, and 2D-hand models largely depend on the quality of the training data.

3.2.2.1 2D-hand Model

A large variety of systems that have been presented in this area use a 2D-hand model. The systems using 2D hand models can be classified into two classes: image property-based models and statistical models.

Image property-based models employ some simple image features. Commonly used image features include geometrical features [26, 46], moment features (using the moments of the image as features) [26, 47, 48, 49, 50, 51, 53], local orientation histograms [52, 54], and local shape features [55].

Wavelets are a relatively new approach and also have become popular in hand gesture recognition in recent years. This technique reflects the image properties in the frequency domain. Oren *et al.* [57] perform non-rigid object detection in still images using the wavelets analysis method.

A large amount of work falls into the class of statistical models, which are based on PCA or its extensions. PCA was first applied in the computer vision community to face recognition by Sirovich and Kirby [17] and later extended by Turk and Pentland [95]. Birk *et al.* [67] have developed a system recognising 25 static gestures of International Sign language. They first perform PCA on sets of training images to generate a Bayes classifier, and then use the classifier in real time. Martin and Crowley [66] describe a system that is used to interact with a virtual world. They address the task of hand gesture recognition in a similar manner to Birk *et al.* They are

the first two research groups applying PCA to hand gesture recognition [36] although the two systems were developed independently.

Heap and Hogg [61] present a two-level hierarchy to extract the existing non-linearity from PC spaces. This approach consists of two main steps: first an initial global PCA is performed on the training data for dimensionality reduction. Second, a *k*-means cluster analysis is carried out to split the data in the global PC space into several clusters in terms of the Eulidean distance. A sub-region is then produced which is centred on the cluster mean, and bounded by a hyperellipsoid with a Mahalanobis radius of some specified value. The idea behind this approach is that a complex, non-linear region can be approximated by a combination of a number of smaller sub-regions, which are treated as linear spaces. Figure 3.1 (taken from [61]) demonstrates the procedure.

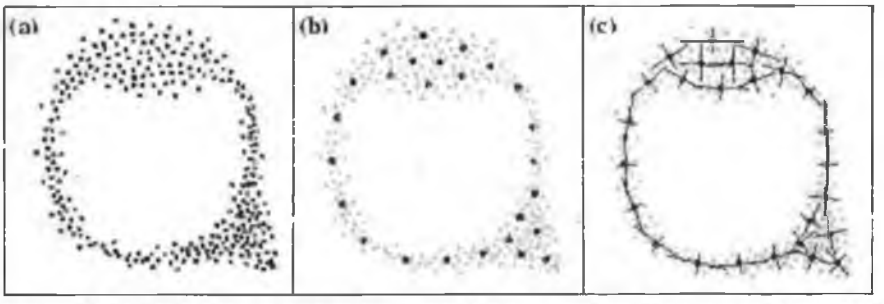


Figure 3.1: Splitting a complex, non-linear region into a number of linear sub-regions (taken from [61]); (a) training data projected into a 2D PC space using PCA, (b) finding out the centers of the sub-regions, (c) principal axes of the sub-regions, which shows that the non-linearity of the original trajectory has been reduced significantly.

Cui and Weng [37] present a general framework to recognise hand signs against a complex background. PCA is applied on the training set to reduce the dimensionality of the feature space. Then, Linear Discriminant Analysis (LDA) is used to automatically select the most discriminating PCs after dimensionality reduction. A space partition tree is used to achieve a logarithmic retrieval time complexity for a database of n items, and a general interpolation scheme to do view inference. The system was tested using 28 hand signs in front of the uniform background. The

experiment required the hand area being manually labelled, and hence is not suitable for a training database with a large number of samples.

Point Distribution Model (PDM), proposed by Cootes [41], is based on the PCA method. Sometimes, PDM is called the Active Shape Model or "smart snakes". In a PDM, objects are defined by landmark points that are placed in the same way on each of a set of examples from the training set. PCA is performed to estimate the mean shape and to find out the main "modes of variation" [41]. Each mode changes the shape by moving the landmarks along straight lines passing through their mean positions. A new image is then expressed as a linear combination of the modes of variation. PDM can represent non-rigid objects, which makes it very suitable for the task of hand gesture recognition, and has already been applied to the area of hand gesture recognition [60, 61, 62, 63].

Many extensions of PCA have been developed. Although these have not yet been applied in hand gesture recognition, these extensions have been used in the field of objection recognition, and can be easily employed for the recognition of human hands.

Probabilistic PCA (PPCA), an extension of standard PCA, has been proposed independently by Moghaddam and Pentland [39], Tipping and Bishop [38] and Roweis [34]. Unlike in standard PCA, where the less important PCs are simply discarded, in PPCA, these PCs are assumed to contain Gaussian noise. [34] and [38] present a noise model and derive an expectation-maximisation (EM) algorithm to estimate the parameters. Moghaddam and Pentland further apply PPCA for face recognition [40].

Independent Component Analysis (ICA) is another useful extension of PCA. In contrast to PCA, which de-correlates the features using the covariance matrix (2^{nd}-order statistics), ICA reduces higher-order statistical dependencies, attempting to make the features as independent as possible [35]. Recently, Moghaddam *et al.* use ICA to recognise objects in cluttered background [42]. Especially, their approach has the ability to model non-rigid objects [42], which makes it possible to extend the method for hand gesture recognition.

3.2.2.2 3D-hand Model

A 3D-hand model is an alternative to 2D-hand models. In this section, we discuss the work that aims to recover 3D articulated pose over time.

3D-hand Models can be classified into two large groups: volumetric models and skeleton-based models. Volumetric models describe the visual appearance of human hands and are often used in the field of computer animation. Because of the huge computational complexity, not many researchers use them to model hands for real-time purposes. However, it may be helpful to construct training databases for statistical models: the performance of statistical models depends on the size of training sets. The more variation within the training sets the better the performance the system can reach. However, constructing a large training database has always been a problem. Computer animation is a possible solution for this. Recently, John *et al.* [68] made an attempt to use computer animated hand shapes in ASL. Their representation of the hand consists of sixteen articulated rigid bodies, which represents the palm and the bones of the fingers and thumb, where the palm is modelled as a single rigid body. See Figure 3.2.

Fig. 3.2: Computer animation effects from [68]: from left to right: an open palm, the letter *D* in ASL, the letter *E* from ASL.

The figure shows that many visual features can be simulated well in this model in terms of both shape and texture, such as the fingers' shadow on the palm in the shapes of *D* and *E* in ASL. The main deficiency is just as the authors indicated in the paper: the absence of the metacarpal bones for the little finger and ring fingers makes it difficult to place the hand in certain configurations where the thumb is touching the little finger.

To reduce the computational complexity of volumetric models, some details in the model have to be eliminated. Often the texture of skin (not to mention the structure of muscles) is not computed. That is, only shape information is taken into account when recognising a gesture. This is reasonable because the deformation of the skin of the human hand does not convey any additional information needed to interpret gesture [3]. For example, humans have no problem to recognise gestures made by cartoon characters even though they are much simpler than real humans.

Some approaches treat the homogeneous hand parts by simpler parameterised geometric shapes, for example cylinders [71, 72]. In [71], Shimada *et al.* use such a model. In their system, first the pose is estimated roughly by silhouette matching; then the rough pose and the initial shape model are refined using some knowledge of human body's shape, pose and motion restrictions. However, because fingertips are used to construct the correspondences between the model and the images, the performance would depend on the accuracy of fingertip detection.

PDM can also be used for 3D modelling. Heap and Hogg [69] apply this idea to track hands with a full 6-DOF in real-time regardless of the hand configuration (10 frames/second on a standard 134MHz Silicon Graphic Indy workstation). A key issue for building their model is the collection of landmark coordinate data from training images. They tackle this problem by capturing training data semi-automatically using a simplex mesh introduced by Delingette [70]. However, this system does not work well when bending points exist.

For real-time purposes, skeleton-based models are often used for the current hand gesture recognition tasks. This type of model is based on anatomical knowledge and treats human hands as articulated objects consisting of a set of sub-objects where each sub-object is rigid. Thus, only the kinematic relations among these sub-objects have to be modelled. Each finger is demonstrated as a series of linked chains, shown in Figure 3.3.

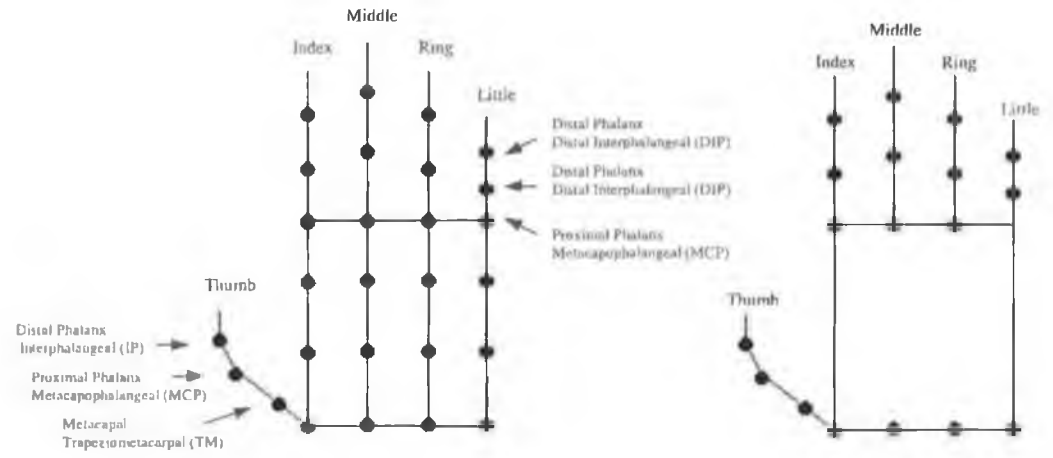


Fig. 3.3: Skeleton-based model of the human hand. The left one mimics all 27 bones of the hand. While the right one is a simplified version that approximates fingers more accurately since fingers have more flexibility than palm.

Although a hand contains 27 bones anatomically, most bones in the palm are not as flexible as in fingers. This gives us a clue that a skeleton-based model can be simplified by ignoring the bones and their joints in the palm. Thus, we have the simplified version of the skeleton-based model, which contains 19 links, to mimic most of the bones and joints and each finger is modelled as a kinematic chain with the palm as its base reference frame.

To estimate the 3D structure of hands precisely with a skeleton-based model, we have to solve the so-called "inverse kinematics problem". That is, to recover the angles of the joints. One way for solving this is to first construct a feature correspondence between 2D images and the 3D model. Once the correspondence has been established, joint angles can be estimated by minimising the distance between the projections from the 3D model and 2D images. Different systems take different image features for the correspondence. The performance of this type of approach greatly depends on the feature detection. If the predefined features can not be decided properly for some reason, for example self-occlusion, the joint angles would not be able to be recovered. Next we review some systems that uses skeleton-based models. We only consider the single camera situation. Thus, 3D structure has to be recovered from a sequence of images, otherwise severe ambiguity problems could happen.

Rehg and Kanade [73] use a skeleton-based model to tackle the self-occluding problem, a problem that always happens during articulated object motion. In their experiments, a simplified skeleton-based model (only 9 DOFs) of the index and middle fingers (3 planar joints per finger) of the hand with full translation is employed. This system does not work in real-time.

Ahmad [75] presents a real-time 3D-hand tracker with 19 DOFs. The tracking speed reaches up to 30 frames/second on a standard workstation with no special image processing hardware other than a frame grabber. The main drawback of the system is that only the hand rotations parallel to camera plane can be recovered, while the rotations of the hand around the other two axes would confuse the system.

Lathuiliere and Herve [76] have developed a system for robot controlling purposes where 26 DOFs of the skeleton are modelled. First the pose of the wrist is computed, and then the value of the finger joint angles is recovered. The fingertips and the middle point of the wrist are used to construct the correspondence between the model and the image.

CREFIT presented by Nolker and Ritter [77, 78] is able to recover 3D hand postures from grey scale images. The recovery of the joint angles is achieved by employing an artificial neural network, which uses a set of non-linear basis manifolds to construct a mapping through a number of "topologically ordered" reference vectors. The features used in this system are the 2D positions of the fingertips, and the DOFs are also restricted to avoid possible ambiguities of joint angles. Therefore, this system can not distinguish differences in depth.

A remarkable success from Rehg and Kanade is "DigitEyes" [15] where a full 27 DOF is recovered at the speed of 10 Hz. However, a special image processing board and two cameras are needed.

One of the issues that is worth mentioning in 3D-hand modelling is that although the hand is highly articulated, its configuration is also highly constrained. That is, it cannot make gestures arbitrarily. These constraints help to reduce the size of the

spatio-temporal space, thus reducing the complexity of gesture recognition. However, the constraints themselves are quite complex. Some of them are due to the anatomy of hands, and some are due to the naturalness of hand motions; some of them can be expressed explicitly, while some can not. Lin *et al.* [29] investigate these constraints thoroughly.

3.2.3 Hand Motion Modelling

Both 2D- and 3D-hand models can only extract spatial parameters of human hands, while a larger number of gestures are performed dynamically. In this section we discuss the attempts that have been made to model the temporal characteristic in motion.

Since a human gesture is a dynamic process, it is important to consider the temporal characteristic of gestures. Although learning the dynamics of hand motion has attracted much attention [85, 89, 91, 92, 93], there are only a very limited number of methods partly because of its complexity. One of the methods is called the Finite State Machine (FSM), which represents dynamic gestures as prototype sequences of states [85, 91]. Recognition is finished by matching an incoming sequence to the prototypes. To handle variations in temporal behaviour, the match is typically computed using some form of Dynamic Time Warping (DTW) [89]. If the prototype is described by statistical tendencies, the time warping is often embedded within an HMM [87, 92, 93]. In this section, we investigate the previous work dealing with the hand-motion modelling issue.

As speech recognition also requires modelling the dynamic features of the speaker, many of the same techniques can be used to model both dynamic hand gestures and speech. DTW is one of those, which enables non-linear time alignment of speech patterns produced with different speech rates. Based on this approach, Darrell *et al.* [89] presents a real-time system recognising three gestures under uniform background.

FSM models gestures as sequences of states in spatio-temporal space. Each state in this model is treated as a multidimensional Gaussian. Thus, each dynamic gesture will

correspond to a sequence of states. Template matching can be used to perform real-time gesture recognition. Hong *et al.* [85] propose a game system based on FSM which ran at the speed of 25 frames/sec on a 350MHz Pentium II PC running Windows NT 4.0. Davis and Shah [91] use FSM to model distinct phases of a generic gesture. In their method, gestures are represented as a list of vectors and are then matched to stored gesture vector models using table lookup based on vector displacements. FSM is also similar to HMMs.

The most successful technique in hand motion recognition is certainly HMMs. Starner and Pentland [92] first used it for the recognition of ASL sentences. They use a simple feature set to describe the hand shape which consists of each hand's x and y position, angle of axis of least interia, and eccentricity of bounding ellipse. In their experiment, 494 sentences are collected. Of which 395 sentences are used as a training set and the remaining 99 as a test set. When the grammar is involved in the recogniser, 99.2% recognition rate was achieved, while without the grammar, the recognition rate is 91.3%.

Vogler and Metaxas [93] also use HMMs to recognise ASL. They couple 3D-motion analysis with a context-dependent HMM to recognise 53 gestures. In their system, HMMs and 3D-motion analysis are two separate modules. Given a sequence of images, 3D-motion analysis is performed to constrain the recognition process in the HMM. In their experiment, 486 sentences in total are collected, of which 389 examples are used as a training set and the remaining 97 as a test set. 87.71% recognition rate is achieved in their experiments when coupling 3D constraints with HMMs, against 83.63% accuracy using 2D constraints. However, three cameras have to be used to get 3D information.

The standard HMM [92, 93] is extended by Wilson and Bobick [87] to have the ability to recognise parametric gestures. A parametric gesture is a movement that exhibits meaningful, systematic variation [87]. Since the standard HMM approach tends to model the meaningful variation as noise, it would not be able to handle this type of gesture. Wilson and Bobick include a global parametric variation in the output probabilities of the states of the HMM and they formulate an EM method for training

the parametric HMM. During the online recognition, the parametric HMM simultaneously recognises the gestures and estimates the quantifying parameters.

HMMs is similar to FSM as they both model dynamic gestures as sequences of state in spatio-temporal space. However, a couple of differences exist between them. First, the number of states and the topological structure of an HMM has to be fixed before training whereas FSM does not have this requirement; second, a few FSMs can be concatenated together to form a more complex FSM. For example, if gesture C is the combination of gesture A and B, then the FSM C is constructed by appending the FSM B to the end of FSM A. While for HMMs, the simple relationship does not hold.

Kalman filters are a relatively old method for modelling hand motion. They allow the system to predict future behaviour given a spatial hand model to provide some sense of error correction. Also it enables the system to operate in real-time because the computer only examines the part of the image where the filter predicts the hand must be. The Extended Kalman filtering (EKF) technique is an extension of the standard Kalman filter under the assumption of small motion of the object of interest. Shimada [71] uses a similar technique in his system. A 3D-hand model is used to capture the spatial characteristics, and then EKF is used to estimate the hand motion. However, the Kalman filtering technique is based on the spatial-temporal continuity assumption, which often fails to hold in real life.

Neural Networks are another alternative for modelling temporal characteristics. Gao *et al.* [94] present a Sign language recogniser combining neural networks with HMMs to recognise continuous large vocabulary in Chinese Sign language. During the training phase, Neural Networks and DTW are used to supervise the temporal segmentation of training data, which is very important for an HMM-based system and always affects the system performance greatly.

3.3 Summary

In this chapter, we have examined past development of the research of hand gesture recognition. To recognise a hand gesture properly, both spatial and temporal

characteristics need to be modelled. Although many methods have been shown, none of them is the best. Different methods are good for different tasks.

For a full and successful gesture recognition system, it is very important to incorporate face recognition and other parts of body, because to recognise some gestures correctly, the facial expression and the relative position between the hand and body are essential.

"Overall, at the current state of the art, vision-based gesture tracking and recognition are still in their infancy. In order to develop a natural and reliable hand gesture interface, substantial research efforts in computer vision, graphics, machine learning, and psychology should be made."

--Wu and Thomas [88]

Chapter 4 System Overview

To design a hand gesture recognition system has never been an easy task. It provides a lot of challenges. Even if we ignore the variability of lighting, shading and complex scenes containing cluttered background, human hands still produce considerably differing views depending on what they are performing and their position and orientation. A general gesture interpreter needs a broad range of contextual information, general knowledge, cultural background and linguistic capability [37], which is far beyond current computational power.

In this chapter, we first introduce our objectives and give some assumptions in Section 4.1; then we discuss the hand model, i.e. 2D or 3D, chosen for our system in Section 4.2. Section 4.3 will give the general framework of our system. We summarise in Section 4.4.

4.1 Objective and Constraints

In order to make our system widely acceptable, we are aiming at developing a hand gesture recognition system working on an entry-level workstation, which uses only one camera and can recognise hand gestures using computer vision principles without the need of any other special image processing hardware. To make a system widely acceptable, two factors are important: price and performance. On the one hand, our system only uses a single camera, and this camera can be as cheap as a common webcam (in fact that is what we are using for our performance evaluation). The system also works on an entry-level workstation. These guarantee the cheap price of the system. On the other hand, our algorithm has low computational cost, which grows logarithmically proportional to the size of the vocabulary. Even without the help of any grammar restriction, the recognition accuracy is very high. The above two points give the system potential for popularity.

The aims are quite ambitious. Some constraints have to be made to limit our research to a reasonable range. Firstly, our main research interests are in the hand modelling

and recognition problems. Although segmentation of the hand from a complex background is also an important issue for a successful system, it is not what we are concerned about here. Thus we assume the system works under normal office illumination conditions, which simplifies the task of hand tracking. Secondly, we employ statistical methods in our system. Such methods need a large amount of training data, which is another difficult issue to be solved. As in the case of speech recognition, nearly all the commercial products (for example: IBM ViaVoice, Vocalis Speechware, *etc.*) are based on HMMs. The training database of these systems involved hundreds or even thousands of people's hard work. In our study, we are more interested in reducing the computational costs. That is, a user-independent system is not our target yet. In contrast, a user-dependent system will be shown at the end of the thesis.

One thing which should be noted is that the computational cost in this thesis is measured by the time complexity because time complexity is a machine- and language-independent measurement.

4.2 Hand Model selection: 2D or 3D

An important step in the system designation is to choose an appropriate hand model. In the literature review, we have already shown systems that modelled hands either in 2D or in 3D. 2D modelling is an obvious solution for a single camera system, but 3D modelling is also possible since the hand structure could be recovered from a sequence of images. In this section, we discuss which one we should choose.

Classical computer vision theory suggests constructing a 3D model of the human hand, statistically or anatomically, and then recovering the parameters during the recognition phase. This is the approach used in [73, 75, 78]. Accuracy and flexibility are the two most obvious advantages of this type of model. In theory, a 3D model can recover movements of finger joints precisely, but there are some difficulties. Firstly, such systems estimate 3D parameters by fitting the 3D model to images based on feature correspondence. However, in many cases, for example when hands move quickly or self-occlusion exists, it is very difficult to detect such features. Failure to

detect predefined features would severely affect the system performance or even cause the failure of the whole system. Unfortunately, in a single camera system, occlusion and self-occlusion often happen or are even unavoidable [56]. Secondly, even if the features can be found reliably, to recover the parameters is still not easy. The gesture needs to be determined by generating the "possible pose" candidates, and then projecting candidates onto the image plane to find the best match with respect to a certain similarity measurement. This is essentially a search and optimisation problem, to which the best solution has not been found yet because the search space is so huge. Due to the high DOF of the human hand it is nearly impossible to avoid being trapped in a local optimum. Even finding a sub-optimal solution is computationally intensive. Consequently, it is infeasible for a system that is aiming at working on an entry-level computer. Apart from the above two reasons, many 3D modelling systems require *a priori* knowledge about the internal camera parameters, which means the camera has to be carefully calibrated. This makes the 3D model-based systems dependent on users and impossible for portable applications.

To overcome all these limitations, we choose a 2D (or appearance-based) model. In this case, no *a priori* model is used, and the model has to be learned from a collection of data. After the learning phase, a number of gestures can be recognised. 2D hand model-based systems benefit from high speed and low computational cost.

4.3 Outline of the System

4.3.1 Hardware Set-up

In our system, the video camera is set up in front of the user. See Figure 4.1. The user can move his/her hand freely in a certain range. The camera acquires images and sends them to the machine, which then processes the images using computer vision principles. The name of the recognised gesture will be shown on the monitor.

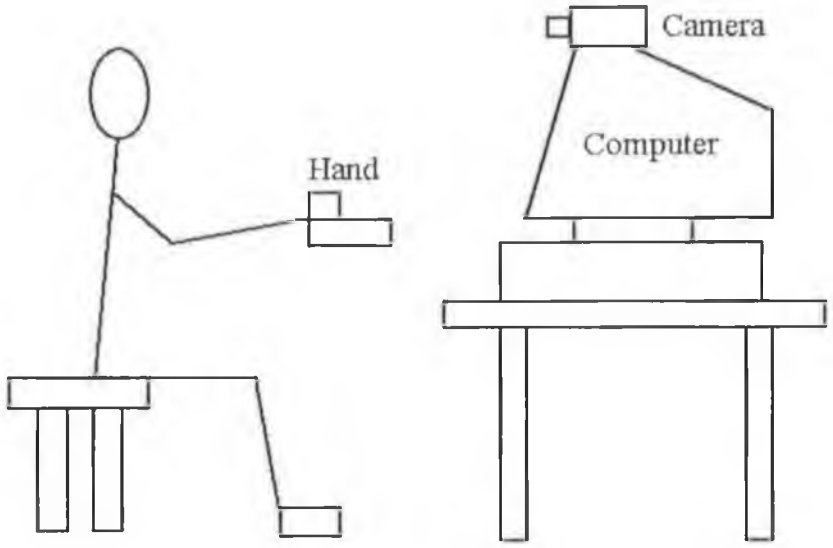


Figure 4.1: System hardware setting up.

4.3.2 Software Architecture

We now describe our software framework. The whole system is divided into three modules: 1. pre-processing, 2. training and 3. recognition. The first module is responsible for acquiring the visual data and pre-processing it including segmentation, noise removal and normalisation. The training module constructs a hierarchical decision tree and an HMM recogniser. The last module is the recognition module and responsible for the classification of unknown gestures. When the recognition module starts up after training, it initialises the HMMs and reads in the hierarchical decision tree. Then for each frame, it gets the hand data from the data acquisition and pre-processing module and recognises the gesture made by the user. If there is a meaningful gesture, it outputs its name. The whole framework is illustrated in Figure 4.2.

As shown in the figure, the system modules are highly structural. Each module is composed of a few sub-modules, and each sub-module is only connected with one previous step and sends the results to one following sub-module. This type of framework gives the system expandability. Each module and sub-module can be substituted as long as the input and output data structures obey the same syntax. For instance, the construction of the decision tree affects the final performance of the

system. If desired, this part can be easily replaced without change to the code of other parts of the system.

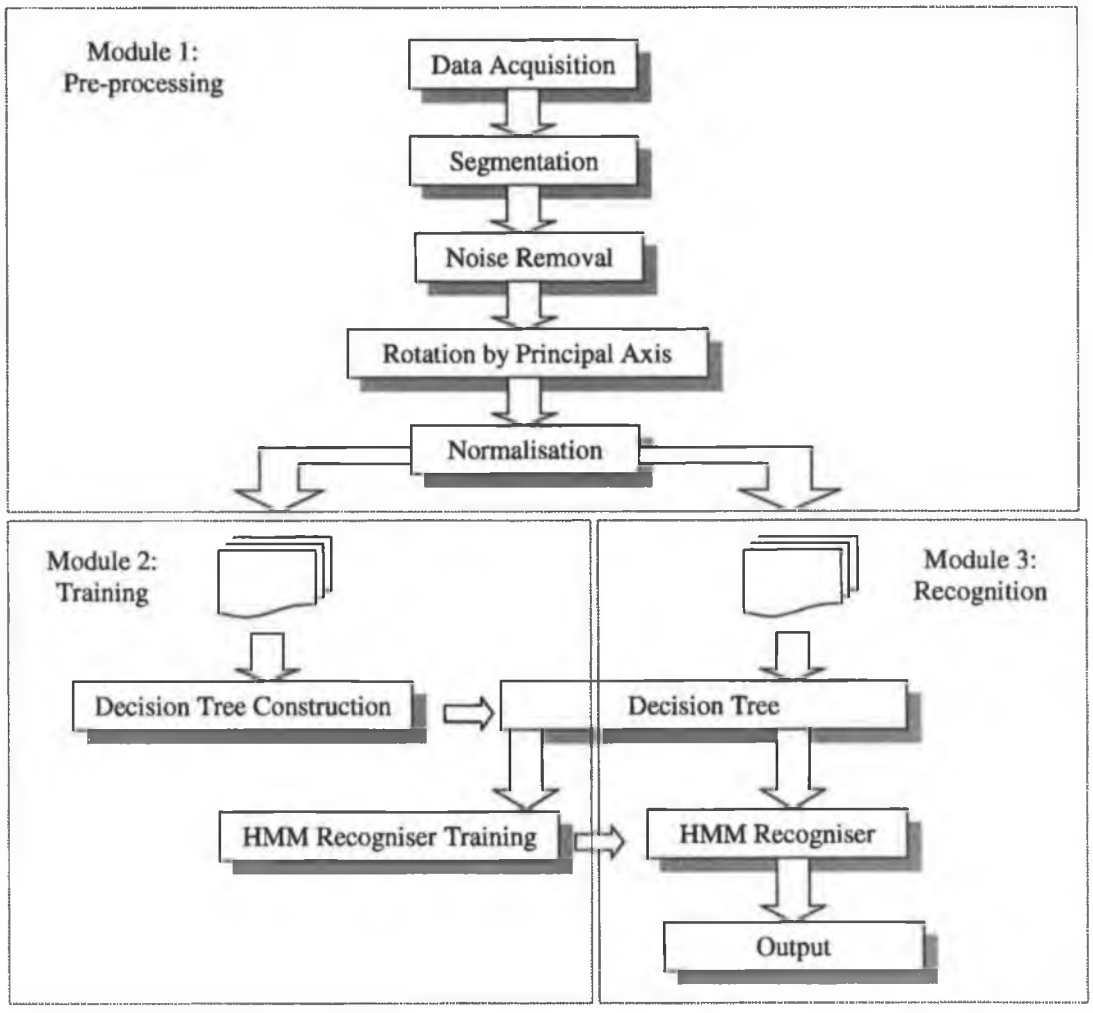


Figure 4.2: System software architecture.

4.4 Summary

In this chapter, we discussed our objectives and made a few assumptions in the first section. We chose the 2D-hand model for our system, and we showed the hardware set-up. The software architecture has been demonstrated, which is composed of three modules: data acquisition and pre-processing, training and recognition. These three modules include most of our work.

Chapter 5 Static Gesture Recognition Using PCA

5.1 Introduction

Our work in this chapter is similar to Birk and Martin's systems [66, 67](see Section 3.2.2.1). We use PCA to recognise the finger-spelling gestures in ISL. The performance will be evaluated using three types of classifier.

The remainder of the chapter is divided into three parts. We first introduce the pre-processing procedure in Section 5.2. We then explain how to construct image vectors, and apply PCA to a set of training images in Section 5.3. The evaluation results of the recognition of finger-spelling gestures of ISL using three types of classifiers are presented in Section 5.4. Finally we summarise.

5.2 Pre-processing Procedure

The original images from the camera may contain many objects and various backgrounds. The first task of any gesture recognition system is to find the hand object while eliminating the influences from other objects and the background, i.e. to follow the position of the human hand. This task is accomplished by the pre-processing module in our system.

The pre-processing procedure includes four steps, which are summarised here. Details can be found in Appendix A.

Step 1: Standard colour segmentation is applied to the original image.

Step 2: Noise is then removed from the image.

Step 3: The hand is rotated according to its principal axis.

Step 4: The area of the image is scaled so that its resolution is reduced to 32×32, and the centroid of the image is moved to 16×16.

Figure 5.1 shows the effects of the above procedure.

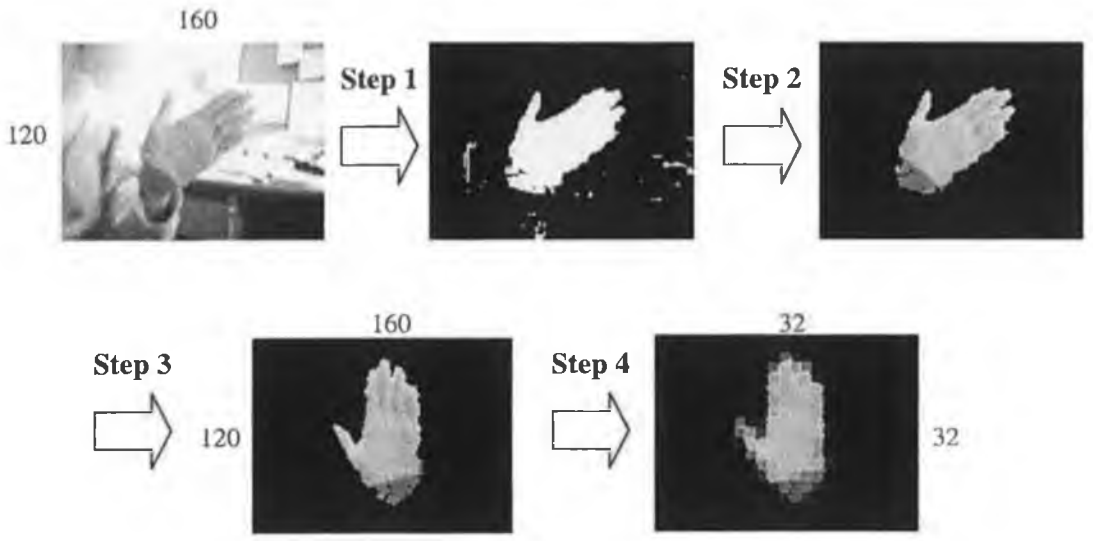


Figure 5.1: Pre-processing procedure. The numbers beside the images are the resolution.

5.3 Construction of Image Vectors

Consider an image composed of $m \times n$ pixels, where m and n are the dimensions of the image. We construct a vector by concatenating the image pixel row by row. See figure 5.2.

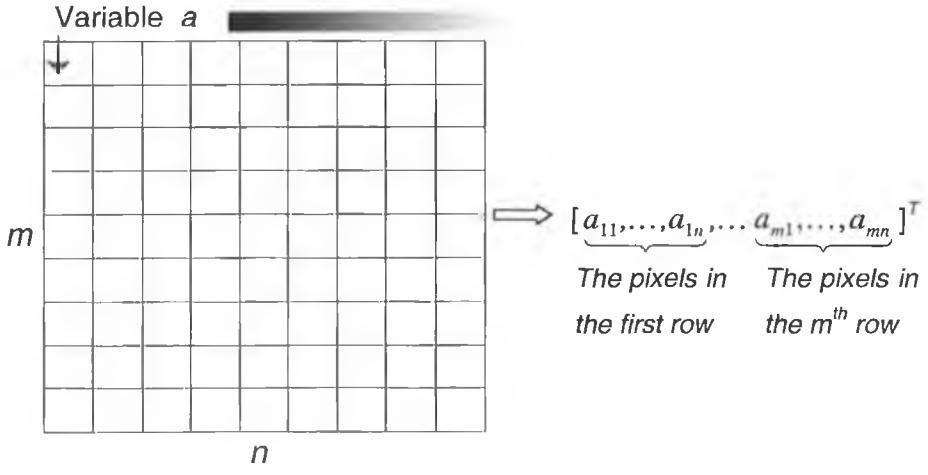


Figure 5.2: Construction of image vector. The grid represents a 32×32 image. Each pixel can be treated as a variable varying from 0 to 255, where a_{mn} is the grey value of the pixel at mth row and nth column.

Suppose the grey value of the pixel at ith row and jth column is represented by a_{ij} . The image vector can be written as $\mathbf{f} = [a_{11},\ldots,a_{1n},\ldots,a_{m1},\ldots a_{mn}]$. In our case, $m = n =$

32, hence the length of the vector is fixed as 1024 (32×32). As the numbers of rows and columns are not important any more here, we rewrite the image vector into $\mathbf{f} = [a_1,\ldots,a_N]$, where $N = mn = 1024$. Each pixel in such an image vector can be treated as a random variable varying from 0 to 255, i.e. from total darkness to total whiteness. Therefore the whole image vector is a random vector containing 1024 random variables.

Each image vector corresponds to a single point in a 1024-dimensional feature space. The dimensions in the feature space are not independent from each other since correlation exists between the pixels. Using the strategy of PCA (see Section 2.2), we transform the feature space into a new PC space whose dimensionality is much lower than the original feature space.

Every PC in the PC space represents a mode of variation of the hand structure. The first PC corresponds to the variation of a large-scale structure where the largest eigenvalue is its weight, the second PC corresponds to the variation of a less large-scale structure where the second largest eigenvalue is its weight, and so on. Figure 5.3 gives an example. The PC space used in the example was trained by the training set containing 1200 frames with 600 from class "C" and 600 from class "F". The variations of the first three PCs are shown. Some typical reconstruction of images using the first three PCs are given in Figure 5.4, where the two static gestures on the left side are the original images of "C" and "F" in ISL respectively, while the two images on the right are their reconstruction.

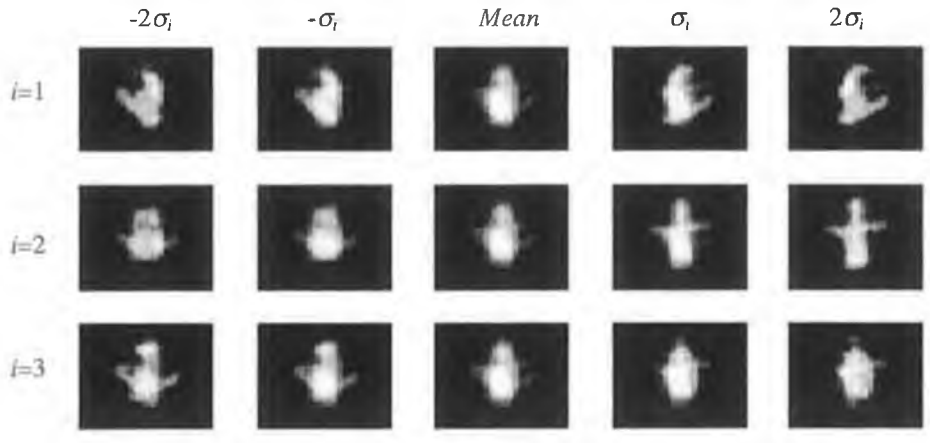


Figure 5.3: The first three PCs. σ_i is the standard deviation along the ith PC, and $\sigma_i = \sqrt{\lambda_i}$.

Figure 5.4: Image reconstruction using PCA. The left two images represent "C" and "F" in ISL respectively, while the two images on the right are their reconstruction.

In the example, the first 22 PCs included more than 90% of the energy. Therefore we can reduce the dimensionality from 1024 to 22 with only 10% loss of energy. Furthermore, the projection of the images on these 22 PCs can be used as features for classification purpose. Consequently, the processing time is reduced by a factor of 46 (1024/22).

When applying PCA, one problem is to decide the dimensionality of the feature space, i.e. how many PCs to retain. Generally speaking the task of determining how many PCs to retain is a matter of representing as much information as possible. The more PCs the more energy can be preserved. However, retaining more PCs also means more computation and less dimensionality reduction, and sometimes more PCs will bring in more noise, such as the variation of the illumination condition or other white noise. Hence, this is a tradeoff between the loss of the energy and the amount of computation.

In practice, we take the following strategy: the number of PCs to be retained is decided using a simple threshold T. The criterion is formalised as

$$E_{remain}(K) > T \qquad 0<T<1. \qquad \ldots(5.1)$$

where $E_{remain}(K)$ is defined in Equation 2.8. Empirically, 0.95 would be a fair threshold for many applications.

5.4 Static Gesture Recognition

We now evaluate this methodology by recognising the 23 finger-spelling gestures. The illustration of these gestures is given in Appendix B.

5.4.1 Training Phase

The first thing is to construct the image database, from which both training and test sets will be extracted. The acquisition of image data is a key issue in the system evaluation, especially for a system using an appearance-based hand model. If most of the images in the database concentrate on one or some specific view angles, the eventual recognition rate would be quite good. However, the system would be sensitive to the changes of the hand configurations. Hence the recognition rate would not measure the performance of the system well.

To do a fair test, we use the following strategies:

1) We construct a separate database for each of the 23 finger-spelling gestures, with each database containing 1200 images. Different view angles are included in the database as much as possible.
2) A *holdout* method [43] is applied for selecting the training and test sets from the databases. First a number of images, say 120, are extracted randomly from each database for training, then another group of images, say 120, are extracted from the rest of the images in each database. The procedure assures the independence between training and test samples.
3) We then train the classifier using the training set and calculate its accuracy using the test set.
4) We repeat the step 2 to 3 for 10 times and work out the average accuracy.

Figure 5.5 gives some examples that are selected from the training sets randomly. It can be seen clearly that the samples cover a wide range of view angles.

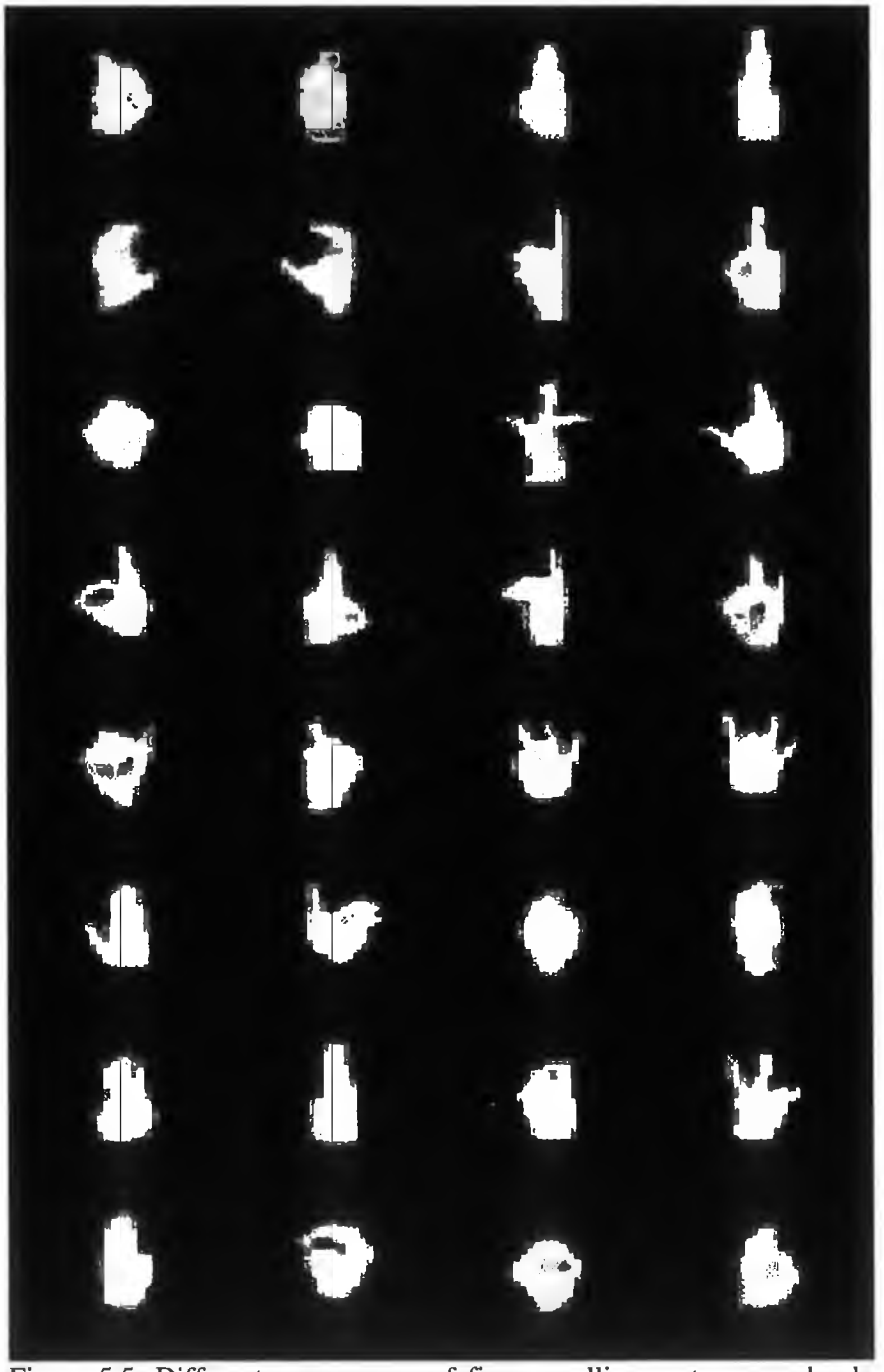


Figure 5.5: Different appearances of finger-spelling gestures randomly extracted from the training set.

5.4.2 Classification Using Three Types of Classifier

In this section, we discuss three types of classifier. For the first two classifiers, we calculate a single PC space which contains all 23 finger-spelling gestures. We calculate the mean vector of each gesture class in the PC space. For the third classifier, we use a different PC space for each gesture.

(i) Single-space Euclidean Classifier

When confronted with a classification problem, the easiest method is to assign the unknown object into the cluster with the shortest Euclidean distance between the unknown object and the mean of the cluster. Suppose we have an unknown feature vector $\mathbf{p} = [p_1, p_2, \ldots, p_K]$ and the ith class is characterised by its mean vector $\mathbf{w}_i = [w_{i1}, w_{i2}, \ldots, w_{iK}]$, where K is the dimensionality of the feature space. The Euclidean distance between them is defined as:

$$d_E(\mathbf{p},\mathbf{w}_i) = \| \mathbf{p} - \mathbf{w}_i \| = \left[\sum_{j=1}^{K} (p_j - w_{ij})^2 \right]^{1/2} \quad \ldots(5.2)$$

The classification criterion thus can be formulated as:

$$w_c = \min_i (d_E(\mathbf{p},\mathbf{w}_i)) \quad \ldots(5.3)$$

where w_c is an integer number representing the chosen class. This is a simple criterion with very little computation. We call the classifier that uses the above classification criterion a *single-space Euclidean classifier* since only a single PC space is computed from the overall training samples. When applying it in the experiment, a reasonable result is achieved, as shown in Figure 5.6.

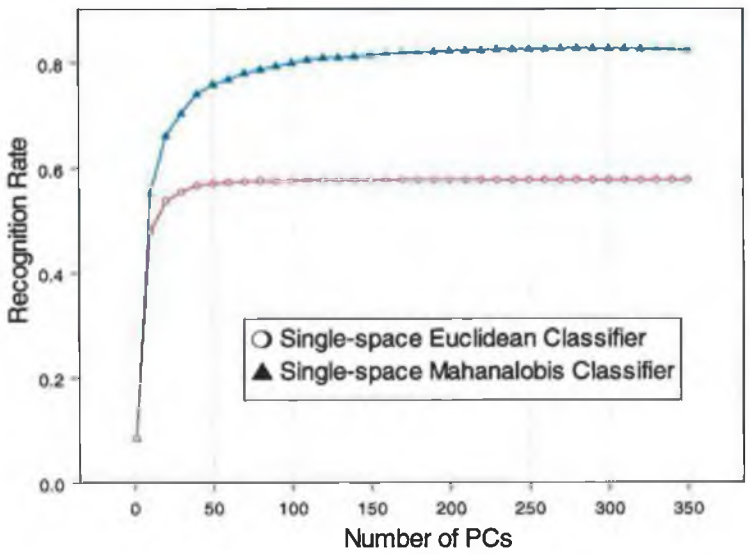


Figure 5.6: System performance test: single-space Euclidean classifier vs. single-space Mahalanobis classifier.

(ii) Single-space Mahalanobis Classifier

The single-space Euclidean classifier treats the distance along every dimension of the PC space in the same manner and does not take into consideration the different variance along different directions. See Figure 5.7. To overcome this problem, we take advantage of another classifier: the *single-space Mahalanobis classifier*. It also works in a single PC space, but uses the Mahalanobis distance, defined in Equation 5.4, rather than the Euclidean distance in the PC space.

$$d_M(\mathbf{p},\mathbf{w}_i) = |\mathbf{p}-\mathbf{w}_i| = \left[\sum_{j=1}^{K}\left(\frac{p_j - w_{ij}}{\sigma_j}\right)^2\right]^{1/2} \quad \ldots(5.4)$$

where σ_j is the standard deviation along the jth PC. The single-space Mahalanobis classifier can be formalised by the following equation:

$$w_c = \min_i(d_M(\mathbf{p},\mathbf{w}_i)) \quad \ldots(5.5)$$

Since in the feature space, the jth eigenvalue, λ_j, represents the data variance along the jth PC, the above classification criterion can be rewritten as:

$$w_c = \min_i\left[\sum_{j=1}^{K}\frac{(p_j - w_{ij})^2}{\lambda_j}\right]^{1/2} \quad \ldots(5.6)$$

Compared to the Euclidean distance, only a simple scaling factor is added to each term, which does not make the computation much more complex. Figure 5.6 gives the performance tests using the two criteria. Each curve represents the average recognition rate of ten trials. Up to 25% improvement has been achieved with the classifier using the single-space Mahalanobis classifier compared to the single-space Euclidean classifier.

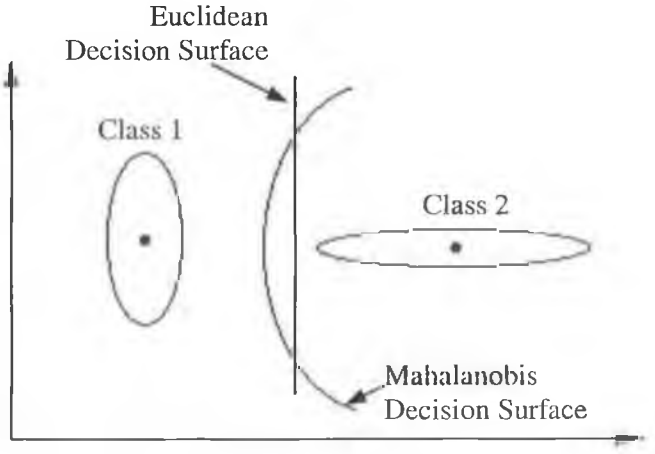


Figure 5.7: Difference between the single-space Euclidean classifier and the single-space Mahalanobis classifier.

It is worth noting once again that choosing different number of PCs can affect the ultimate performance greatly. For example, in terms of the Mahalanobis classifier, 70% of test images were correctly classified when 30 PCs corresponding to the 30 largest eigenvalues were used, while 82% can be correctly recognised when 160 PCs were used. A 12% improvement is achieved with the cost of about 5.3 (160/30) times more computation. There is a tradeoff between the recognition rate and the computational speed. When too many PCs are included, noise will affect the final result and thus cause a drop in recognition rate. For example, in Figure 5.6, using the Mahalanobis classifier, when more than 330 PCs are included, there is a reduction of the recognition rate. When the images contain much noise, the drop will become even clearer. On the other hand, it is always desirable to have less computation. Increasing the number of PCs means an increase of the online computation. When the improvement of the performance is not good enough, we prefer a system with less computation and little reduction in accuracy. That is, we have to ask the question: is it worth achieving 12% improvement with 5.3 times more computation? There is no absolute answer and it depends on the working environment. In our experiment, we use the first few PCs that contain at least 95% overall energy. In this case, the average dimensionality of the feature space is 87, and the recognition rates from the Euclidean classifier and the Mahalanobis classifier are 57% and 79% respectively.

(iii) Multiple-subspace Classifier

The above procedure gives us an at most 83% recognition rate, which is not a satisfactory result. An improvement is possible by employing a multiple-PC-space classifier, which is similar to the subspace pattern recognition method, a statistical method proposed originally by Watanabe [44, 45]. Instead of using a single PC space, we use a set of PC spaces, each corresponding to one class. Since each PC space is a subspace in the original 1024-dimensional global space, we call it a *multiple-subspace classifier*. In practice, given N finger-spelling gestures to recognise, we construct one subspace for each gesture using PCA, and name them $L_1, L_2, L_3, \ldots, L_N$.

Given an image vector $\mathbf{f}$, the distance between $\mathbf{f}$ and the ith subspace, L_i, is defined using the original image vector $\mathbf{f}$ and the reconstructed image vector $\mathbf{f}_i'$ from L_i:

$$d_S(\mathbf{f}, L_i) = \| \mathbf{f} - \mathbf{f}_i' \|^2 \qquad \ldots(5.7)$$

The multiple-subspace classifier can be formalised as:

$$w_c = \min_i(d_S(\mathbf{f}, L_i)) \qquad \ldots(5.8)$$

Again, w_c is an integer number representing the chosen class. The procedure is visualised in Figure 5.8. Given an unknown image $\mathbf{f}$, it will be classified into L_1, i.e. subspace 1, since the Euclidean distance between $\mathbf{f}$ and $\mathbf{f}_1'$ is shorter than the one between $\mathbf{f}$ and $\mathbf{f}_2$.

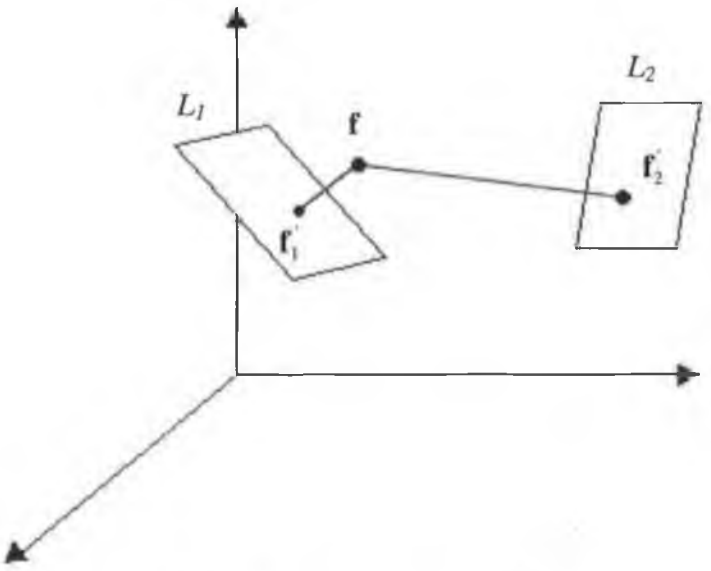


Figure 5.8: The multiple-subspace classifier.

Figure 5.9 shows the average recognition rate for the overall 23 finger-spelling gestures. Note for visualisation purposes, that the average result in the figure is worked out under the condition that every subspace has the same dimensionality, whereas in practice we decide the dimensionality of each subspace separately using Equation 5.1. That is, for each feature space we reserve the first few PCs that involve at least 95% overall energy. In this case, the dimensionalities of the subspaces vary from 22 to 56, and the average recognition rate is 98%, which is very close to the best performance. A drop in recognition rate can be found when more than 30 PCs are used. Again, this is because adding more PCs adds more noise.

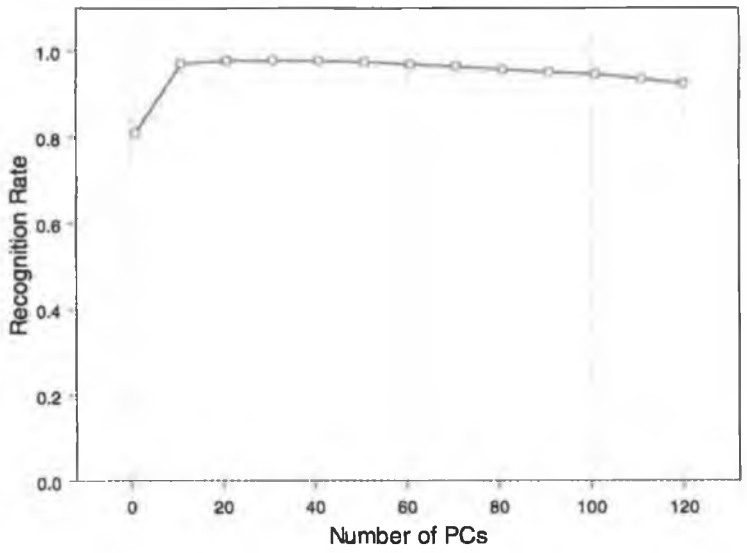


Figure 5.9: System performance test using multiple-subspace classifier.

5.4.3 Time Complexity of the Classifiers

In terms of the computational complexity, the most expensive computation is the scalar product operation, i.e. the multiplication between the image vector and the PCs. Each classifier's time complexity would be bounded by the number of scalar product operations. In other words, the time complexity can be computed with respect to the number of the scalar product operations. See Table 5.1:

Table 5.1: Time complexity of the classifiers

Classifier	Time Complexity
Single-space Euclidean Classifier	$O(K)$
Single-space Mahalanobis Classifier	$O(K)$
Multiple-subspace Classifier	$O(N_c \times \max_i(K_i))$

- K is the dimensionality of the single feature space.
- K_i is the dimensionality of the ith subspace in the multiple-subspace methods.
- N_c is the number of finger-spelling gestures to be recognised, in our case, $N_c = 23$.

In practice, the time complexity of the multiple-subspace classifier would grow roughly linearly with respect to the number of classes to be recognised, because an

additional finger-spelling gesture would not affect the dimensionalities of the other subspaces thus $\max_i(K_i)$ would hardly change much.

5.5 Summary

In this chapter, a standard PCA-based system has been designed. First we introduced the pre-processing procedure. Then, a single PC space or multiple subspaces are constructed according to the type of the classifier employed. A number of classifiers have shown quite different accuracy results. The multiple-subspace classifier outperforms the single-space classifiers.

The time complexity of the multiple-subspace classifier is a linear function of the number of classes. This would become unacceptably slow when there are many gestures to be recognised. By utilising a hierarchical decision tree combined with multi-scale theory, in the next chapter we will develop a fast and reliable system.

Chapter 6 Decision Tree and Multi-scale Theory

6.1 Introduction

In the previous chapter, a good recognition rate for the finger-spelling gestures in ISL was obtained using the multiple-subspace classifier. However, the processing time is linearly proportional to the number of gestures. For a large set of gestures, the speed would be unacceptably slow to perform any real-time recognition. This chapter introduces a novel methodology to reduce the online recognition time by combining standard PCA with multi-scale theory. A hierarchical decision tree is constructed and the processing time is only proportional to the depth of the decision tree.

The core technique in this chapter is a hierarchical decision tree, which is somewhat similar to the two-level hierarchy presented by Heap and Hogg [61] (see Section 3.2.2.1). Their approach is similar to ours in two aspects. First, PDM (Point Distribution Model) is a statistical method based on PCA. Similarly, our approach is also based on PCA. Second, the way they find the linear sub-regions in the global PC space, *k*-means, is the one we employ here.

Differences between Heap and Hogg's method and ours are as follows: first and foremost, their hierarchy contains only a two-level tree and thus can only handle single non-linearity. In other words, the efficiency decreases as the number of hand shapes increases. In comparison to this, we present a multi-layer decision tree, which contributes the ability to handle the complex non-linearity in a large database and to recognise gestures quickly. Second, unlike Heap and Hogg's system where the number of sub-regions has to be decided manually, the construction of our system is fully automated, which makes the training and expansion of the system much easier. Thirdly, our approach will be applied to extract the intermediate hand configurations from dynamic gestures, so that the configurations can be further fed into a DHMM (discrete HMM) for the recognition of dynamic gestures. The third point is an important aspect of the hierarchical decision tree, but to our knowledge has not been mentioned by either Heap and Hogg or other researchers before. We will discuss the third point thoroughly in the next chapter together with the DHMMs since the

recognition of dynamic gestures needs coordination between the decision tree and DHMMs.

Similar work has been done by Cui and Weng [37] (see Section 3.2.2.1). Their scheme is also similar to our hierarchical decision tree. The differences between our systems are 1) no multi-scale theory is embedded in their system and 2) our decision tree is applied to extract the intermediate hand configurations from dynamic gestures, while Cui and Weng only use it as a way to manage a large number of training samples.

In this chapter, we discuss the use of multi-scale theory in our system. We introduce the construction of the hierarchical decision tree, and consider the selection of the scale parameters. We evaluate our approach with the finger-spelling gestures. Finally, we have the summary.

6.2 Why Multi-scale Theory?

Blurring an image gives us two advantages:

1. It reduces the noise level. Since noise is unavoidable when acquiring the training data, and the quality of training data affects the final performance very much, we would like to train our system with a set of "clean" data. Most of the unavoidable noise is white, i.e. independent from the "real" gesture information. In the frequency domain, the frequency of white noise distributes evenly on all the frequency bands. In contrast, the energy of the hand images is distributed in the low and middle frequency bands. Thus most of the high frequencies are caused by the noise. In the field of digital image processing, a Gaussian kernel is treated as a low-pass filter, which removes the high frequency elements from the original signal. Consequently, convolving an image with a Gaussian kernel reduces the level of the white noise.

2. Blurring with a Gaussian kernel reduces the dimensionality of the PC space. The bigger the scale parameter the smaller the dimensionality of the PC space. This is because the blurring procedure suppresses the noise and minor structure in the image,

which makes it possible to use fewer PCs to represent more energy. Empirically, a dimensionality reduction of more than 50% can be achieved. See Figure 6.1.

We use a training set composed of 5 subsets, i.e. 5 gestures, each of which has 300 samples. Without any blurring, the dimensionality of the PC space is 50. As the scale parameter increases, the curve drops fairly quickly. When σ is 1.3, the dimensionality of the PC space is reduced to 21. That is, about 58% dimensionality reduction is achieved with respect to the original PC space.

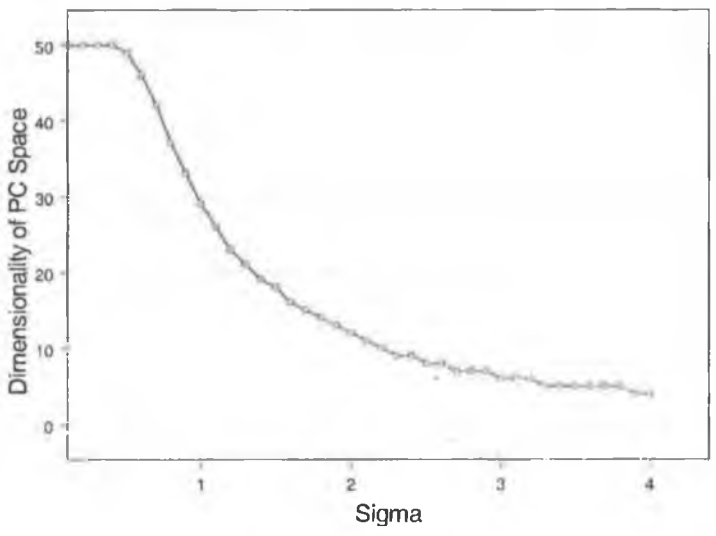


Figure 6.1: Dimensionality of PC space vs. Sigma

Recall the fact that PCA assumes that the projection of the training samples in the PC space is bounded by a hyperellipsoid (see Figure 2.3). From the hyperellipsoid point of view, the dimensionality reduction of the PC space means that the shape of the hyperellipsoid changes according to the scale parameter, i.e. the data distributes on fewer dimensions.

We can analyse the shape of the hyperellipsoid by examining the shape, i.e. the curvature, of the eigenvalue curve since each eigenvalue reflects the axis length of the hyperellipsoid along the corresponding PC. To compare the curvatures of the eigenvalue curve under different scale parameters, we normalise the eigenvalue curve to remove the variation in absolute magnitude:

$$\lambda_i' = \lambda_i / \lambda_1 \qquad \ldots(6.1)$$

Since the dimensionality of the PC space is reduced when the scale parameter increases, which implies that the image energy focuses on the first few PCs, consequently, the curvature of the normalised eigenvalue curve becomes steeper, as shown in Figure 6.2. The normalised eigenvalues along some less important PCs are approaching zero (for example, the normalised 10^{th} eigenvalues are 0.10, 0.07 and 0.01 corresponding to the scale parameters of 0, 2.0 and 4.0).

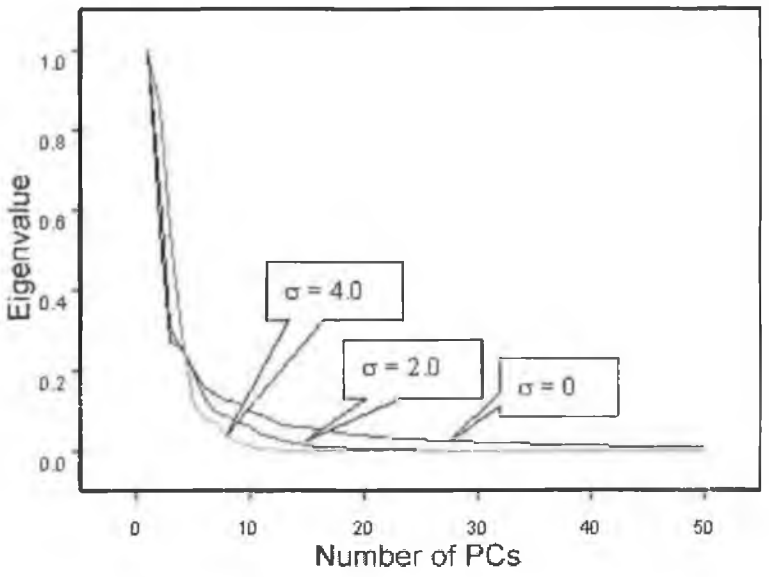


Figure 6.2: Varying σ, the scale parameter, affects the curvature of eigenvalue curves. Note only the first 50 eigenvalues are shown.

The dimensionality reduction of the PC space gives us some benefits. First, The dimensionality reduction of the PC space would improve the speed of both the training and the recognition phase since it reduces the total amount of computation of the transform from the original feature space into the PC space. Second, even without considering the speed, the dimensionality reduction improves the stability of the *k*-means algorithm. A disadvantage of the *k*-means algorithm is that it is easy to get trapped in local maxima. The higher the dimensionality of the PC space, the more local maxima exist. The blurring procedure smoothes the images, hence can reduce the number of local maxima in the PC space and improve the stability of the *k*-means algorithm.

6.3 Construction of Decision Tree

As stated in the preceding chapter, the computational workload grows linearly according to the number of gestures when a multiple-subspace classifier is applied. For a system with a large vocabulary, this could be a crucial disadvantage since hundreds of gestures would need to be recognised, and the processing speed would become unacceptably slow. Another drawback of the multiple-subspace method is that the overlaps among the subspaces would become more and more severe when the number of gestures increases. Thus the system performance would decrease significantly. There is a need to develop a new methodology that is able to process the images fast as well as reliably.

In ISL, some gestures are very similar to each other. The differences between their images are in the fine structure. This fine structure will be eliminated when the image is blurred by a Gaussian kernel with a small scale parameter. On the other hand, some gestures are very different from each other. In this case, the differences between their images are in the coarse structure. Only when the image is convolved with a Gaussian kernel with a large scale parameter, can the blurring operation affect the coarse structure. To illustrate this, we give an example below. In ISL, the hand shape representing "L" is similar to "B", and dissimilar to "F". See figure 6.3:

Figure 6.3: Hand shapes in ISL corresponding to three English characters. From left to right: "L", "B" and "F". For "L", one fully opens one's palm. For "B", one opens the palm with the thumb closed. For "F", one crosses one's thumb and index finger, and opens the other three fingers to form the shape of "F". Note, the shape of "L" is shown in a side view, while the other two are front views.

These three images will be projected to three different points in PC space. Increasing the scale parameter of the Gaussian kernel means that they would become more and more similar. Consequently, the points in the PC space would move closer to each other, as shown in Figure 6.4. Given a threshold T, we ignore the difference between

the gestures and say the two hand shapes are the same when the distance between their corresponding points falls below T. As can been seen in the figure, "L" and "B" would be treated as the same gesture when the scale parameter reaches 0.9, while the value for treating "F" and "L" as the same is about 2.8. In other words, when $\sigma > 0.9$, the differences in the fine structure between "L" and "B" have been eliminated, and when $\sigma > 2.8$, the differences in the coarse structure between "L" and "F" are eliminated,

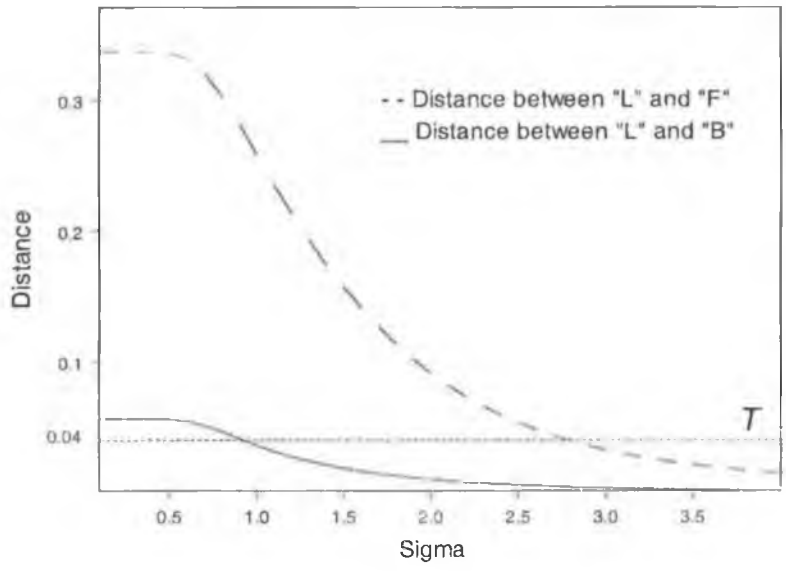


Figure 6.4: Distance vs. Sigma. The horizontal axis is the scale parameter of the Gaussian kernel, while the vertical axis is the distance between the projection point "L" and other two. Threshold T is set up to 0.04.

Based on the above thoughts, recall the two facts:

1. the large PCs correspond to the variation of the large-scale, i.e. coarse, structures, while the PCs corresponding to the small eigenvalues represent the variation of the small-scale, i.e. fine, structures (see Section 5.3).
2. if we convolve a set of images with a Gaussian kernel with a large σ, only the first few PCs will be retained in the PC space. However, if we convolve them with Gaussian kernel with a smaller σ, more PCs will be retained (see Section 6.2).

We can construct a hierarchical decision tree: given a training set, we convolve the training images with a Gaussian kernel with a large σ so that these images can be split into several subgroups according to the differences of the first few PCs, i.e. coarse structures; then we reduce the value of σ, and split each subgroup into more

subgroups according to more PCs. As the process continues, a tree-like structure would be formed, where each node is a PC space. See Figure 6.5:

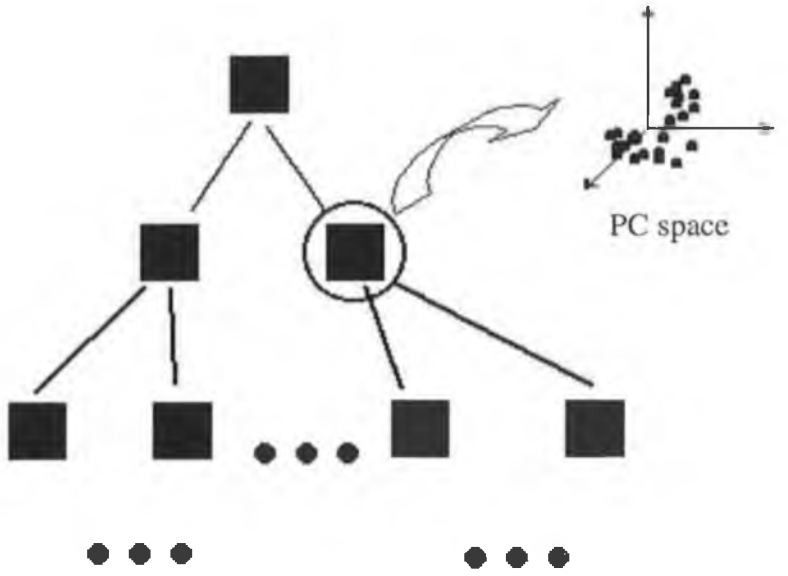


Figure 6.5: Illustration of a decision tree.

The detailed procedure of constructing a hierarchical decision tree is as follows: given a training set $\mathbf{X}=\{\mathbf{f}_1, \mathbf{f}_2,\ldots, \mathbf{f}_N\}$, where $\mathbf{f}_1, \mathbf{f}_2,\ldots, \mathbf{f}_N$ are the image vectors,

1. Every sample in the training set $\mathbf{X}$ is convolved with a two-dimensional Gaussian kernel whose scale parameter is σ, $G(x,y,\sigma)$:

$$\mathbf{X}' = \{\mathbf{f}_1', \mathbf{f}_2', ..., \mathbf{f}_N'\} \qquad \ldots(6.2)$$

where $\mathbf{f}_i'$ is given by:

$$\mathbf{f}_i' = \mathbf{f}_i * G(x, y, \sigma) \qquad (1<i<N) \qquad \ldots(6.3)$$

where * defines a convolution. This step blurs the differences between the images and reduces their separation in the PC space. This reduces the number of eigenvectors needed to describe the data as well.

2. A PC space is computed from $\mathbf{X}'$:

   1). Computing the covariance matrix of $\mathbf{X}'$ using Equation 2.1.

2). A PC space is then computed by solving the eigenvalue decomposition problem (see Equation 2.4).

3). The dimensionality of the PC space is decided by retaining the first few PCs so that at least 95% of energy is retained (see Equation 2.8).

3. The standard *k*-means algorithm is then applied to the data in the PC space, dividing them into C clusters according to what type of tree is wanted, i.e. for a binary tree $C=2$, for a quad-tree, $C=4$, and so on. The original training set $\mathbf{X}$ is then split into C groups: $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_C$.
4. For each of the C clusters, check if the stop criterion is satisfied. If it is, mark it as a leaf, and if all the clusters at the current level are leaves, stop the splitting process. Otherwise, for each $\mathbf{X}_i$ ($1<i<C$), repeat step 1 to 4 with a smaller scale parameter σ' ($\sigma'<\sigma$).

An important parameter during the construction of a decision tree is the termination condition. Currently we choose the data variance in the PC space as the termination condition. When the variance of image projections in the PC space reduces to a certain level, 0.5 for example, the node will be marked as a leaf, and no further splitting operation will be done on this node. When all the nodes on the current level are leaves, the construction is stopped and the learning process is finished.

After the construction of the decision tree, the original training set has been divided into leaves, with each leaf containing some data projections of the training samples. The data projections in the leaf form several clusters. We need to label these clusters, i.e. we have to indicate which cluster represents which gesture. The labelling procedure is implemented in the following way: we search every leaf in the tree. Given a leaf, we check the training images grouped into that leaf, find the data clusters included, and label the clusters accordingly. For instance, suppose a leaf contains 100 training images in which, say 35 frames belong to "C" and the rest "L", this leaf then contains two data clusters, one of which be labelled as "C" and the other "F". The centres of the cluster "C" and the "F" would be recorded respectively. Given an unknown frame, classified into this leaf by the decision tree, we compute its Euclidean distance to "C" and "F" and classify it to the one with the shorter distance.

In our evaluation, we found that most of the leaves contains only one data cluster. That is, either "C" or "F" will exist in one leaf but not both. Figure 6.6 shows some example mean images of the leaves. Most of the images are quite clear, for example, image (1) represents "O" in ISL, (2) represents "F", (3) represent "C", and so on. To some extent, the purity of the leaves reflects the power of our system – the hand shapes can be effectively extracted even if the training set contains a large variety of different shapes from different view angles.

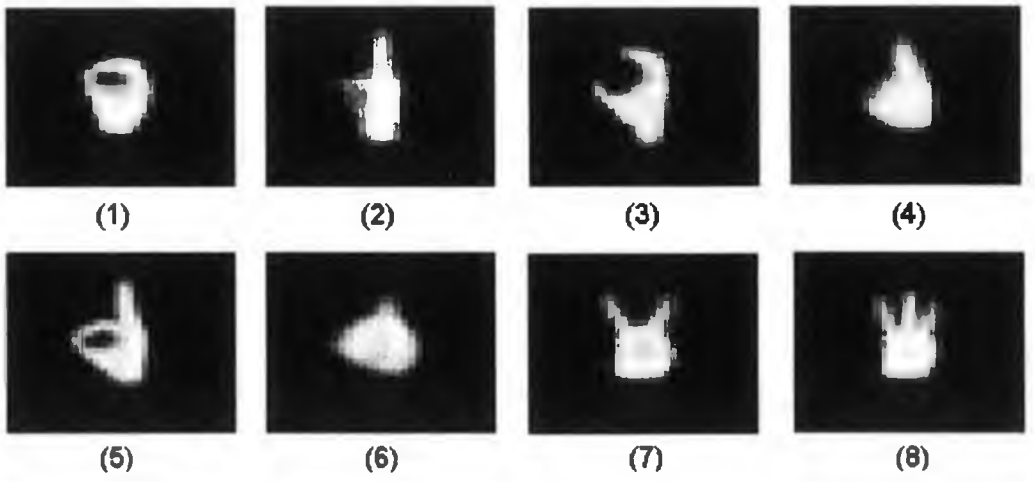


Figure 6.6: Mean images of the leaves from the decision tree made from the training data. These clear images illustrate that most of the leaves contain a single gesture. These images are not picked up deliberately for the thesis, but selected randomly from the leaves.

The decision tree reduces the time complexity of the gesture recognition task significantly. Suppose the training set is composed of N_c subsets, i.e. N_c gestures, and each of the subsets contains M samples. Let d be the depth of the decision tree constructed from the training set. When doing recognition using a standard multiple-subspace classifier, the time complexity is bounded by the number of scalar multiplication operations needed, which is $O(c \times max(D^S_i))$ (see Table 5.1), where D^S_i represents the dimensionality of the PC space built from the *ith* subset and $1 < i < c$. However, in the decision tree method, the depth of the decision tree, d, is only logarithmically proportional to the number of the training samples, cM [42]. The time complexity of the online recognition is $O(\log(c) \times max(D^D_j)))$, where D^D_j stands for the dimensionality of any PC space in the decision tree, and j is greater than 1 and smaller than the total number of the nodes. Empirically, the ratio between $max(D^D_j)$ and

$max(D^S_i)$ is bounded from above by 3 (and the lower boundary is not of interest). Hence the ratio between two time complexities becomes $O(c)/O(\log(c))$. Logarithmic time saving is achieved by taking advantage of the decision tree method. The more gestures to be recognised, the more time is saved.

6.4 Performance Evaluation

To evaluate the methodology we introduced above, we still use the 23 finger-spelling gestures as we did in the last chapter.

Again, the first thing is to construct the training set and test set. We select 300 images for each gesture from the same database used in the last chapter. For the test set, we select another 300 frames from each gesture, i.e. 6900 frames, excluding those that have been used for training purposes. This procedure assures the independence between the training images and the test images, and can reflect the system performance better. But before we can start to build the decision tree, a few crucial problems need to be cleared up.

6.4.1 The Selection of Scale Parameters

One important thing, however, is how to select the proper decrement for the scale parameter. In other words, since the number of PCs of the PC space is increasing from the root to the leaves, i.e. the dimensionality of the PC space is increasing, we attempt to address the problem: how fast should the dimensionality increase?

By observing eigenvalue curves such as those in Figure 6.7, we found that empirically the curve of the normalised eigenvalues fits the following power function:

$$f(x) = x^{-n} \qquad \ldots(6.4)$$

where x is the order of the PCs – the horizontal axis in the plot of eigenvalue curves.

For fixed n, the power function is strictly decreasing. The larger the value of n, the more steeply the function descends. A feature of the power function is that its curvature increases as n increases. Figure 6.8 depicts this situation. In our

experimentation, given the eigenvalue curve, it can always be fitted by the power function surprisingly well (for example, see Figure 6.7).

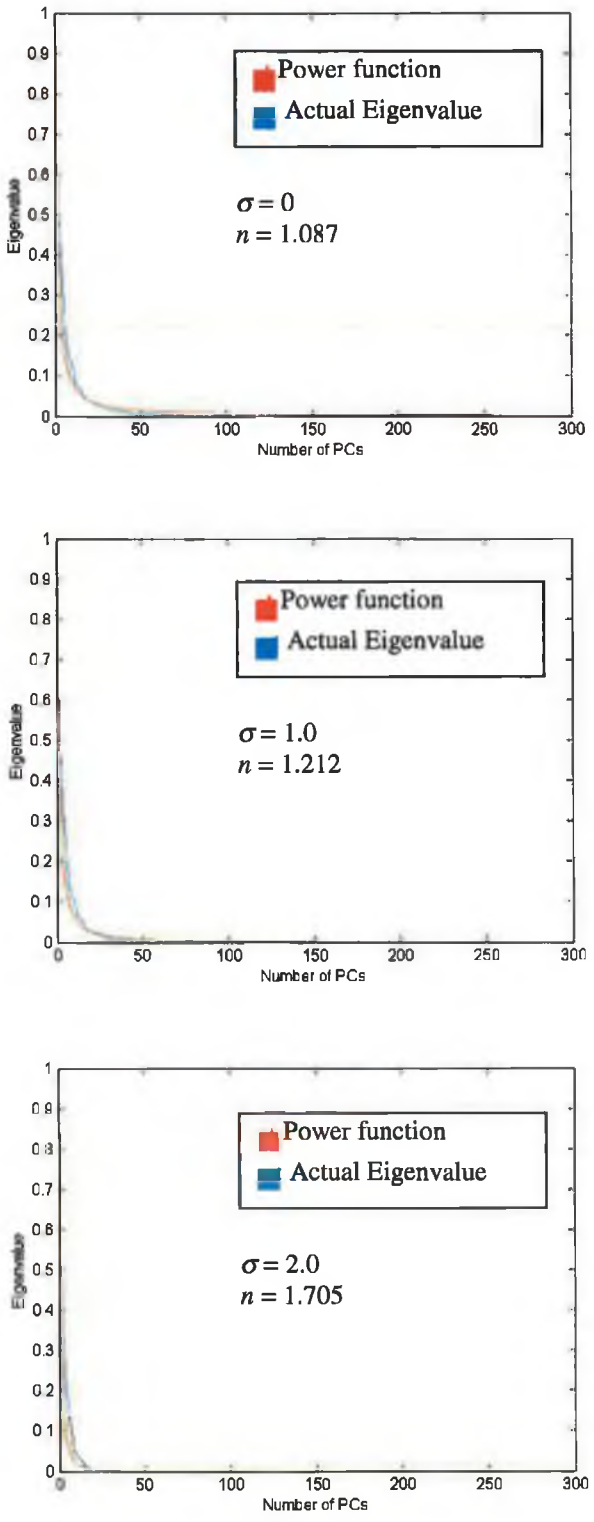


Figure 6.7: Results of curve fitting: power function in Eqation 6.4 is used to fit three eigenvalue curves in figure 6.2. Only first 300 PCs are shown for the ease of visualisation.

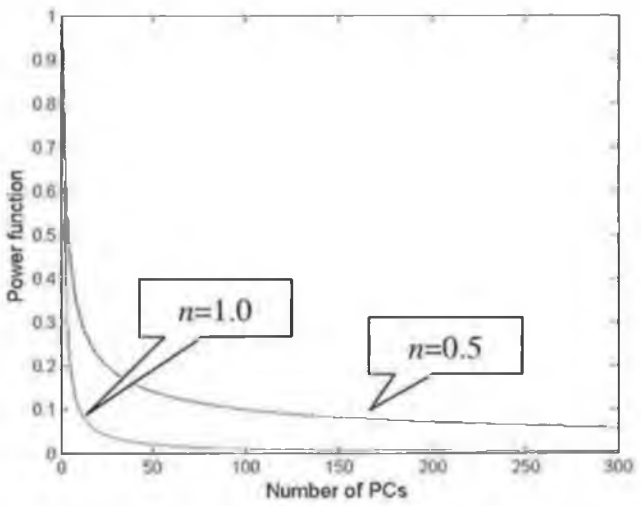


Figure 6.8: An example of the power function. Only first 300 PCs are shown for the visualisation reason.

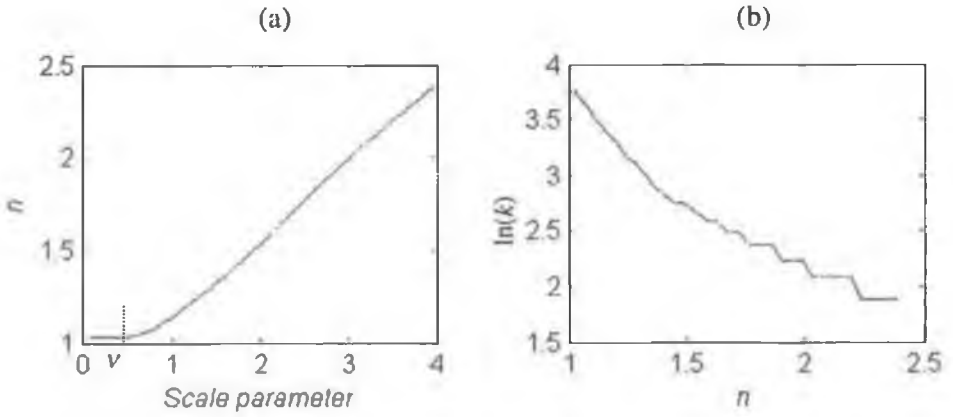


Figure 6.9: Relationship between σ, n, and k. ln(k) in (b) stands for the natural logarithm of k.

Varying the scale parameter, the eigenvalue curve will change, hence affecting the value of n in Equation 6.4. The change of the eigenvalue curve will also affect k, the dimensionality of the PC space. Figure 6.9 shows that there exists a simple relationship between the scale parameter, n, and k. It reveals the following three facts:

1). See (a). When σ is smaller than a certain value v, n keeps constant. At this stage, noise is being removed but the underlying structure remains unaltered.

2). See (a). When σ is greater than the value v, n increases with respect to σ. At this stage, the image information is concentrated on the PCs corresponding to the first few greatest eigenvalues. As σ increases, the dimensionality is reduced and the eigenvalue curve becomes steeper, therefore n increase.

3). See (b). n increases linearly as the value of $ln(k)$ decreases. Hence, σ also increases linearly with respect to the decreasing of $ln(k)$: an exponential relationship exists between σ and k. We assume that k should increase linearly so that the fine structures can also appear in a linear way. Based on this, we have the conclusion that σ should be reduced logarithmically. In practice, we define a new variable t to replace σ:

$$t = c \log \frac{\sigma}{\varepsilon} \qquad \ldots(6.5)$$

where c and ε are two scaling constants (empirical values $c = 0.1$ and $\varepsilon = 20$ are used in our experiments). The change of t causes the logarithmic change of σ.

Although we have revealed the relationship between σ and k, we do not know yet the best value of σ to start at the root layer. Neither do we know the best reduction step, i.e. how much should we reduce the value of t from one layer of the tree to the next layer. There is no available background theory for these two questions. In our system, we solve them by trial and error. In practice, we take the value of t=54 to start at the root level, and then reduce the value by 6 for each next layer. If t reaches zero while the construction of the decision tree is still continuing, the value of t remains zero, i.e., the images will not be further blurred. In our case, t = 54 and the decrement step is 6, so that after the ninth layer, the data will not be blurred any more since there are few data left after many splits.

6.4.2 Static Gesture Recognition

Using the principles introduced above, we can construct a hierarchical decision tree fully automatically based on the training images. Although the system has the potential for a very large vocabulary, we only show the recognition result for the 23 finger-spelling gestures in ISL. The evaluation results are shown in Figure 6.10.

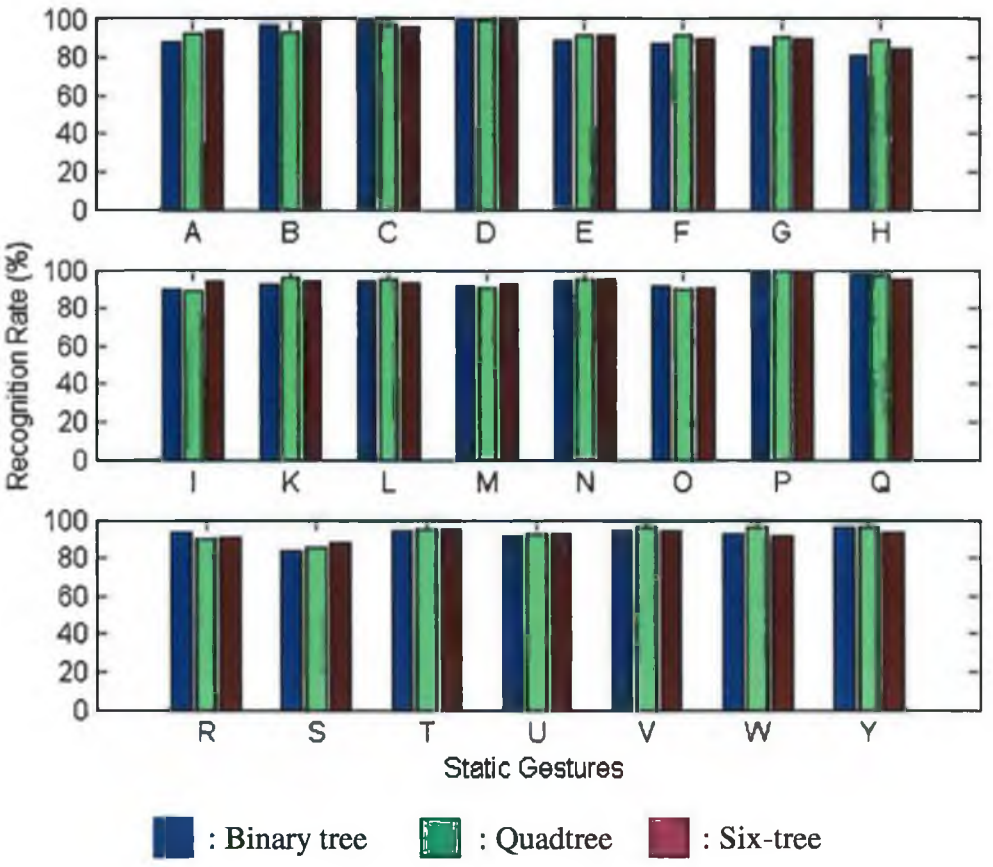


Figure 6.10: Evaluation results of the hierarchical decision tree using 23 static gestures taken from ISL.

Various types of decision tree were used, including binary tree, quadtree, and six-tree. The depths of the decision trees are 11, 4, and 3 respectively. The experimental results are 92.2%, 93.4% and 93.3% respectively. The multiple branch trees show a small advantage against the binary tree. We will use quadtree for our future research, specifically in the next chapter to recognise dynamic gestures because it obtained the best recognition rate.

Values of t other than 54 are also possible. In our experience, as long as the value of t is restricted between 50 and 60, no significant change occurs in the test results. In terms of the decrement step, empirically, values between 4 to 7 are good choices. When the scale parameter has reduced to zero, i.e. t =0, even if the decision tree still keeps splitting data, the data stop blurring.

The confusion matrix of the classification results using the quadtree is plotted as an image in figure 6.11. The rows represent the true classes of the test images, while the columns represent the classes that the images have been classified into. The intensity of each square pixel indicates the recognition rate. Ideally, only pixels on the diagonal should be highlighted. That is, the off-diagonal pixels imply the misclassified images. Note that for visualisation reasons the intensities of the misclassified ones have been exaggerated.

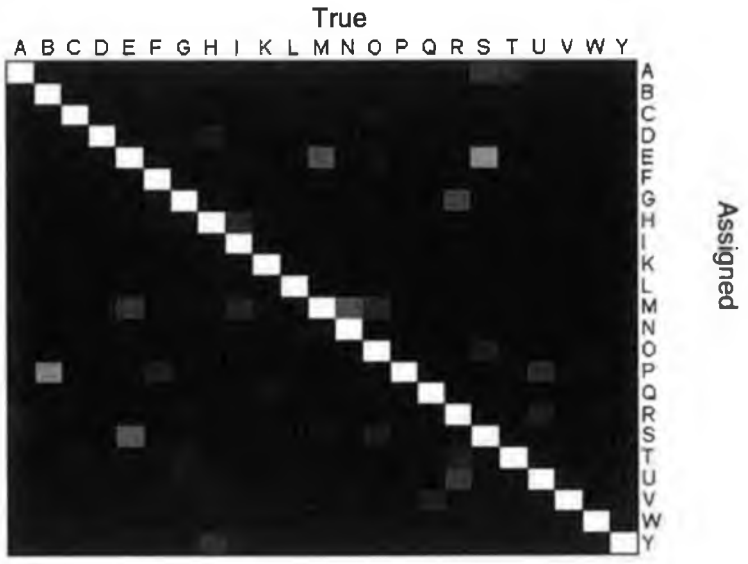


Figure 6.11: Visualisation of the confusion matrix.

- **Error analysis**

The recognition error is caused by visual similarity and can be divided into two types:

1. Intrinsically, similar gestures will have similar images, which will cause error. In the above test, most of the errors in "S" were misclassified into "E" and "A". Figure 6.12 illustrates the three hand shapes.

Figure 6.12: Visually similar gestures. From left to right: A, E and S. The finger-spelling of these three signs are similar. (Pictures taken from [21].)

2. Using a single camera, from a specific view angle, two intrinsically dissimilar gestures could look very similar. The vast majority of the errors are due to this reason. For example, the side views of "H" and "D" are very hard to distinguish since the little finger would be overlapped with the index finger from the side view when performing "H". Another example is "B" and "P". These look rather similar from the front view. Figure 6.13 illustrates this type of error.

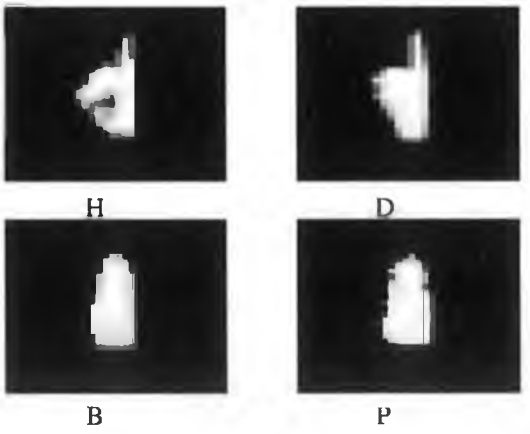


Figure 6.13: Two examples to show that some gestures look similar from a specific view angle. From the side view, "H" looks similar to "D", and from the front view, "B" looks similar to "P".

The error caused by the visual similarity is very hard to eliminate since either gestures are inherently similar or appear so because of use of a single camera. However, apart from these errors, the recognition rate is high and the computational cost is little. One of our main aims is to recognise dynamic gestures. As we will see in the next chapter, when applied to the recognition of dynamic gestures, the error caused by the visual similarity is not a problem any more, because most of the dynamic gestures are made

either by very different hand shapes or by similar hand shapes combined with completely different hand movements.

- **Results Comparison**

We now compare the evaluation result using the hierarchical decision tree method with the other two types of method: multiple-subspace classifier and hierarchical decision tree method without blurring. Figure 6.14 illustrates the comparison:

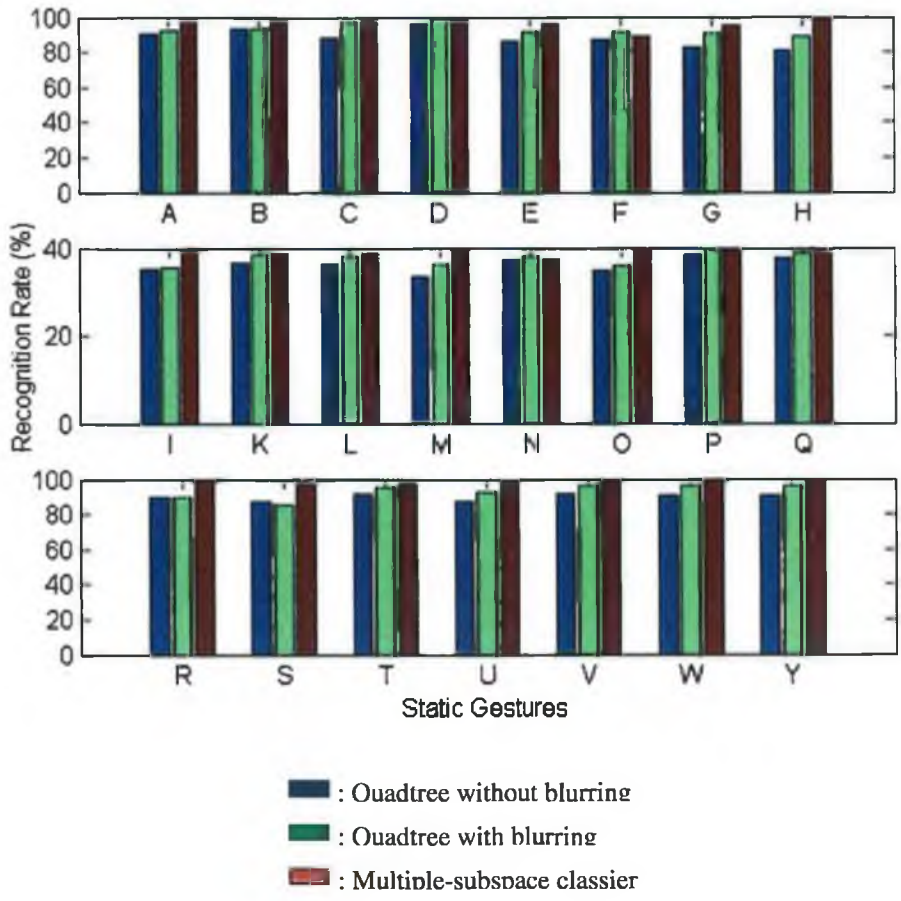


Figure 6.14: Performance comparison using three types of method.

The average recognition rate using quadtree without blurring is 89.5%, comparing to 93.4% when blurring is used, 3.9% improvement is achieved. When multiple-subspace classifier is tested, the dimensionality of each subspace is decided by Equation 2.8, that is, at least 95% of energy is retained. In this case, the average recognition rate is 97.8%, which outperforms the hierarchical decision tree method by 4.3% even when blurring is used. However, the recognition time is reduced logarithmically.

6.5 Summary

In this chapter, the hierarchical decision tree was combined with multi-scale theory to speed up the recognition procedure. Logarithmic time saving is achieved in comparison with the system using subspace classifiers. We discussed the construction of the hierarchical decision tree. A power function was used to model the eigenvalue curves. The direct result of this model is that increasing the scale parameter logarithmically reduces the structure in the images – an important point, not only for gesture recognition but also for any other image processing tasks that needs multi-scale theory.

All the above ideas are novel and as far as the author knows these ideas have never been reported by anybody else in any format. More importantly, as we will see in the next chapter, the hierarchical decision tree plays an important role in the recognition of dynamic gestures. It will be used as a hand configuration extractor to extract important static hand shapes from a sequence of images: another new idea presented in this thesis. We will discuss this in the next chapter.

Chapter 7 Dynamic Gesture Recognition

7.1 Introduction

In the previous chapters, we discussed how to recognise static gestures as well as improve the recognition speed using the hierarchical decision tree combined with multi-scale theory. However in reality people use dynamic gestures to communicate and express themselves much more often than static gestures. The recognition of dynamic gestures is an essential task for a good gesture recognition system.

In recent years HMMs have attracted increasing attention. Since Starner and Pentland [92] applied them to recognition of ASL sentences many other researchers have employed them in their systems, such as Vogler and Metaxas [93] and Wilson and Bobick [87]. In fact, HMMs are a rather general mathematical model that are able to handle the temporal variability of a dynamic process. Different researchers use various features as the input, hence the differences between their systems. Many of them only use basic geometric parameters of the hands or other simple features. For instance, Starner and Pentland use a simple feature set to describe the hand shape which consists of the x and y position of each hand, angle of axis of least interia, and eccentricity of bounding ellipse. Lee and Kim [86] took advantage of the hand centroid, divided the 2D plane in the image into 16 directions, and used the direction of the movement of the hand centre as the feature vector. Naturally, we can imagine that simple feature vectors could cause problems because different complex gestures could have very similar simple features. In this case, when the size of the vocabulary increases, the coincidence between features will become more severe. However, using complex features will increase the workload of the system, and thus slow down the real-time performance.

In this chapter, we describe a novel methodology that combines the hierarchical decision tree recogniser and HMMs together to achieve good recognition rate as well as retaining low time complexity. Three factors are crucial for any dynamic gesture: hand configuration, hand movement trajectory and the relative position of the hand and other parts of body. Currently, we only focus on the recognition of the first two factors. We have developed our algorithm based on the idea that we can consider

these separately. Firstly, we consider the hand configuration factor. We treat each frame as a static hand-shape. We do not use information from neighbouring frames to recognise the shape, although it will be considered in future work. We use the decision tree recogniser to extract the spatial configurations from the hand image since each leaf of the tree represents a specific hand shape. Secondly, we represent the global movement of the hand with a direction code which can only take one of the eight values: right, up right, up, up left, left, down left, down and down right. If we label each leaf in the decision tree, the label of the leaf and the direction code compose a new two-dimensional vector that is then used as the input to the DHMM (Discrete HMM) recogniser. Since the decision tree has the potential to handle even complex hand configurations, it overcomes the shortcoming of using simple feature vectors as used in many of the systems discussed above. In the meanwhile, it still retains the simplicity of the input feature vector of the DHMM, only two dimensions in our case; hence the time complexity is low. Both the decision tree recogniser and DHMM are so fast that theoretically there is no problem for the system to deal with large numbers of gestures in real-time.

The remainder of the chapter is divided into three parts. Section 7.2 discusses the construction of the feature vector, including the use of the hierarchical decision tree as a hand configuration extractor and the direction code of the hand movement. Section 7.3 gives the experimental results where we use our system to recognise 35 dynamic gestures selected from ISL. Finally we summarise.

7.2 Feature Extraction for the DHMM Recogniser

Different systems use different features. We may classify the previous approaches based on HMMs into three categories according to the form of their features: continuous HMMs, semi-continuous HMMs, and DHMMs. The systems based on both continuous HMMs and semi-continuous HMMs use features with continuous formats. That is, these features are described by probability density functions, for instance Guassian or mixed Gaussian. In contrast, the systems based on DHMMs use features with discrete format. That is, the feature vector space is partitioned into several regions, and every vector is replaced by the identifier, such as 1,2,... or A, B,..., of the region it belongs to. Most of the previous systems [87, 92, 93] are of

continuous or semi-continuous types due to their ability to deal with complex situations. DHMMs were only used in systems with very small vocabulary [86]. In terms of the speed, DHMMs have the advantage over the other two types.

To achieve both low computational cost and robustness, we have used DHMMs in our system, and have developed our own feature vector, which is completely different from the single features used in work by others [92, 93].

Figure 7.1: Three examples from ISL. Pictures are taken from [21]. All examples have specific meanings in ISL, however, this is not important for us. Thus we refer them as "gesture 1", "gesture 2", and "gesture 3".

Let's start from a few dynamic gesture examples that are illustrated in Figure 7.1. When a tutor teaches others these gestures, he would use the following sentences to describe them [21]:

- Gesture 1: Hold the hand in the "T" position at chest level, then move it to the right changing to the "V" position.
- Gesture 2: Hold the hand in the "A" position beside left cheek, then move it to the right.
- Gesture 3: Hold the hand in the "L" position, then swing it to the right.

("T", "V", "A", and "L" positions are in terms of the static gestures in ISL)

The above teaching method shows three key issues when describing a dynamic gesture:

1. The hand configuration is important. For example: "T", "V", "A", or "L" position.
2. The movement of the hand is important. For example: move it to the right, swing it to the right, and so on.
3. The relative position of the hand against other parts of the body is important. For example: chest level, left cheek, and so on.

If our system can fully integrate these points together, we should be able to achieve good results. Unfortunately, the third point is related to segmentation and recognition of other parts of body, which is beyond our current research. Thus, our current system attempts to combine the first and second point into the feature vector. Furthermore, we notice it is harder to handle the first point than the second. People describe the hand movement in a rough way, such as move your left hand to the right. When designing a dynamic gesture, the designer will not describe one gesture as "move your hand to the 63 degree direction up. Be careful, don't do it along the 70 degree, it means something else". As opposed to this, human hands show much more variety in term of its configurations, or shapes. For example, there are about 40 different hand configurations in ISL.

Based on the above thoughts, our feature vector is designed to consist of two elements. One simple direction code represents the global movement of the hand, the other represents the local configuration of the hand. It can be formalised as:

$$I = [I_C, I_D] \qquad \ldots(7.1)$$

where I_C is the hand configuration element, and I_D is the direction element. Each of these is a positive integer number. This two-dimensional discrete feature vector gives the DHMM recogniser the ability to process image sequences at a very fast speed. Meanwhile, it can fully describe even complex hand shapes, and thus can handle complex dynamic gestures. A dynamic gesture is then represented by a sequence of feature vectors. For example, given an example for gesture 1 in figure 7.1, it might be translated into:< [366, 7], [509, 6], [509, 5], [509, 4], [359, 4], [148, 3], [23,3], [23,2] >, while an example of another gesture might be translated into: <[355,1], [509,6], [441, 5], [441, 5], [441, 5]>.

In the rest of this section, we introduce how to apply our hierarchical decision tree as a hand configuration extractor to handle the configuration change, and how to construct the direction code to handle the position change of the hand. By combining the local and global information together, we are able to construct the essential feature vector for the DHMM recogniser.

7.2.1 Hand Configuration Extractor

The decision tree presented in the preceding chapter can separate a set of training data into several groups in a hierarchical way according to their similarities. All the images in one leaf should contain similar gestures: not necessarily the same gesture, but similar in their two-dimensional images. If we apply the same method to a set of training images of dynamic gestures instead of static gestures, the leaves in the tree would contain the different hand shapes, or configurations, that appear in the training images. In other words, the hand configurations are "extracted" from the training images. Hence, we call the hierarchical decision tree a *hand configuration extractor*.

We use gesture 1 as an example. 60 examples with the length varying from 5 frames to 12 frames are acquired continuously. We then compute a hierarchical decision tree based on these data. In total seven leaves are extracted out under the termination condition that the data variance in each leaf is smaller than 0.5. Figure 7.2 shows the mean images of these leaves.

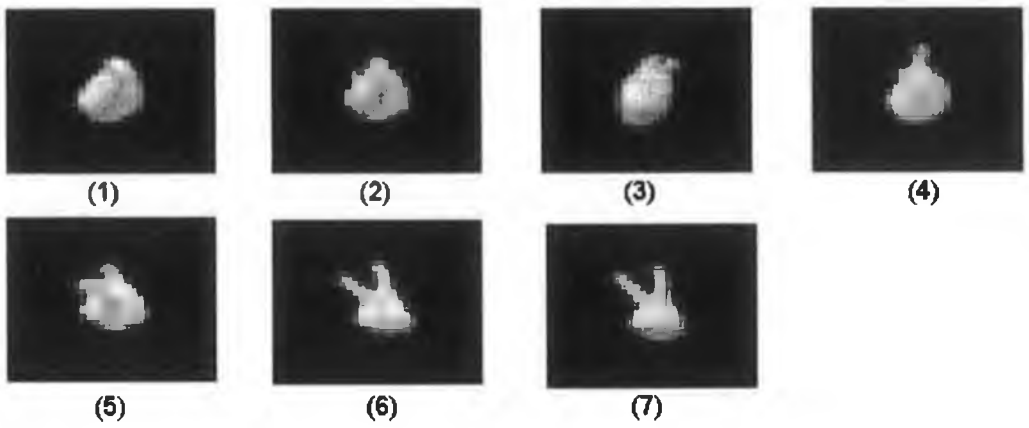


Figure 7.2: Mean images of the decision tree made from gesture 1. The seven images show how one performs the gesture 1: starting with a shape of a fist in (1) and (2), hold hand in "T" position (3), and change into "V" position. (4) and (5) are the intermediate steps of the change.

By looking at the seven images in Figure 7.2, one can have a basic idea of the dynamic procedure of gesture 1: starting with the shape of "T", then changing into the shape of "V". If we number these seven leaves from 1 to 7, each number actually fully reflects one specific hand configuration, and thus can be used as a hand configuration descriptor.

In general, given the training data, we first construct a hierarchical decision tree based on all training data. Then we give a single number to each of the leaves. Thus each number now represents a specific hand shape that has been included in the training gestures. This number is the hand configuration element, I_C, in the two-dimensional feature vector.

7.2.2 Direction Code

The hand configuration extractor is able to extract the hand shapes from the dynamic gestures. However, it does not involve the information on hand movement, which is also important for recognition. Our solution is to treat the hand as a whole object when dealing with the global movement, and use the movement of its centroid to represent the movement of the hand. One thing which needs to be noticed is that the absolute position of the hand centroid is not important. Instead, only the direction of the hand movement is useful for recognition, for example, "move the hand to the right". Thus we define our direction code according to the moving direction of the hand centroid.

Given a sequence of images, we define the direction code I_D based on every two consecutive images f_t and f_{t+1}. Assume the coordinate of the hand centroid in f_t is $<x, y>$, and in f_{t+1} $<x', y'>$. A vector is computed as $<x'-x, y'-y>$. Then we translate this vector into the direction code I_D that is one of 8 directions. See Figure 7.3. Dividing the two dimensional image plane into eight directions is enough since we often say " move to right" or "move up", sometimes we say " move your hand to the up left direction", and we would never say "move your hand along the 63 degree direction".

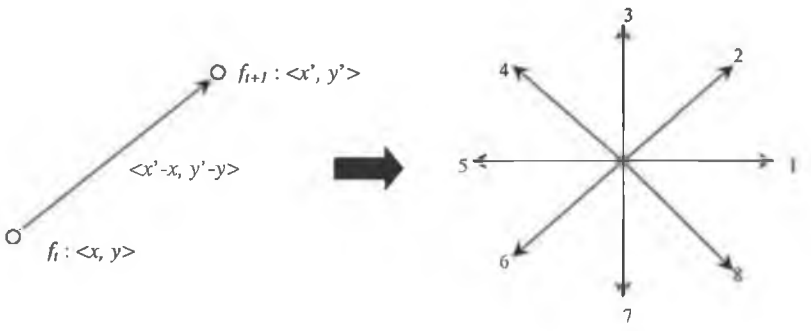


Figure 7.3: Definition of the direction code. Given two consecutive images, a vector <x'-x, y'-y> can be computed, which is then translated into one of the eight directions.

7.3 Evaluation

To evaluate the method presented above, 35 gestures are selected from ISL. Each of them has a specific meaning in ISL. For convenience, we name them in the form of "gesture 1", "gesture 2",..., and "gesture 35". Although many gestures in ISL involve both hands, we only concentrate on the gestures using one single hand.

7.3.1 Training of the system

The complete training procedure is composed of two steps. First the hand configuration extractor needs to be constructed from the given training data. Second, the DHMM recogniser also needs to be trained for the purpose of recognising dynamic gestures. These two steps have to be done sequentially. That is, only after we get the hand configuration extractor, can we train the DHMM recogniser. Figure 7.4 shows the two-step training procedure.

In practice, we took 35 segments of video. Each of which contains one dynamic gesture with 60 examples recorded continuously.

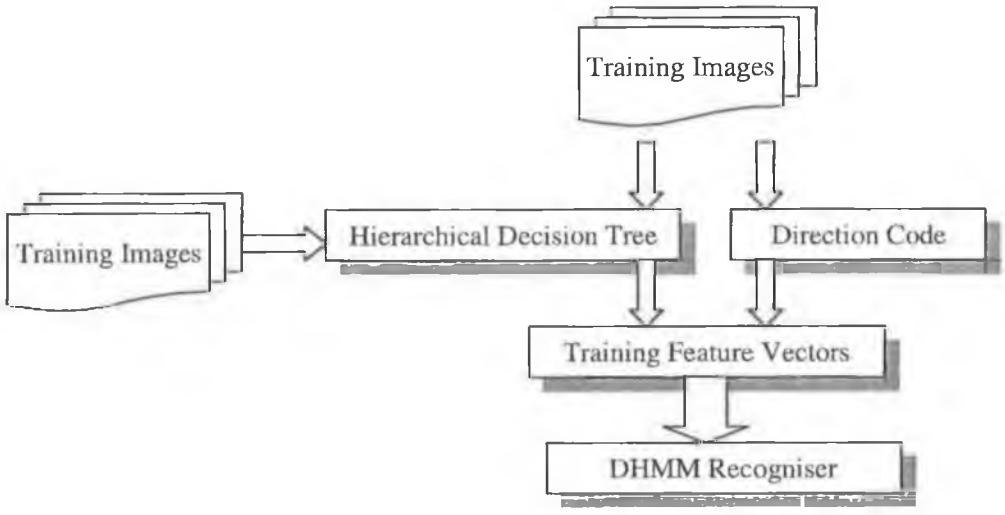


Figure 7.4: The two-step training procedure: first the hierarchical decision tree has to be trained. Then the set of training feature vectors can be made to train the DHMM recogniser.

When training the hand configuration extractor, the procedure introduced in the preceding chapter is applied to the whole training set. The parameters chosen are as follows: t =54 was applied to the root of the quadtree, and the reduction step was set to 6. After training, the depth of the tree was 7.

After the construction of the hand configuration extractor, the same training set is fed into the extractor. Meanwhile, the direction code of every two consecutive images is computed. Combining the outputs of the hand configuration extractor and the direction codes, we have the training feature vectors that will be used to train the DHMM recogniser.

A DHMM model was trained for each of the 35 sets of feature vectors, each representing one dynamic gesture. The DHMM models used here have the same topology as the one in Figure 2.6, i.e. a first-order DHMM with seven states including the start and end states. The Hidden Markov Model Toolkit (HTK) produced by Entropic plc. is used. To avoid the manual segmentation of the training examples, a standard embedded training procedure, explained in [22], was used.

Note during the above procedure, The 35 video segments are acting as a single aggregated training set for the training of the hand configuration extractor. Only one

hand configuration extractor is output from the training procedure. When training for the DHMM models, each gesture should have its own DHMM. Thus 35 DHMM models are needed, and each of them should be trained using its corresponding set of training feature vectors.

7.3.2 Test Results

We grabbed another 35 segments of video; each of which contains one dynamic gesture with 60 examples recorded continuously, which were never used for any portion of the training. The test was evaluated offline by HTK's Viterbi recogniser [22].

Because both the training and test sets consist of continuous video sequences, which have not been manually segmented, it is not possible to easily change the size or membership of the training and test sets. So it would be very difficult to carry out cross-validation or determine the effects of changing sample size.

Given a video sequence, the recognition procedure is as follows: first every frame from the video is input into the hand configuration extractor, a sequence of numbers represents the hand shapes obtained. Meanwhile, a sequence of direction codes is also computed. Using these two sequences of numbers, the feature vectors are calculated. The DHMM recogniser, implemented by the Viterbi recogniser using the Viterbi algorithm (see Section 2.4.3), reads in the 35 DHMM models as well as the feature vectors, and classifies the sequence into the DHMM with highest probability. This sequence might contain a few dynamic gestures. All of them will be extracted by the recogniser.

The recognition results are given in Table 7.1. For convenience, a bar chart of the recognition results is also shown in Figure 7.5.

No grammar was used during the test. Three types of error are presented: deletion (*D*), insertion (*I*), and substitution (*S*). We use an example to illustrate the differences between them: given a sentence "I am a student", the deletion error would be " I a student", the insertion error would be " I am a student student", and the substitution

would be " I am a teacher". Based on the three types of error, two kinds of performance measures are presented: the correct percentage (Corr) and the accuracy percentage (Acc) [22]:

$$Corr = \frac{N - D - S}{N} \times 100\% \quad \ldots(7.2)$$

$$Acc = \frac{N - D - S - I}{N} \times 100\% \quad \ldots(7.3)$$

where N is the total number of the dynamic gestures in each test set, 60 in our case. The absolute values of D, S, I are given in the table, while the values of $Corr$ and Acc are shown in terms of percentage.

Table 7.1: Recognition Results of 35 Dynamic Gestures.

Gesture	Corr(%)	Acc(%)	D	S	I
GESTURE 1	75	75	1	14	0
GESTURE 2	100	100	0	0	0
GESTURE 3	100	100	0	0	0
GESTURE 4	97	97	0	2	0
GESTURE 5	82	82	10	1	0
GESTURE 6	93	93	2	2	0
GESTURE 7	80	80	0	12	0
GESTURE 8	100	100	0	0	0
GESTURE 9	97	97	2	0	0
GESTURE 10	100	98	0	0	1
GESTURE 11	100	98	0	0	1
GESTURE 12	100	100	0	0	0
GESTURE 13	100	100	0	0	0
GESTURE 14	93	93	2	2	0
GESTURE 15	100	98	0	0	1
GESTURE 16	100	100	0	0	0
GESTURE 17	100	87	0	0	8
GESTURE 18	100	98	0	0	1
GESTURE 19	100	100	0	0	0
GESTURE 20	95	95	2	1	0
GESTURE 21	92	92	4	1	0
GESTURE 22	100	100	0	0	0
GESTURE 23	100	98	0	0	1
GESTURE 24	80	80	12	0	0
GESTURE 25	100	100	0	0	0
GESTURE 26	100	100	0	0	0
GESTURE 27	100	98	0	0	1
GESTURE 28	100	98	0	0	1
GESTURE 29	92	92	5	0	0
GESTURE 30	92	92	5	0	0
GESTURE 31	100	90	0	0	6

GESTURE 32	100	100	0	0	0
GESTURE 33	82	78	0	11	2
GESTURE 34	87	52	0	8	21
GESTURE 35	90	90	5	1	0

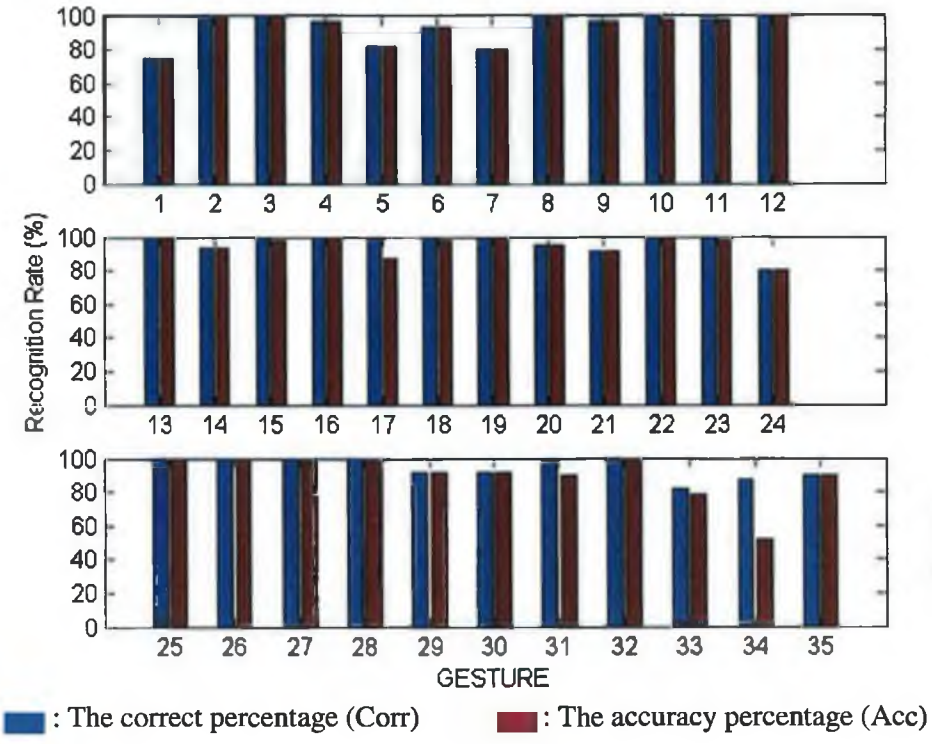


Figure 7.5: The bar chart of the evaluation results of 35 dynamic gestures.

The three types of error deserve different treatments. Insertion error, when it is caused by repetition, is the easiest one to eliminate. A simple restriction, for example no repetition of the same word, reduces it to zero. However, other types of insertion, for instance the sentence is interpreted as "I am a teacher student", are hard to solve. The deletion problem is more severe than the repetition error but less severe than the substitution problem. For example, the computer might be able to finish the sentence "I a student" automatically, while it is nearly impossible to ask the computer to change "I am a teacher" into "I am a student". The substitution problem can not be improved by giving grammar restrictions. Under this criterion, gesture 1, 7 and 33 were badly recognised. 14, 12 and 11 substitution errors happened out of a total number of 60 respectively.

Most of the errors in gesture 1 were misclassifed into gesture 27 because of the similarity between their hand configurations. Gesture 1 holds the hand in the shape of "D". Gesture 2 holds the hand in the shape of "W". From a specific viewpoint, the two shapes are similar. The ambiguity is essentially caused by using a single camera. No good solution is available in the current situation.

Most of the errors in gesture 7 were misclassified into gesture 34. Both gestures have the same change in terms of the hand configurations. The difference is that gesture 7 is moving the hand backwards, while gesture 34 needs to move the hand to the right. This error shows the weakness of our system on handling 3D hand movements. Since no depth information was considered during the recognition, the backward movement sometimes was treated the same as moving to the right. The information on hand area is not much help either, because the backward movement is not very significant compared to the distance from the hand to the camera. 3D depth recognition is another open problem for our system, and more research has to be done in future.

The error in gesture 33 is a different type: most of the errors were misclassifed into gesture 1. Both gestures hold the hand in the same hand shape of "D", and both of them perform similar hand movements: from the upright to down left. The differences between them is that the hand movement is a curve and is performed at chest level, while gesture 33 should perform as a straight line at face level. It could be improved effectively if the system had the ability to recognise the relative positions of the hand against other parts of body.

7.4 Summary

In this chapter, we presented a novel approach that is able to recognise dynamic gestures using a hierarchical decision tree combined with multi-scale theory and DHMMs. Given several training sets, each representing one dynamic gesture, first all of them are treated as a single but aggregated training set to construct a hierarchical decision tree using the method discussed in the last chapter. Then the training sets of feature vectors are computed with the decision tree and direction codes. A number of DHMMs thus can be trained. The recognition procedure is as follows: given a sequence of images, each image will be input to the decision tree, and a number that

represents the leaf to which the image belongs will be obtained. Meanwhile, a set of direction codes is also computed from the image sequence. The feature vector set is thus constructed, and is fed into the DHMM recogniser. The images will be classified into the DHMM(s) with the highest probability.

The Viterbi algorithm is used for recognition. Hence the time complexity of the recognition using the DHMM recogniser is the same as that of the Viterbi algorithm: $O(N^2T)$, where N is the number of states in DHMM (7 in our case), and T is the length of the image sequence about to be recognised. Since N is a constant, the time complexity then becomes $O(T)$.

Chapter 8 Conclusion and Perspective

8.1 Summary

This thesis has attempted to address the problem of gesture recognition for low computational cost using only a single camera, in order to make the system as widely available as possible. Current approaches are either restricted to recognising a limited number of gestures so that simple methods can be adopted, or recognising a large vocabulary using complex methods. Simple methods limit the capability of systems, while complex methods imply high computational costs so that the applicability would be restricted. The author has attempted to take into consideration both issues to develop a system that has the potential to recognise a large set of vocabulary and maintains as low a computational cost as possible. Given this aim, a gesture recognition system has been presented, based on a novel method combining a hierarchical decision tree with multi-scale theory and DHMMs.

Chapter 2 gave the definitions and notations of the terms. It also introduced the background knowledge of PCA, multi-scale theory, and DHMMs. Chapter 3 examined the past developments in the research of hand gesture recognition.

Chapter 4 to 7 involved most of the work done by the author. A general overview of the system was given in Chapter 4. Chapter 5 demonstrates using standard PCA to recognise 23 finger-spelling gestures in terms of three types of classifier. The evaluation results of this chapter show that the multiple-subspace classifier outperformed the other two classifiers which used a single PC space. The disadvantage of the multiple-subspace classifier is that its time complexity grows linearly with respect to number of gestures, which makes the recognition hard for a large vocabulary on an entry-level computer.

To address the linear time complexity problem, we presented a novel method in Chapter 6: a hierarchical decision tree was constructed from the training set, with each node being a PC space. The multi-scale theory was integrated into the construction of the tree to reduce the noise level in the training images and the dimensionality of PC spaces. When applied to the online recognition, the time complexity of the

hierarchical decision tree grows logarithmically according to the number of gestures. This gives our system the potential to recognise a large number of gestures without increasing the processing time much. During this chapter, a power function model was proposed for the purpose of scale parameter selection. We revealed that the scale parameter of Gaussian kernels should be reduced logarithmically from the top to the leaf level during the construction of the tree. The errors in the experiments were caused by two reasons: either two gestures are intrinsically similar or appear so due to the use of a single camera.

To recognise the dynamic gestures, the hierarchical decision tree was further used as a hand configuration extractor to get the intermediate hand shapes from dynamic gestures. The movement of the hand was represented by the direction code. A dynamic gesture was thus represented by a sequence of two-dimensional vectors, which was fed into a DHMM recogniser, where it would be classified into the DHMM with the highest probability. The approach was then evaluated by 35 dynamic gestures selected from ISL. The errors were caused by three reasons: similarity of hand shapes, similarity of hand movements, or lack of the ability to recognise other parts of body.

Looking through the recognition procedure of a gesture, the time complexity of the whole system is composed of three parts: data pre-processing, hierarchical decision tree, and the DHMM recogniser. Since the time spent on the data preprocessing is a constant c_1, i.e. it does not change with the number of gestures, the time complexity on this part can be ignored. As stated in the last section of chapter 7, given a sequence of images, the time complexity on each DHMM model is $O(T)$, where T is the length of the image sequence. Thus for each frame in the sequence, the average time complexity is another constant c_2 ($O(T)/T$). If the size of the vocabulary is fixed to be X, the average time spent on each frame would be fixed as Xc_2. Hence the time complexity of the whole system is only based on the hierarchical decision tree, which is logarithmically proportional to the size of the vocabulary.

8.2 Future Work

First of all, we assumed our system worked under normal office illumination conditions (see Section 4.1), and currently the user has to wear a glove. More research

needs to be done to remove these two constraints. That is, we will attempt to segment the hand against complex background without requiring the coloured glove. A relatively new technique could be taken into consideration: the Condensation algorithm: which is based on sampling theory and can track objects in real-time even in an environment where many similar colours exist [96]. The algorithm also shows robustness against the variation of illumination condition [96].

Second, in the current hierarchical decision tree, a still image is classified into one of the leaves without considering its previous images. The knowledge of the neighbouring frames could speed up the process, and improve the accuracy of the classification. For example, given a frame, we can ignore some of the branches in the decision tree given information from its previous frames. We are attempting to integrate this feature into our system.

Thirdly, to eliminating the error caused by the similarities of the hand movements during the recognition of dynamic gestures, we should recover the 3-D world co-ordinates of the hand rather than 2-D image co-ordinates (as we are doing now). Using the information of hand area, centroid and neighbouring images, this should be possible.

Apart from the above, there are many other possible research areas. For instance, one difficulty in statistical approaches is the acquisition of training data (see Section 3.2.2.2). The training database of speech recognition systems often includes the contributions from hundreds of people and years of work. We think computer animation techniques might be able to help us with this issue. By using computer animation, it is possible to construct a large training database. Since the advantage of our system will be more obvious when applied to a large vocabulary, we would be able to make the system recognise hundreds of dynamic gestures with low computational cost.

Currently, the DHMM recogniser has a fairly simple topology: first-order DHMMs. A couple of improvements will be researched in future:

1. Adding extra transition paths, i.e. higher order DHMMs, will make the DHMM recogniser able to deal with more complex situations, such as partially occluded

gestures. See figure 8.1, where a second-order HMM is given and transitions can happen between state 1 and 3, 2 and 4, 3 and 5 without passing their intermediate states. Hence even an occlusion appears, say at state 3, the recognition can still continue by jumping from 2 to 4 directly. In fact, other members of the research group are developing a system to recognise partially occluded gestures using graph matching. We are attempting to assess the advantages and disadvantages of both techniques so that better understanding of the occluded gestures can be obtained.

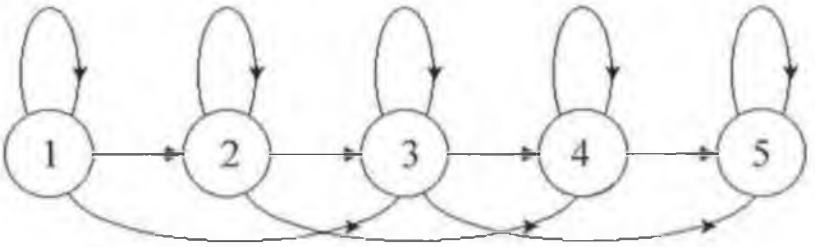


Figure 8.1: A second-order HMM.

2. Another problem is co-articulation. This is the phenomenon whereby each gesture is influenced by the preceding and following gestures in a continuous sequence. This problem also occurs in speech recognition, and requires some adaptation of the HMMs. We will have to predict transitional forms between one gesture and another, i.e., predict the transition probability from the end state of one DHMM to the start state of another DHMM. Our ultimate target is to establish a DHMM network where each node is an individual DHMM, the links between the paths are probabilistic and decided by some constraints, grammar for example.

Finally, as we stated in Chapter 3, face and the other parts of body are also very important for the hand gesture recognition. Thus, it is worthwhile to put efforts into the recognition of these. On the other hand, the methodology developed in this thesis can be naturally extended for the recognition of face and other parts of body (or even in broader areas, for example, hand-written character recognition, vehicle detection, target recognition, and so on) since all of the tasks fall into the general field of object recognition.

References

[1] Y. Wu, T. S. Huang, "Vision-Based Gesture Recognition: A Review", in *Proceedings of the International Gesture Recognition Workshop*, GifsurYvette, France, 1999, 103-115.

[2] D. M. Gavrila, "The Visual Analysis of Human Movement: A Survey", in *Computer Vision and Image Understanding,* 73(1), 1999, 82-98.

[3] V. Pavlovic, R. Sharma, and T. S. Huang, "Visual interpretation of hand gestures for human-computer interaction: A review", *IEEE Transaction on Pattern Analysis and Machine Intelligence,* 19(7), 1997, 677-695.

[4] K. P. Murphy, "Passively learning finite automata", *working paper 96-04-017,* Santa Fe Institute, USA, 1996.

[5] R. A. Bolt, "Eyes at the interface", *in Proceeding of ACM Human Factors in Computing Systems Conferenc*e, Gaithersburg, USA, 1982, 360–362.

[6] I. Pitas, "Digital image processing algorithms ", *Prentice Hall*, New York, USA, 1993.

[7] S. Theodoridis, K. Koutroumbas, "Pattern Recognition", *Academic Press*, San Diego, USA, 1998.

[8] E. Gose, R. Johnsonbaugh, and S. Jost, "Pattern Recognition and Image Analysis", *Prentice Hall*, New Jersey, USA, 1996.

[9] D. D. Lee, H. S. Seung, "Learning in Intelligent Embedded Systems", in *Proceedings of the Embedded Systems Workshop,* Cambridge, Massachusetts, USA, 1999.

[10] K. Trish, D. M. Sylvia, and S. H. Ali, "SNAP & TELL: A Vision-Based wearable system to support 'Web-On-The-World' Applications", in *Proceedings of the conference on Digital Image Computing - Techniques and Applications (DICTA),* Melbourne, Australia, 2002.

[11] A. Utsumi, J. Ohya, "Direct Manipulation Interface Using Multiple Cameras for Hand Gesture Recognition", in *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, Austin, Texas, USA, 1998, 264-267.

[12] L. Jin, P. Siegmund, S. Katharina, and H. Jörn, "Three-dimensional PC: toward novel forms of human-computer interaction", in *Proceedings SPIE Photonics East Critical Review Conference on Three-dimensional Video and Display,* Vol. CR76, Boston, USA, 2000, 250-281.

[13] C. S. Lee, K. M. Oh, and C. J. Park, "Virtual Environment Interaction Based on Gesture Recognition and Hand Cursor", in *Proceedings of International Conference on Mechatronic Technology (ICMT)*, Pusan, Korea, 1999, 398-403.

[14] C. S. Lee, S. W. Ghyme, C. J. Park, and K. Y. Wohn, "The Control of Avatar Motion Using Hand Gesture", in *Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST)*, Taipei, Taiwan, 1999, 59-65.

[15] J. M. Rehg, T. Kanade. "DigitEyes: Vision-Based Human Hand Tracking", *Technical Report CMU-CS-93-220,* School of Computer Science, Carnegie Mellon University, USA, 1993.

[16] R.M. Voyles, "Toward Gesture-Based Programming: Agent-Based Haptic Skill Acquisition and Interpretation", Ph.D. Thesis, Robotics Institute, Carnegie Mellon University, USA, 1997.

[17] G.Iannizzotto, M. Villari, and L. Vita. "Hand tracking for human-computer interaction with Graylevel VisualGlove Turning back to the simple way". *ACM*

Workshop on Perceptive User Interfaces, Orlando, Florida, ACM Digital Library, 2001, ISBN 1-58113-448-7.

[18] Francis Quek, David McNeill, Robert Bryll *etc.*, "Gesture and Speech Multimodal Conversational Interaction", *Vision Interfaces and Systems Laboratory (VISLab) Report: VISLab-01-01*, Wright State University, USA, 2001.

[19] J. Martin, V. Devin, and J. Crowley. "Active hand tracking". In *Proceedings of IEEE 3rd International Conference on Automatic Face and Gesture Recognition*, Nara, Japan, 1998.

[20] J. B. Hans, B. Anja, B. U. Dietrich, K. Markus, and G. H. Micheal, "Neural Architecture for Gesture-Based Human-Machine-Interaction", in *Proceedings of International Gesture Workshop,* Bielefeld, Germany, 1997, 219-232

[21] Sign Language Association of Ireland , "Sign on: Basic signs used by Irish deaf people", Dublin, Ireland, 1995.

[22] S.Young, G. Evermann, D. Kershaw, G. Moore, J. Odell, D. Ollason, V. Valtchev, and P. Woodland, "The HTK Book", Version 3.1, Cambridge, U.K. *http:// htk. eng.cam.ac.uk/ prot-docs/htk_book.shtml*, 2001.

[23] T. A. Mysliwiec, " FingerMouse: A Freehand Computer Pointing Interface", *Technical Report VISLab-94-001*, Vision Interfaces and Systems Laboratory (VISLab), Electrical Engineering and Computer Science Department, The University of Illinois at Chicago, 1994.

[24] S. Ressler, B. Antonishek, Q. M. Wang, A. Godil, and K. Stouffer, "When Worlds Collide - Interactions between the Virtual and the Real", in *Proceedings on 15th Twente Workshop on Language Technology; Interactions in Virtual Worlds*, Enschede, Netherlands, 1999.

[25] M. C. Moy, "Gesture-Based Interaction with a Pet Robot", in *Proceedings of the 16th National Conference on Artificial Intelligence,* Orlando, USA, 1999, 628-633

[26] T. Starner, A. Pentland, "Real Time American Sign Language Recognition Using Desk and Wearable Computer-Based Video recognition", in *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 20(12), 1998.

[27] H. Wu, A. Sutherland, "Dynamic Gesture Recognition Using PCA with Multiscale Theory and HMMs", in *Proceedings of The 2nd SPIE International Symposium on Multispectral Image Processing and Pattern Recognition*, Wuhan, China, 2001, 132-139.

[28] H. Wu, A. Sutherland, "Irish Sign Language Recognition Using Hierarchical PCA", in *Proceedings of Irish Machine Vision and Image Processing Conference (IMVIP 2001)*, Maynooth, Ireland, 2001, 158-165.

[29] J. Lin, Y. Wu, and T. S. Huang, "Modeling the Constraints of Human Hand Motion", in *Proceedings of IEEE Human Motion Workshop,* Austin, Texas, 2000. 121-126.

[30] R. Rosales, V. Athitsos, and S. Sclaroff, "3D Hand Pose Reconstruction Using Specialized Mappings", in *Proceedings of IEEE International Conference on Computer Vision (ICCV),* Vancouver, Canada, 2000

[31] A. Pentland, "Perceptual Intelligence", in *Communications of the ACM (CACM),* 43(3), 2000, 35-44.

[32] R. Collins, A. Lipton, T. Kanade, H. Fujiyoshi, D. Duggins, Y. Tsin, D. Tolliver, N. Enomoto, and O. Hasegawa, "A System for Video Surveillance and Monitoring: VSAM Final Report", *Technical report CMU-RI-TR-00-12*, Robotics Institute, Carnegie Mellon University, USA, 2000.

[33] I. Haritaoglu, D. Harwood, and L. Davis, "real-time surveillance of people and their activities", in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8), 2000, 809-830.

[34] S. Roweis, "EM algorithms for PCA and SPCA", in M.I. Jordan, M.J. Kearns, and S.A. Solla, editors, "Advances in Neural Information Processing Systems", Vol. 10, MIT Press, Cambridge, MA, USA, 1997.

[35] T. W. Lee, "Independent Component Analysis: Theory and Applications", *Kluwer Academic Publishers*, 1998

[36] J. LaViola, "Whole-hand and speech input in virtual environments", *Master's thesis*, CS-99-15, Department of Computer Science, Brown University, USA, 1999

[37] Y. Cui, D. Swets, and J. Weng, "Learning-based Hand Sign Recognition using SHOSLIF-M", in *Proceedings of 5th International Conference on Computer Vision*, Boston, USA, 1995, 631-636.

[38] M .T. Chan, Y. Zhang, and T. S. Huang, "Real-time lip tracking and bimodal continuous speech recognition", In *Proceedings of IEEE Signal Processing Society 1998 Workshop on Multimedia Signal Processing*, Los Angeles, USA, 1998, 65-70.

[39] B. Moghaddam, A. Pentland, "Probabilistic visual learning for object representation", in *IEEE Transaction on Pattern Analysis and Machine Intelligence,* 19(7), 1997, 696-710.

[40] B. Moghaddam, W. Wahid, and A. Pentland, "Beyond Eigenfaces: Probabilistic Matching for Face Recognition", in *Proceedings of 3rd IEEE International Conference on Automatic Face & Gesture Recognition*, Nara, Japan, 1998, 30-35.

[41] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham, "Active Shape Models-Their Training and Application", in *Computer Vision and Image Understanding,* 61(1), 1995, 38-59.

[42] B. Moghaddam, X. Zhou, and T. S. Huang, "ICA-based Probabilistic Local Appearance Models", in *Proceedings of International Conference on Image Processing (ICIP'01)*, Thessaloniki, Greece, October, 2001.

[43] M. Shridhar, A. Badreldin, "High accuracy character recognition algorithm using fourier and topological descriptors", in *Pattern Recognition*, 17(5), 1984, 515-524.

[44] A. Kuehnle, "Symmetry-based recognition of vehicle rears", in *Pattern Recognition Letters,* 12(4), 1991, 249-258.

[45] F.J. Iannarilli, S.H. Jones, H.E. Scott, and P. Kebabian, "Polarimetric Spectral Intensity Modulation (P-SIM): Enabling simultaneous hyperspectral and polarimetric imaging", in *Proceedings of SPIE,* vol: 3698, 1999.

[46] M.M. Krueger, "Environmental Technology: Making the Real World Virtual", *Communications of the ACM,* 36(7), 1993, 36-37.

[47] E. Hunter, J. Schlenzig, and R. Jain, "Posture Estimation in Reduced-Model Gesture Input Systems", in *Proceedings Of IEEE International Workshop on Automatic Face and Gesture Recognition,* Zurich, Switzerland, 1995, 290-295.

[48] W. Freeman, K. Tanaka, J. Ohta, and K. Kyuman, "Computer Vision for Computer Games", in *Proceedings Of IEEE International Conference on Automatic Face and Gesture Recognition,* Killington, Vermont, USA, 1996, 100-105.

[49] C. Charayaphan, A. Marble, "Image Processing System for Interpreting Motion in American Sign Language", in *Journal of Biomedical Engineering*, 14(15), Vienna, 1996, 419-440.

[50] J. Davis, M. Shah, "Gesture Recognition", *Technical Report CS-TR-93-11*, University of Central Florida, USA, 1993.

[51] S. Tamura, S. Kawasaki, "Recognition of Sign Language Motion Images", in *Pattern Recognition,* 21(4), 1988, 343-353.

[52] W. T. Freeman, M. Roth, "Orientation Histograms for Hand Gesture Recognition", in *International Workshop on Automatic Face and Gesture Recognition (IWAFGR95)*, Zurich, Switzerland, 1995.

[53] C. Bregler, J. Malik, "Learning appearance based models: Mixtures of second moment experts", in *Advances in Neural Information Processing Systems*, Vol. 9, *M. Mozer, M. Jordan, T. Petsche ed.* MIT Press 1996.

[54] W. T. Freeman, D. B. Anderson, P. A. Beardsley, C. N. Dodge, M. Roth, C. D. Weissman, W. S. Yerazunis, H. Kage, K. Kyuma, Y. Miyake, and K. Tanaka, "Computer Vision for Interactive Computer Graphics", in *IEEE Computer Graphics and Applications*, 18(3), 1998, 42-53.

[55] K. Cho, S. M. Dunn, "Learning Shape Classes", in *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 16(9), 1994, 882-888.

[56] A. Shamaie, A. Sutherland, "Recognition of Partially-occluded Hand Gestures", in *Engineering and Physical Sciences Research Council (EPSRC) Summer School on Machine Vision (poster session)*, University of Surrey, Guildford, UK, 2001

[57] F. Kjeldsen, "Visual Interpretation of Hand Gestures as a Practical Interface Modality", Ph.D. Thesis, Columbia University, USA, 1997.

[58] W. Huang and R. Mariani, "Face Detection and Precise Eyes Location", in *Proceedings of the IEEE International Conference on Pattern Recognition (ICPR'00)*, Vol. 4, Barcelona, Spain, 2000, 722-727.

[59] C. Becchetti, L. P. Ricotti, "Speech recognition : theory and C++ implementation", New York : Wiley, 1999.

[60] A. Heap, F. Samaria, "Real-time hand tracking and gesture recognition using smart snakes", in *Proceedings of Interface to Real and Virtual Worlds*, Montpellier, France, 1995, 261-271.

[61] A. Heap, D.C. Hogg. "Improving specificity in PDMs using a hierarchical approach". In *Proceedings of British Machine Vision Conference*, Vol: 1, Essex, UK, 1997, 80-89.

[62] R. H. Davies, T. F. Cootes, and C. J. Taylor, "Optimising Statistical Shape Models Using a Minimum Description Length Approach", in *Proceedings of Medical Image Understanding and Analysis (MIUA 2001)*, Birmingham, UK, 2001.

[63] A. Hill, C.Taylor, "Automatic landmark generation for point distribution models", in *Proceedings of British Machine Vision Conference (BMVC)*, Birmingham, UK, 1994, 429-438.

[64] H. Bunke, T. Caelli ed. "Hidden Markov Models - Applications in Computer Vision", *World Scientific Publishing*, Singapore, 2001.

[65] L. R. Rabiner, B. H. Juang, "Fundamentals of Speech Recognition", *PTR Prentice Hall,* New Jersey, USA, 1993.

[66] J. Martin, J. Crowley, "An appearance-based approach to gesture recognition", in *Proceedings of 9th International Conference on Image Analysis and Processing*, Florence, Italy, 1997, 340-347.

[67] H. Birk, T. B. Moeslund, and C. B. Madsen, "Real-Time Recognition of Hand Alphabet Gestures Using Principal Component Analysis", in *Proceedings of 10th Scandinavian Conference on Image Analysis*, Lappeenranta, Finland, 1997, 261 – 268.

[68] J. McDonald, J. Toro, K. Alkoby, A. Berthiaume, P. Chomwong, J. Christopher, M. J. Davidson, J. Furst, B. Konie, G. Lancaster, S. Lytinen, L. Roychoudhuri, E. Sedgwick, N. Tomuro, and R. Wolfe, "An Improved Articulated Model of the Human

Hand", in *Proceedings of the 8th International Conference in Central Europe on Computer Graphics, Visualization and Interactive Digital Media*, Plzen-Bory, Czech, 2000, 306 – 313.

[69] T. Heap, D. Hogg, "Towards, 3D Hand Tracking Using a Deformable Model", in *Proceedings of International Conference on Automatic Face and Gesture Recognition,* Killington, USA, 1996, 140-145.

[70] H. Delingette, "Simplex meshes: A general representation for 3D shape reconstruction", in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR '94)*, Seattle, USA, 1994, 856-857.

[71] N. Shimada, Y. Shirai, Y. Kuno, and J. Miura, "Hand Gesture Estimation and Model Refinement Using Monocular Camera- Ambiguity Limitation by Inequality Constraints", in *Proceedings of 3rd IEEE International Conference on Automatic Face and Gesture Recognition*, Nara, Japan, 1998, 268—273.

[72] H.Ouhaddi, P.Horain, "3D Hand Gesture Tracking by Model Registration", in *Proceedings of International Workshop on Synthetic - Natural Hybrid Coding and Three Dimensional Imaging (IWSNHC3DI'99)*, Santorini, Greece, 1999, 70-73.

[73] J. Rehg, T. Kanade, "Model-based tracking of self-occluding articulated objects", in *Proceedings of 5th IEEE International Conference on Computer Vision (ICCV 95)*, Boston, USA, 1995, 612-617.

[74] P. Baldi, S. Brunak, "Bioinformatics: The Machine Learning Approach", *The MIT Press*, Cambridge, MA, USA, 1998.

[75] S. Ahmad, "A Usable Real-time 3D Hand Tracker", in *Proceedings of 28th Asilomar Conference on Signals, Systems and Computers*, IEEE Computer Society Press, Utah, USA, 1995, 1257-1261.

[76] F. Lathuiliere, J. Y. Herve, "Visual Hand Posture Tracking in a Gripper Guiding Application", in *IEEE International Conference on Robotics and Automation (ICRA 2000)*, Vol. 2, 2000, 1688-1694.

[77] C. Nolker, H. Ritter, "CREFIT: Visual Recognition of Hand Postures", in *Proceedings of 3rd International Gesture Workshop (GW'99)*, Gif-sur-Yvette, France, 1999, 61-72.

[78] C. Nölker, H. Ritter, "Parametrized SOMs for Hand Posture Reconstruction", in *Proceedings of Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN'2000)*, Vol. 4, Como, Italy, 139-144.

[79] T. Lindeberg: "Scale-Space Theory in Computer Vision", Kluwer Academic Publishers, Dordrecht, Netherlands, 1994.

[80] S. McKenna, S. Gong, "Gesture recognition for visually mediated interaction using probabilistic event trajectories", in *Proceedings of 9th British Machine Vision Conference (BMVC)*, Southampton, England, 1998.

[81] M. Walter, A. Psarrou, and S. Gong, "Auto-clustering for unsupervised learning of atomic gesture components using Minimum Description Length", in *Proceedings of IEEE ICCV Workshop on Recognition, Analysis and Tracking of Faces and Gestures in Real-time Systems*, Vancouver, Canada, 2001, 157-163.

[82] F. de la Torre, S. Gong, and S. McKenna, "View alignment with dynamically updated affine tracking", in *Proceedings of 3rd IEEE International Conference on Automatic Face and Gesture Recognition*, Nara, Japan, 1998, 510-515.

[83] L. K. Hansen, J. Larsen, "Unsupervised Learning and Generalization", in *Proceedings of the IEEE International Conference on Neural Networks*, Washington DC, USA, Vol. 1, 1996, 25-30.

[84] K. I. Diamantaras, S. Y. Kung, "Principal Component Neural Networks: Theory and Applications", *John Wiley & Sons, Inc.* New York, USA, 1996.

[85] P. Y. Hong, T. S. Huang, "Gesture Modeling and Recognition Using Finite State Machines", in *Proceedings of the 4th IEEE International Conference on Automatic Face and Gesture Recognition,* Grenoble, France, 2000.

[86] H. K. Lee, J. H. Kim, "An HMM-based Threshold Model Approach for Gesture Recognition", in *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 21(10), 1999, 961-973.

[87] A. Wilson, A. Bobick, "Recognition and interpretation of parametric gesture", in *Proceedings of IEEE International Conference on Computer Vision*, Bombay, India, 1998, 329-336

[88] Y. Wu. T. S. Huang, "Hand Modeling, Analysis, and Recognition", in *IEEE Signal Processing Magazine,* 18(3), 2001, 51-60.

[89] T. Darrell, I.A. Essa, and A. Pentland, "Task-specific gesture analysis in real-time using interpolated views", *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 18(12), 1996, 1236-1242.

[90] Y. Wu, T. S. Huang, "Capturing articulated human hand motion: A divide-and-conquer approach", in *Proceedings of IEEE International Conference on Computer Vision*, Corfu, Greece, 1999, 606—611.

[91] J.Davis, M.Shah, "Visual Gesture Recognition", in *Vision, Image, and Signal Processing*, 141(2), 1994, 101-106.

[92] T. Starner, "Visual recognition of American Sign Language Using Hidden Markov Models", Master's thesis, MIT Media Lab, USA, 1995.

[93] C. Vogler, D. Metaxas, "ASL Recognition Based on a Coupling Between HMMs and 3D Motion Analysis", in *Proceedings of IEEE International Conference on Computer Vision*, Bombay, India, 1998, 363–369.

[94] W. Gao, J. Y. Ma, J. Q. Wu, and C. L. Wang, "Sign Language Recognition Based on HMM/ANN/DP", in *International Journal of Pattern Recognition and Artificial Intelligence*, 14(5), 2000, 587-602.

[95] M. Turk, A. Pentland, "Eigenfaces for Recognition", in *Journal of Cognitive Neuroscience*, 3(1), 1991, 71-86.

[96] M. Isard, A. Blake, "Condensation – conditional density propagation for visual tracking", in *International Journal of Computer Vision,* 29(1), 1998, 5-28.

[97] R. Bowden, "Learning non-linear Models of Shape and Motion", *PhD Thesis*, Department of Systems Engineering, Brunel University, Uxbridge, Middlesex, UK, 2000.

[98] S. J. Foran, "Irish Sign Language", Revised Edition, National Association for the Deaf, Dublin, Ireland, 1996.

Appendix A

In this appendix, we will demonstrate how colour can be used to track the hand reliably without high computational cost, including hand segmentation, noise removal, rotation with respect to the principal axis, and normalisation.

A.1 Hand Segmentation Using Colour

The colour content of an image is an important attribute. It contains a rich amount of information that can be used to segment objects. In addition, to apply colour to object segmentation is computationally inexpensive. Thus, for a real-time system, it is the first choice.

Various colour spaces have been used for image processing. In computer graphics, the most commonly used space is the RGB space. In this method, each pixel of the image contains three primary channels: red, green and blue. All colours can be expressed by combinations of these three primary colours. So every pixel is represented by an individual point in the three dimensional RGB space. Other colour spaces include HSV space and HLS space. They are transformations of RGB space that can describe colours in terms more natural to an artist. The name HSV stands for hue, saturation, and value. HLS stands for hue, lightness, and saturation [97]. Our system adopts the RGB space. We use the following notation: 1) r, g, b are the red, green and blue values of an individual pixel in the image; 2) given a two-dimensional image, r_{ij}, g_{ij} and b_{ij} represent the red, green and blue values of the pixel at ith row and jth column in the image. The same notations apply throughout the rest of the appendix.

In our system, the user is asked to wear a coloured glove whose colour is distinct in the background. The colour of the glove is thus a point in RGB space. Since the colour is distinct, we can segment the area with similar colours out of the RGB space with a certain threshold. Suppose the glove's colour is $< R_{glove}, G_{glove}, B_{glove} >$. The area with similar colours is formulated by:

$$\sqrt{(r - R_{glove})^2 + (g - G_{glove})^2 (b - B_{glove})^2} < T \qquad \ldots(A.1)$$

where T is a predefined threshold. This equation tells us that the shape of the area with similar colours of the glove is a sphere.

The above algorithm assumes an ideal environment. However, often the condition of the working environment varies from time to time. In practice especially the illumination condition is different during the daytime from the night. Such a simple model can not handle this situation. We deal with this problem by replacing the absolute RGB value with intensity, which provides a more robust method for separating colours since it would result in changes in intensity but not colour when colours change from shading or lighting difference. The procedure is listed as follows:

1) We extract a rectangular area as the sample area from the hand area in the image manually. The size of the sample area is m by n. Note this only needs to be done once if the colour of the glove doesn't change. See Figure A.1.

Figure A.1: Sample area of the hand. The white square in the hand area is marked manually to get the sample area of the glove.

First the average grey value A of the sample area is computed by the following equation:

$$A = \sqrt{A_R^2 + A_G^2 + A_B^2} \qquad \ldots(A.2)$$

where A_R, A_G and A_B are the average colours of red, green, blue channel in the sample area respectively, and are defined as follows:

$$A_R = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} r_{ij}}{S} \qquad \ldots(A.3)$$

$$A_G = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} g_{ij}}{S} \quad \ldots(A.4)$$

$$A_B = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} b_{ij}}{S} \quad \ldots(A.5)$$

where S represents the area of the sample area and is given by $m \times n$.

2) Second, the average colour intensities of each channel in the sample area can be computed by:

$$I_R = A_R / A \quad \ldots(A.6)$$

$$I_G = A_G / A \quad \ldots(A.7)$$

$$I_B = A_B / A \quad \ldots(A.8)$$

The colour segmentation is then based on the relative intensities rather than the absolute RGB values. The glove's colour intensity is thus given by $<I_R, I_G, I_B>$.

3) The whole image is scanned pixel by pixel and the colour of each pixel is compared to the sample intensities. Given a threshold set up by hand, the pixels with similar colours of the glove will remain, while the rest are set to zero. This is formulated by:

$$\frac{<r,g,b> \bullet <I_R, I_G, I_B>}{\sqrt{r^2+g^2+b^2}} < T' \quad \ldots(A.9)$$

where $<r, g, b>$ is the colour of an individual pixel; T' is the threshold set up by the user which ranges from 0 to 1; and $\bullet$ denotes the dot product.

Replacing the absolute colour with intensity essentially changes the shape of the area, with similar colours of the glove, from a sphere into a cone, as shown in Figure A.2:

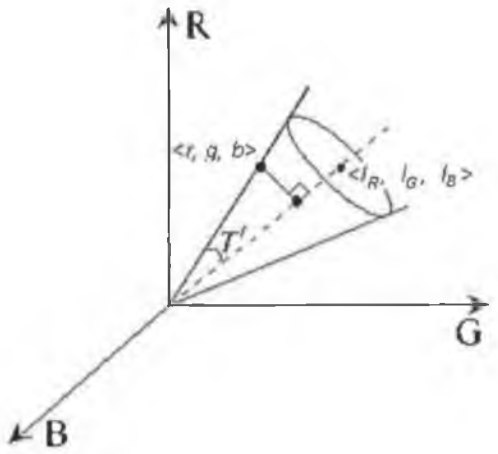


Figure A.2: The RGB cone in the 3D RGB space. R, G and B are the axes of red, green and blue, respectively.

The diagram depicts the RGB cone in the RGB space. The apex of the cone being the origin, $<I_R, I_G, I_B>$ represents the average intensity of the sample area, whose magnitude is 1. Thus it is a unit vector in the space. Colours that fall into the cone represent similar colours to the glove. T' represents the threshold.

The segmentation, using the average intensity, gives the system extra robustness against the illumination variation. As the lighting condition changes, the unit vector C would move along the axis of the cone. Consequently, the size of the cone varies to include less or more colours in the RGB space.

We present the result in Figure A.3. The left image gives the original scene. The white square in the hand area is marked manually to get the sample area of the glove. The right image shows the result, after segmentation using the method introduced above. For visualisation reasons, we increased the contrast of the remaining area against the background, while in practice the remaining area still keeps the same colour as in the original image. Due to the automatic colour balancing of the camera, some colours in the original image have been distorted, which is a usual disadvantage using a cheap webcam; (however, we have to deal with it since we want to the system to be as cheap as possible).

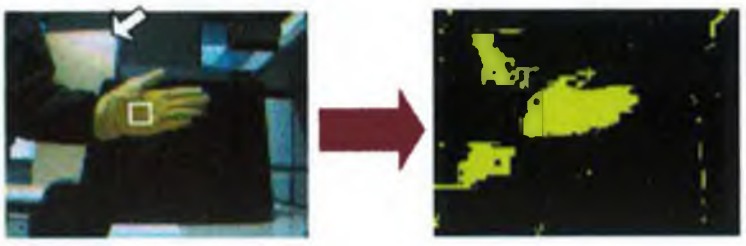


Figure A.3: Hand Segmentation using colour cue. The colour of the area pointed by the arrow in the left image is distorted by the colour balancing of the webcam.

Noise exists in the processed image, which is to be expected. However, the largest object in the image is still the hand. To remove these noises, The *Grassfire* algorithm can be used. Details of this algorithm are introduced by Pitas [6]. The algorithm treats the hand as a connected area in the image, so are the other noises. Under the assumption that the hand area is the largest connected area; (this is fair because the hand is the main object in the image), *Grassfire* can be used to find the area of each connected region, and the largest area is be assumed as the hand, as depicted in Figure A.4.

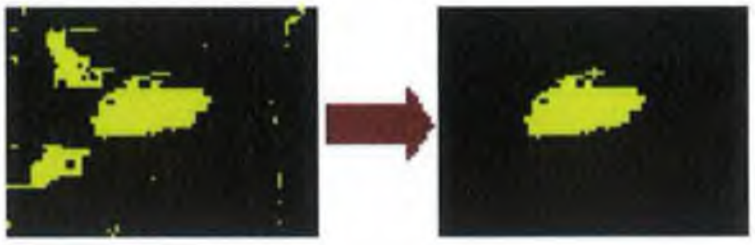


Figure A.4: Hand Segmentation using colour cue. The left image is taken from the right image of Figure 4.2.

A.2 Rotation According to the Principal Axis

Next we describe how to reduce the variance of images caused by the hand rotation in a 2D plane.

When performing gestures, users often rotate their hands as well. Figure A.5 illustrates this situation. Although both gestures mean "c" in ISL, they would not be

projected onto the same point in the feature space since the image vectors would be completely different.

Figure A.5: Variance caused by hand rotation. Both postures mean "C" in ISL, while they look rather different after rotated a certain angle for a computer.

To handle this type of variance, PCA can be applied to find out the principal axis of the hand, and then rotate the hand picture according to the angle of the principal axis. By principal axis, we mean the main axis of the hand posture. It can be computed in the following way: First rather than concatenating the grey values of an image row by row, we construct a hand coordinates matrix **H** using the hand pixels' coordinates, i.e. those pixels in the image whose grey value is greater than zero, expressed by the following equation:

$$\mathbf{H} = \{...,(i - c_x, c_y - j),...\} \qquad ...(A.10)$$

where i and j are the hand pixel's coordinates in the original image, and c_x and c_y are the coordinates of the centroid.

We normalise the hand coordinates in terms of the centroid. **H** is thus an $A \times 2$ matrix, where A is the area of the hand in terms of image pixel. The whole procedure is clearly depicted in Figure A.6.

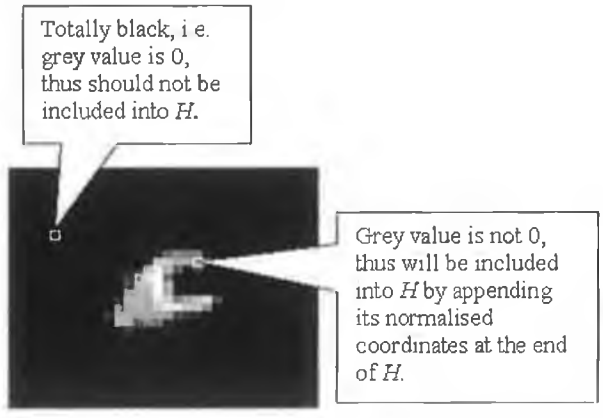


Figure A.6: Construction of the hand coordinates

We first compute the covariance matrix of the hand coordinates matrix, and then using the eigenvalue decomposition algorithm introduced in the last section to find its two eigenvalues (λ_1, λ_2) and two eigenvectors ($\mathbf{e}_1$, $\mathbf{e}_2$). The eigenvector, say $\mathbf{e}_1$, corresponding to the greater eigenvalue, say λ_1, is the principal axis of the hand sign, whose slope S is:

$$S = e_{21}/e_{11} \qquad \ldots(A.11)$$

where $[e_{11}, e_{21}]^T$ is $\mathbf{e}_1$.

Figure A.7 depicts the effect of the rotation in terms of the principal axis. As shown in the figure, the two postures look quite different before the rotation. Consequently, their projections are far away from each other. While after the rotation by their principal axes respectively, the two postures look more or less the same, and the distance in the feature space becomes so small that they could be roughly treated as the same one.

Figure A.7: Illustration of the rotation by the principal axis. The left two images represent the same postures, and the right two are the rotation in terms of the principal axes of the left postures respectively.

The rotation with respect to the principal axis effectively reduces the variance caused by the hand rotation in x-y plane. Therefore, reduces the workload of the recognition as well because the number of PCs needed to maintain the major energy of the images is also reduced greatly.

A.3 Normalisation

So far the user's hand has been extracted from the background, but it is not enough to proceed to the feature extraction step. When performing gestures, one would always intend to move ones hand forwards or backwards with respect to the camera. Consequently, the area of the hand varies significantly during the period. While this is not a problem for human eyes, which can automatically focus on the moving hand, it does disturb the computer classification. This problem can be solved by performing normalisation.

The first thing we need to do is to find the smallest bounding rectangle of the hand area, as shown in Figure A.8. Suppose its dimension is m_r by n_r, the centroid (C_x, C_y) of the hand is derived by the following equations:

$$C_x = \sum_{i=1}^{m_r}\sum_{j=1}^{n_r}(D_{ij} \times i) \Big/ \sum_{i=1}^{m_r}\sum_{j=1}^{n_r} D_{ij} \qquad \ldots(A.12)$$

$$C_y = \sum_{i=1}^{m_r}\sum_{j=1}^{n_r}(D_{ij} \times j) \Big/ \sum_{i=1}^{m_r}\sum_{j=1}^{n_r} D_{ij} \qquad \ldots(A.13)$$

where D_{ij} is the average pixel intensity which is expressed as

$$D_{ij} = \frac{r_{ij} + g_{ij} + b_{ij}}{3} \qquad \ldots(A.14)$$

Once the centroid of the hand has been found, the hand can be normalised by its area. Figure A.8 gives the effect after the normalisation.

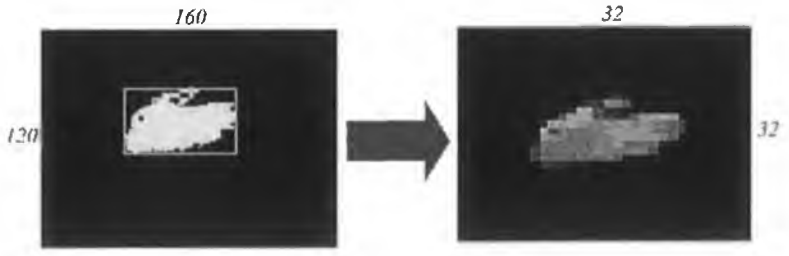


Figure A.8: Normalisation by hand area. The white rectangle in the left image is the smallest bounded rectangle of the hand area. Note the resolution is decreased from 160×120 to 32×32 after the normalisation.

After the normalisation, the image resolution is scaled down from 160×120, the original image size, to 32×32. There are two advantages for doing this. First and foremost, it makes the hand appearance constant with respect to the computer.

Second, the decreased resolution also reduces the recognition time, an important factor for real-time system. Different resolutions other than 32×32 are possible, however, empirically this is enough for ISL recognition since the subtle variation of the hand texture is not important, i.e., only obvious shape change would make sense for the audiences.

The 32×32 image needs to be further normalised by energy using the following equation:

$$D_{ij}^{'} = D_{ij} \Big/ \sqrt{\sum_{i=1}^{32}\sum_{j=1}^{32} D_{ij}^{2}} \qquad \ldots(A.15)$$

where D_{ij} is again the average pixel intensity defined above. The normalisation by energy makes the later classification procedure more robust against the variation caused by the illumination, because it unifies the energy over all images.

A.4 Summary

This appendix has demonstrated how colour can be used to track the hand effectively without high computational cost. In a word, we first use the colour to segment the hand from the background and remove the noise using the *Grassfire* algorithm. Second, we rotate the hand according to the angle of the principal axis to eliminate the variance caused by the hand rotation in a 2D plane. Third, we normalise the hand according to the hand centroid, which keeps the hand area constant with respect to the camera when the hand moves forward or backwards along the camera axis. Figure A.9 gives the complete pre-processing procedure:

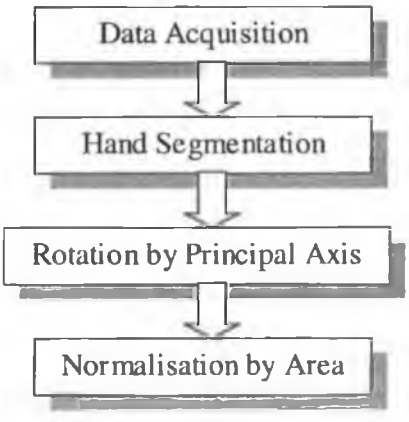
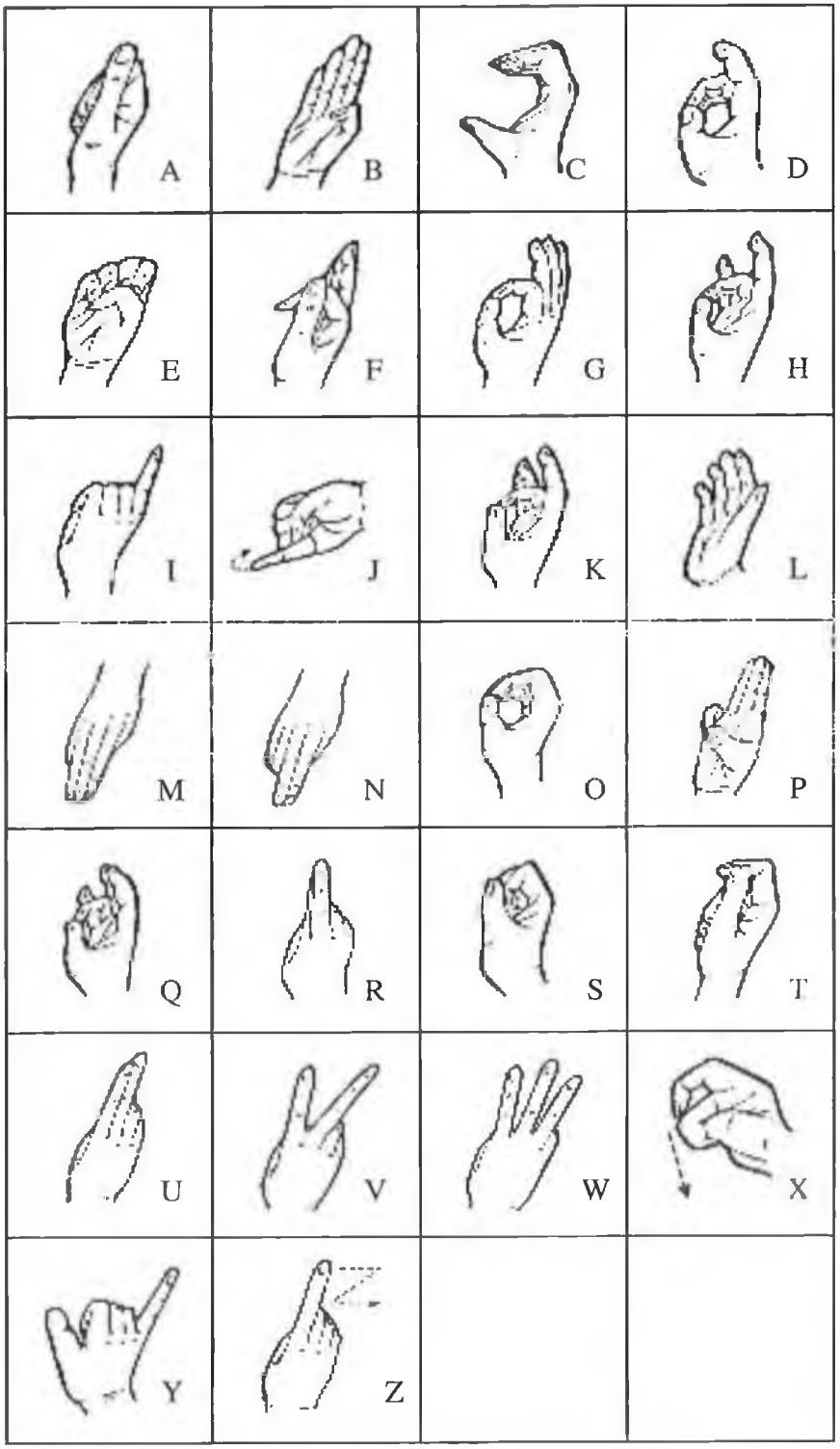


Figure A.9: Diagram for the pre-processing procedure.

Appendix B

This appendix illustrates the 26 gestures in ISL corresponding to the 26 letters in English alphabet. All the pictures are taken from http://myparent.com.



Appendix C

This appendix illustrates the 35 dynamic gestures used for the evaluation purpose in chapter 7. Each of them has a specific meaning in ISL. For instance, Gesture 1, 3, 10 stand for "X", "J" and "Z". The pictures of Gesture 1, 3 and 10 are taken from http://myparent.com. The pictures of Gesture 4, 20 and 28 are taken from [21]. All the rest are taken from [98].

Gesture 1: Open the index finger of any hand, close the rest. Held out at chest level, and move it to self.

Gesture 2: Right hand in the shape of open "C", palm up, opposite the waist. Change "C" to "O" as the hand moves up.

Gesture 3: Any hand in the shape of "I", palm to self, held out. Move the hand to self.

Gesture 4: Right hand in the shape of "T" at chest level, palm sideaway, move it to right changing to "V" and palm outwards.

Gesture 5: Right hand in the shape of "R", palm right-wards, held out at shoulder level.

Gesture 6: Right hand in the shape of "D", palm to self, Move tip of the right index finger along the left neck to the throat.

Gesture 7: Right hand opens in the shape of "20", palm outwards, held out. Move the right hand backwards, changing to clenched "G".

Gesture 8: The tip of the right hand at the forehead. Right hand in the shape of "D", palm sidewards. Move the right hand out, turning the "D" downwards.

Gesture 9: Right hand in the shape of "C", palm sidewards, opposite to the heart. Move it down to the right, turning palm down.

Gesture 10: Any hand in the shape of "D", held out. Move the hand to the right, then move down left, then right. (In the sign of "Z")

Gesture 11: Right hand in the shape of "B", held out. Make the sign of the Cross.

Gesture 12: Right hand in the shape of "B", palm sidewards, held out at the chest level. Move the right hand to the chest.

Gesture 13: Right hand in the shape of "A", palm outwards, opposite to the right shoulder. Zigzag it downwards.

Gesture 14: Right hand in the shape of "C", palm sidewards, held out, draw it to the chest, changing "C" to "A".

Gesture 15: Right hand in the shape of "D", palm to the right. Move the right hand down slowly on centre of itself.

Gesture 16: Right hand in the shape of "D", palm down. Move the tip of "D" up from along the left arm from the wrist.

Gesture 17: Right hand in the shape of "O", palm to self, at the chin. Change "O" to "5", as the right hand is moved outwards.

Gesture 18: Right hand in the shape of "2", palm sidewards, thumb-top at the right cheek. Move the right hand out and down briefly.

Gesture 19: Right hand in the shape of "E", palm outwards, opposite to the right shoulder. Change "E" to "5", swivelling it up.

Gesture 20: Move the right hand in the shape of "F", palm down from right to left with a slightly upward arc.

Gesture 21: Right hand in the shape of "5", fingers down. Move the right hand up, and change it to the shape of "O".

Gesture 22: Right hand in the shape of "D", palm to self, at waist level. Move the right hand down briefly.

Gesture 23: Right hand in the shape of "F", palm down-sides, at left shoulder. Changing to "H" as the right hand is moved out.

Gesture 24: Right hand in the shape of "V", fingertips at the right waist.

Gesture 25: Right hand in the shape of bent "L", palm to self, fingertips at the right chest. Move the right hand down to the right waist.

Gesture 26: Right hand in the shape of "W", palm to self. Move the right hand on self from left chest to left waist.

Gesture 27: Right hand in the shape of "W", palm sidewards. Held out at side level. Turn palm down sharply.

Gesture 28: Right hand in the shape of "A", palm to self on forehead. Move down to chest, then to left shoulder and right. (As in the sign of the cross)

Gesture 29: Right hand in the shape of "P", palm to self on chin. Move the right hand rightwise.

Gesture 30: Right hand in the shape of bent "L", palm to self. Move the fingertips of the right hand to self.

Gesture 31: Right hand in the shape of "Y", palm sidewards. Swivel it briskly.

Gesture 32: Right hand in the shape of "P", palm outwards opposite right shoulders. Shake the right hand slightly.

Gesture 33: Right hand in the shape of "D", palm side wards, at the right cheek. Turn "D" down to the left jaw, turning palm down.

Gesture 34: Right hand in the shape of "G", palm side wards. Move the right hand down opposite the nose and lips, then from left to right at the lips.

Gesture 35: Right hand in the shape of "T", palm sidewards, at the left shoulder. Move the right hand out.