GESTURE RECOGNITION AS A MEANS OF HUMAN-MACHINE INTERFACE

RODNEY D. HALE

GESTURE RECOGNITION AS A MEANS OF HUMAN-MACHINE INTERFACE

BY
@RODNEY D. HALE, B.ENG.

A THESIS SUBMITTED TO THE SCHOOL OF GRADUATE STUDIES IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF ENGINEERING

FACULTY OF ENGINEERING AND APPLIED SCIENCE
MEMORIAL UNIVERSITY OF NEWFOUNDLAND
JULY, 1998

ST. JOHN'S NEWFOUNDLAND CANADA

Abstract

The development of a reliable multi-modal human-machine interface has many potential applications. The interface with a personal computer has become very common yet many disabled users have limited access due to the restrictiveness of the current interface. An improved interface would improve the quality of life for disabled users and has applications in controlling machinery in an industrial setting. Many different types of gestures ranging from head gestures, headpointing, hand and arm gestures are being investigated. A wide variety of classification techniques are available. These techniques range from simple clustering routines to complex adaptive routines. This work compares the recognition results of four pattern recognition techniques, the k-nearest neighbor, a Mahalanobis distance classifier, a rule based classifier and hidden Markov models. The techniques were tested on a set of six hand gestures captured using The Flock of Birds data collection system. The best average recognition result was 97% obtained from the k-nearest neighbor classifier, the Mahalanobis distance classifier had an average recognition rate at 92%, the rule based classifier had an average recognition rate at 89% and the hidden Markov models had the lowest average recognition results at 83%. The hidden Markov models are the most complex of the four techniques studied. Although the average recognition results were lower, they are rich in mathematical structure and can be used to model very complex observations.

Acknowledgments

The pursuit of graduate work has many challenges, both financial and academic. The completion of this work was made possible in part by the help of colleagues, friends, and family. I would specifically like to thank Dr. Ray Gosine for his academic supervision and financial support, Dr. John Molgaard for his academic supervision, the Faculty of Engineering and Applied Science and the School of Graduate Studies for partial financial support. I owe many thanks to my colleagues, including Chris Fowler and Blair Carroll, who provided constant support and tolerance. Finally, I would like to thank Jennifer and my family for continued support in my pursuits.

Contents

List of Figures

List of Tables

List of Symbols

Symbol	Description
FOB	Flock of Birds
FIFO	First In First Out filter
T	overall tremor level
N	buffer length
pdf	probability density function
α,β	probability error
$\vec{f}$	feature vector
ρ	correlation
σ	covariance
ac	correlation matrix
HMM	Hidden Markov model
A	state transition probability matrix
B	current state probability matrix
π	initial state probability matrix
N	number of model states
Q	number of possible observation symbols
λ	Hidden Markov Model parameter description
O	observation sequence
P(O|M)	probability of O given M
α_t	probability of first t symbols given O and M
β_t	probability of symbols t to T given the model state, O and M
γ_t	state probability given O and M
$\hat{\pi}$	re-estimation of π
ξ_t	consecutive state probabilities given O and M
$\hat{a}$	re-estimation of elements of matrix A
$\hat{b}$	re-estimation of elements of matrix B
V	volume
p_y	Parzen density estimate
$d(\hat{y},y)$	distance function
N_i	total number of samples in class i
K_i	class covariance matrix
m_i	class mean
q	N level k-dimensional mapping
A_q	reproduction alphabet or codebook
S	partition
$d(x,\hat{x})$	mapping distortion
A_0	initial reproduction alphabet or codebook
$P(A_m)$	minimum distortion partition
D_m	average distortion

ac_k	a set of m features
$P(\underline{ac}\|C_j)$	multivariate conditional density function given class C_j
w	window height
h	step size

CHAPTER 1

INTRODUCTION

1.0 Background

Researchers for a number of years have been searching for alternate improved means of human-machine interaction. While a common human-machine interface today is the interface with a personal computer, the communications link between the user and the processor is still quite restrictive. It is argued that the full potential of using personal computers has not been realized due in part to the restrictions associated with the common interfaces such as keyboards and mice. Despite many advances in computing power, the user's main interface has not changed significantly since the early typewriters.

Improvements in the interface with a computer can benefit both able-bodied and disabled persons. The conventional keyboard and mouse computer interface can be very difficult to operate if one has even a mild motor control disability. The development of an improved interface would not only improve the access, but would reduce the handicap of not having access to a computer, and therefore significantly improve productivity.

If we consider the possibility of controlling other machines through an improved human-computer interface, the possible applications broaden significantly. The improved access to conventional computer applications related to office work now extends to include the control of robotic work cells which can enhance the quality of life for disabled persons, and to the control of machinery in harsh or dangerous work places providing an obvious improvement in work safety.

Researchers have been studying many different hardware and software arrangements. The hardware include systems for tracking hand/finger gestures, head tracking, eye-gaze tracking and head pointing. These hardware systems range from completely nonintrusive tracking systems, which include video tape tracking similar to some gait analysis systems to attaching sensors to the subject's head or hand. Each of these systems is more or less adaptable to different applications depending on the requirements of that particular application. Generally, if tracking is required over a fairly large area, motion tracking with attached sensors can be cumbersome. However, if precise motion tracking over small ranges is required, attaching sensors can provide large amounts of precise data.

The variation in the software systems being developed is even more diverse than the hardware systems. There are several issues which must be addressed by software designers when building a human-machine interface. Signal Processing techniques used for gesture segmentation and data filtering must be addressed. The type of hardware used largely dictates the approach; video tracking requires image processing techniques while

attached sensor tracking requires the analysis of streams of geometric and dynamic data. The next problem is choosing an applicable pattern recognition technique. There are many options and the choice depends on the sophistication of the system. The common approach is to build a system which can be easily transferred to a wide range of users. This requirement involves designing adaptive software systems, making the system complex.

Despite the amount of research over the past years, there are currently no commercial systems capable of fulfilling the various demands for an improved human-machine interface. Systems have been developed for specific users but these are generally expensive and not easily transferred to others.

1.1 Scope of Research

The goal of this research is to outline some of the potential pattern recognition techniques applicable to a sensor based gesture tracking system. Four techniques are described and tested on a set of six hand gestures. Recognition results for these four techniques are compared. A brief discussion of some of the implementation issues relating to these pattern recognition techniques is included. All gestures were analyzed off-line and no attempt was made to optimize any of the recognition systems.

1.2 Thesis Layout

The following section, chapter 2, includes a discussion of relevant publications in the area

of pattern recognition. Specifically, literature related to the areas of gesture, speech, word and script recognition are included. Chapter 3 is an introduction to the major components of a gesture recognition system. Chapter 4 provides a detailed description of the four pattern recognition techniques tested. The testing procedure and recognition results for the six gestures studied are described in chapter 5, followed by concluding remarks in chapter 6.

CHAPTER 2

LITERATURE REVIEW

2.0 Introduction

Throughout the research community there is a growing number of projects exploring gesture, speech, and character recognition. This chapter will provide an overview of some of the research performed in the field of pattern/gesture recognition, specifically research applicable to the development of a human-machine interface.

2.1 Gesture Recognition

Much research has focused on recognizing gestures performed by different parts of the human body. The gesture types studied have varied considerably, as have the techniques used for recognition. Researchers at the University of Oxford, have investigated recognizing arm gestures using accelerometers. The goal is to use natural arm gestures, measured using accelerometers, to aid people with athetoid cerebral palsy who have impaired speech and motor control (Harrington et al., 1995).

Six single axis accelerometers are attached to the subject's forearm to record the gesture

movement. Accelerometers have been chosen for people with athetoid cerebral palsy who have difficulty targeting specifically with limbs but have sufficient control to make dynamic gestures that are recognizable. The computer recognition uses template matching which measures the difference between the test gesture and the reference templates. The class corresponding to the reference template which best matches the test gesture is recognized. Initial results with able-bodied users familiar with the system had recognition rates around 95% and approximately 80% for the able-bodied users unfamiliar with the system. Results from users with athetoid cerebral palsy had lower recognition rates at approximately 60% (Harrington et al., 1995).

Takakashi et al. at the Tsukuba Research Center, have proposed a spotting algorithm to recognize the meaning of human gestures from motion images. Human gestures are observed as a sequence of images through a camera, and a spatio-temporal vector field (edge feature images) is derived. To make the features robust, a spatial-reduction, temporal-averaging operation, and saturation operation is adapted to each vector element (Takahashi et al., 1994). The system consists of an algorithm which extracts spatial-temporal vector fields (composed of three dimensional edges) from input image sequences and makes standard sequence patterns that correspond to gestures. The recognition process consists of spotting recognition by Continuous Dynamic Programming. Some experiments have been performed to study the influence of clothing and background variability. The initial results indicate the method is robust against variations in clothing and background.

Birk and Moeslund at the Aalborg University in Denmark have investigated using principal component analysis as a means of hand gesture recognition (Birk & Moeslund, 1996). A training set of low resolution images for 25 letters from the hand alphabet were collected and analyzed. They report off-line recognition results at 99%. No recognition rates were quoted for the online implementation. The gestures studied were static gestures where the only object in the image was the users' hand.

A standard form of human-human communication for people with hearing disabilities is the use of the American Sign Language. This language contains approximately 6000 gestures representing common words and an option of finger spelling for other less common words. Starner and Pentland at the Massachusetts Institute of Technology have investigated using video capture for real-time recognition of a subset of gestures from the American Sign Language (Starner & Pentland, 1995). Gestures are recognized using a set of Hidden Markov models which do not explicitly model the individual fingers.

Two experiments were performed using a test set of 500 sentences and a 40 gesture or word lexicon. In the first experiment the user wore colored gloves, a yellow glove on the right hand, and a red glove on the left hand. The video images were of the whole person and not specifically the hands. Wearing the colored gloves made locating and segmentation of the hands less complicated. This experiment resulted in a recognition rate of 96%. The second experiment involved video capture of the same set of gestures without the gloves. In this experiment, the hue and saturation specifically associated with

skin tone was used to locate and segment the hands. Recognition rates for this experiment were somewhat lower at 83% (Starner & Pentland, 1995).

Kim at the IBM Research Center, has investigated the on-line recognition of hand gestures using feature analysis (Kim, 1988). The goal of the research was to improve the use of spreadsheet applications by providing a gestural means of command control. A number of gestures are required for a single command. The gesture data is filtered in a number of stages and finally directional features are used for recognition. Testing was performed on 18 gestures collected from twenty subjects with a reported recognition rate of about 80% (Kim, 1988).

Brown et al. have developed an eyegaze and headpointing system to aid persons with physical and/or mental disabilities to communicate in a learning environment. The goal of the project was to determine features, which are currently unavailable, in eyegaze/headpointing assistive technology. The project designed and developed eyegaze and headtracking hardware and software based on proven educational and training theory of people with mental retardation (Brown et al., 1991). Testing was performed on subjects with varying degrees of mental retardation and physical disabilities. The results with this new eygaze technique are much improved from earlier versions but is still somewhat primitive (Brown et al., 1991).

Shein et al. at the Hugh MacMillan Medical Center, have developed a software

application that displays an on-screen keyboard, WIVIK (Windows Visual Keyboard) (Shein et al., 1991). The user can enter text using the WIVIK keyboard which is displayed in a movable, re-sizeable window, with any pointing device that emulates a Microsoft Mouse. The possibilities for the pointing device include touchscreens and headpointing. All window's options are available and the keyboard can be customized. No testing was performed using the WIVIK. In-house technical evaluations have been carried out to ensure the WIVIK does function as desired (Shein et al., 1991).

Tew and Gray, with the Rehabilitation Technology Research Group, University of York, have used an electronic pointing device to emulate several mouse operations (Tew & Gray, 1993). The pointing device replaces the mouse movements and the special gesture movements of the pointer replace the manual operation of the mouse buttons. The special gesture movements referred to are a left button click, a right button click, hold down mouse button, cancel last gesture, pause and stationary pointer. The dynamic programming gesture recognition algorithm utilizes a reference template. The incoming signals are compared to a set of templates. Tests were conducted on 10 normal subjects with recognition rates ranging from 96-99% for the various gestures observed.

Harwin, at the University of Cambridge, has used The Polhemus 3Space Isotrack to capture the head gestures of subjects with cerebral palsy. The subjects spelled words, with the aid of a virtual head stick which conveyed their intention. Harwin then recognized these gestures using Hidden Markov models. Testing indicated recognition

rates were around 90% (Harwin, 1991).

Perricos, also at the University of Cambridge, has also used the Polhemus 3Space Isotrack to capture the head gestures of subjects with cerebral palsy (Perricos, 1995). The subjects performed a series of eight gestures which could be used to emulate mouse movements and mouse button clicking. Perricos used Dynamic Time Warping to recognize these gestures. Testing indicated recognition rates between 85 and 90%.

Cairns, at the University of Dundee, compared the recognition rates of Dynamic Programming techniques, Hidden Markov models, and Artificial Neural Networks for hand and head gesture data collected using the Polhemus 3Space Isotrack (Cairns, 1993). Able-bodied subjects performed five hand gestures and the disabled subjects performed gestures they found easy to make. The recognition rates using the Artificial Neural Networks on data collected from able bodied subjects were not encouraging and further testing was not performed. The two dynamic recognition methods, Dynamic Programming and hidden Markov models, performed fairly well on data collected from disabled users with 77% and 70% recognition rates respectively.

A wide variety of systems have been developed and tested with some success in laboratory settings. There is however a gap in the research which links the many techniques to a system which is applicable to a wide variety of people with varying needs.

2.2 Speech Recognition

Researchers have been searching for a fast and reliable means of speech recognition for a number of years. One of the more obvious uses of a speech recognition interface is the fast and easy access to computer word processing applications. Other uses could include an improved means of access for many disabled people. Commercial speech recognition systems are available and computer systems can now be purchased with speech recognition capabilities. Many of these systems require excessive demands on processing capabilities making practical use impossible. These systems are generally subject to interference from background noise and have limited vocabularies. Much research is still underway to improve these systems.

Companies producing popular software packages like Miscosoft® Windows are researching speech recognition techniques as additions to their platform packages. Huang et al. with the Microsoft® corporation have been refining and extending Sponx-II technologies since 1993 in order to develop practical speech recognition at Microsoft®. They have developed Whisper (Windows Highly Intelligent Speech Recognizer) which offers speech input capabilities and can be scaled to meet different computer platform configurations (Huang et al., 1995). Some features of the system include continuous speech recognition, speaker-independence, noise robustness, on-line adaptation, dynamic vocabularies and grammars. Results based on testing of a 260 word Windows continuous command-and-control task, produced an average speaker-independent word recognition error rate of 1.4% on a 1160 utterance testing set (Huang et al., 1995). The system is

reported to run in real-time.

Other researchers have investigated sophisticated pattern recognition techniques and their application to speech recognition. Morgan and Bourlard, with the International Computer Science Institute, Berkeley, CA, have studied the use of hybrid hidden Markov models and Artificial Neural Networks to speech recognition problems (Morgan & Bourlard, 1995). The attraction to more sophisticated algorithms is to have applicability to more complex systems that can incorporate deep acoustic and linguistic context. These hybrid systems have been implemented on fairly simple systems only with error rates based on 1000 word vocabulary tests around 5.8%.

Other application areas for speech recognition research is the improved transmission and recognition of vocal signals, specifically over telephone lines. deVeth and Boves, with the University of Nijmegen, are investigating techniques which reduce the recognition degradation due to transfer characteristics of the microphone and the transfer channel (deVeth & Boves, 1997). This research involves the improvement of an existing method of noise reduction which in turn leads to higher recognition rates. A data set of 911 utterances of a connected six digit utterance was tested using a hidden Markov model classifier. The reported recognition rate was approximately 98%.

In addition to improvements to the recognition algorithms, researchers are investigating improved voice acquisition systems in an attempt to improve speech recognition rates.

Omologo et al. are investigating the use of different numbers of sensors and their placements (Omologo et al., 1997). An array of directional microphones were arranged and tested in quiet and noisy environments. A set of Continuous Density Hidden Markov models were used. The results with the microphone array in the noisy environment showed an improvement in recognition results relative to results obtained using a single microphone. Testing using a test set of 2316 words characterized by a word dictionary of 343 words indicated recognition rates at 84%.

2.3 Character/Script Recognition

Areas which can benefit from the improvements in script recognition range from automated processing of banking documents to an improved means of computer data entry. Due to the large amount of variation in the character shapes and the overlapping nature of handwritten words, automatic recognition is a difficult problem. Many variations of segmentation based word recognition, and context based word recognition are currently being explored. Mohamed and Gader, with the University of Missouri, have investigated using segmentation-free hidden Markov modeling and segmentation-based dynamic programming techniques (Mohamed & Gader, 1996). The segmentation-free technique is attractive and can improve recognition results significantly when handwritten words are not easily segmented. This technique uses continuous density Markov character models to construct word models. The dynamic programming segmentation-based technique is lexicon based. Each word is further segmented into primitives which represent single characters. When both of these techniques were tested together, a

recognition rate of 97% was reported (Mohamed & Gader, 1996).

Cursive word recognition can also be quite useful when sorting mail by recognizing ZIP codes. Gillies, at the Environmental Research Institute of Michigan, has researched discrete hidden Markov models for recognizing handwritten address blocks from the US Mail service (Gillies, 1992). The approach is to develop hidden Markov models for individual letters and to combine these models to form a model for each word in the lexicon. Through a process of feature detection and vector quantization the cursive word images are transformed into observation sequences which are then analyzed by the hidden Markov models. Testing was performed on a set of 296 cursive words with a recognition rate of 73% (Gillies, 1992).

Another popular technique used in pattern recognition and script recognition is Artificial Neural Networks. Wu et al., with the University of Sydney, are studying a special network known as the self-organizing map (Wu et al., 1994). The self-organizing map is a sheet-like network which builds representatives by analyzing input patterns and determining the most appropriate model. Using classifier based self-organizing maps can be viewed as a k-nearest neighbor classifier (Wu et al., 1994). The training set included a set of 10,426 binary images of handwritten digits. This set contained the characters 0, 1,2, 3, 4, 5, 6, 7, 8 and 9. Sixty-four features were extracted from these images which were used in training the neural network. Specifically, Wu et al. proposed using two-layer self-organizing maps. The first layer classifies the input patterns into subspaces,

similar input patterns are self-arranged to be near each other in the map. The second layer then self-organizes the input patterns and are differentiated into specified classes using the higher resolution maps located in the second layer. Testing was performed using a further 10,426 handwritten digits. A recognition accuracy of 97% was reported based on this test set (Wu et al., 1994).

Other pattern recognition techniques have more recently been applied to handwritten script recognition including variations of fuzzy logic classifiers. Malaviya et al., at the German National Research Center for Information Technology, have investigated using automatic fuzzy generated rules for recognition of handwritten script (Malaviya et al., 1996). The fuzzy rules used for recognition contain the feature information extracted from the training data set. The approach is based on a multi-level fuzzy rule based paradigm. The segmentation of the handwritten script is performed by the first layer or the low-level layer of the recognition scheme. Malaviya et al. have developed a dedicated fuzzy language FOHDEL (Fuzzy On-Line Handwriting Description Language) to overcome some syntactical ambiguities involved with the fuzzy structural features. The hybrid approach utilizes fuzzy statistical measures and neural networks in combination with expert knowledge to enhance the final classification decision. Test set used consisted of 1800 distinct symbols from ten different writers with a recognition performance of 92% (Malaviya et al., 1996).

Past research has been extensive and much research is continuing, all pushing towards

attainment of an improved human-machine interface. Some commercial systems are available, specifically in the area of speech and script recognition. There is considerable room for the improvement of existing systems and specifically the development of new systems for new applications.

CHAPTER 3

PATTERN RECOGNITION SYSTEMS

3.0 Introduction

Most pattern recognition systems have a similar underlying structure regardless of the application. The purpose of this chapter is to provide an overview of typical pattern recognition systems, specifically hand gesture recognition systems. Section 3.1 provides a description of a generalized gesture recognition system, section 3.2 provides a brief description of the different types of gestures, section 3.3 provides a description of segmentation techniques, and section 3.4 provides a description of feature extraction and segmentation through windowing.

3.1 Gesture Recognition Systems

The function of a gesture recognition system is to provide a means of taking the transduced gestures from the user and of producing some meaningful input or action. There are a number of common stages to any gesture recognition process as shown in figure 3.1.

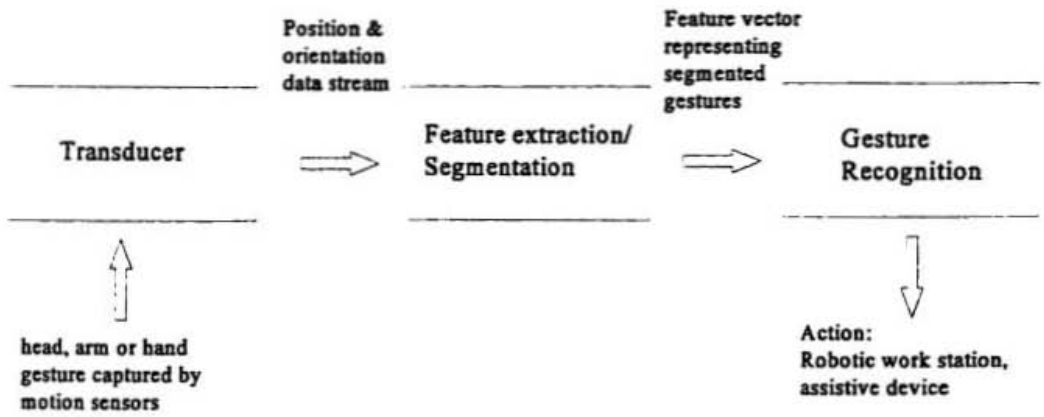


Figure 3.1 - Generalized Gesture Recognition System

The role of each of these stages is as follows:

3.1.1 Transducer

This stage converts movements into a representation useful for further computer processing. In the work reported here, the gestures are tracked using a six degree-of-freedom measuring device called the Flock of Birds (FOB) configured to track the position and orientation of the receiver. Each receiver is capable of making from 10 to 144 measurements per second of each of the three positions, x, y and z, and the three orientations, roll, pitch and yaw, when the receiver is within 3 feet of the transmitter (Ascension, 1995).

The FOB determines position and orientation by transmitting a pulsed DC magnetic field that is simultaneously measured by the sensors in the flock. Each receiver independently

computes it's position and orientation from the measured magnetic field characteristics and transfers this information to the host computer.

3.1.2 Feature extraction/segmentation

This stage transforms the data stream representing the position and orientation of the transduced gesture into data representing single gestures. The segmentation and feature extraction process can be very difficult and can affect the overall performance of a gesture recognition system. There are several options to consider in this stage:

i. Segment the transduced data stream into individual gestures by means of a tremor filter (section 3.3) and extract appropriate features which represent the gesture. This produces a single feature vector representing a single gesture.

ii. Extract features from the transduced data stream by means of windowing (section 3.4) and segment into individual gestures based on the features extracted *or* segment the data stream using a tremor filter and extract features from the segmented data stream by means of windowing. This produces a sequence of feature vectors which represent a gesture.

The extraction of appropriate features can be challenging. Choosing the features which best describe the gestures can involve the use of conventional statistical feature extraction by Principal Component Analysis or by selecting features which intuitively represent the gestures.

3.1.3 Gesture recognition

This stage takes the features representing an individual gesture and classifies it as a particular gesture or as not recognized (spurious). This is usually performed by comparing it to a predefined set of gestures determined off-line. There are many techniques available and the choice depends on the system requirements. This stage of the process is usually the most time intensive for any recognition system (Cairns, 1993). Typically, the classifier will be restricted to run in real-time which can limit the number of gestures it can recognize and the degree of complexity of each gesture.

3.1.4 Action

Once the gesture has been recognized, the user's intention can be relayed to an application which then performs some task. This stage of the process is application dependent. For example, hand gestures can be used to control wheel chairs, robotic work stations or machining equipment.

3.2 Types of Gestures - Static and Dynamic

There are two basic types of gestures: static and dynamic gestures. Static gestures are the simplest form and represent gestures such as finger spelling languages, where each letter is represented by a static position and orientation of the hand and fingers. Time varying movements are not involved in static gestures. Each gesture is represented by a single vector containing position and orientation data.

Dynamic gestures are more complex and represent gestures such as the British Sign Language where each word is formed by the movement of the hands over a time period. Each gesture is represented by a sequence of vectors detailing the formation of the gesture over time.

3.3 Tremor Filter

Gestures are a difficult form of data to represent visually especially when they consist of both positional and angular measurements. Before gestures can be analyzed by most recognition algorithms they must be segmented. This involves determining the beginning and the end of each individual gesture.

The average level of movement of the transduced hand position readily equates with a measure of tremor or movement which can be used as a means of gesture segmentation. This approach involves having the user intentionally provide periods of little or no movement between gestures. The tremor filter then detects these periods of reduced movements or "still" periods at the start or end of each gesture (figure 3.2).

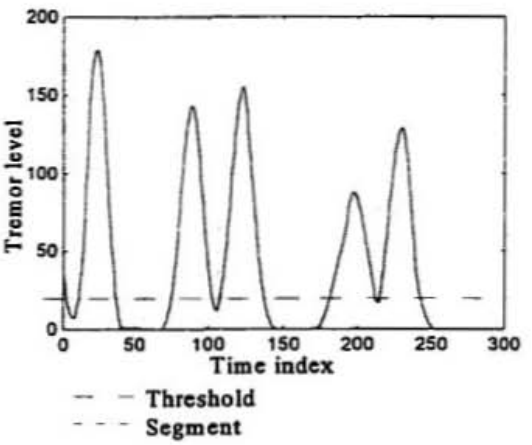


Figure 3.2 - Tremor Filter

The tremor filter consists of a fixed length FIFO (first in, first out) buffer. The tremor in each axis is measured by taking the variance of the transducer position over the fixed time span corresponding to the buffer length (Perricos, 1995). The tremor level is defined as:

$$T_j = \frac{1}{N}\sum_{i=1}^{N}(x_{ij} - \bar{x}_j)^2 \qquad 3.1$$

where: N → buffer length

j → current dimension of the transduced data vector

x_i → i^{th} element of the buffer

$\bar{x}_j$ → mean value of x over the buffer length

If the position does not change in a particular direction or dimension $x_{ij} = \bar{x}_j$, therefore $T_j = 0$. The overall tremor level can be expressed as a Euclidean distance

$$T = \sqrt{\sum_{j=1}^{6} T_j^2} \qquad 3.2$$

3.3.1 Optimizing a Tremor Filter

Two variables affect the performance of a tremor filter; the threshold value T which distinguishes intentional hand movement from mere tremor and the buffer length N. The simplest tremor filter is a single threshold filter. A fixed threshold is set and anything above this threshold is considered movement while everything below this threshold is

considered still (figure 3.3). The optimum tremor level here is the level at which the still probability density function (pdf) and the movement pdf intersects.

A disadvantage to this filter is the possibility that the tremor level may fall below this threshold during a gesture. Assuming that both distributions are normal, the probability

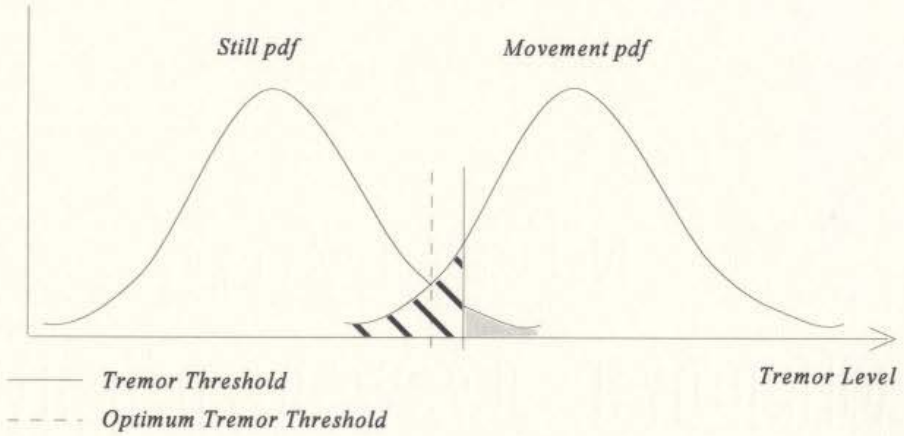


Figure 3.3 - Error Probability in Single Threshold Movement Detection

of being in the moving state and classifying the hand as still is an α type error (Devore, 1987) represented by the hatched portion in figure 3.3. The shaded region in figure 3.3 represents the probability of classifying the hand as moving while in the still state or β type error (Devore, 1987). It should be clear that in this instance it is not possible to reduce the probability of a beta type error without increasing the probability of an alpha type error.

From a probabilistic view, a single tremor threshold appears to present the optimum

solution to the problem. In practice however, errors occurring in the segmentation task that could lead to a recognition error will result from the tremor level briefly crossing the threshold level when the hand is in a given state. There are two possibilities here:

i. the tremor in the still state briefly rises above the threshold into the moving state as would occur in the case of jitter. In this case the jitter would be considered a gesture.

ii. the tremor in the moving state briefly falls below the threshold value in the middle of a gesture. In this case the movement is prematurely segmented as a gesture.

What is required for a reliable tremor filter is the maximization of the probability of remaining in a given state once the hand is in that state. In probability theory this requires minimizing the beta type errors for each state. A practical solution is to have two tremor thresholds as in figure 3.4. If the hand is in a still state, it is considered still

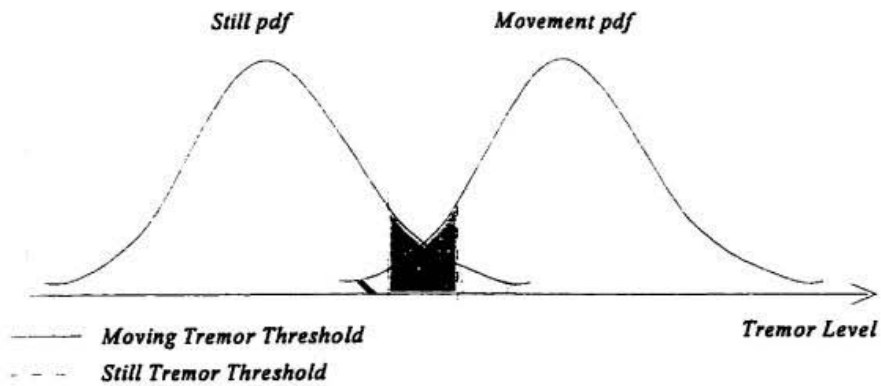


Figure 3.4 - Error Probability in Dual Threshold Movement Detection

until the tremor level surpasses the still threshold. If the hand is in a movement state, it is

considered moving until the tremor level crosses the moving threshold. Gestures are now required to be more distinct. The hand must be more still at the end of a gesture than with the single threshold case and the beginning of the gesture will be detected at a larger hand tremor than with the single threshold case.

The second variable which affects the performance of a tremor filter is the buffer length. The appropriate buffer length can be determined experimentally. It should be noted that for a given buffer length the two tremor thresholds are unique and must be recalculated if the buffer length changes.

3.4 Windowing - Feature Extraction and Segmentation

Analyzing a portion or window of the data stream is widely used in signal and image

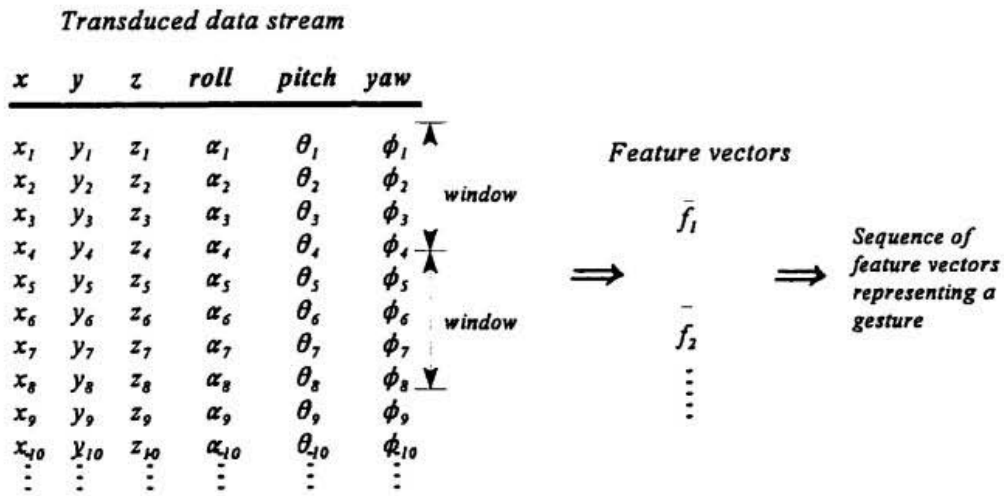


Figure 3.5 - Windowing/feature extraction

processing. The process of analyzing only a portion of the signal at a time has also been used in automatic speech recognition systems. A portion of the signal from the sensor, possibly a microphone, is generally transformed into the frequency domain for further analysis. This process can also be applied to the area of gesture recognition. Portions (a window) of the data stream from the transducer can be transformed (feature extraction) prior to analysis by a recognition system (figure 3.5). This process can allow the recognition of more complex gestures without having to use more complex features. It is intuitive that describing a portion of a gesture requires fewer features than the description of an entire complex gesture.

The window size is an important factor which can affect performance. It is clear that the window size can not exceed the length of a single gesture. This will make recognizing individual gestures impossible. The window size should be kept as small as possible without adding computational expense by over characterizing a gesture. The appropriate window size can be determined experimentally.

3.5 Feature Space Reduction

Feature selection can prove difficult for a number of reasons. Selecting the appropriate features which will give the best classification will often lead to selecting a large number of features. Although this may provide better classification results, with large numbers of features, computations may become unmanageable and impractical. As well, large numbers of features (increasing the dimensionality of the feature space) will make the

feature space increasingly sparse if the number of training vectors is not increased.

A number of solutions to this problem exist. Two common solutions are the use of factor analysis or principal component analysis (Duda & Hart, 1973). Both the areas of factor analysis and principal component analysis are extensive and will not be discussed here. Components of each can be used to find a lower dimensional representation by combining or removing features which are highly correlated and by combining or removing features which have low inter-feature variance.

A hierarchical clustering procedure can be used to reduce the dimensionality using the correlation between two features. The correlation between two features can be defined as (Duda & Hart, 1973)

$$\rho_{ij} = \frac{\sigma_{ij}}{\sqrt{\sigma_{ii}\sigma_{jj}}} \qquad \textbf{3.4}$$

where σ_{ij} is the covariance of the features i and j; σ_{ii} and σ_{jj} are the variances of features i and j respectively. Completely uncorrelated features will have $\rho_{ij}^2=0$ and completely correlated features will have $\rho_{ij}^2=1$. If the correlation between two features is above a specified threshold, those features are combined, or both can be removed without combining, if sufficient data representation is conserved. A hierarchical dimensional reduction procedure can be followed (Gamage, 1993).

1. Set *d* or the dimensionality as the number of features in the data set.

2. Calculate the correlation matrix of features (ac_i).

3. Determine the most correlated pair (ac_i and ac_j).

4. Combine ac_i and ac_j, remove one; or remove one or both initially.

5. Adjust dimensionality *d* appropriately.

6. Go to step 2.

This procedure can be followed until the desired dimensionality is reached or until the largest correlation between any two features falls below a preset value. In this thesis, this procedure was followed until the largest correlation between features fell below a desired level and then further dimensional reduction was performed by removing features with low inter-feature variance.

CHAPTER 4

TECHNIQUES USED IN GESTURE RECOGNITION

4.0 Introduction

The role of the gesture recognition algorithm is to classify the pre-processed gestures as one of a set of previously defined gestures, or as a non gesture. There are many pattern recognition techniques available, examples include heuristic methods, statistical classification, clustering and template matching, and syntactical methods. Sections 4.1 to 4.5 describe the techniques considered in this research.

4.1 Hidden Markov Models

Hidden Markov models (HMM) have been used extensively in the area of speech recognition. Recently, however, the use of hidden Markov models has become more wide spread. The models are very rich in mathematical structure and hence can form the theoretical basis for use in a wide range of applications (Rabiner, 1989). They have been used recently in the area of head gesture recognition (Harwin, 1991 and Cairns, 1993).

There are different forms of HMM's with varying architecture depending on the

application. The simplest form of HMM and the most widely used is the discrete HMM. In this instance we consider the process we wish to model to have emitted or produced a set of discrete observations which can be modeled. The discrete HMM will be discussed in more detail in following sections.

In certain applications, the limitations imposed by a discrete HMM may not be acceptable. For these cases a continuous HMM can be used. The principal difference is that continuous HMM's have continuous observation density functions.

4.1.1 A Description of Hidden Markov Models

Before introducing the mathematical representation of a HMM, begin with a conceptual representation of what a HMM represents. Perhaps the easiest way to illustrate a HMM conceptually is through the use of Levinson's genie (Levinson et al., 1983). Consider a genie who has several urns at his disposal. Each urn holds a number of different colored balls. The genie picks a ball from one of the urns and shows it to you (a single observation symbol). He then replaces the ball and selects another ball from another urn (possibly from the same urn). He does this N times after which you have a discrete observation sequence O, of N observations. The genie does have a preference in choosing an urn which we do not know.

We now wish to determine, based on the set of observations shown to us by the genie, the number of urns at his disposal and the proportion of colored balls in each urn (model

estimation). Once we have determined the rules the genie used to produce the observation sequence, we can now determine the probability that another observation sequence was produced by the same genie using the same number of urns containing the same proportion of colored balls (recognition).

Consider next a simplified speech recognition example. If the word is broken into a set of phonemes, the sequence of these phonemes can be recognized as a particular word. A different set of phonemes will represent a different word. A subset of these phonemes taken as an observation sequence can be modeled and used for recognition. It should be clear at this point that this approach can be applied to many temporal events. The only requirement is that we have a set of observations which we know represents a word or gesture (model estimation). We can then use this model to determine the probability that a different set of observations (pattern) was produced by that model.

4.1.2 A Mathematical Description of Hidden Markov Models

A hidden Markov model can be described mathematically by a set of matrices A, B and π. The matrix A is the matrix of state transition probabilities, the matrix B is the matrix of probabilities of observing a particular symbol while in that state, and the π matrix is the matrix of probabilities of being in a particular state initially (Rabiner, 1989). Determining these matrices is the problem of model estimation and training.

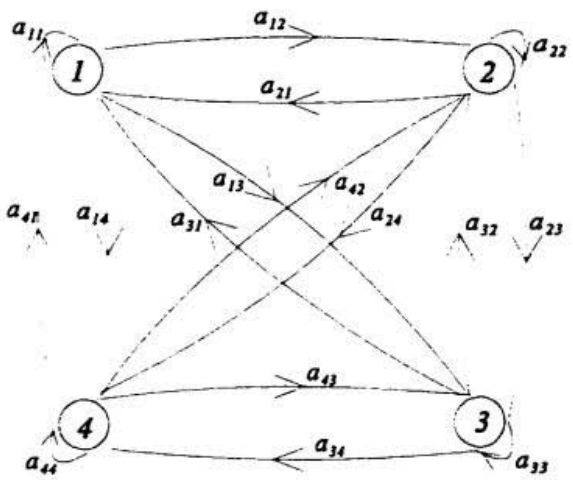


Figure 4.1 - Generalized HMM Architecture (4 states)

Consider a system with 4 possible states (figure 4.1). In the most general model architecture, it is possible to move from any given state to another at a given time (ergodic model). It is also possible to remain in the current state. An arbitrary model for this system will have a NxN matrix A given by:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ \cdot & \cdot & \cdot & \cdot \\ a_{N1} & a_{N2} & \cdots & a_{NN} \end{bmatrix} \qquad \textbf{4.1}$$

where a_{12} is the probability of moving from state 1 to 2 given you are in state 1. Assuming that while in each state it is possible to observe one of Q symbols (O_1, O_2,..., O_Q), the model will have an NxQ matrix B given by:

$$B = \begin{bmatrix} b_1(O_1) & b_1(O_2) & \cdots & b_1(O_Q) \\ b_2(O_1) & b_2(O_2) & \cdots & b_2(O_Q) \\ \cdot & \cdot & \cdot & \cdot \\ b_N(O_1) & b_N(O_2) & \cdots & b_N(O_Q) \end{bmatrix} \quad \textbf{4.2}$$

where $b_1(O_2)$ is the probability of observing symbol O_2 while in state 1. Finally π, the initial state distribution, can be given by:

$$\pi = \begin{bmatrix} \pi_1 \\ \pi_2 \\ \cdot\cdot \\ \pi_N \end{bmatrix} \quad \textbf{4.3}$$

where π_1 is the probability of being in state 1 initially.

It is now possible to completely define a particular HMM (λ) using the characteristic matrices:

$$\lambda = (A, B, \pi) \quad \textbf{4.4}$$

Given an observation sequence $O=\{O_1, O_2, ..., O_T\}$, model parameters A, B and π can be estimated which define a new HMM, or O can be used with an existing HMM model to determine the probability that the observation sequence was produced by that model, P(O|M). This leads to the issue of how to compute these probabilities, P(O|M), and how to use a given observation sequence to estimate, or re-estimate, A, B, and π.

4.1.3 Recognition

Begin by defining the forward probability of the first t symbols occurring and the model being in state i at time t, given the model M.

$$\alpha_t(i) = P(O_1 O_2 \ldots O_t, state(t) = i | M) \qquad 4.5$$

A recursive method can then be used to calculate the probability of the sequences of symbols up to t+1 occurring conditional on the state, the observation sequence O, and the model M.

$$\alpha_{t+1}(j) = \left(\sum_{i=1}^{N} \alpha_t(i) a_{ij} \right) b_j(O_{t+1}) \quad 1 \le t \le T-1; \; 1 \le j \le N \qquad 4.6$$

Initialization, or the probability of the first observation symbol is given by:

$$\alpha_1(i) = \pi_i b_i(O_1) \quad 1 \le i \le N \qquad 4.7$$

Once the preceding forward probability calculation has been performed and α_T has been calculated, the probability of the observation sequence O being produced given the model M can be calculated as:

$$P(O|M) = \sum_{i=1}^{N} \alpha_T(i) \qquad 4.8$$

4.1.4 Model re-estimation, training

Model re-estimation is performed using the Baum-Welch algorithm (Rabiner, 1989).

Define the backward probability, the probability of the symbol sequence from t to T occurring given the state, the observation sequence O and the model M.

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j) \quad 1 \le t \le T-1 \tag{4.9}$$

β_T is initially set to a N X 1 matrix of 1's.

Combining 4.6 and 4.9; the probability of being in state i at time t given the observation sequence O and the model M can be calculated as:

$$\alpha_t(i)\beta_t(i) = P(O|M) \times P(state(t) = i|O,M)$$
$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{P(O|M)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)} \tag{4.10}$$

To follow a proper probability mass function

$$\sum_{i=1}^{N} \gamma_t(i) = 1 \tag{4.11}$$

The re-estimation of π_i, or the probability of occupying state i at t=1 can be written as:

$$\hat{\pi}_i = \gamma_1(i) \tag{4.12}$$

Define now another variable ξ as the probability of being in state S_i at time t and state S_j at time $t+1$ given the observation sequence O and the model M:

$$\xi_t(i,j) = \frac{\alpha_t(i)\, a_{ij}\, b_j(O_{t+1})\, \beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_t(i)\, a_{ij}\, b_j(O_{t+1})\, \beta_{t+1}(j)} \qquad \textbf{4.13}$$

The model parameter A can now be re-estimated as:

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(j)} \qquad \textbf{4.14}$$

and B can be re-estimated as:

$$\hat{b}_j(k) = \frac{\sum_{t=1,\ s.t.\ O_t = v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \qquad \textbf{4.15}$$

Again to follow a proper probability mass function

$$\begin{aligned} \sum_{j=1}^{N} \hat{a}_{ij} &= 1 \qquad 1 \le i \le N \\ \sum_{k=1}^{Q} \hat{b}_j(k) &= 1 \qquad 1 \le j \le N \end{aligned} \qquad \textbf{4.16}$$

The probability of an observation sequence O being produced by the Model M can now be calculated based on the re-estimated model matrices A and B. For a more detailed discussion of the theory of hidden Markov models and some implementation issues, refer

to Rabiner, 1989.

4.2 The Nearest Neighbor Algorithm

One of the decision rules used in this thesis is called the k-nearest neighbor decision rule. It is a decision rule based on nonparametric estimation of the probability density for a number of categories. The nonparametric estimation procedure is called the Parzen estimation which provides an estimate of the class conditional densities. For a set of N independent samples y^1, y^2, ..., y^N, a volume V centered at $\hat{y}$ and k being the number of neighbors in consideration, it can be shown that the density estimate can be expressed as (Therrien, 1989)

$$\bar{P}_y(\hat{y}) = \frac{k}{NV} \qquad 4.17$$

We can fix the number of samples, k, and let the volume V be determined so that the region around $\hat{y}$ contain just k samples. We can then define V as that of an m-dimensional "hypersphere" centered around $\hat{y}$. V is the volume of a region in space defined by

$$d(\hat{y}, y) \leq r \qquad 4.18$$

where the distance function d is the Euclidean distance and r is the radius of the hyper sphere.

Let the radius of the hyper sphere for each class be determined so that the hypersphere

encloses just k samples of the class, that is r be the distance to the k^{th} nearest neighbor of $\hat{y}$. Let N_i be the total number of samples of class i and represent the density function using equation 4.17.

The decision rule for minimum probability of errors becomes (Therrien, 1989)

$$\frac{k/N_1V_1}{k/N_2V_2} \underset{w_2}{\overset{w_1}{\gtrless}} \frac{N_2/(N_1+N_2)}{N_1/(N_1+N_2)} \qquad 4.19$$

$$or \quad \frac{V_2}{V_1} \underset{w_2}{\overset{w_1}{\gtrless}} 1$$

This decision rule fixes the number of samples of each class and compares the volumes of the two hyper spheres. This is the grouped form of the nearest neighbor decision rule.

If V_i is smaller than V_2, this implies that there are more samples of w_1 in the vicinity of $\hat{y}$ than are samples of w_2. The point $\hat{y}$ is therefore assigned to w_1.

Another form of the k-nearest neighbor rule, and probably the most common, is the pooled form. The hyper sphere around $\hat{y}$ is determined to include k total samples regardless of class. This procedure results in equal volumes for the two classes and a

different number of samples k_i for each class. The decision rule becomes

$$\frac{k_1}{k_2} \begin{matrix} > \\ < \end{matrix} 1 \quad \begin{matrix} w_1 \\ w_2 \end{matrix} \qquad \textbf{4.20}$$

$\hat{y}$ is assigned to the class that has the largest number of samples in the hyper sphere (k should be taken to be an odd number to avoid ties).

There is the possibility here to have weighted decision rules. That is if there is loss associated with choosing a particular class, the algorithm can be weighted or biased to only choose the class which may cause damages when the vector is clearly a member of that group.

4.3 Mahalanobis Distance Classifier

The Mahalanobis distance classifier (Therrien, 1989) uses a discriminate function to classify an unknown gesture. This parametric classifier uses the training data for parameter estimation only and this data is not used during classification. The assumption here is that the underlying data is normally distributed along each feature axes. This leads to the assumption that the class clusters are hyper-ellipsoids. The discriminate function is defined by:

$$dist_i = (\vec{x} - \vec{m}_i)^T K_i^{-1} (\vec{x} - \vec{m}_i) \qquad \textbf{4.21}$$

where: $\vec{x}$ is the unknown gesture

$\vec{m}_i$ is the mean of class i

K_i^{-1} is the inverse of class i covariance matrix

The Mahalanobis distance to the mean of each class is calculated, and the unknown gesture is classified as the class corresponding to the minimum distance. The inclusion of the class covariance matrix K_i in the distance calculation provides a measure of the variance in each feature axis and the correlation between features. Feature axes with high variation will factor differently than those with low variation.

The training gestures are used only to determine the class means, m_i, and the class covariance matrices, K_i. This classifier is much faster than the k-nearest neighbor classifier.

4.4 Vector Quantization

Vector quantization is an important application related to clustering which has been widely used in the field of digital signal processing and communication. It is often used as a preprocessor to a Dynamic Programming or hidden Markov model classification techniques used in many speech recognition systems. The range of possibilities which describe a gesture can be divided into a number of levels. Given a sample gesture, it will be denoted by a discrete value representing the particular level which best represents it. Figure 4.2a illustrates how a continuous event can be transformed into a series of discrete

values. Figure 4.2b illustrates how vector quantization can be used to map each discrete value to a reproduction vector.

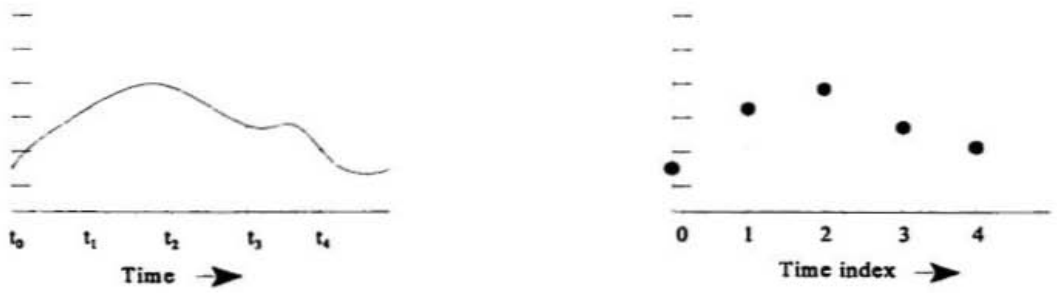


Figure 4.2a - Sampling and Quantization of a Signal (Therrien, 1989)

Consider an N level k-dimensional quantizer as a mapping, q, that assigns to each input vector, $x=(x_0, ..., x_{k-1})$, a reproduction vector, $x = q(x)$, drawn from a finite reproduction alphabet, $A_q=\{y_i; i=1, ..., N\}$. The reproduction alphabet (or codebook) A_q and the partition, $S=\{Si; i=1, ..., N\}$ of the input vector space into the sets $Si=\{x: q(x)=y_i\}$ of input vectors mapping into the i^{th} reproduction vector (or codeword) completely describes the quantizer q (Linde et al., 1980).

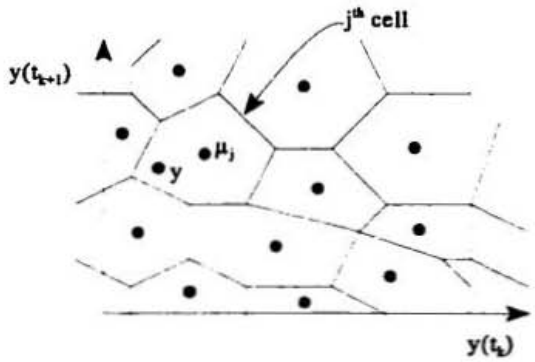


Figure 4.2b - Vector Quantization (Therrien, 1989)

Assuming the distortion caused by the mapping of the input vectors is some non-negative value denoted $d(x,\hat{x})$, the goal is to minimize this distortion during mapping. There are

many different distortion measures. The most common for mathematical convenience is the squared-error distortion given by:

$$d(x,\hat{x}) = \sum_{i=0}^{k-1} |x_i - \hat{x}_i|^2 \qquad \textbf{4.22}$$

The process of mapping an input vector follows four steps.

(1) Initialization: For an N level quantizer, with a distortion threshold $\epsilon \geq 0$, an initial N-level reproduction alphabet A_o, and a training sequence $\{x_j; j=0, ..., n-1\}$, set m=0 and $D_1=\infty$.

(2) Given $A_m=\{y_i; i=1, ..., n-1\}$, find the minimum distortion partition $P(A_m)=\{S_i; i=1, ..., N\}$ of the training sequence: $x_j \in S_i$ if $d(x_j, y_i) \leq d(x_j, y_n)$ for all *n*. Find the average distortion

$$D_m = D(\{A_m, P(A_m)\}) = \frac{1}{n}\sum_{j=0}^{n-1} \min_{y \in A_m} d(x_j, y) \qquad \textbf{4.23}$$

(3) Determine if minimum distortion is reached:

$$(D_{m-1} - D_m)/D_m \leq \epsilon \qquad \textbf{4.24}$$

If the distortion is below the minimum threshold, halt with A_m as the final reproduction alphabet, otherwise continue to step 4.

(4) Find the optimal reproduction alphabet $x(P(A_m))=\{x(S_i); i=1, ..., N\}$ for $P(A_m)$. Set $A_{m+1} \triangleq x(P(A_m))$. Replace m by m+1 and go to 2.

While designing the quantizer, only partitions of the input alphabet or training sequence are considered. Once the final codebook A_m is obtained, it is used on new data outside the training sequence with the optimum nearest-neighbor rule or an optimum partition of k-dimensional Euclidean space.

4.4.1 Choosing an Initial Reproduction Alphabet

There are several ways to choose the initial reproduction alphabet A_o. One method is to use a uniform quantizer over all or most of the source alphabet. A second technique, "splitting", is useful when a quantizer is to be designed of successively higher rates until achieving an acceptable level of distortion. Consider a M-level quantizer with $M=2^R$, R=0, 1, ... until an initial guess for an appropriate N-level quantizer is achieved. A simple algorithm using this technique (Linde et al., 1980) is:

(1) Initially set R=0 (M=1) and define $A_o(1)=x(A)$, the centroid of the entire alphabet.

(2) Given the reproduction alphabet $A_o(M)$ containing M vectors $\{y_i; i=1, ..., M\}$, "split" each vector y_i into two close vectors $y_i+\epsilon$ and $y_i-\epsilon$, where ϵ is a fixed perturbation vector. The collection A of $\{y_i+\epsilon, y_i-\epsilon, i=1, ..., M\}$ has 2M vectors. Replace M by 2M.

(3) If M=N, set $A_o=A_q(M)$ and that is the initial reproduction alphabet. If not, run the same procedure for an M-level quantizer on $A_q(M)$ to produce a good reproduction alphabet $A_o(M)$ and return to step 2.

Using this splitting algorithm on a training sequence, the initial quantizer is a one-level

quantizer consisting of the centroid of the training sequence. This vector is then "split" into two vectors and a two-level quantizer algorithm is run on this pair to obtain a good two-level quantizer. Each of these two vectors is then split and the algorithm is run to produce a good four-level quantizer and so on resulting in fixed-point quantizers for 1, 2, 4, 8, ..., N levels.

4.5 Rule-Based Gesture Recognition

This section describes the generation of rules which can be used to recognize gestures. The statistical theory required to estimate the underlying probability density functions (pdf) for each class is outlined initially. The process leading to the generation of rules is then described. A short description of the application of these techniques as applied to known underlying distributions (the Gaussian distribution) is included to illustrate the effect of varying sampling parameters and to demonstrate the technique.

4.5.1 Estimating the Underlying PDF

Consider a set of m features ac_k, k=1,2,...,m. This set of features adequately describes a single gesture, C_j. There are n samples of this gesture; the feature vector representing this gesture is m columns wide and n rows long; $\underline{ac}=(\underline{ac_1}, \underline{ac_2}, ..., \underline{ac_m})$. The rules for this gesture will be centered around the high density regions of the R^m feature space. It is therefore necessary to estimate the density function. The procedure for estimating the density function is described below (Gamage et al., 1996).

The probability P that a feature vector $\underline{ac}$ of group C_j belonging to the region R is given by

$$P = \int_R p(\underline{ac}|C_j)d\underline{ac} \qquad \textbf{4.25}$$

where $p(\underline{ac}|C_j)$ is the multivariate conditional probability density function, given that the class is C_j, the probability density function of $\underline{ac}$. With n samples of class C_j drawn from the density function $p(\underline{ac}|C_j)$, the probability that k samples belong to the region R is given by the binomial law

$$P_k = \binom{n}{k} P^k (1-P)^{n-k} \qquad \textbf{4.26}$$

The expected value of k is given by E(k)=nP. The maximum likelihood estimate for P is given as

$$\hat{P} = \frac{k}{n} \qquad \textbf{4.27}$$

If the region is assumed to be small, P can be approximated as

$$P = \int_R p(\underline{ac}|C_j)\,d\underline{ac} = p(\underline{ac}|C_j)V \qquad \textbf{4.28}$$

where $\underline{ac}$ is the point within the region R and V is the volume of the hyper cube enclosed by R. Combining 4.25, 4.27 and 4.28 $p(\underline{ac}|C_j)$ can be estimated as

$$\hat{p}(\underline{ac}|C_j) = \frac{k/n}{V} \qquad \textbf{4.29}$$

If the sample size is large, k/n will converge to P. For fixed volume V, $\hat{p}(\underline{ac}|C_j)$ is an average of $p(\underline{ac}|C_j)$

$$\hat{p}(\underline{ac}|C_j) = \frac{P}{V} = \frac{\int_R p(\underline{ac}|C_j)d\underline{ac}}{\int_R d\underline{ac}} \qquad \textbf{4.30}$$

With a finite sample, V can be expressed as a function of n. Generally, as n increases V should decrease. For a univariate case the length h should decrease as $n^{-1/5}$ (Gamage et al., 1996). Generalized for m-dimensions, the volume can be expressed as

$$V_n = \frac{h^m}{n^{\frac{m}{5}}} \qquad \textbf{4.31}$$

The h in equation 4.31 is the size of each subrange. With all feature vectors normalized over their respective ranges, h can be the same for each class. In m-dimensional space h can also be considered as the length of each side of the hyper cube. Determining the size of h is shown in an illustrative example of the reproduction of several Gaussian distributions.

Define the function $\phi(u)$ as

$$\phi(u) = \begin{cases} 1 & |u_k| \le \frac{1}{2} \quad k=1,2,...,m \\ 0 & otherwise \end{cases} \qquad \textbf{4.32}$$

This function $\phi(u)$ defines a unit hyper cube in m-dimensional space centered at the origin. The function

$$\phi\left(\frac{\underline{ac} - \underline{ac_i}}{h}\right) \quad where\ \underline{ac},\ \underline{ac_i} \in C_j \qquad \textbf{4.33}$$

is a unit hyper cube centered at $\underline{ac}$ and will be unity if $\underline{ac_i}$ belongs to the hyper cube of volume V_n. The function will be zero otherwise. The number of samples k_n in the hyper cube can now be determined as

$$k_n = \sum_{i=1}^{n} \phi\left(\frac{\underline{ac} - \underline{ac_i}}{h}\right) \quad where\ \underline{ac},\ \underline{ac_i} \in C_j \qquad \textbf{4.34}$$

The estimate of $\hat{p}(\underline{ac}|C_j)$ can now be written as

$$\hat{p}(\underline{ac}|C_j) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{V_n} \phi\left(\frac{\underline{ac} - \underline{ac_i}}{h}\right) \qquad \textbf{4.35}$$

The window size defined by the sub-range height h is an important factor in determining the estimate of $\hat{p}(\underline{ac}|C_j)$. The number of times this window is sampled over the feature data is also important, known as the step size. The effect of the variation of these

variables on the ability to reproduce a known distribution is discussed in section 4.5.3.

The high density regions will correspond to the local maxima of $\hat{p}_n(\underline{ac}|C_j)$, which can be determined by taking the partial derivatives of $\hat{p}_n(\underline{ac}|C_j)$ with respect to each feature ac_k and equating to zero. The second partial derivatives with respect to all features, must be negative for a maximum. This can be written as

$$\frac{\partial \hat{p}_n(\underline{ac}|C_j)}{\partial ac_k} = 0 \quad for\ k=1,2,...m$$
$$\frac{\partial^2 \hat{p}_n(\underline{ac}|C_j)}{\partial ac_k^2} < 0 \quad \forall\ k=1,2,...m \qquad \textbf{4.36}$$

4.5.2 Rule Generation

With an estimate of the underlying probability density function for each class, the next step is to generate the rules used for recognition. Using Bayes theorem $P(C_j|\underline{ac}) = \hat{p}_n(\underline{ac}|C_j)P(C_j)$; where $P(C_j)$ is the a priori probability of class C_j. Consider a single mode one dimensional case (a single feature). The estimate of $\hat{p}(\underline{ac}|C_j)$ is illustrated in figure 4.3

The gray shaded region between ac_1 and ac_2 represents the rule for this class. The area of the shaded region is set equal to the area under the curve from ac_1 to ac_2. That is, the window height w is given by

$$w = \frac{1}{(ac_2 - ac_1)} \int_{ac_1}^{ac_2} p(ac|C_1)P(C_1)dac \qquad \textbf{4.37}$$

The single rule for this class will be IF $ac_1 < ac < ac_2$ then the class is C_1. The intersection between the curve and a predefined threshold determine the values of ac_1 and

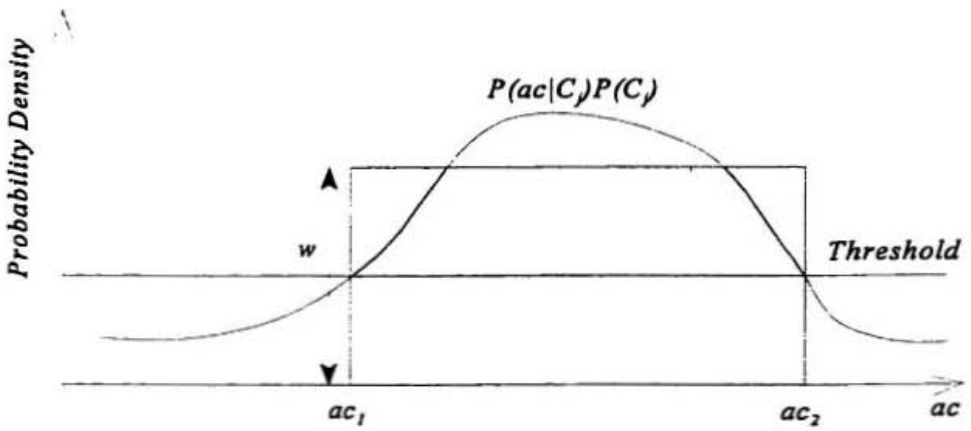


Figure 4.3 - Probability Density Function of Single-mode, Single Class and Selection Window for Rule Generation (Gamage, 1993)

ac_2. The threshold can be determined through an iterative process. A minimum allowable error is selected and compared to the error associated with the set threshold level. The error is given by

$$error = 1 - \int_{ac_1}^{ac_2} p(ac|C_1)P(C_1) \qquad \textbf{4.38}$$

This is the probability that a given feature vector *ac*, will fall outside the rule window and not get classified.

A slightly different approach to rule generation is outlined by Gamage (Gamage, 1993). This technique uses the possible relationships among the features within each group to generate rules. The process is illustrated in figure 4.4 by a connectivity diagram. The set of m features are arranged in columns of ascending order. The data values of each feature

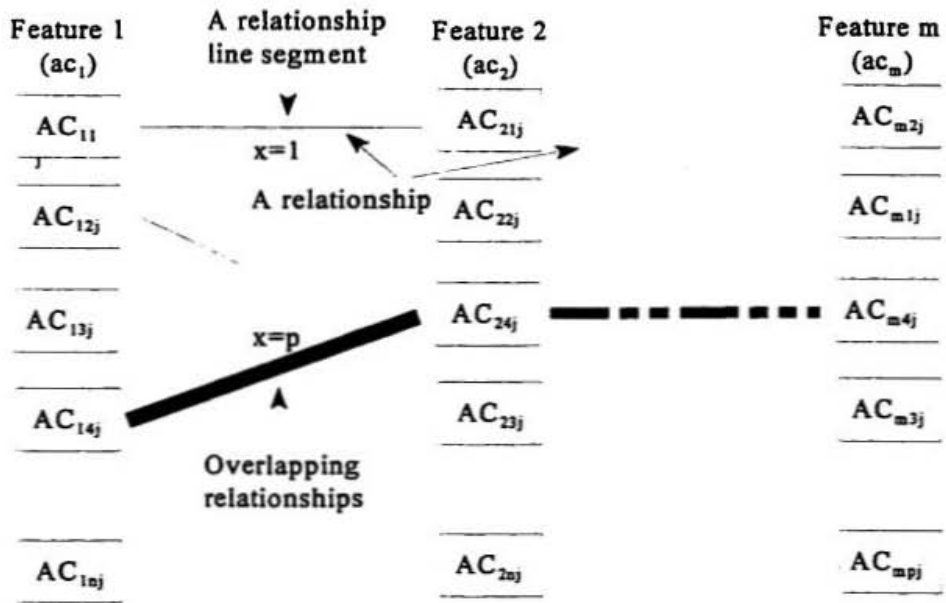


Figure 4.4 - A Generalized Connectivity Diagram for Rule Generation (Gamage, 1993)

for a particular group are plotted in the corresponding column. The feature corresponding to a particular gesture are joined by line segments. A relationship is formed by the m-1

lines connecting a single gesture. The subranges for each column are determined by a window and the number of relationships that fall within that subrange is maximized. The subranges are denoted AC_{kij} where k, i and j are the feature number, the subrange number and the group number respectively. The rules for a particular group are determined by the density of the relationships in that group.

4.5.3 An Illustrative Example

The variation of the sampling window size determined by h and the sampling rate determined by the step size will alter the appearance of the estimated probability density function. As an exercise in predicting appropriate window and step sizes based on the sample size, data was collected from known distributions and the probability density functions estimated. A set of 200 data points were sampled from the normal distribution, normalized to force the data between 0 and 1 and the techniques detailed in section 4.5.1 applied. The resulting distribution estimate is shown in figure 4.5.

The window size h in this example is set to 0.3 and the step size is 10. This means that the data were sampled step/h=33 times from the range from 0 to 1. At each step the density of the window of length h=0.3 was estimated resulting in the curve shown. The combination of this particular h and step size for a sample size of 200 reproduces the normal curve reasonable well. Again this example will have a single rule for each class.

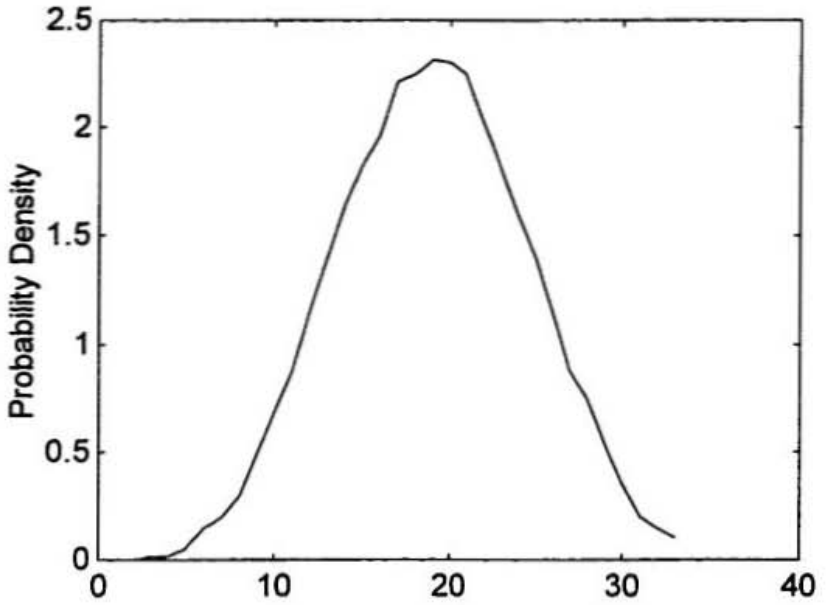


Figure 4.5 - Probability Density Reconstruction of Data Sampled from the Normal Distribution

Next consider a 2-D case; 2 features with 2 modes. Figure 4.6 is a plot of two offset normal distributions reconstructed from a data set 200x2 sampled from the normal distribution.

Again the window height is set to h=0.3 and the step size is 10. Instead of simply moving the sampling window along a single axis as in the 1-D case, now the window estimate of the density is stepped along each axis in turn. The contour lines shown under the curve now represent intersections of a threshold plane and the probability density estimate. The error associated with each of the threshold values is now unity minus the volume under the curve bounded by the contour corresponding to that threshold. For cases with more

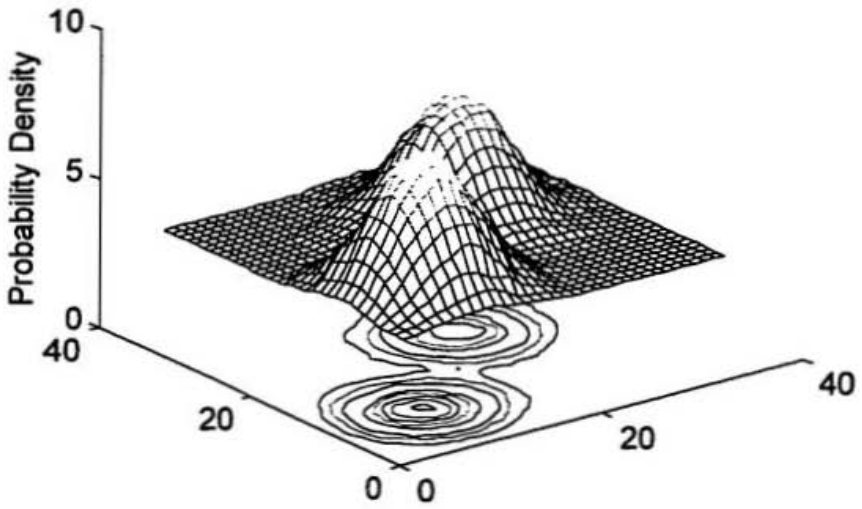


Figure 4.6 - Probability Density Function of a Two-mode, Single Class, Two Feature Case

than two features required to describe the classes to be recognized, it is not possible to visually represent the surfaces or the contours. The probability density estimates will be hyper surfaces with the thresholds as hyper planes. Figure 4.7 details the contour plot for a particular threshold value in the 2-D case shown in figure 4.6.

Again the contour lines correspond to the intersection of the surface to the threshold plane. This class will have two rules (two mode distribution) defined by the rectangular regions overlaying the contour lines in figure 4.5. The area of each rectangular region equals that of the contour it represents. The rules for this example are:

Rule 1: IF $ac_{11} < ac_1 < ac_{12}$ AND $ac_{13} < ac_2 < ac_{14}$

Rule 2: IF $ac_{21} < ac_1 < ac_{22}$ AND $ac_{23} < ac_2 < ac_{24}$

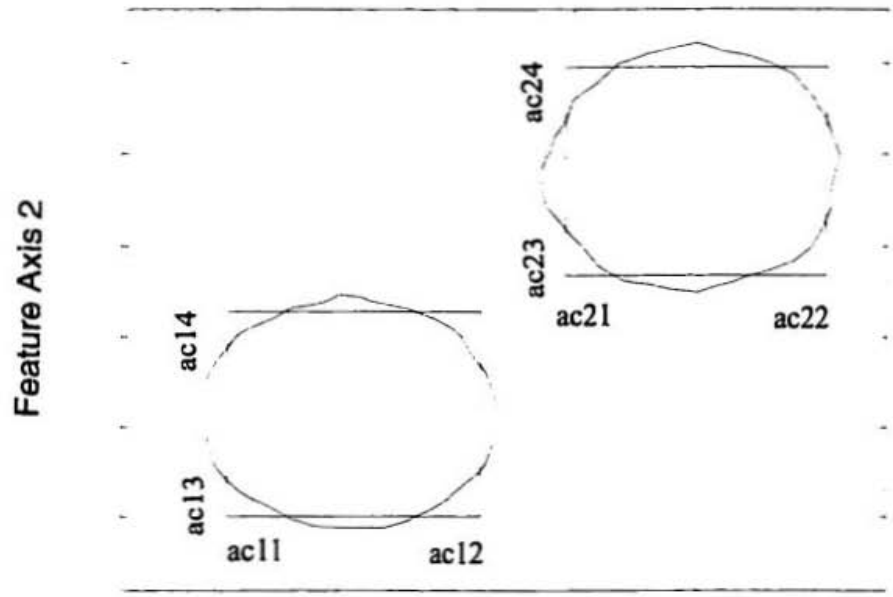


Figure 4.7 - Contour Lines at Threshold Plane with Overlaying Rule Windows

An unknown feature vector $\underline{ac}=[ac_1\ ac_2]$ must satisfy either Rule 1 OR Rule 2 to be classified as class 1. Unclassified feature vectors can be classified as belonging to the class with the highest a priori probability $P(C_j)$.

The error associated with this classification can again be estimated as the probability that $\underline{ac}$ lies outside the acceptance windows orwhere R_i is the region bounded by the contour

$$error = 1 - \sum_{i=1}^{p} \int_{R_i} p(\underline{ac}|C_1)P(C_1)d\underline{ac} \qquad 4.39$$

traced by the intersecting threshold plane and p is the number of regions or the number of high density points intersected by the threshold plane.

CHAPTER 5

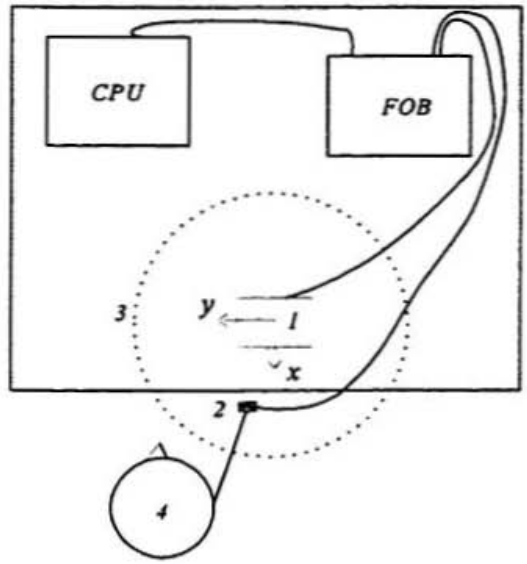
APPLICATION TO HAND GESTURE RECOGNITION

5.0 Introduction

This chapter summarizes the results from the application of the four pattern recognition techniques described in chapter 4. The four techniques are evaluated by comparing recognition results for six hand gestures. A set of four features were selected and used with all four techniques. All the analysis and classification code is written in Matlab version 4.2c.1. The algorithms are included in Appendix B.

5.1 Data Collection

All gestures collected and analyzed during the development of the individual scripts were collected from the author. A single transducer was held in the right hand and each set of gestures was performed. The transmitter was placed on a wooden desk approximately one foot from the user. The sensor was moved in a hemisphere above the plane of the transmitter, remaining within a range of approximately one to two feet from the transmitter at all times. During initial data collection experiments, it was determined that at locations near the plane of origin of the vertical axis (z axis) of the transmitter, an



1 - Emitter
2 - Transducer (bird)
3 - Operating area
4 - User
CPU - Data collection
FOB - Flock of Birds

Figure 5.1 - Data collection setup (top view)

unusually high amount of noise was introduced. This noise was possibly due to field asymmetry at the plane of origin. To avoid this problem, the sensor was operated in a hemisphere above the horizontal plane intersecting the origin of the z axis. Figure 5.1 illustrates the arrangement of the transmitter and the operating volume for the sensor. Care must be taken at all times to minimize the amount of magnetic interference introduced into the operating environment of the sensor. Specifically ferrous material placed between the transmitter and the sensor will cause significant "warping" of the magnetic field generated by the transmitter and can considerably reduce tracking accuracy.

Six hand gestures have been collected and analyzed. Two dimensional representations of these gestures are illustrated in figure 5.2. A number of gestures have been studied by several researchers in the past and the choice here is arbitrary. The six chosen could be

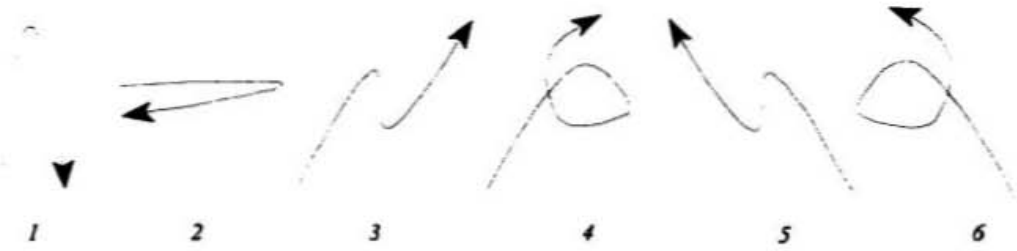


Figure 5.2 - Gesture Types

used for mouse button clicking emulation or as instructions to an assistive device. A total of one hundred samples for each gesture were collected. The selection of an appropriate gesture type is dependent on the nature of the human-machine interface being designed. Hand gestures similar to those studied here may be useful in systems controlling robotic work cells. The six hand gestures chosen here illustrate how some pattern recognition techniques can be applied to gesture recognition problems. All gestures were collected at a sampling rate of 60 Hz, recording both the position and the orientation of the sensor. The gestures generally covered a range of approximately two feet with a duration of approximately 2-3 seconds.

5.1.1 FOB Calibration

The gesture data was collected using the Flock of Birds data collection unit. This tracking system uses a pulsed DC magnetic field and is therefore influenced by surrounding metallic interference. The actual influence on the motion recorded by the unit will depend on the amount of interference. To estimate the amount of distortion in

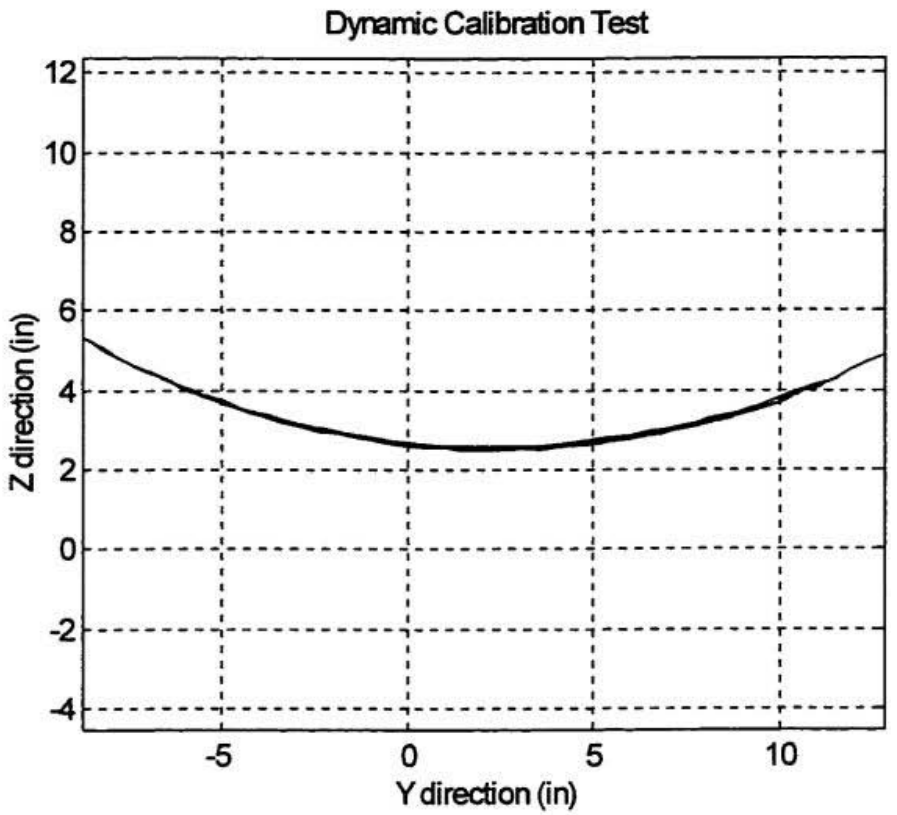


Figure 5.3 - Dynamic Calibration Test

the collected data two calibration tests were performed. Both tests were performed in the same workspace in which the gestures were collected. The first test performed was a dynamic calibration test. This test was performed by allowing the receiver to swing on a pendulum in the y-z plane relative to the transmitter. The data was collected at 60 Hz, the same collection rate used for gesture tracking. The arc created is shown in figure 5.3. The pendulum was allowed to make nine consecutive sweeps. At the bottom of the arc, separation in the path of the sensor between each consecutive pass is evident. Figure 5.4 is a zoom image of the lowest point in the arc shown in figure 5.3. The actual amount of variation in the recorded z position of the sensor is approximately 0.1 inches over a travel

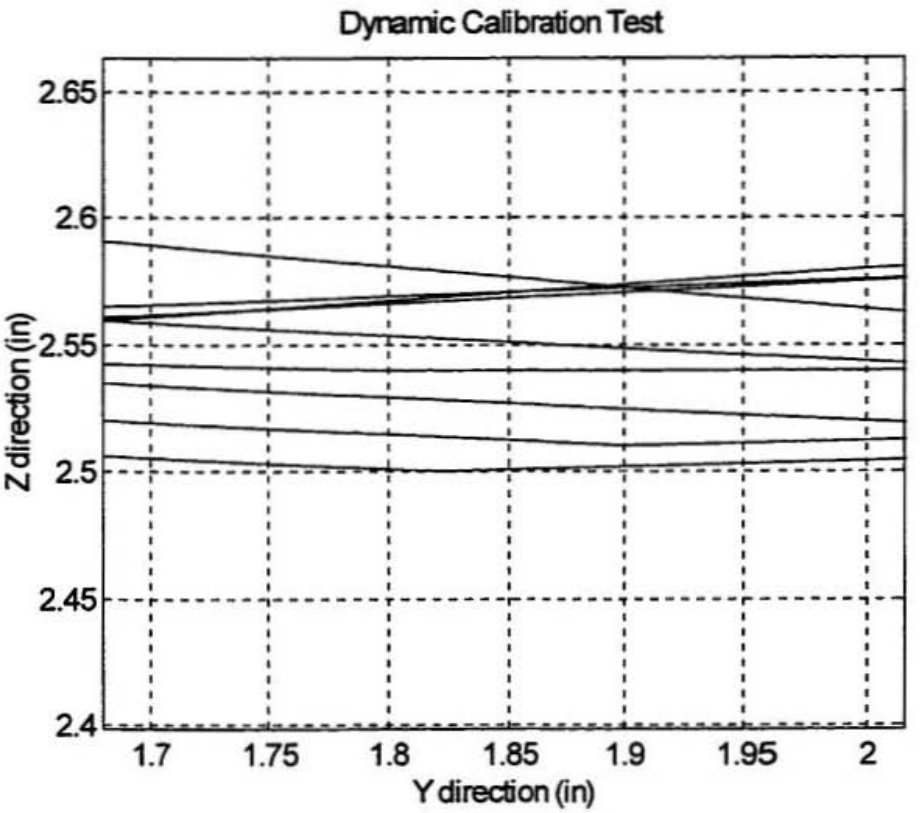


Figure 5.4 - Dynamic Calibration; zoom image

of approximately twenty inches in the y direction. This amount of error is relatively small and can be considered quite acceptable for the purposes of gesture tracking.

A second calibration test was performed to estimate the static accuracy of the data collection system. The sensor is moved a known distance and the difference between the recorded distance traveled and the actual distance is compared. This test was performed by mounting the sensor on a plastic vernier caliper. The caliper was then opened to 2 inches. The data was collected while the calipers were slowly closed from the 2 inch

opening to the fully closed position (zero inches). This was repeated ten times for both the 2 inch opening and a 4 inch opening. The difference between the beginning position and the final positions for each test is calculated using equation 5.1 and compared to the actual distance traveled.

$$dist = \sqrt{(x_f - x_i)^2 + (y_f - y_i)^2 + (z_f - z_i)^2} \qquad 5.1$$

where x_f, y_f and z_f are the final x, y and z positions respectively and x_i, y_i and z_i are the initial x, y and z positions respectively. Table 5.1 is a summary of the test results.

Table 5.1 - Calibration Results

2 inch travel (in)	4 inch travel (in)
1.92	3.92
1.99	3.93
2.01	3.95
1.96	3.88
1.97	3.89
1.93	3.97
2.00	3.91
1.94	3.89
1.97	3.89
1.91	3.88
mean: 1.96	mean: 3.91
max: 2.01	max: 3.97
min: 1.91	min: 3.88
standard deviation: 0.0341	standard deviation: 0.0309

The vernier calipers used have an error of ±1/128 (0.008) inches. Including user error, it is reasonable to expect an error of ±0.01 inches. The results indicate that the sensor in both cases has underestimated the actual distance by approximately 2%. Since gesture tracking involves monitoring positions over relatively large movements, this error level is quite acceptable.

5.2 Gesture Segmentation

Each of the one hundred individual samples of all six of the gestures, are segmented using a two threshold tremor filter. The success of a tremor filter is partially controlled by the user during data collection. The user must intentionally hold the transducer still for approximately a second at the end of each gesture. The upper and lower tremor thresholds were optimized manually. Plotting the tremor level over several gestures will provide initial threshold levels. Knowing the number of gestures in a particular gesture set, the threshold levels were then adjusted until the desired segmentation was obtained. The threshold levels used in the tremor filter are:

$$T_{upper} = 1$$

$$T_{lower} = 0.1$$

The buffer length is largely determined by the sampling rate used during data collection. Using a sampling rate of 60 Hz and a buffer length of 30 with the above thresholds resulted in good gesture segmentation. Each of the individual segmented gestures were then mapped into a set of feature vectors.

5.3 Feature Extraction

The feature extraction and selection process begins with the extraction of a large feature set. This feature set is then analyzed and redundant features are removed. The initial feature set was:

1) maximum position in the x direction
2) maximum position in the y direction
3) maximum position in the z direction
4) minimum position in the x direction
5) minimum position in the y direction
6) minimum position in the z direction
7) maximum difference in the x rotation
8) maximum difference in the y rotation
9) maximum difference in the z rotation
10) mean curvature estimate

These features are simple to calculate and are spatial measures which can be used intuitively to describe motions. These features are calculated relative to the axis of the field emitter.

Features 1,2,6, and 8 were removed due to a high correlation between the feature sets. Feature sets with a correlation greater than $\rho=0.8$ had one of the two highly correlated features removed and the remaining feature set was rechecked. Again feature sets with a correlation greater than $\rho=0.8$ had one of the two highly correlated features removed. This procedure was continued until no two feature sets had a correlation coefficient above $\rho=0.8$. The remaining six features were then compared in terms of their variance. The two features with a lowest variance were removed leaving the final feature set:

1) maximum position in the z direction
2) minimum position in the x direction

3) minimum position in the y direction
4) maximum difference in the z rotation

This feature set was used for the four pattern recognition techniques studied. The number of features is somewhat restricted by the size of the training set. The size of the feature set should be kept as small as possible to avoid making the training data in feature space too sparse. Based on a training set of 100 for each gesture, a four dimensional feature space maintain's a sufficiently dense feature space.

Feature extraction is a very important component of gesture recognition systems and often involves an iterative process of choosing a number of features, feature space reduction, and testing using the chosen pattern recognition technique. For the purposes of comparing techniques this iterative process is unnecessary. The goal of this research is not to obtain the highest possible recognition rates for this set of gestures, but rather to evaluate the performance of each technique relative to the other for the same gesture set using the same set of features.

The k-nearest neighbor classifier, Mahalanobis distance classifier, and the rule based classifier all classify gestures based on features extracted from the entire gesture. This results in a feature set representation for the average of the gesture data. Each gesture, which contains approximately two hundred data points, is now represented by a set of four features. The six hundred gestures (one hundred gesture 1, one hundred gesture 2 etc.) are described by a matrix of numbers 600x4; 600 rows for the gestures and four

columns for the features.

The hidden Markov models use a different approach to pattern recognition which must be considered at the feature extraction stage. The HMM method of pattern recognition accepts a sequence of observations which represent a single gesture. The first stage in transforming the transduced gesture data into sequences is to extract features at intervals along the gesture. The segmented gesture data is windowed at intervals through the duration of the gesture. At each interval, the gesture data over a certain range or window size is used to extract the four features. The interval is increased and again the gesture data in the windowed region is used to extract the four features. This process is continued to the end of the gesture and results in a series of feature sets for each gesture. It is obvious that a single window over an interval of the entire gesture will produce the same result as the feature extraction process for the k-nearest neighbor classifier, the Mahalanobis distance classifier and the rule based classifier.

The windowing interval will depend on a number of factors. A gesture which involves many small detailed movements over relatively short distances or time spans will require a very small window size. Gestures such as the six studied here contain no small detailed movements, allowing a larger window size. The window size was chosen as a function of the gesture length, specifically 1/8 times the gesture length resulting in a series of eight feature sets. This series of features is then analyzed by a vector quantizer to produce the observation sequences required by the hidden Markov models.

5.4 K-Nearest Neighbor

The k-nearest neighbor is a non-parametric classifier which uses all the training data when classifying an unknown gesture. This classifier is possibly the simplest to understand and to implement. The results of a k-nearest neighbor test provide a good indication of how well the classes cluster and how much separation between the class clusters is present.

The feature data from the feature extraction algorithm must be normalized prior to testing with the k-nearest neighbor algorithm. A simple method of normalizing the feature set is to divide each of the four features by its maximum possible value. The same normalization factor is used for all gestures in the training set and for subsequent unknown gestures. Normalizing will force all feature vectors in the range from 0 to 1 in all dimensions.

The decision rule for the k-nearest neighbor rule is a distance comparison. The Euclidian distance from an unknown gesture in feature space (feature vector) is compared to all other known feature vectors. The unknown gesture is classified as the class containing closest *k* feature vectors. To avoid ties, k is chosen to be an odd number. Testing of the k-nearest neighbor algorithm was done setting k=1, k=3 and k=5. The recognition results for k=1 are given in table 5.2 in a confusion matrix. The general form of a confusion matrix can be written as:

$$C = \begin{bmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,6} \\ c_{2,1} & c_{2,2} & \cdots & c_{2,6} \\ . & . & . & . \\ c_{6,1} & c_{6,2} & \cdots & c_{6,6} \end{bmatrix}$$

The diagonal elements ($c_{1,1}$, $c_{2,2}$, $c_{3,3}$, $c_{4,4}$, $c_{5,5}$, $c_{6,6}$) of the matrix are the number from each class correctly classified. For example $c_{1,1}$ is the number correctly classified as class 1, $c_{2,2}$ is the number correctly classified as class 2 and so on. The remaining elements represent the numbers incorrectly classified as other classes. The element $c_{1,2}$ is the number of gestures from class 1 which are incorrectly classified as class 2, $c_{2,1}$ is the number of gestures from class 2 incorrectly classified as class 1.

All recognition results for the k-nearest neighbor tests were obtained using the "leave-one-out" technique. This technique allows the use of all training data for testing without biasing the recognition results. Each gesture is removed from the training set and classified using the remaining training set. It is then replaced and the next gesture is removed and it is classified using the remaining training set. This process is continued until all gestures have been removed and classified.

The results for the k-nearest neighbor test with, k=1 neighbor, given in table 5.2 are for one hundred of each of the six gestures. Gestures 1 and 2 have the highest recognition rates at 100% and 99% respectively. Gesture 3 has the lowest recognition rate at 88%. The highest confusion rate is between gesture 3 and 4 at a rate of 8%.

Table 5.2 - Confusion Matrix; KNN - k=1

100	0	0	0	0	0
0	99	0	1	0	0
0	0	88	8	4	0
0	0	7	93	0	0
1	0	3	0	96	1
0	0	0	0	3	97

The results of the k-nearest neighbor test with, k=3 neighbors, given in table 5.3 again are for one hundred of each of the six gestures. Gestures 1 and 2 have the highest recognition rates at 100% and 99% respectively. Gesture 5 has the lowest recognition rate at 93%. Gestures 4 and 5 are confused with gesture 3 at a rate of 5%. The next highest confusion rate is between gestures 3 and 5 which are confused with each other 3% of the time.

Table 5.3 - Confusion Matrix; KNN - k=3

100	0	0	0	0	0
0	99	0	1	0	0
0	0	96	1	3	0
0	0	5	95	0	0
1	0	5	0	93	1
0	0	0	0	2	98

Gestures 1 and 2 are the simplest and most distinguishable of the six gestures which is indicated by the perfect recognition rates for both. The confusion of gesture 4 as gesture 3, and of gestures 3 and 5 with each other, is somewhat expected as these gestures are

similar differing only in the mid-point movement of the gesture.

The results for k=5 neighbors in table 5.4 are very similar to those obtained for k=3 neighbors. The recognition rate for gesture 5 increased slightly to 94%. The lowest recognition rate is now for gesture 3 at 91%. The greatest confusion remains between gestures 4 and 3 and has increased slightly to 7%. The highest recognition rate is 100% for gesture 1.

Table 5.4 - Confusion Matrix; KNN - k=5

100	0	0	0	0	0
0	99	0	1	0	0
0	0	91	7	2	0
0	0	6	94	0	0
1	0	4	0	94	1
0	0	0	0	2	98

The results of both tests of the k-nearest neighbor classifier, with k=3 neighbors and k=5 neighbors, indicates that the data cluster reasonably well. Increasing the number of neighbors, k, did not improve the recognition results. The recognition results for k=7 are 100%, 99%, 87%, 86%, 92% and 97% for gestures 1 to 6 respectively. The average recognition rate is 94% which is a decrease in performance.

5.5 Mahalanobis Distance Classifier

The Mahalanobis distance classifier is a parametric classifier. The training data is used only for class parameter estimation. It is similar to the k-nearest neighbor classifier in that it uses a distance measure to classify unknown gestures. With this classifier however, the distance used is not the Euclidian distance and the distance to each of the feature vectors in the training set is not calculated. Instead the Mahalanobis distance to the mean of each of the class clusters is determined. The unknown gesture is classified as the class with the minimum distance.

The parameters used are the class means and the class covariance matrices. As with the k-nearest neighbor classifier, the features were first normalized. The class means were estimated by simply taking the mean of each feature set for all six gestures. The class covariance matrices were then estimated. The diagonal elements of the matrix are the variances of each feature set (matrix column). The square root of the diagonal elements is the standard deviation of each feature set. The remaining elements are an indication of the correlation between the features.

Although the features were normalized prior to the calculation of the class means and the class covariance matrices, this method can compensate for some scale changes between the features. However, if the scale differences are significant or the feature sets are highly correlated, the inverse of the class covariance matrices can become unstable resulting in unreliable recognition results. A check is to first calculate the condition of

the covariance matrices before inverting and to remove or combine highly correlated features.

The recognition results for the Mahalanobis distance classifier given in table 5.5 generally show a slight decrease relative to the results obtained from the k-nearest neighbor classifier test. The highest recognition rate is 100% for gesture 1 and the lowest recognition rate is for gesture 3 at 70%. The confusion rate for incorrectly classifying gesture 3 as gesture 4 at 29% is somewhat surprising. Although these gestures are similar, the k-nearest neighbor classifier recorded a 1% confusion rate. This result is a clear indication that the parametric representation for the cluster of data representing gesture 3 data has lost valuable information. The use of the class mean and covariance matrix assumes the data are distributed normally along each feature axis.

Table 5.5 - Confusion Matrix; Mahalanobis distance

100	0	0	0	0	0
0	99	0	1	0	0
0	0	70	29	1	0
0	0	2	98	0	0
0	0	6	0	90	4
0	0	0	0	7	93

The variance along each axis can differ meaning the clusters are not assumed to be hyper spheres. In fact, using these parameters to represent the class data assumes the class clusters are hyper ellipsoids. A possible explanation for the low recognition results for

gesture 3 is that the data may be skewed along one or more of the feature axis. Using the normal assumption in this case will misrepresent the data and may cause an overlap in the hyper ellipsoids representing gesture 3 and gesture 4, which would in turn lead to classification errors. Figures 5.5 and 5.6 are normal probability plots for features 1 and 2 of gesture 3. Both plots indicate that the gesture data is not normal. The data would lie along the line in each plot if the normal assumption were valid. Also at issue here is the question of whether the classes are convex. The parametric representation of the class clusters is possible following assumptions regarding the shape of these clusters. The decision rule which ideally separates these clusters also assumes a shape. The low recognition rates here may be caused by the actual shape of the class clusters not following the assumptions made regarding their shape.

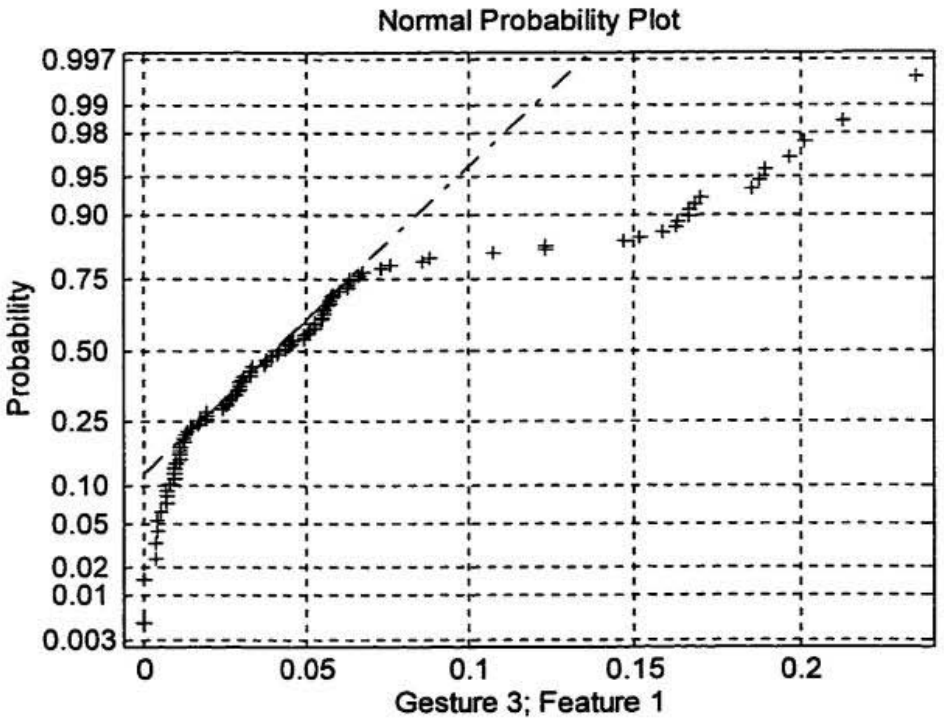


Figure 5.5 - Normal Probability Plot, Gesture 3, Feature 1

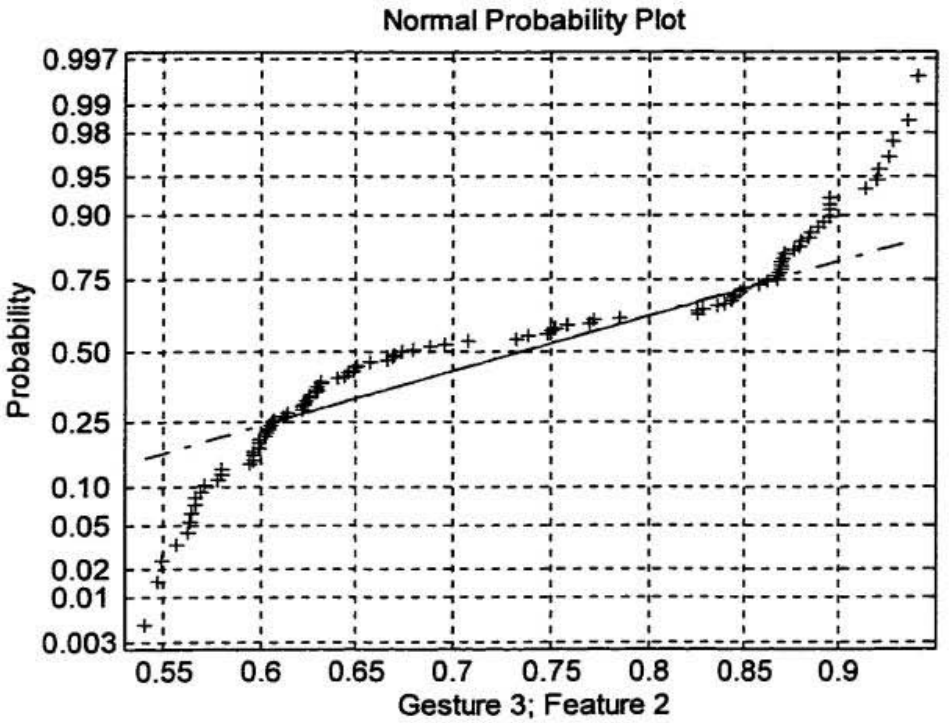


Figure 5.6 - Normal Probability Plot, Gesture 3, Feature 2

5.6 Rule Based Recognition

The rule based recognition technique involves estimating the probability density functions (pdf) of each class. The pdf is estimated for each feature pair. The plots of the pdf estimates for each feature pair for all six gestures are given in Appendix A. Using four features the feature pairs are: features 1 & 2, features 1 & 3, features 1 & 4, features 2 & 3, features 2 & 4, and features 3 & 4. Each gesture has six pdf estimate plots for all possible feature pairs. An example of a pdf estimate for gesture 3, using features 2 & 3, is given in figure 5.7.

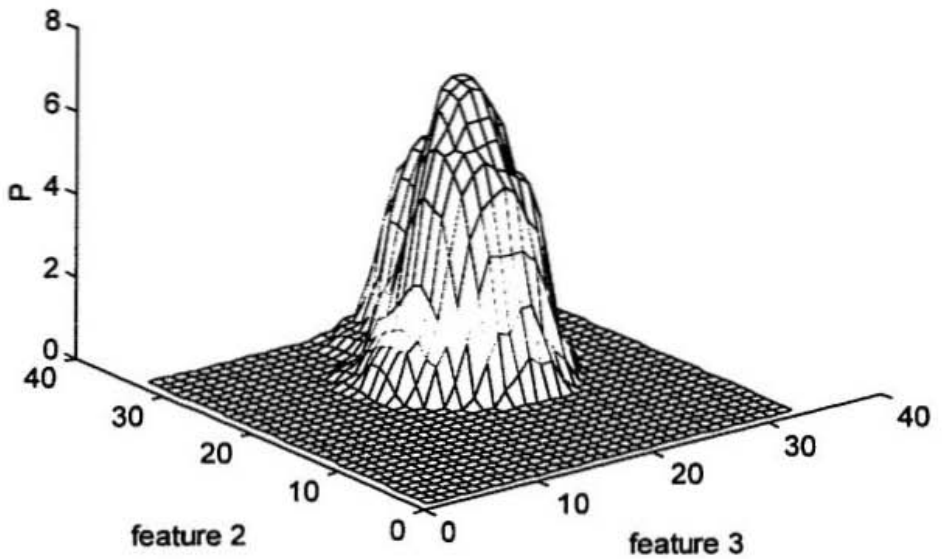


Figure 5.7 - pdf estimate; gesture 3 (features 2 & 3)

The step size used was 10 with a window size h=0.3 as proposed in the illustrative example in section 4.5.3. This combination adequately reproduced a known pdf shown in figure 4.6 and produces relatively smooth pdf estimate plots for each of the feature sets for the six gestures.

The pdf estimate shown in figure 5.7 is single mode and has its peak located in a corner of the horizontal plane. It should be noted that the horizontal axis labels range from 0 to 40. Although all input feature vectors were normalized between the range from 0 to 1, the combination of a step size of 10 and a window size h=0.3 means the window will step 10/0.3=33 times to cover the range from 0 to 1. The axis labels shown in the figure are

step indices and not feature values.

The rules are generated from the pdf estimate plots by mapping the intersection contour of a threshold plane parallel to the feature axis plane and the surface. The threshold level is set at 0.6 times the peak value of the surface. The intersection of the threshold plane and the surface for the pdf estimate in figure 5.7 is shown in figure 5.8.
Since the pdf estimates and the rules are generated automatically, there is no need to approximate the intersecting area with a rectangle of the same area, as described in section 4.5.3 and figure 4.7. The data used to generate the contour shown in figure 5.8 is output from the rule generation algorithm in the form of a 33 x 33 matrix of zeros and ones. At locations inside the contour region, the matrix entries are equal to one. All other entries in the matrix are zeros. This provides a simple method for determining if an unknown feature vector falls in the region bounded by the contour plot or outside. If the matrix entry is one, the unknown feature vector will pass the rule. If the matrix entry is zero, it will not.

The remaining rules are generated for all six gestures using this procedure. Each gesture has six two dimensional rules, similar to that in figure 5.8, corresponding to all possible combinations of feature pairs. It is possible that when a pdf estimate is multi-modal, there will be more than one intersecting region. In such a case, figure 5.8 would have several acceptance regions. Feature vectors inside either of the regions will pass that rule. This method provides a means of generating rules that have no boundary shape

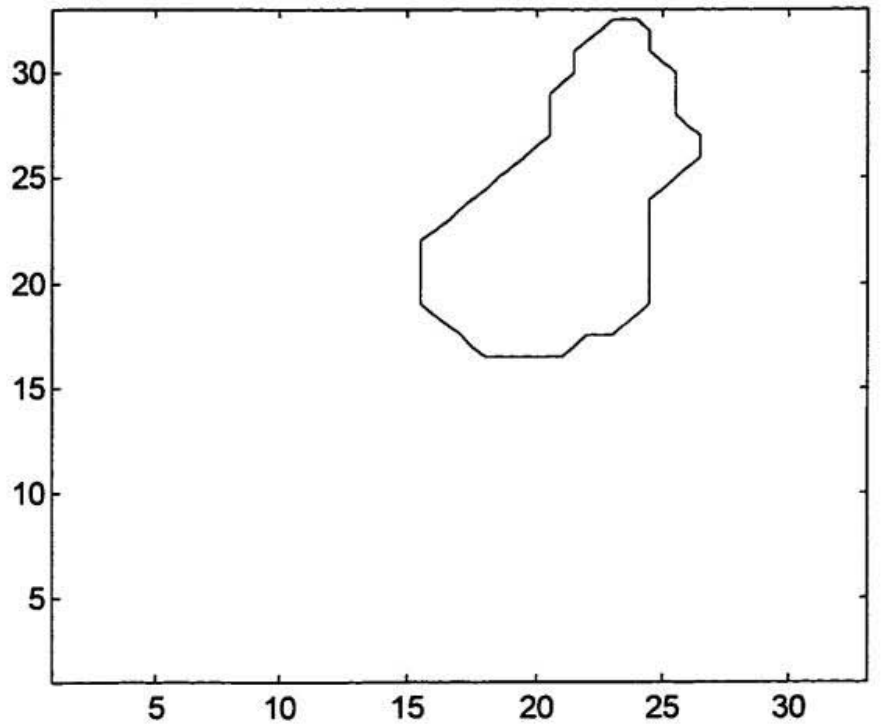


Figure 5.8 - Threshold Intersection

restrictions or restrictions on the possible number of acceptance regions.

The unknown gestures are tested using all six rules for each of the possible six classes. Classification is performed by assigning the unknown gesture to the class which has passed the most rules. The recognition rates are given in table 5.6. The highest recognition rate at 100% is for gesture 1. Gesture 3 has the lowest recognition rate at 65%. The highest confusion rate is 21%, gesture 3 incorrectly classified as gesture 5.

Table 5.6 - Confusion Matrix; Rule based recognition

100	0	0	0	0	0
0	99	0	1	0	0
0	0	65	14	21	0
2	1	6	88	2	1
2	0	1	1	88	8
0	0	0	0	6	94

An investigation of the pdf estimate plots for gestures 1, 2 and 6 can explain the high recognition rates. Figure 5.9 is a pdf estimate for gesture 1 using features 1 & 3, figure 5.10 is for gesture 2 using feature 1 & 2 and figure 5.11 is for gesture 6 using features 1

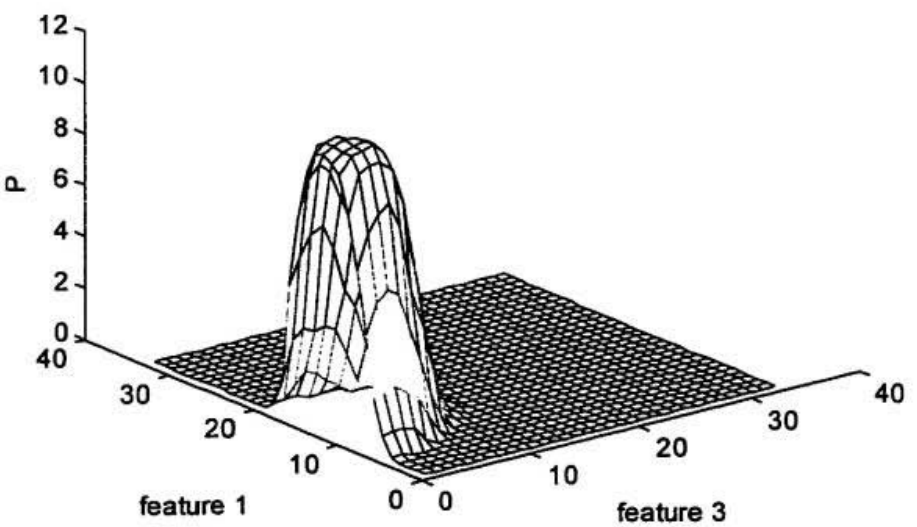


Figure 5.9 - pdf estimate; gesture 1 (features 1 & 3)

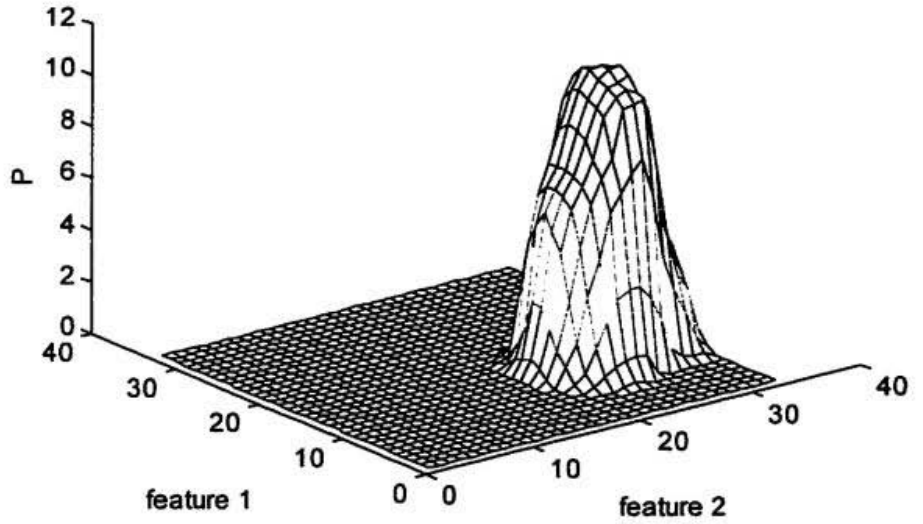


Figure 5.10 - pdf estimate; gesture 2 (features 1 & 2)

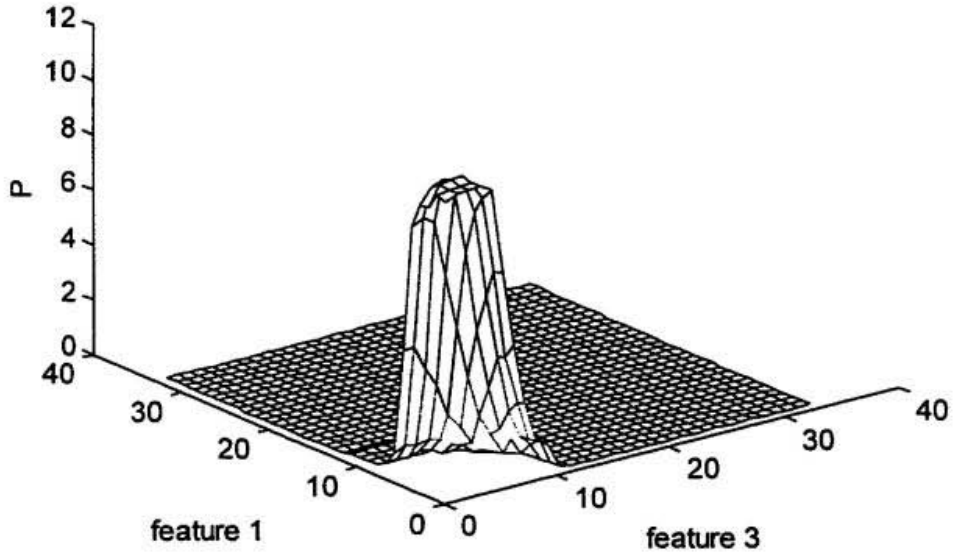


Figure 5.11 - pdf estimate; gesture 6 (features 1 & 3)

& 3. These plots generally show high narrow surfaces located at the corners or the edge of the feature axis plane. The resulting rules will contain acceptance regions which are small and lie at the corners or the edge of the feature axis plane. This arrangement makes these gestures distinctive and other gestures are unlikely to pass all these rules leading to classification errors. The pdf estimate plots for gesture 5 in comparison show two pdf estimate surfaces which are wide and are generally located at or near the center of the feature axis plane (figures 5.12 and 5.13). These pdf estimate plots will result in a set of rules with large acceptance regions near the center of the feature axis plane. These gestures are not distinctive and gestures from other classes easily satisfy these rules causing classification errors.

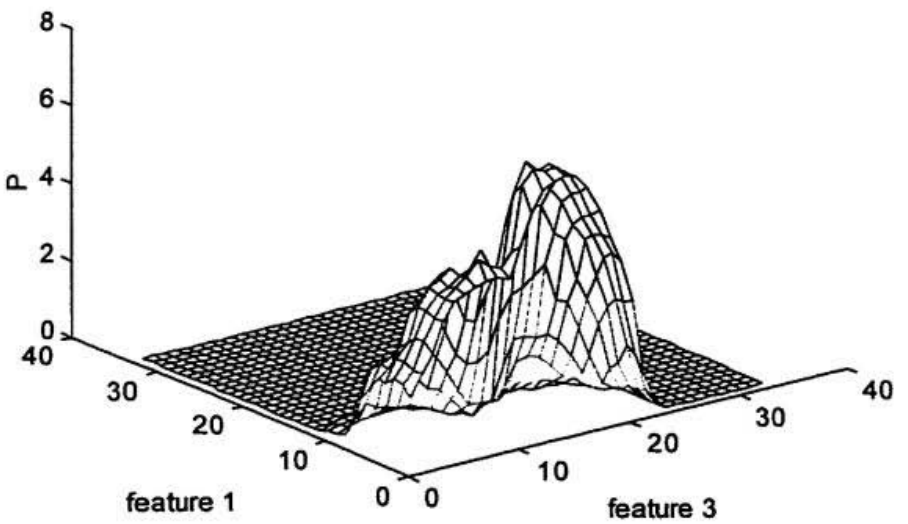


Figure 5.12 - pdf estimate; gesture 5 (features 1 & 3)

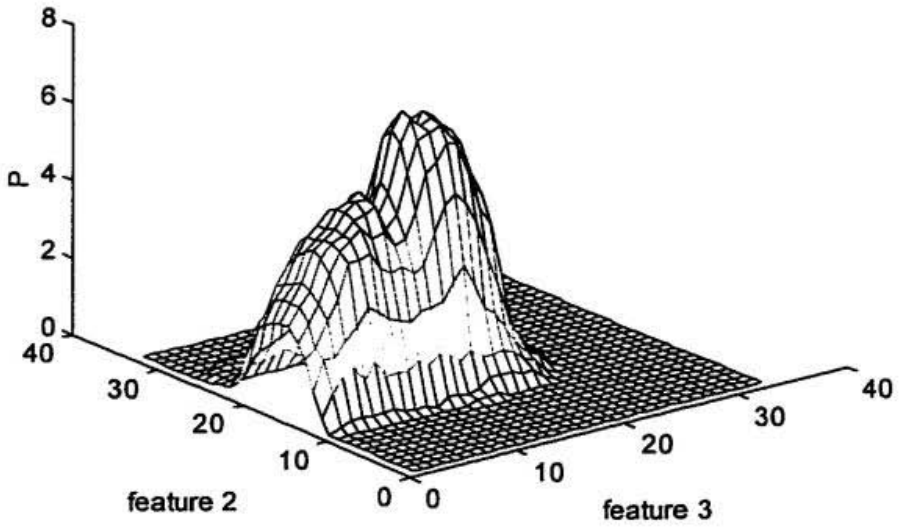


Figure 5.13 - pdf estimate; gesture 5 (features 2 & 3)

It is possible that the recognition results can be improved if a set of rules are chosen from the six rules which minimizes the number large general rules which may cause classification error. The other alternative is to select a new feature set which minimizes the overlap in the pdf estimate plots. A more sophisticated solution is to analyze the pdf plots in *n* dimensional space (*n* being the number of features, four in this case). Given the relatively good recognition results from the k-nearest neighbor classifier, it is likely the overlap in the feature space will be reduced with the addition of other dimensions. The projection of clusters onto feature axis planes may result in overlap which can be avoided if the pdf estimates are analyzed directly in four dimensional space. Conceptually this approach is possible. The 3D surfaces shown in Appendix A will be hyper surfaces and the rules are generated by an intersecting hyper plane. Practically however, the resulting

multi dimensional matrices would be very difficult to manage using the Matlab version 4.2c.1 platform.

The purpose of this research is to compare the four pattern recognition techniques when applied to a set of six hand gestures. Other than minor adjustments to the threshold level, the results reported here have not been optimized. If the threshold is set too high, the resulting acceptance regions may become small making passing a sufficient number of rules to be correctly classified difficult. Conversely, if the threshold is too low, the acceptance regions may become large and dominate the feature space. Many feature vectors will mistakenly be classified into that class.

5.7 Hidden Markov Models

The HMM algorithm requires an additional feature preparation process. The feature set for each of the gestures in the training set is represented by a set of features' output from the window feature extraction algorithm. This feature set must now be converted to a set of observation sequences by a process of vector quantization.

The vector quantization process transforms a multi-dimensional string of feature vectors into a single vector or observation sequence using a reproduction alphabet. The initial reproduction alphabet is chosen and optimized using the first feature vector in the set. This reproduction alphabet is then used throughout the remaining quantization. The result is a set of one hundred observation sequences for each of the six gestures in the

training set. The number of possible observations output from the vector quantization algorithm is set to Q=8. An example of a number of observation sequences is given below.

Gesture (1):	O_1=7	7	8	7	7	7	7	7	
	O_2=7	8	2	7	5	6			
Gesture (2):	O_1=6	6	8	6	6	6	4	6	
	O_2=6	6	8	5	3	8	6	6	6

These examples illustrate the differences between the observation sequences for each gesture after the vector quantization process. The number of observations in a sequence can vary depending on the length of the gesture.

Training of the hidden Markov models is sequential beginning with initial estimates of the model parameters A, B and π. These matrices are initially estimated by a random number generator. Each element of the matrices is randomly chosen from a uniform distribution of numbers ranging from 0 to 1. The size of the matrices is determined by the number of states in the hidden Markov models (N) and the number of possible observations or state outputs (Q). With N=8 and Q=8 the matrix A is an 8x8 (NxN), B is an 8x8 (NxQ), and π is an 8x1 (Nx1). The matrices are then normalized to form proper probability mass function estimates. Each row in the state transition matrix A and each row in the observation probability matrix B are normalized by dividing each element in a row by the sum of all elements in that row. This ensures that:

$$\sum_{j=1}^{N} a_{ij} = 1 \;;\; \sum_{k=1}^{Q} b_j(k) = 1 \qquad 5.2$$

The column elements of the initial state probabilities matrix π are divided by the sum of the column ensuring that:

$$\sum_{i=1}^{N} \pi_i = 1 \qquad 5.3$$

Training of the model parameters corresponding to gesture 1 or $\lambda_1=(A_1,B_1,\pi_1)$ begins by using the initial estimates of A_1, B_1, and π_1 to determine the probability P_1, that the first observation sequence for gesture 1, O_1, was produced by model λ_1. This probability and the observation sequence are then used to update the model parameters A_1, B_1, and π_1. Using the second observation sequence, O_2, the process is repeated. A second set of model parameters A_2, B_2, and π_2 are randomly chosen from a uniform distribution and the probability that this sequence was produced by the latest randomly generated model parameters is calculated P_2. The observation sequence O_2 and P_2 are then used to produce a second updated estimate of the model parameters A_2, B_2, and π_2. This process continues for all observation sequences used for training. At each iteration a set of model parameters (A_1, B_1, π_1, A_2, B_2, π_2, etc.) and the corresponding probabilities P_1, P_2, etc. are stored. These are then combined to produce the final estimate of the model parameters for model $\lambda_1=(A_1,B_1,\pi_1)$ corresponding to gesture 1.

This process is then repeated to obtain model parameters for the remaining five gesture classes, $\lambda_2=(A_2,B_2,\pi_2)$, $\lambda_3=(A_3,B_3,\pi_3)$, $\lambda_4=(A_4,B_4,\pi_4)$, $\lambda_5=(A_5,B_5,\pi_5)$ and $\lambda_6=(A_6,B_6,\pi_6)$. Training involved using all 100 observation sequences for each of the six gestures to obtain the final model parameter estimates. Testing was performed using much the same process used to test the rule based algorithm. The first observation sequence for gesture 1 is used with each of the model estimates λ_1, λ_2, λ_3, λ_4, λ_5 and λ_6 to obtain six probability estimates P_1, P_2, P_3, P_4, P_5, and P_6 respectively. The probability P_1 is the probability that the observation sequence was produced by model λ_1 and so on. The observation sequence is classified as the gesture class associated with the highest probability. This process is continued for all 100 observation sequences for each gesture. The results of this test are detailed in table 5.7.

Table 5.7 - Confusion Matrix; HMM - N=8, Q=8

87	5	0	4	0	4
16	84	0	0	0	0
0	0	65	29	0	6
1	0	26	71	0	2
0	0	1	0	99	0
3	0	8	0	0	89

The highest recognition rate was 99% for gesture 5 with the lowest recognition rate for gesture 3 at 65%. The highest confusion rate was for gesture 3 incorrectly classified as gesture 4 at 29%. These results strongly indicate that the observation sequences

corresponding to gesture 3 and those corresponding to gesture 4 are very similar. Figure 5.14 is a plot of the first ten observation sequences from gestures 3 and 4. This plot clearly shows the similarities between the observation sequences for gesture 3 and those for gesture 4. Each start low, increasing to between 6 and 8 in the center and reduce to 1 or 2 at the end. These two gestures are very similar geometrically and this similarity is evident in the observation sequences produced by the vector quantization process.

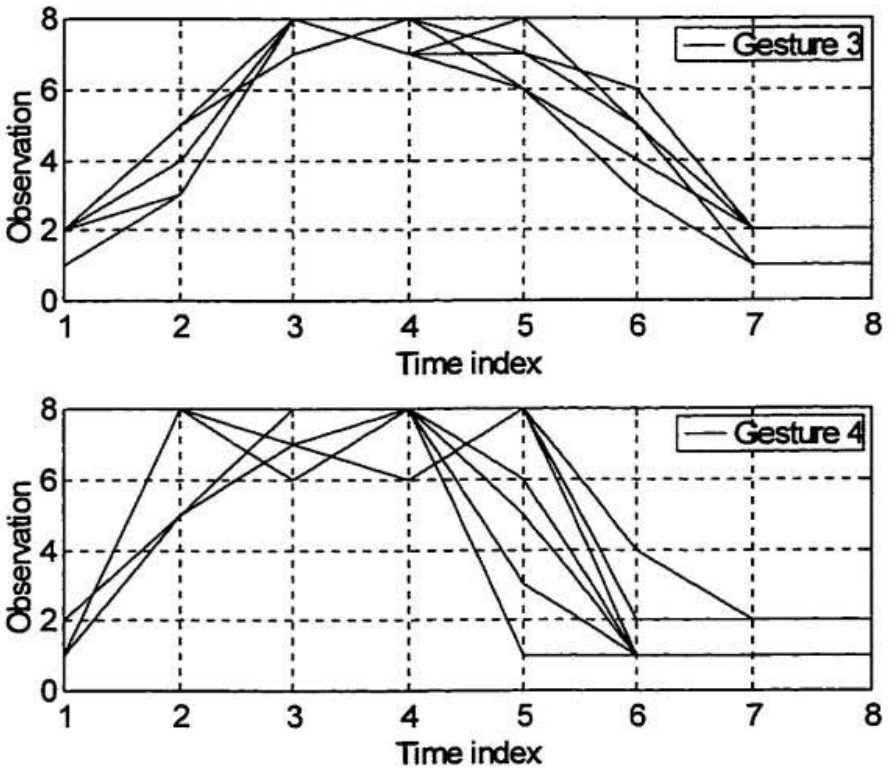


Figure 5.14 - Observation Sequence Comparison; gestures 3 & 4

A similar result might be expected for gestures 5 & 6. These are geometrically similar to each other, as are gestures 3 & 4. A look at the plot of ten of the observation sequences

for these two gestures (figure 5.15) however, illustrates that in fact these sets of observation sequences are quite dissimilar as indicated by the good recognition results. Many of the observation sequences for gesture 5 are exactly the same and overlay each other in the plot. This again is reflected in the near perfect recognition results for gesture 5.

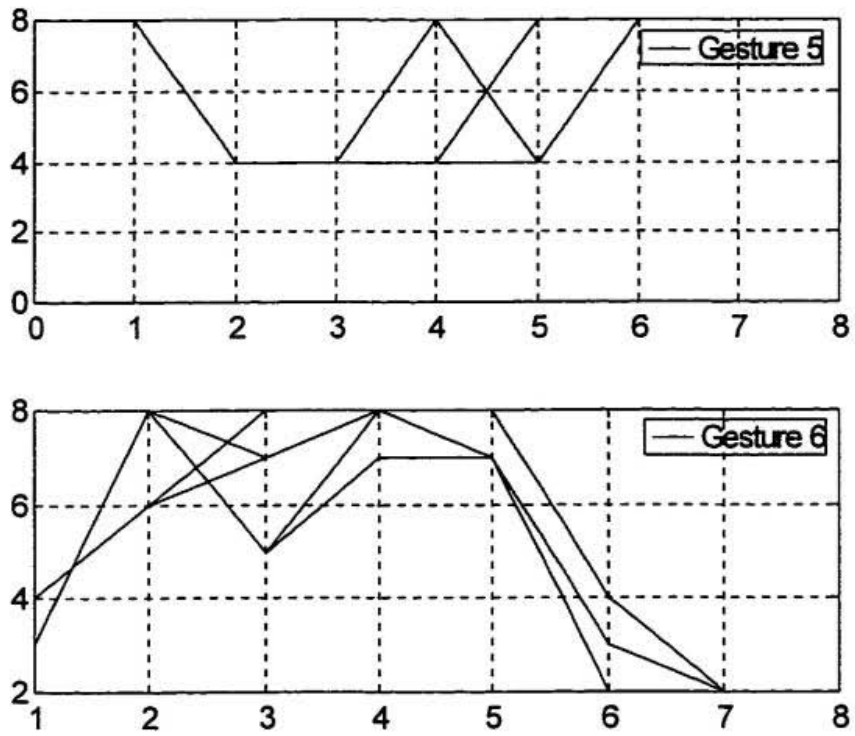


Figure 5.15 - Observation Sequence Comparison; gestures 5 & 6

Although all gestures were collected from the author it is possible that a difference exists between the collection process of gestures 5 & 6 and during the collection process of gestures 3 & 4. The movements for both gestures 3 & 4 are to the right while the movements for gestures 5 & 6 are to the left. The left movement was more unnatural and

required more concentration possibly leading to a more precise movement which resulted in a set of more distinct gestures. Results may be improved if the number of possible observation symbols were increased. This would allow a wider range of possible outcomes and may produce more distinct observation sequence sets for each gesture. There are many other variables to adjust throughout the process beginning with the selection of other features through to adjusting the number of possible states in the hidden Markov model estimations.

5.8 Discussion

The development and implementation of most pattern recognition systems to applications involves issues other than the recognition rates alone. This applies to practically all applications including development of assistive devices or the automation of an industrial process. The scope of this thesis does not include an extensive study of implementation issues but a comparison of recognition rates alone may be misleading without mention of some practical implementation issues.

A significant issue in most applications, specifically those requiring real-time processing, is the issue of algorithm speed. The actual speed constraints are completely application dependent. Vision based automation processes such as defect detection in lumber and lumber products generally involve incorporation into an existing system at existing production speeds. This places serious limitations on the time available for image processing. This will translate to a limitation on the number and complexity of features

available for subsequent classification algorithms.

Gesture recognition applications requiring real-time processing have similar limitations. The hardware interface will largely dictate the actual pattern recognition algorithm selection. The sensor based system used during data collection here requires the analysis of both position and orientation data. The types of features are limited to the positional and dynamic features discussed previously.

Table 5.8 summarizes the recognition results for the four classification techniques. Recognition results are relatively high using all four classification techniques for gestures 1, 2, 5, and 6. Gestures 3 and 4 have a decrease in recognition rates for the rule based classifier and the hidden Markov model classifier.

Table 5.8 - Recognition Results (per gesture)

	Gesture 1	Gesture 2	Gesture 3	Gesture 4	Gesture 5	Gesture 6
KNN;k=1	100	99	88	93	96	97
KNN;k=3	100	99	96	95	93	98
KNN;k=5	100	99	91	94	94	98
Mal. dist.	100	99	70	98	90	93
Rule	100	99	65	88	88	94
HMM	87	84	65	71	99	89

The best overall recognition rate for all gestures was obtained using the k-nearest

neighbor classifier with k=3 (table 5.9). The recognition results obtained from the hidden Markov model classifier was the lowest at 83% overall recognition accuracy for all six gestures.

Table 5.9 - Average Recognition Results (all gestures)

KNN;k=1	KNN;k=3	KNN;k=5	Mal. dist.	Rule	HMM
96%	97%	96%	92%	89%	83%

The choice of a classification technique for these six gestures based on speed and performance would be the Mahalanobis distance classifier. Although the k-nearest neighbor classifier has a higher overall recognition rate, the fact that it is non-parametric requires the use of the complete training set during classification. This means that each unknown gesture is compared in four dimensional feature space to all other 600 known gestures. The distance to each of the 600 known gestures is calculated and the minimum *k* distances are used for classification. Even with this relatively small training set and only four features this can be quite time consuming. With the addition of more gesture types, bigger training sets, and higher dimensional feature space this process becomes very slow. There are search methods such as k-d trees which speed up the performance of k-nearest neighbor classifiers but the gain in speed is generally balanced by some loss in recognition performance.

The Mahalanobis distance classifier is parametric and uses the training data only during training. Classification is simply a matter of calculating a single Mahalanobis distance to

each class mean and choosing the minimum. Both the Mahalanobis distance classifier and the k-nearest neighbor classifier are attractive in that they are simple to implement and can provide good recognition results. One of the disadvantages of these algorithms is the lack of tuning parameters. Complex practical problems rarely yield acceptable recognition results with the first implementation of a classification algorithm. The rule based classifier and the hidden Markov model classifier may be more appropriate for these applications. With sufficient optimization, both are capable of recognizing more complex gestures.

There are implementation issues other than algorithm speed and performance worth mentioning. A large field of research is devoted to the development of adaptive pattern recognition systems. An application example is a commercial speech recognition system. Initially, the system performs moderately well for a wide range of users. With sufficient use, the system can adapt to a specific user with increased performance. The hidden Markov model classifier has been used extensively in the area of speech recognition largely because it can be taught to be adaptive.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

6.0 Conclusions

This research investigated four pattern recognition techniques and their application to the area of hand gesture recognition. The four pattern techniques studied were the k-nearest neighbor classifier, a Mahalanobis distance classifier, a rule based classifier, and hidden Markov model classification. A set of six hand gestures were used to test each of the four techniques. The six hand gestures were collected from the author using a sensor based motion tracking system called "The Flock of Birds". The gesture data contained both three dimensional position information and three dimensional orientation information. The hand gestures were general movements with a range of approximately one foot.

A test set of 100 of each of the gestures was collected. The gestures were then segmented using a two threshold tremor filter. The segmented gestures were than mapped into a four dimensional feature space. This set of four features were used during the testing of all four pattern recognition techniques.

The k-nearest neighbor classifier was tested using k=1, k=3 and k=5 neighbors. The best recognition results observed were using k=3 neighbors. The average recognition rate for all six gestures was 97%. The test with both k=1 and k=5 neighbors resulted in an average recognition rate of 96% for all six gestures. The best recognition rate with both k=1 and k=5 neighbors was gesture 1 at 100%. The lowest recognition rate for k=1 neighbor was for gesture 3 at 88%; the lowest recognition rate for k=3 neighbors was for gesture 5 at 93% and the lowest recognition rate for k=5 neighbors was for gesture 3 at 91%. An immediate conclusion is that the classes in the four dimensional feature space cluster fairly well. The high recognition rates for gesture 1 is not surprising. Both gestures 1 and 2 are simple movements, vertically and horizontally respectively.

The k-nearest neighbor classifier is very simple and is perhaps the easiest to implement. It is commonly used as a basis for comparison of other classification techniques. The k-nearest neighbor classifier can however be very slow in applications with large numbers of training sets and a relatively high dimensional feature space. Perhaps more significantly, due to the simplicity of the algorithm, there is little opportunity to optimize the results other than to adjust the number of neighbors or to select different feature sets.

The Mahalanobis distance classifier produced an average recognition rate for all six gestures of 92%. Again gesture 1 had the highest recognition rate at 100%. The lowest recognition rate was for gesture 3 at 70%. This result may initially be a little surprising considering the good recognition results from the k-nearest neighbor classifier for gesture

3. However, the Mahalanobis distance classifier is a parametric classifier which makes the underlying assumption that the data along each feature axis is normally distributed. It is likely that this assumption is in part responsible for this low recognition rate.

The Mahalanobis distance classifier has a speed advantage over the k-nearest neighbor classifier. Its parametric nature excludes the need to use all training data during the classification of each unknown gesture. The speed issue is very important when considering classifier development intended for incorporation into real-time processing.

The rule based classifier produced an average recognition rate of 89%. The highest recognition rate was for gesture 1 at 100%. The lowest recognition rate observed was for gesture 3 at 65%. The results of this test indicate a significant confusion rate of 21% between gestures 3 and 5. These gestures are similar and have overlap in the cluster projections onto the feature axis.

The rule based classifier uses statistical information about the distribution estimates of the class data to automatically generate rules. The rules express this statistical information. This approach does not make any assumptions regarding the class clusters. A number of variables make the rule based classification technique tunable which can prove important for more complex problems. The threshold level for rule generation can be adjusted and the sets of rules during classification can be altered, both adjustments will change recognition results.

The hidden Markov model classifier has the lowest overall average recognition rate at 83%. The highest recognition rate was 99% for gesture 5. The lowest recognition rate was for gesture 3 again at 65%. The highest confusion rate was between gestures 3 and 4 at 29%. The observation sequences for gestures 3 and 4 are noticeably similar. The hidden Markov model classifier uses only the observation sequence representation of the gestures, making recognition difficult when observation sequences are similar for two or more gestures.

The hidden Markov model classifier provides the most opportunity for optimization. The vector quantization algorithm required to map the feature sets into observation sequences can be adjusted. The number of possible observation outcomes and the reproduction alphabet used can be changed, both of which will likely change the recognition rates. In the hidden Markov models themselves, the number of states and the model architecture can be adjusted. The hidden Markov models used during testing were ergodic models. This generalized form allows state transitions in all directions. Other specific architectures like the left-to-right models may improve recognition results for some applications.

Each of the four classification techniques performed reasonably well on the six gestures tested. The complexity of each technique varies widely from the k-nearest neighbor classifier to the hidden Markov model classifier. Considering both recognition rates and the speed issue, the Mahalanobis distance classifier is best suited to classifying the six

gestures tested. With the addition of a larger more complex gesture set, a more sophisticated approach like the rule based or the hidden Markov model classification may be required.

6.1 Future Work

The rule based approach discussed in this thesis can be improved with the development of a process of analyzing the feature data in multi-dimensional space. Currently each feature pair is processed individually resulting in a number of two dimensional rules. Analysis in multi dimensional space would eliminate the projections onto two dimensional axis planes and utilize the full cluster separation found in the multi dimensional space.

The hidden Markov Models studied in this thesis were discrete models. Segmentation of the gesture data stream is necessary when using discrete hidden Markov Models. The segmentation process can cause recognition errors if gestures are segmented prematurely. The continuous form of the hidden Markov Models accept the continuous gesture data stream alleviating the segmentation step. The development of continuous hidden Markov Models could potentially provide a sophisticated real time gesture recognition algorithm.

References

Ascension Technology Corp., (1995). *The Flock of Birds Installation and Operation Guide*, 123 p.

Birk, H. and Moeslund, T.B. (1996). "Recognizing Gestures from the Hand Alphabet Using Principal Component Analysis", *Master's Thesis*, Aalborg University, Denmark, 169 p.

Brown, C., Kostraba, J. R. and Cavalier, A. (1991). "Eyegaze and Headpointing Technology for Children and Adults with Mental Retardation", *RESNA 14th Annual Conference*, Kanas City, MO., pp. 383-385.

Cairns, A.Y. (1993). "Towards the Automatic Recognition of Gesture", *PhD Thesis*, University of Dundee, 163 p.

deVeth, J. and Boves, L. (1997). "Phase-Corrected Rasta for Automatic Speech Recognition over the Phone", *Proc. ICACCP-97*, Munich, Germany, pp. 1239-1252.

Devore, J.L. (1987). *Probability and Statistics for Engineering and the Sciences*, 2nd Edition, Brooks/Cole Publishing Company, Monterey, CA.,, 672 p.

Duda, R.O. and Hart, P.E. (1973). *Pattern Classification and Scene Analysis*, Wiley, New York, 273 p.

Gamage, L.B., Gosine, R.G. and deSilva, C.W. (1996). "Extraction of rules from natural objects for automated mechanical processing", *IEEE Transactions on Systems, Man and Cybernetics*, 20(1), pp. 105-120.

Gamage, L.D.K.B. (1993). "A Model-Based Approach to Complex Contour Generation for Process Automation using Computer Vision", *Phd Thesis*, University of British Columbia, Canada, 210 p.

Gillies, A.M. (1992). "Cursive Word Recognition Using Hidden Markov Models", *United States Postal Service: Advanced Technology Conference*, Vol. 1, pp. 557-562.

Harrington, M. E., Daniel, R. W. and Kyberd, P. J. (1995). "The European Context for Assistive Technology", *Proc. 2nd Tide Congress*, pp. 432-435.

Harwin, W. (1991). "Computer Recognition of the Unconstrained and Intentional Head Gesture of Physically Disabled People", *Phd Thesis*, University of Cambridge, UK., 139 p.

Huang, X., Acero, A., Alleva, F., Hwang, M-Y., Jiang, L. and Mahajan, M. (1995). "Microsoft Windows Highly Intelligent Speech Recognizer: Whisper", *IEEE International Conference on Acoustics, Speech and Signal Processing*, USA.

Kim, J. (1988). "On-Line Gesture Recognition by Feature Analysis", *Proceedings of Vision Interface*, Edmonton, Alberta, pp. 51-55.

Levinson, S.E., Rabiner, L.R. and Sondhi, M.M. (1983). "An Introduction to the Application of the Theory of Probabilistic Functions of a Markov Process to Automatic Speech Recognition", *The Bell Systems Technical Journal*, no. 62, pp. 1035-1074.

Linde, Y., Buzo,A. and Gray, R.M. (1980). "An Algorithm for Vector Quantizer Design", *IEEE Transactions on Communications*, Vol. 28, no. 1, pp. 84-95.

Malaviya, A., Leja, C. and Peters, L. (1996). "A Hybrid Approach of Automatic Fuzzy Rule Generation for Handwriting Recognition", *Pre-proceedings of the Fifth Workshop on the Frontiers of Handwriting Recognition (IWFHR5)*, Colchester.

Mohamed, M. and Gader, P. (1996). "Handwritten Word Recognition Using Segmentation-Free Hidden Markov Modeling and Segmentation-Based Dynamic Programming Techniques", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 18, no. 5, pp. 548-554.

Morgan, N. and Bourlard, H. (1995). "An Introduction to Hybrid HMM/Connectionist Approach", *IEEE Signal Processing Magazine*, May 1995, pp. 25-40.

Omologo, M., Matassoni, M., Svaizer, P. and Giuliani, D. (1997). "Hands-Free Speech Recognition in a Noisy and Reverberant Environment", *Proc. Of the ESCA-NATO Workshop on Robust Speech Recognition for Unknown Communication Channels*, Point-a-Mousson, France, pp. 195-198.

Perricos, C. (1995). "Head Gestures as a Means of Human-Computer Communication in Rehabilitation Applications", *PhD Thesis*, University of Cambridge, UK., 191 p.

Rabiner, L.R. (1989). "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition", *Proceedings of the IEEE*, Vol. 77, no. 2, pp. 257-285.

Shein, F., Hamann, G., Brownlow, N., Treviranus, J., Milner, M. and Parnes, P. (1991). "WIVIK: A Visual Keyboard for Windows 3.0", *RESNA 14th Annual Conference*, Kanas City, MO., pp. 160-162.

Starner, T. and Pentland, A. (1995). "Real-Time American Sign Language Recognition

from Video Using Hidden Markov Models", *M.I.T. Media Laboratory Perceptual Computing Section Technical Report*, no. 375.

Takahashi, K., Seki, S. and Oka, R. (1994). "Time-varying Image Processing and Moving Recognition", *Proceedings of the 4th International Workshop*, Amsterdam, London, pp. 65-71.

Tew, A.I. and Gray, C.J. (1993). "A Real-time Gesture Recognizer Based on Dynamic Programming", *Journal of Biomedical Engineering*, Vol. 15, pp. 181-187.

Therrien, C. W. (1989). *Decision Estimation and Classification*, John Wiley & Sons, Inc., 251 p.

Wu, J., Yan, H. and Chalmers, A. (1994). "Handwritten Digit Recognition using Two-Layered Self-Organizing Maps", *International Journal of Neural Systems*, Vol. 5, no. 4, pp. 357-362.

APPENDIX A
PDF ESTIMATE PLOTS

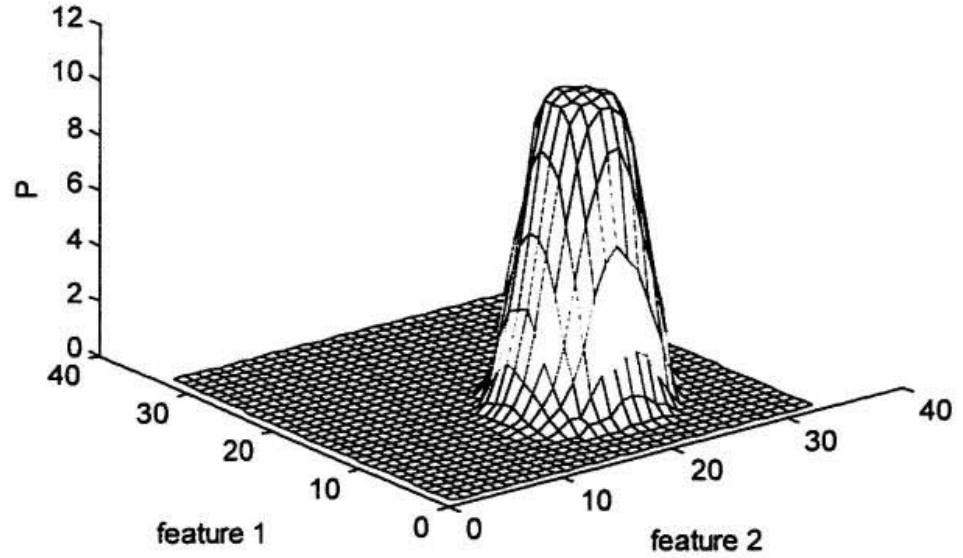


Figure A1 - pdf estimate; gesture 1 (features 1 & 2)

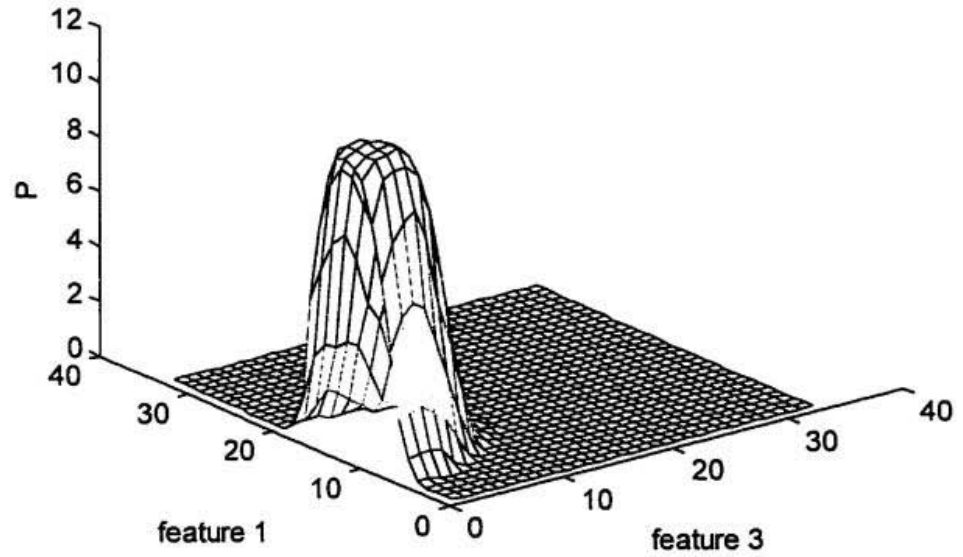


Figure A2 - pdf estimate; gesture 1 (features 1 & 3)

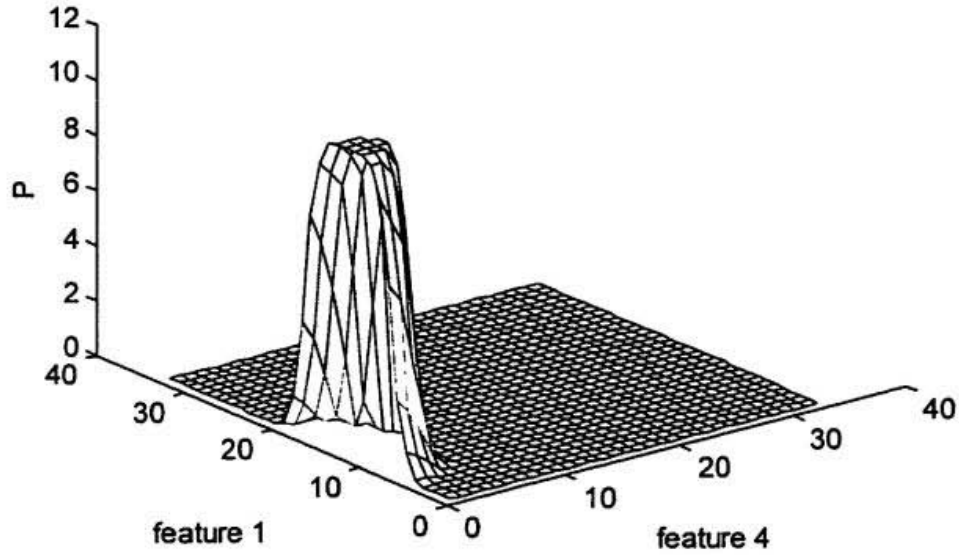


Figure A3 - pdf estimate; gesture 1 (features 1 & 4)

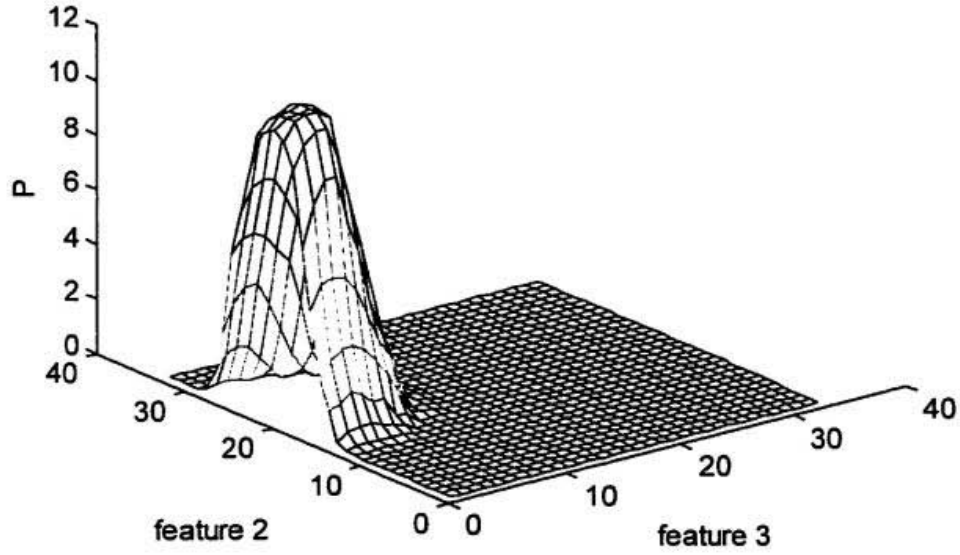


Figure A4 - pdf estimate; gesture 1 (features 2 & 3)

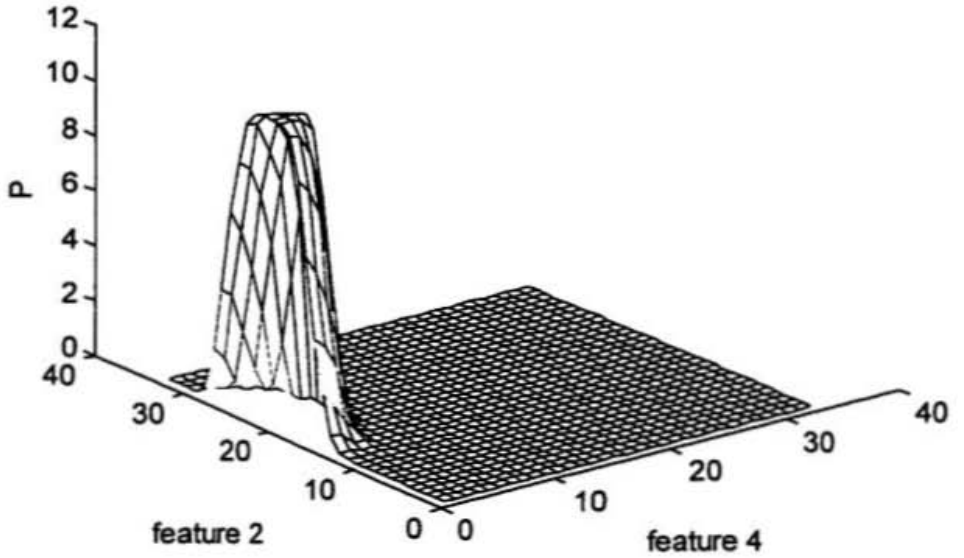


Figure A5 - pdf estimate; gesture 1 (features 2 & 4)

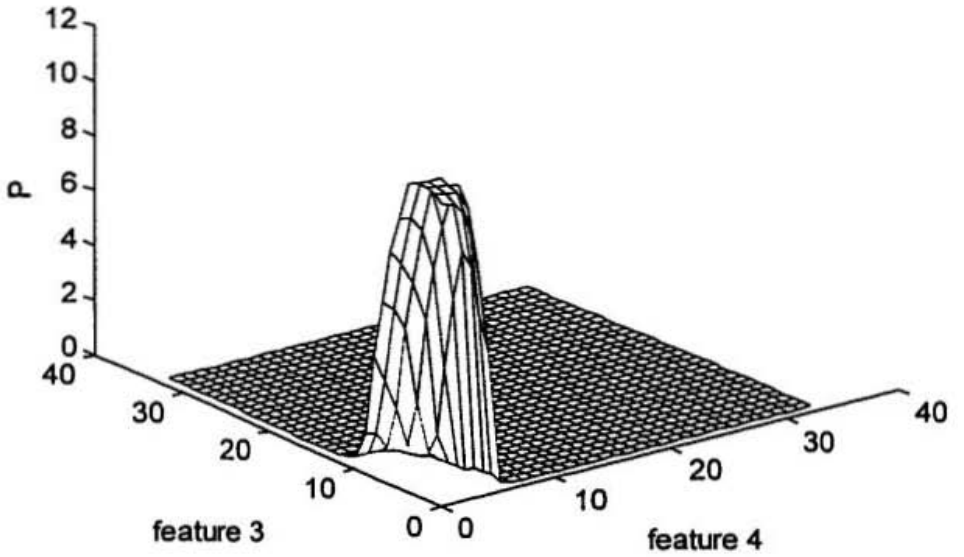


Figure A6 - pdf estimate; gesture 1 (features 3 & 4)

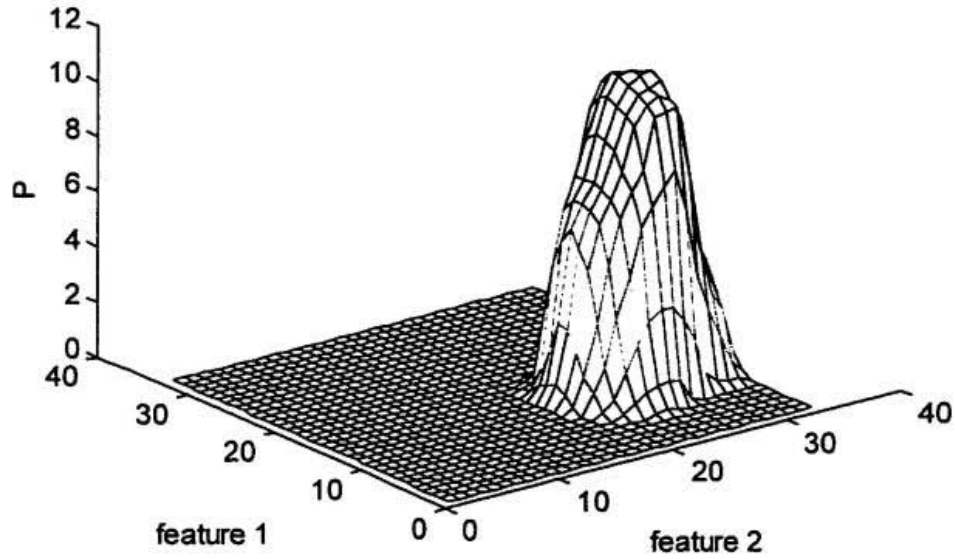


Figure A7 - pdf estimate; gesture 2 (features 1 & 2)

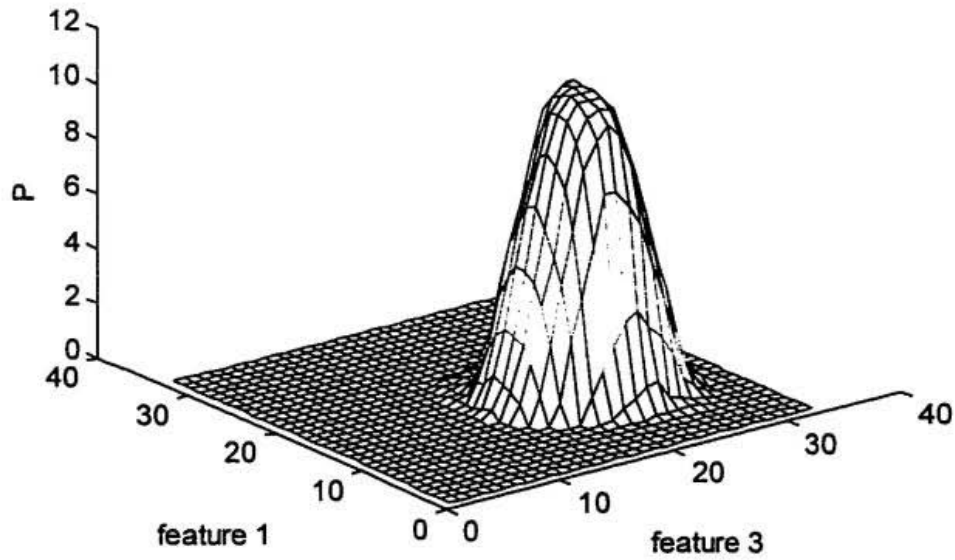


Figure A8 - pdf estimate; gesture 2 (features 1 & 3)

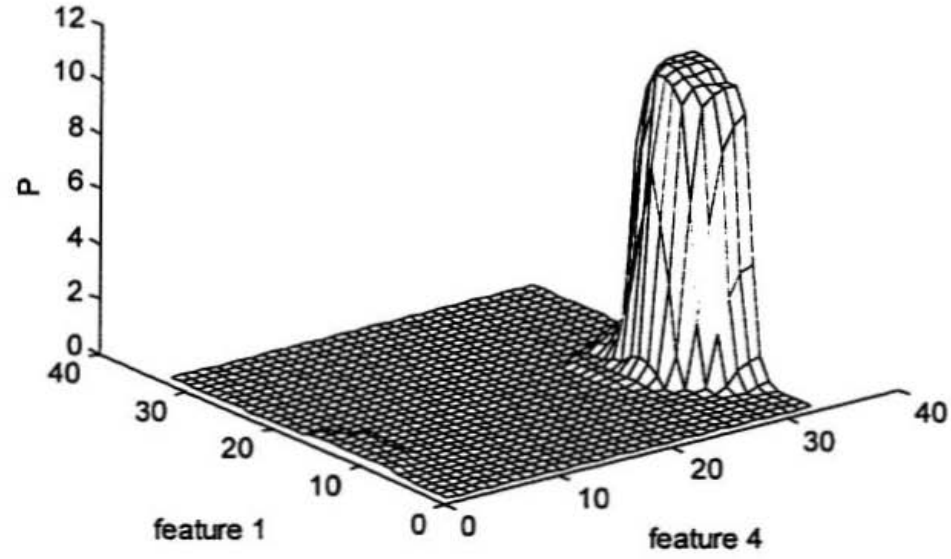


Figure A9 - pdf estimate; gesture 2 (features 1 & 4)

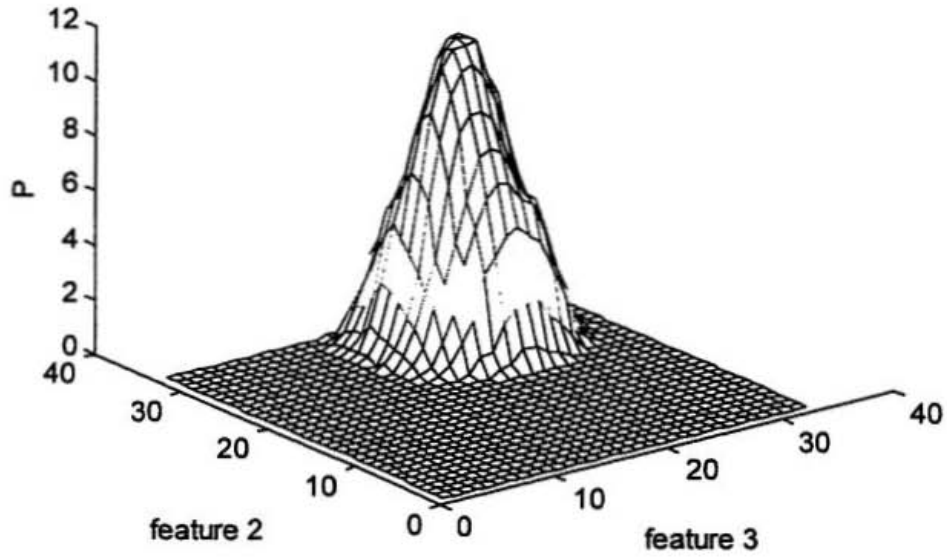


Figure A10 - pdf estimate; gesture 2 (features 2 & 3)

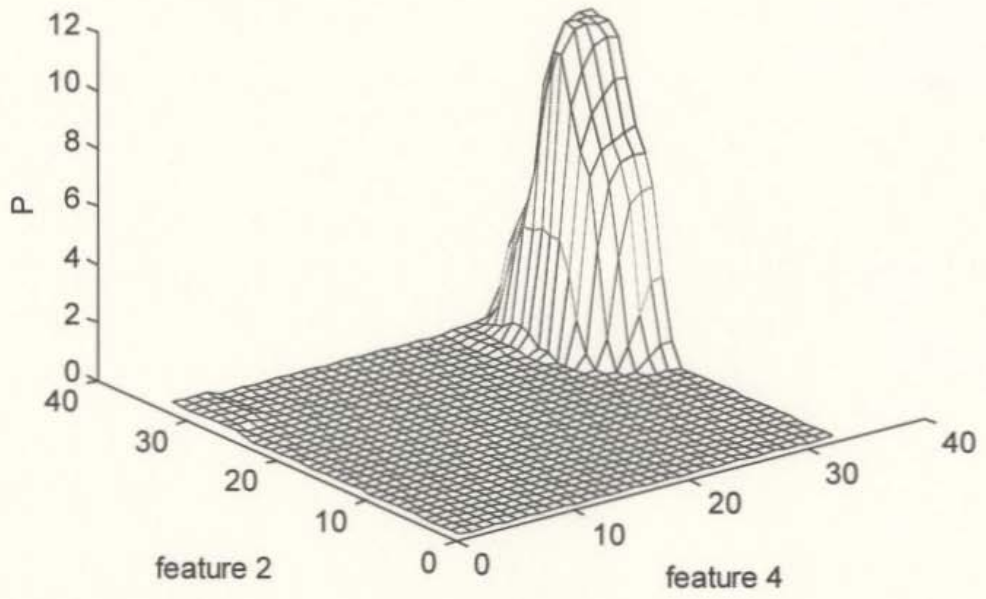


Figure A11 - pdf estimate; gesture 2 (features 2 & 4)

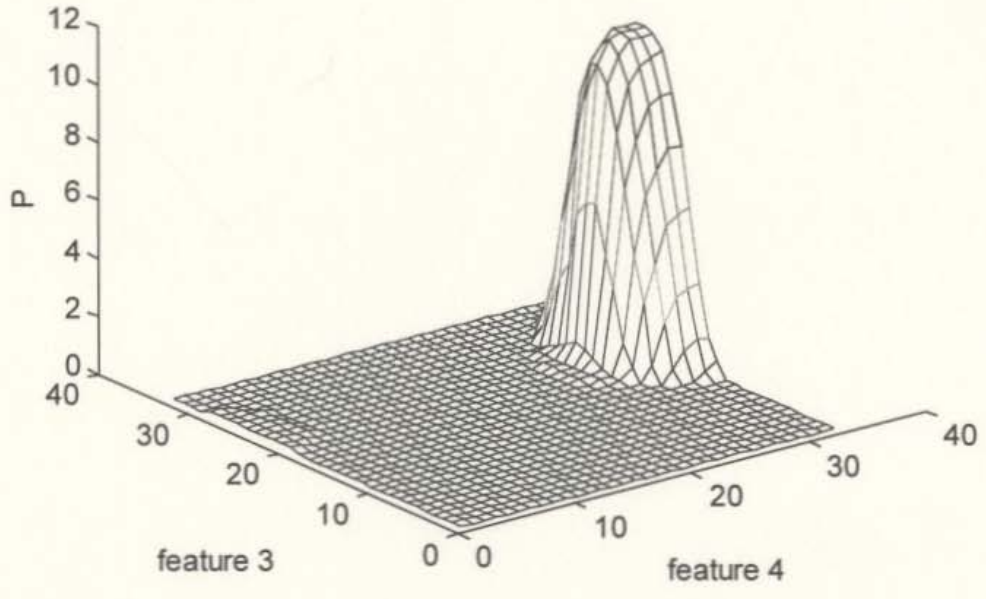


Figure A12 - pdf estimate; gesture 2 (features 3 & 4)

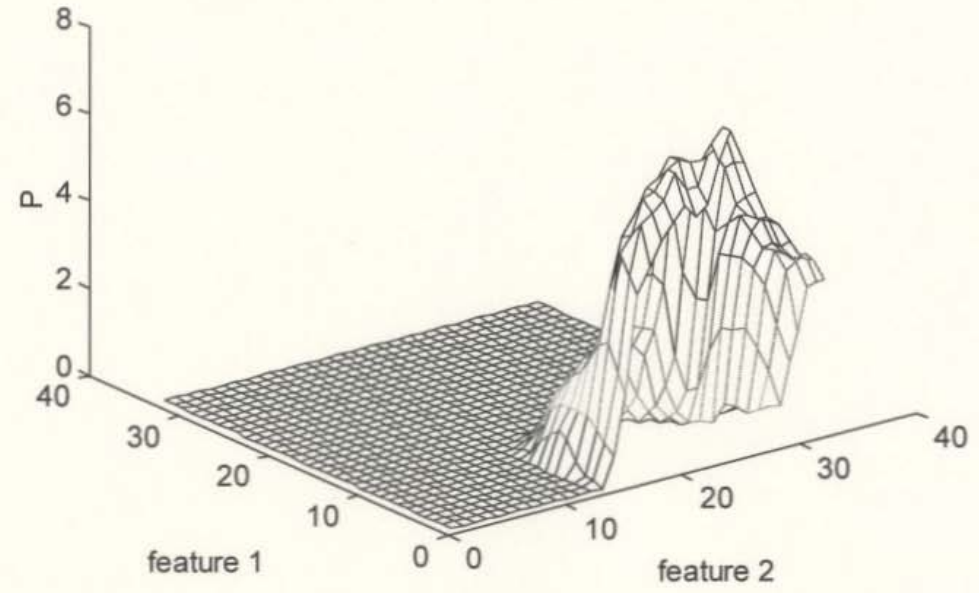


Figure A13 - pdf estimate; gesture 3 (features 1 & 2)

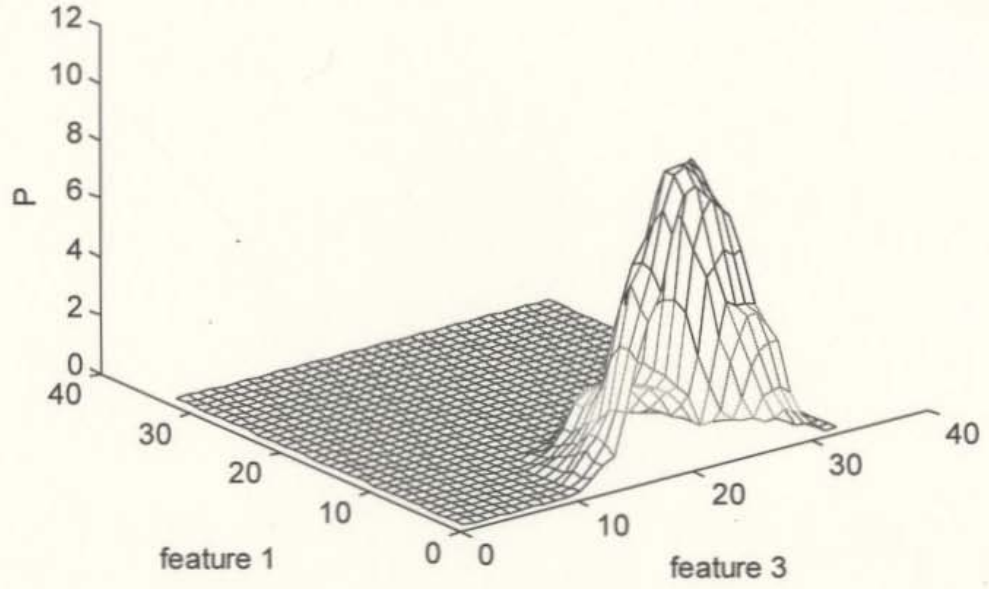


Figure A14 - pdf estimate; gesture 3 (features 1 & 3)

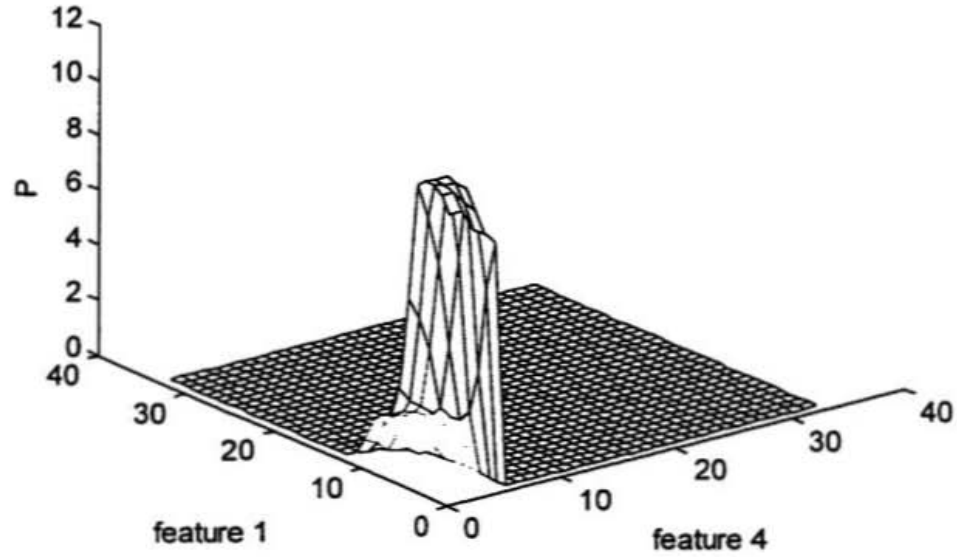


Figure A15 - pdf estimate; gesture 3 (features 1 & 4)

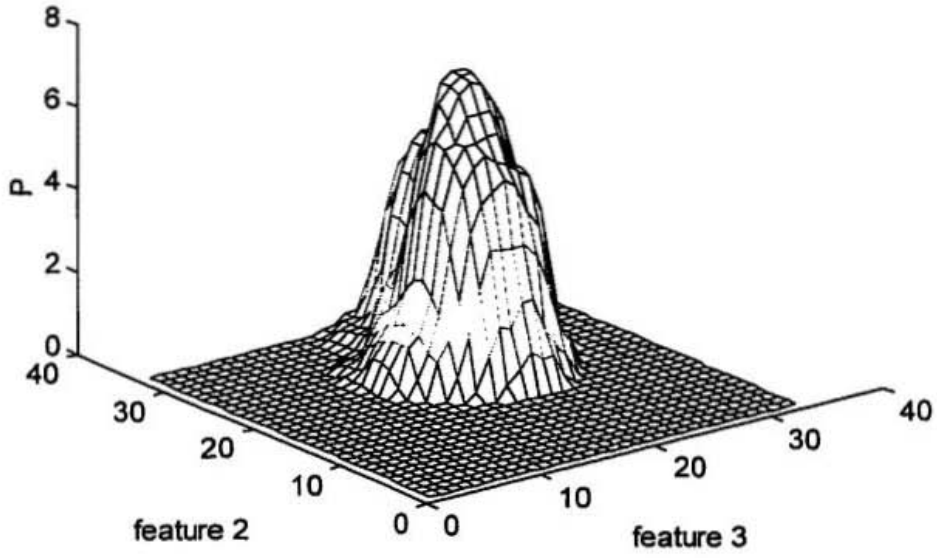


Figure A16 - pdf estimate; gesture 3 (features 2 & 3)

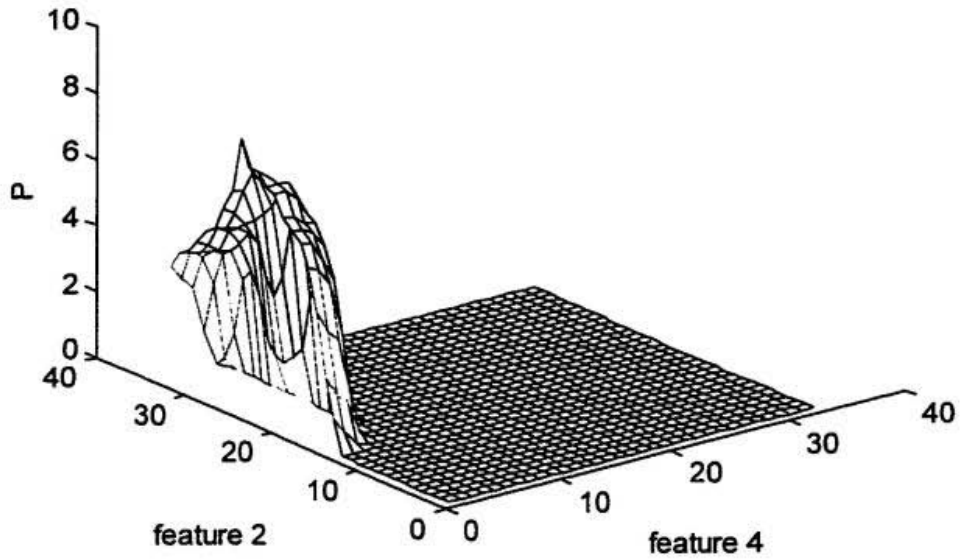


Figure A17 - pdf estimate; gesture 3 (features 2 & 4)

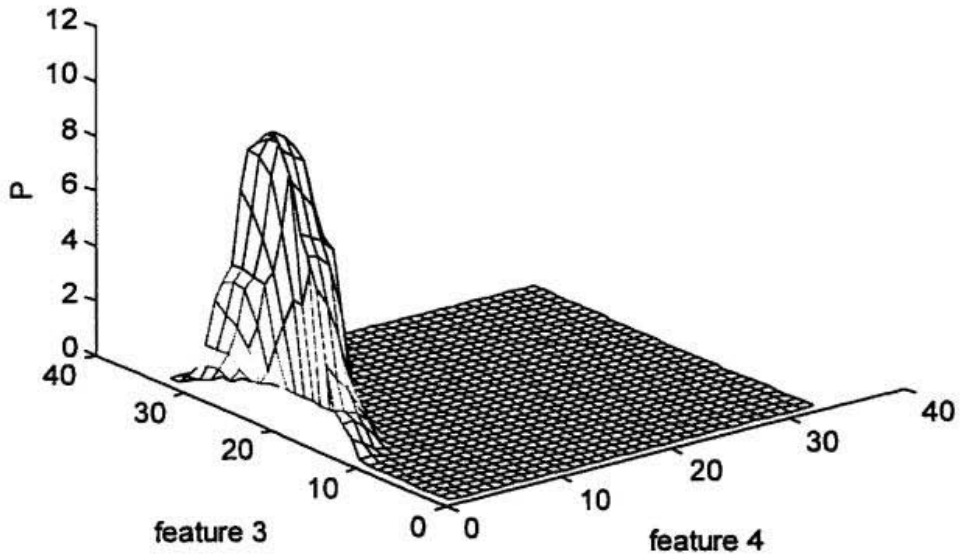


Figure A18 - pdf estimate; gesture 3 (features 3 & 4)

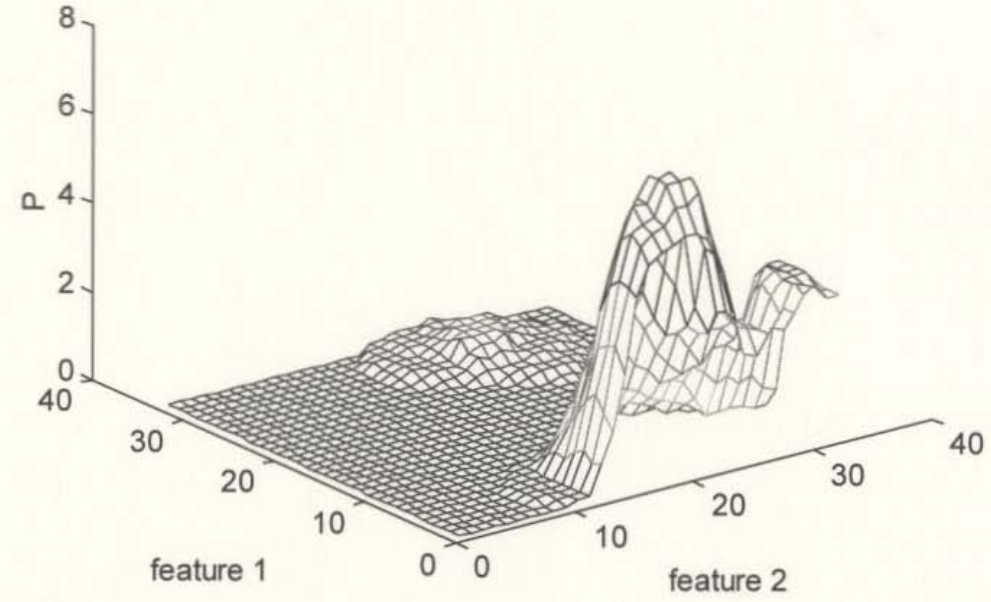


Figure A19 - pdf estimate; gesture 4 (features 1 & 2)

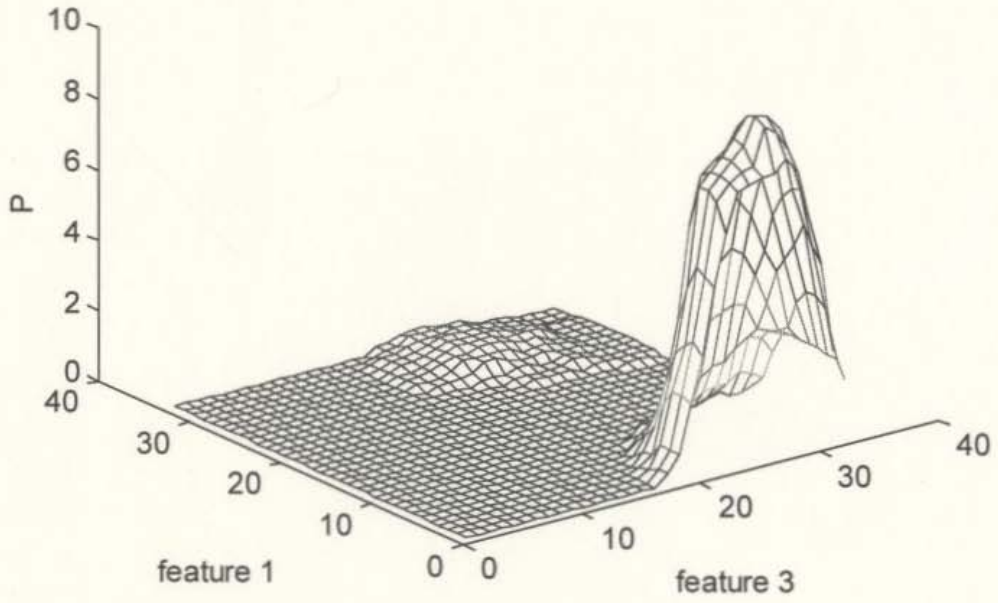


Figure A20 - pdf estimate; gesture 4 (features 1 & 3)

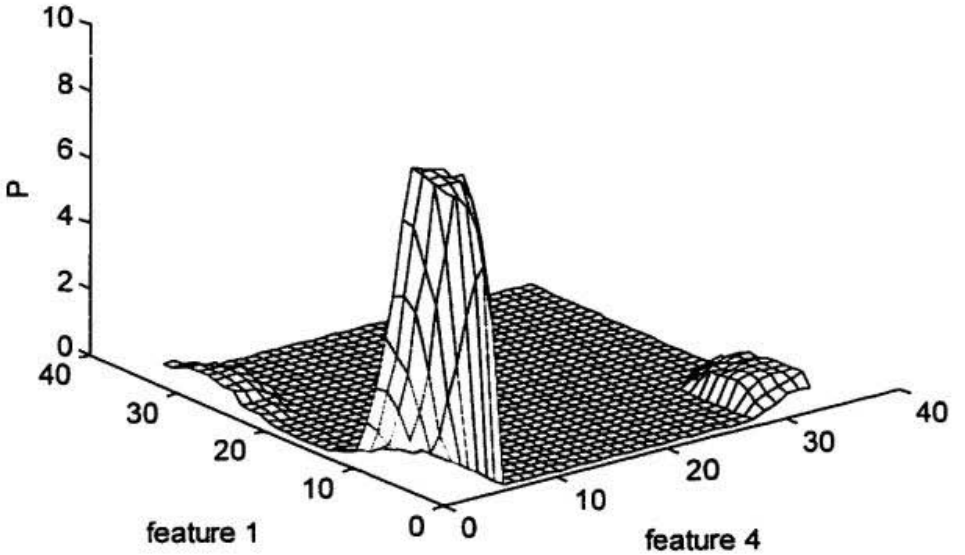


Figure A21 - pdf estimate; gesture 4 (features 1 & 4)

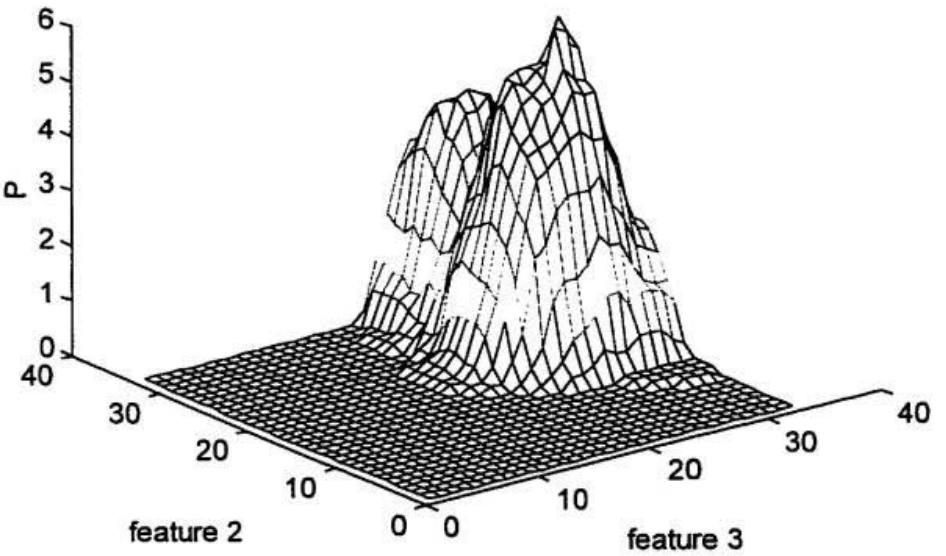


Figure A22 - pdf estimate; gesture 4 (features 2 & 3)

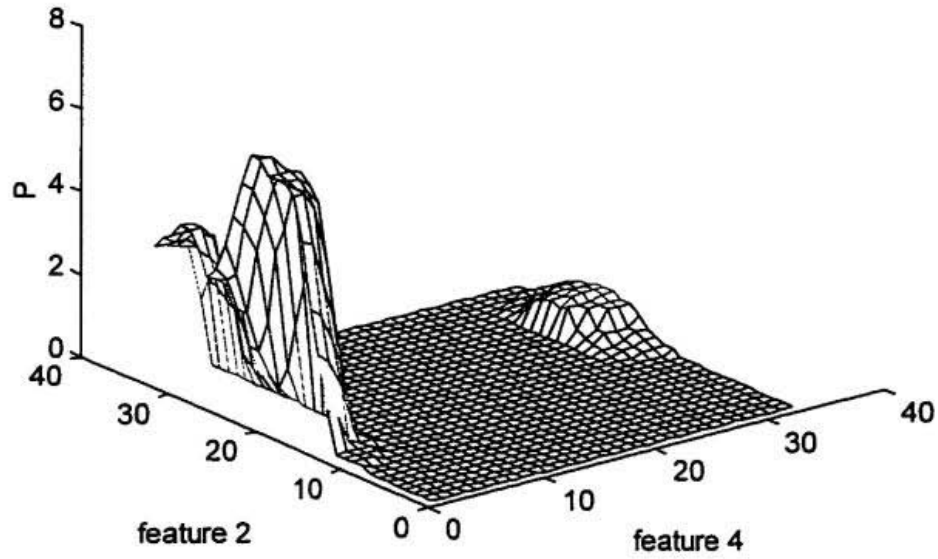


Figure A23 - pdf estimate; gesture 4 (features 2 & 4)

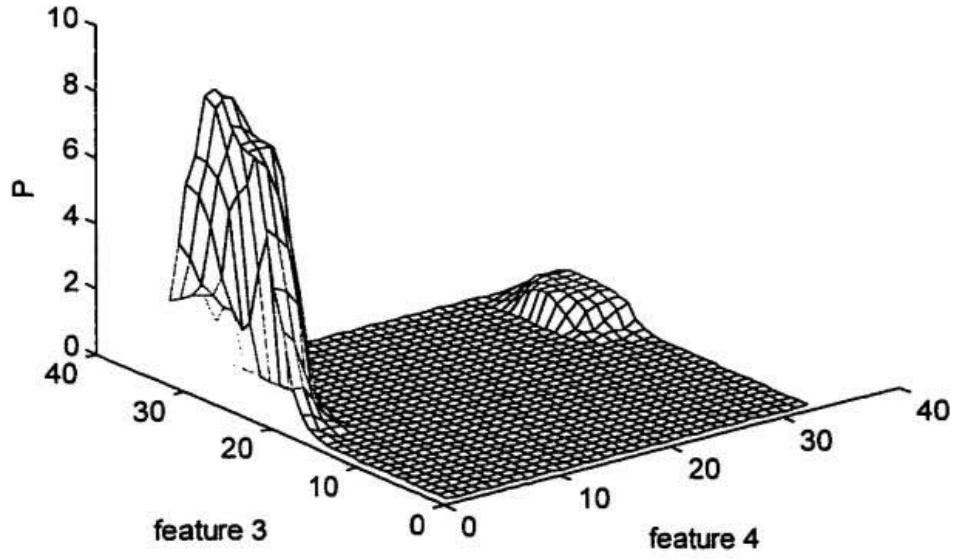


Figure A24 - pdf estimate; gesture 4 (features 3 & 4)

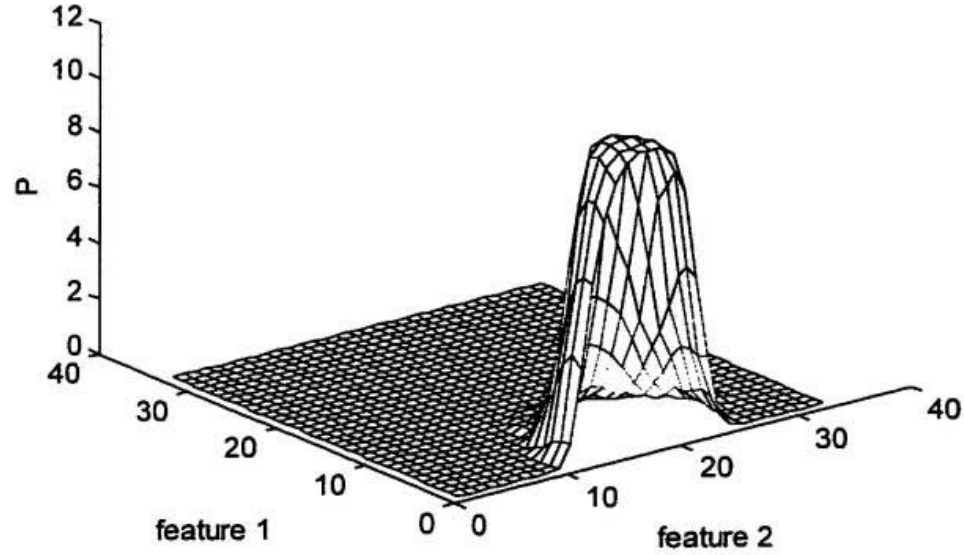


Figure A25 - pdf estimate; gesture 5 (features 1 & 2)

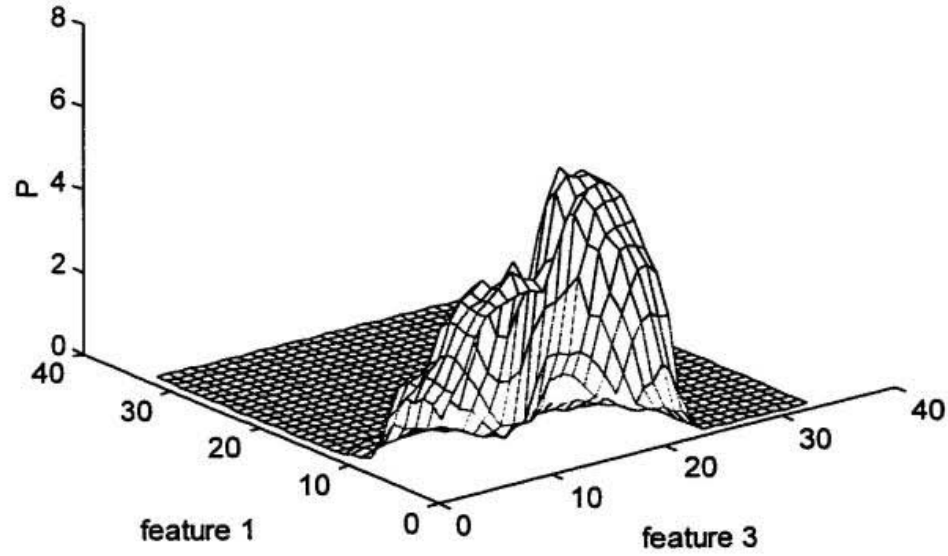


Figure A26 - pdf estimate; gesture 5 (features 1 & 3)

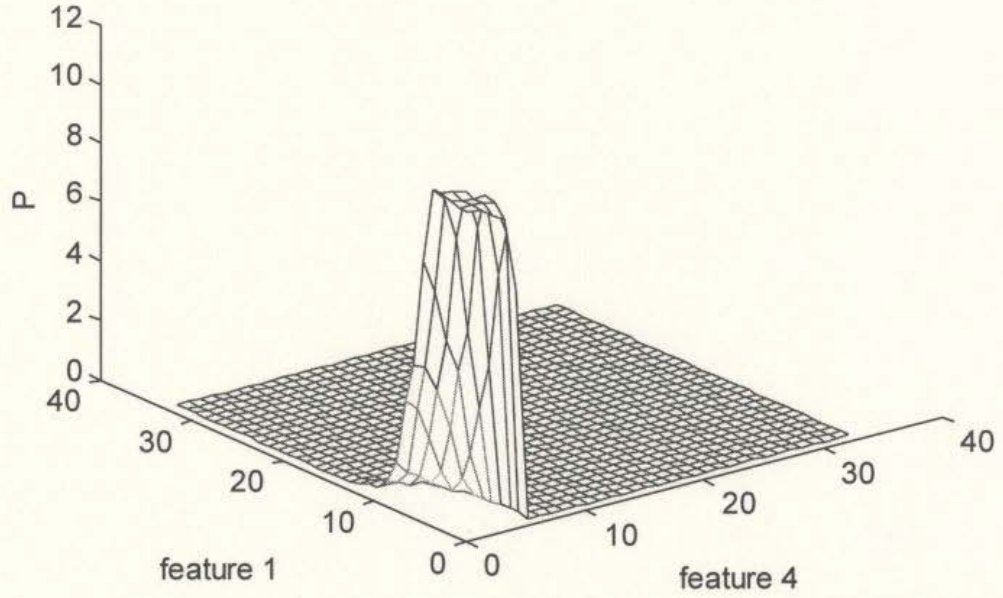


Figure A27 - pdf estimate; gesture 5 (features 1 & 4)

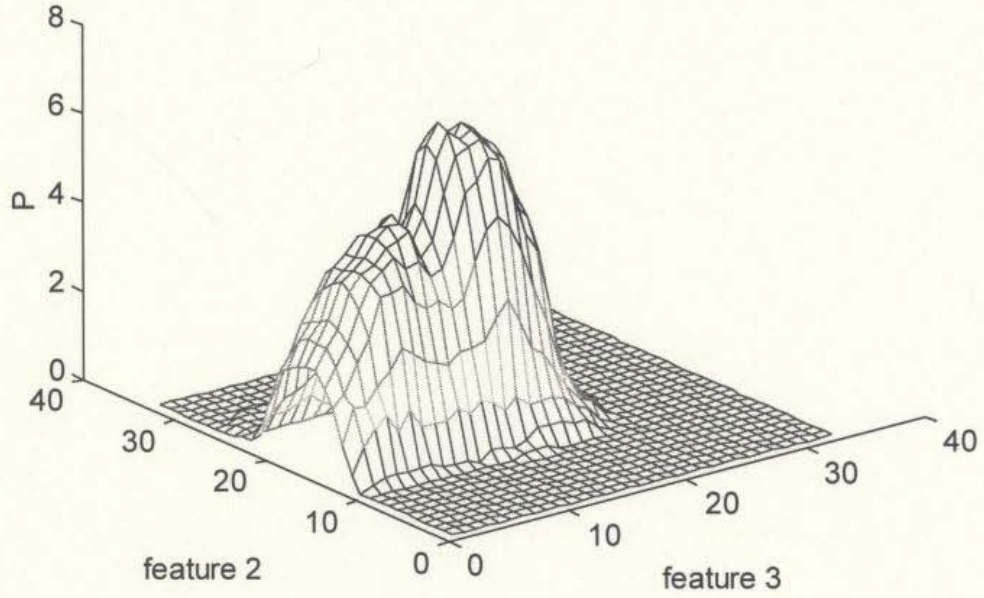


Figure A28 - pdf estimate; gesture 5 (features 2 & 3)

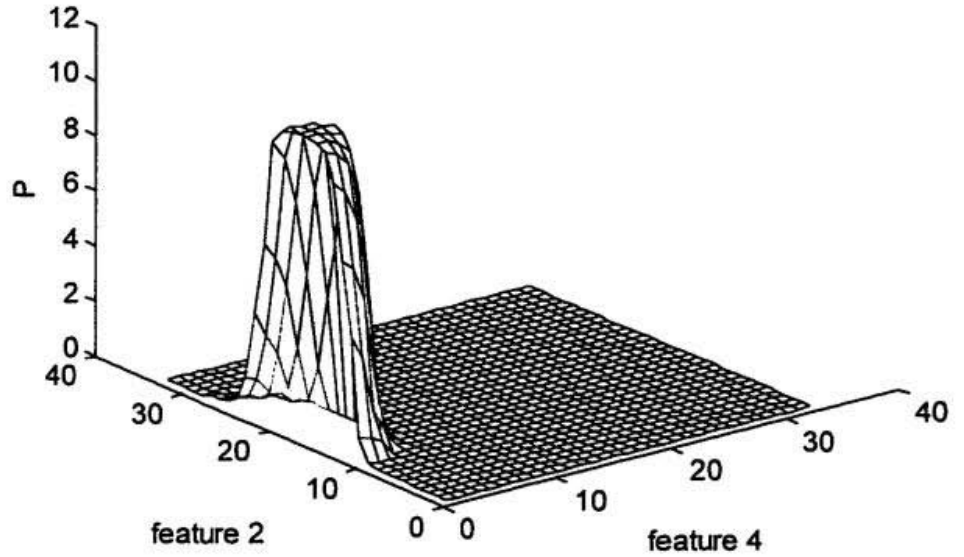


Figure A29 - pdf estimate; gesture 5 (features 2 & 4)

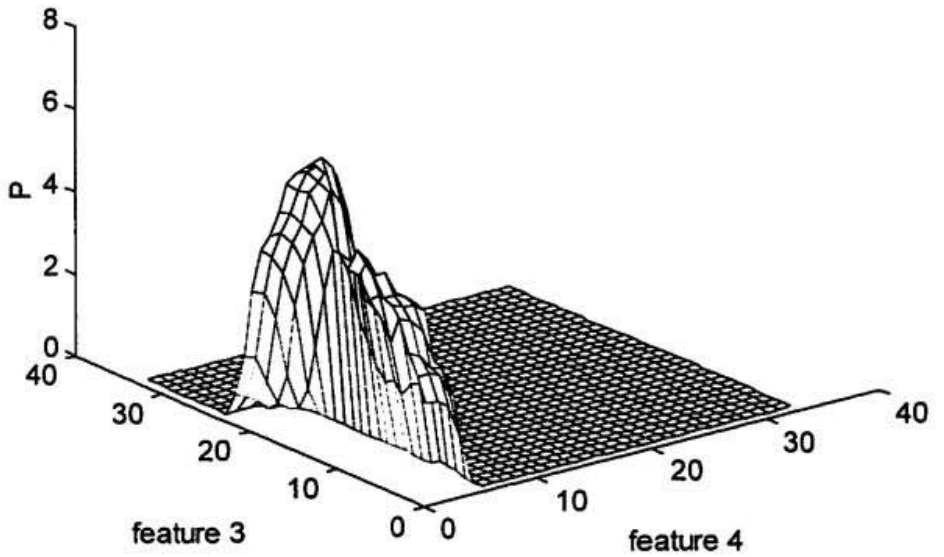


Figure A30 - pdf estimate; gesture 5 (features 3 & 4)

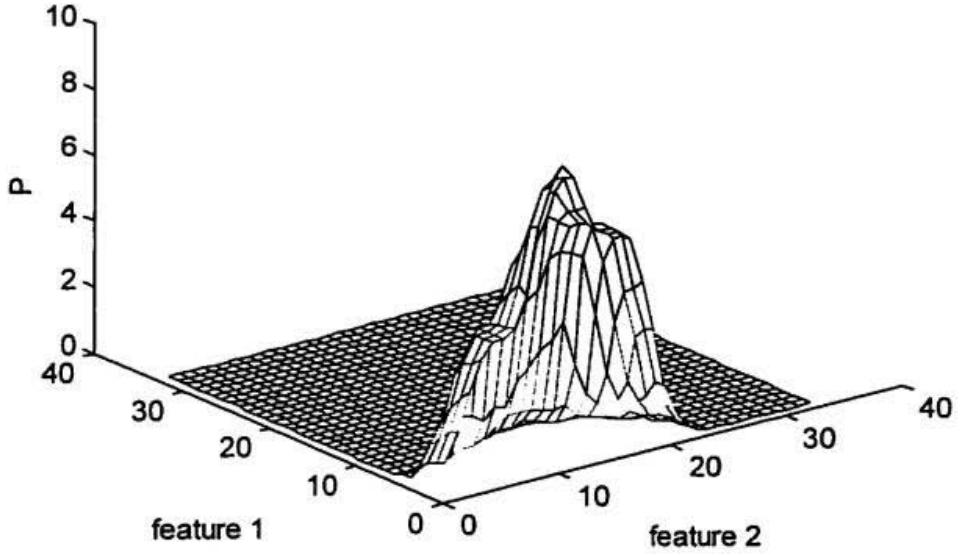


Figure A31 - pdf estimate; gesture 6 (features 1 & 2)

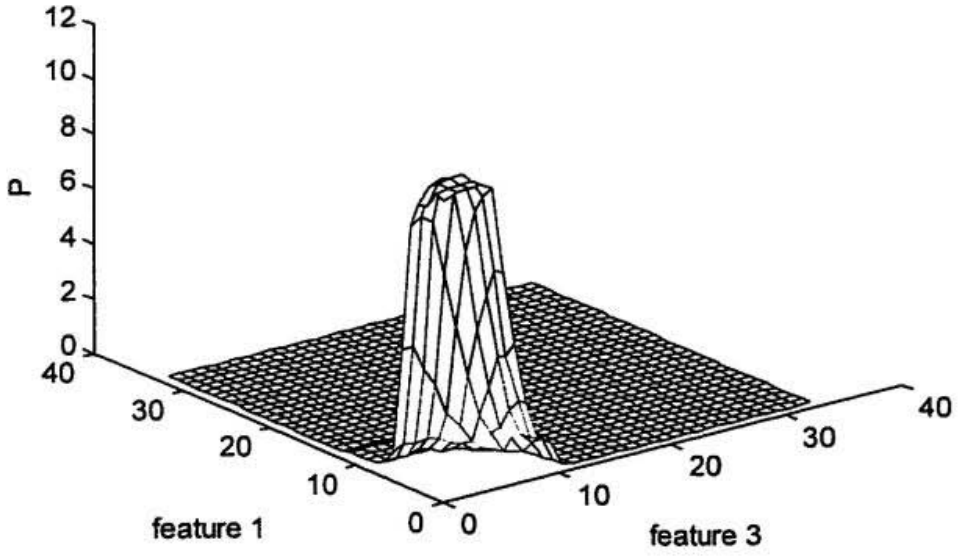


Figure A32 - pdf estimate; gesture 6 (features 1 & 3)

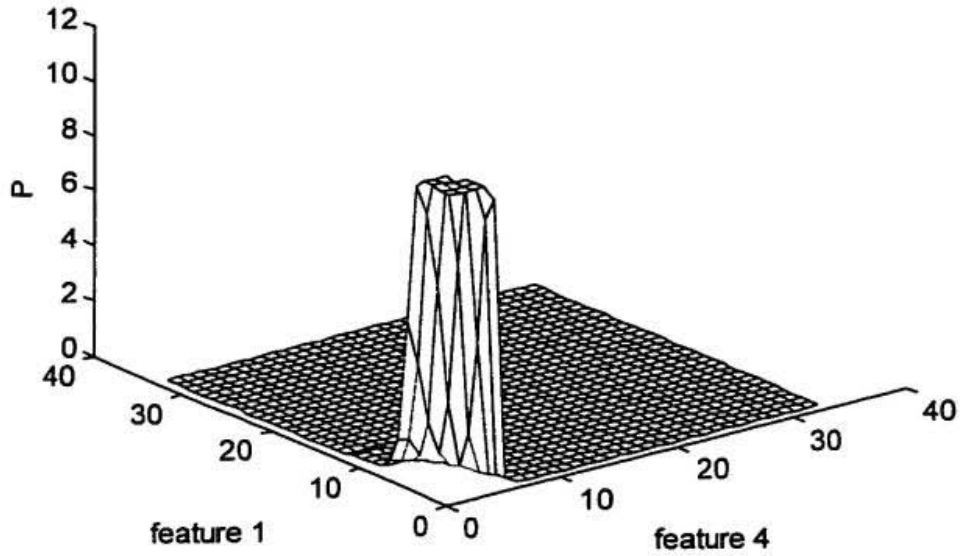


Figure A33 - pdf estimate; gesture 6 (features 1 & 4)

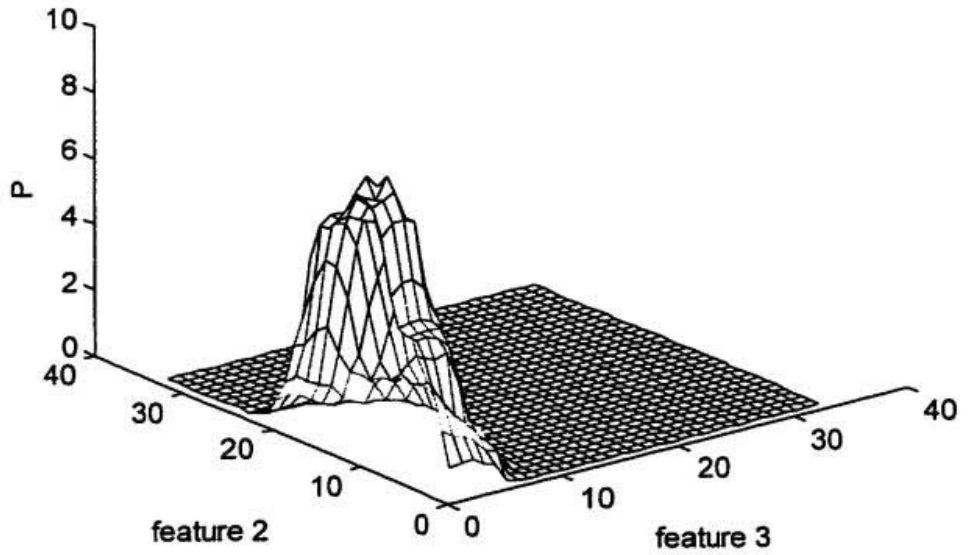


Figure A34 - pdf estimate; gesture 6 (features 2 & 3)

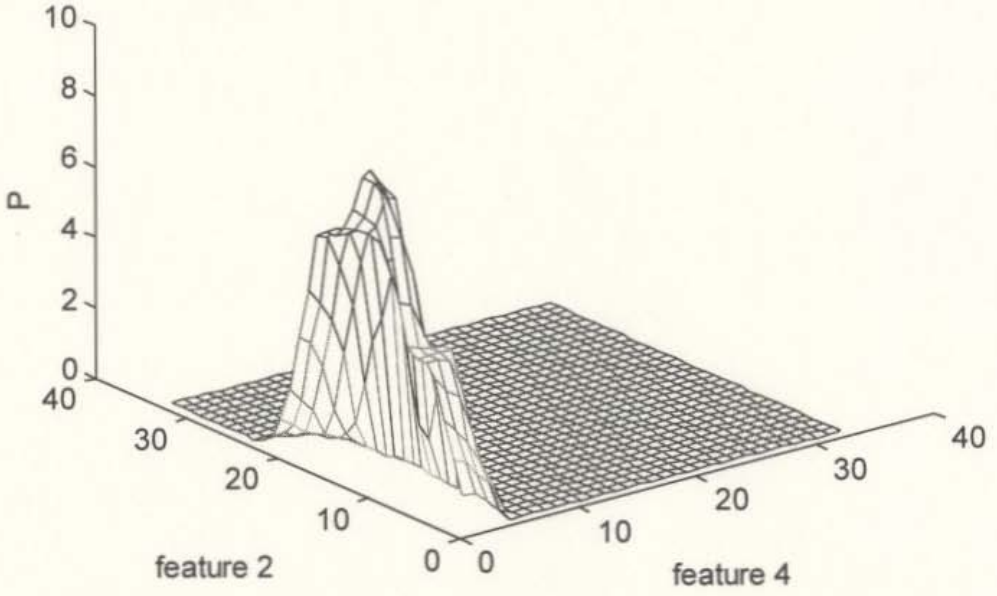


Figure A35 - pdf estimate; gesture 6 (features 2 &4)

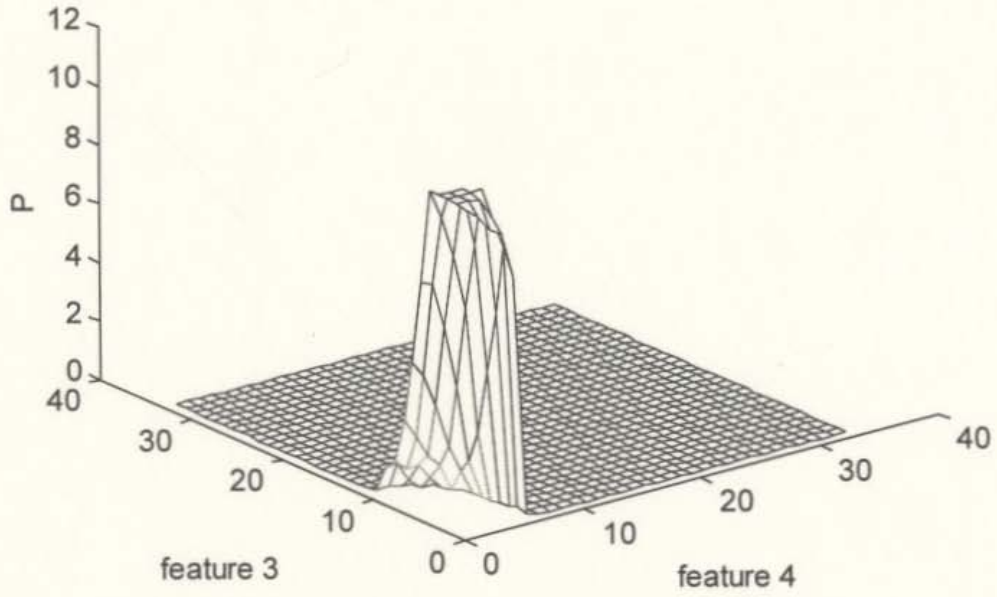


Figure A36 - pdf estimate; gesture 6 (features 3 & 4)

APPENDIX B
MATLAB SCRIPT FILES

RULE.M

```
function [P,location]=rule(fvectors)
% Rule generation and rule based recognition; feature vectors for a number
% of training samples will be used to produce a number of rules which
% will be used for recognition
% R. Hale - March 1997

% determine the size of the input vector; this will determine the size
% of the matrix of possible window centers
%[ordered index]=sort(fvectors);
[n m]=size(fvectors);

% Initially set the height of the window
h=.3;

% Increment to step ac at each interval
step=10;

% The volume of the hypercube V_n
V_n=h^m;

% build ac vector step/h long and m columns
for i=1:step/h
   for j=1:m
      ac(i,j)=h*i;
   end
end
ac=ac/step;

% build vector with all possible combinations for ac;
% begin with a general matrix which will be the proper size
% and have the first column with correct values
%
% the matrix ac_comb will contain all possible combinations for the
% ac vectors based on the number of dimensions (m), the window height (h)
% and the step size (step)

for i=1:fix(step/h)^(m-1)
   ac_comb=[ac_comb;ac];
end

% take the second column and sort the values to produce the
```

```
% desired sequence
if m>1
   for i=1:fix(step/h)^2:fix(step/h)^m
      ac_comb(i:i+fix(step/h)^2-1,2)=sort(ac_comb(i:i+fix(step/h)^2-1,2));
   end
end

% remaining 3 to m columns can be sorted
if m>2
   for j=3:m
      for i=1:fix(step/h)^j:fix(step/h)^m
         ac_comb(i:i+fix(step/h)^j-1,j)=sort(ac_comb(i:i+fix(step/h)^j-1,j));
      end
   end
end


% Determine the function phi for all elements of ac_comb compared to all
% vectors in input vector

[rows cols]=size(ac_comb);

for i=1:rows
   % loop through for all vectors in input vector
   for k=1:n
      phi(k)=norm((ac_comb(i,:)-fvectors(k,:))/h);
   end

   less=find(phi<=.5);
   p_ac(i)=1/(n*V_n)*length(less);
end

% put probabilities in matrix where each column represents each range
% movements of the window
count=1;
for i=1:fix(step/h):fix(step/h)^m
   P(:,count)=p_ac(i:(i+fix(step/h)-1))';
   count=count+1;
end

% determine the appropriate rules
location=threspdf(P);
% plot the regions of interest if dimensionally possibly
```

```
%figure;
%axis([0 step/h 0 step/h])
%hold;
%plot(I,J,'k-')
```

THRESPDF.M

```
function out=threspdf(pdf)
[rows cols]=size(pdf);
threshold=max(max(pdf))*0.3;
for i=1:rows
  for j=1:rows
    if pdf(i,j)>=threshold
      out(i,j)=1;
    else
      out(i,j)=0;
    end
  end
end
```

GAUSS1D.M

```
% create a vector R which is normally distributed
R=normrnd(1.2,.3,200,1);
R=abs(R/max(R));

% Show the didtribution of R is normal
%figure(1)
%normplot(R)

% Run the rule generation script

[P,L]=rule(R);
figure(2)
plot(P);
```

GAUSS2D.M

```
% create a vector R which is normally distributed
R=normrnd(1.5,.4,100,2);
B=normrnd(3.5,.5,200,2);
R=[R;B];
```

```
R=R/max(max(R));

% Run the rule generation script

[P,L]=rule(R);

% plot the surface
%figure
%mesh(P)
```

TREMOR.M

```
function seg=tremor(gesture)
echo off;
% Script which segments a 6 DOF data stream from The Flock
% of Birds using tremor filter (FIFO buffer)
% Rodney Hale July 1996
% Segmentation of data file gesture.dat stored in sgement matrix
% Runs dist.m which extracts overall distance feature & stores in matrix
% called distance


tic          % set the timer

N = 15;        % (length of buffer)

% determine optimum threshold levels from plot of the threshold, T, value
upthres=4;    % (upper threshold value)
lowthres=0.3;    % (lower threshold value)
countm=0;

% load the file of gestures
%load gesture.dat;
[rows columns]=size(gesture);

% calculate variance of buffer for 3 DOF; x, y & z position

for j=1:rows-N
   for i=1:columns-4 % use x y & z positional data
      buffer=gesture(j:j+N,i+1);
      Tj(i)=(var(buffer))^2;
   end
   T(j)=(Tj(1)+Tj(2)+Tj(3))^0.5; %+Tj(4)+Tj(5)+Tj(6))^0.5; % overall tremor
```

```
    if T(j) < lowthres
        status(j)=1;    % 1 = still
    elseif T(j) > upthres
        status(j)=2;    % 2 = movement
    end
end

% determine the state, still state when status=1 and until status =2;
% movement state when status=2 and until status =1

move=find(status==2);
still=find(status==1);
inter=find(status==0);

% using the difference between successive status elements, the beginning
% and the end points of the gestures can be determined; 2 (or 1) in the
% difference matrix indicates the beginning of a gesture, -1 (or 1) indicates
% the end of a gesture

begin_end=diff(status);
Begins=find(begin_end==2);

seg=[];

for i=1:length(status)
    if status(i)==1
        tmp=find(status(i:length(status))==2);
        if length(tmp)~=0
            diff1=diff(status(i:tmp(1)+i-1));
            if diff1(1)==-1 & diff1(length(diff1))==2
                seg=[seg; i];
            end
        end
    end
end

% save the segmented gestures
%save features.dat distance -ascii
toc   % elapsed time
```

SEG_FEAT.M

```
function features=seg_feat(file,segdata)
% This function calls the feature extraction function feat.m to return
% a feature set for a file, file, containing unsegmented gestures using the
% segmenting data, segdata, from the tremor.m function

for i=1:length(segdata)-1
   features(i,:)=feat(file(segdata(i):segdata(i+1),:));
end
```

FEAT.M

```
function f=feat(file)
   [n m]=size(file);
   % find maximum position values for each direction
   maximum=max(file);
%   f1=maximum(2); % high corr.
%   f2=maximum(3); % high corr.
   f3=maximum(4);
   % find minimum position values for each direction
   minimum=min(file);
   f4=minimum(2);
   f5=minimum(3);
%   f6=minimum(4); % high corr.
   % find maximum difference of consecutive orientations for each orientation
   maxdiff=max(diff(file));
%   f7=maxdiff(5); % removed July 28; low inter var.
%   f8=maxdiff(6); % high corr.
   f9=maxdiff(7);
   % use distance from point to line as indication of curvature
   count=0;
   for i=1:n-2
      a=file(i+2,2:4)-file(i,2:4);
      b=file(i+1,2:4)-file(i,2:4);
      if norm(a)~=0
         count=count+1;
         d(count,1)=norm(cross(a,b))/norm(a);
      end
   end
%   f10=mean(d); % removed July July 28; low inter var.
   % pass back matrix containing all feature vectors
   f=[f3 f4 f5 f9];
```

```
f=abs(f);
```

PDFPLOT.M

```
figure(1)
[P1_2,L1_2]=rule([n_feats1(1:100,1) n_feats2(1:100,2)]);
mesh(P1_2)
ylabel('feature 1')
xlabel('feature 2')
zlabel('P')
figure(2)
[P1_3,L1_3]=rule([n_feats1(1:100,1) n_feats2(1:100,3)]);
mesh(P1_3)
ylabel('feature 1')
xlabel('feature 3')
zlabel('P')
figure(3)
[P1_4,L1_4]=rule([n_feats1(1:100,1) n_feats2(1:100,4)]);
mesh(P1_4)
ylabel('feature 1')
xlabel('feature 4')
zlabel('P')
figure(4)
[P2_3,L2_3]=rule([n_feats1(1:100,2) n_feats2(1:100,3)]);
mesh(P2_3)
ylabel('feature 2')
xlabel('feature 3')
zlabel('P')
figure(5)
[P2_4,L2_4]=rule([n_feats1(1:100,2) n_feats2(1:100,4)]);
mesh(P2_4)
ylabel('feature 2')
xlabel('feature 4')
zlabel('P')
figure(6)
[P3_4,L3_4]=rule([n_feats1(1:100,3) n_feats2(1:100,4)]);
mesh(P3_4)
ylabel('feature 3')
xlabel('feature 4')
zlabel('P')
```

PASSRULE.M

```
function check=passrule(f,L)
f=round(f.*32)+1;
if L(f(1),f(2))==1
  check=1;
else
  check=0;
end
```

RULECLSS.M

```
function C=ruleclass(f1,f2,f3,f4,f5,f6);

% create all pdf estimates and rules
[P1_1_2,L1_1_2]=rule([f1(:,1) f1(:,2)]);
[P1_1_3,L1_1_3]=rule([f1(:,1) f1(:,3)]);
[P1_1_4,L1_1_4]=rule([f1(:,1) f1(:,4)]);
[P1_2_3,L1_2_3]=rule([f1(:,2) f1(:,3)]);
[P1_2_4,L1_2_4]=rule([f1(:,2) f1(:,4)]);
[P1_3_4,L1_3_4]=rule([f1(:,3) f1(:,4)]);

[P2_1_2,L2_1_2]=rule([f2(:,1) f2(:,2)]);
[P2_1_3,L2_1_3]=rule([f2(:,1) f2(:,3)]);
[P2_1_4,L2_1_4]=rule([f2(:,1) f2(:,4)]);
[P2_2_3,L2_2_3]=rule([f2(:,2) f2(:,3)]);
[P2_2_4,L2_2_4]=rule([f2(:,2) f2(:,4)]);
[P2_3_4,L2_3_4]=rule([f2(:,3) f2(:,4)]);

[P3_1_2,L3_1_2]=rule([f3(:,1) f3(:,2)]);
[P3_1_3,L3_1_3]=rule([f3(:,1) f3(:,3)]);
[P3_1_4,L3_1_4]=rule([f3(:,1) f3(:,4)]);
[P3_2_3,L3_2_3]=rule([f3(:,2) f3(:,3)]);
[P3_2_4,L3_2_4]=rule([f3(:,2) f3(:,4)]);
[P3_3_4,L3_3_4]=rule([f3(:,3) f3(:,4)]);

[P4_1_2,L4_1_2]=rule([f4(:,1) f4(:,2)]);
[P4_1_3,L4_1_3]=rule([f4(:,1) f4(:,3)]);
[P4_1_4,L4_1_4]=rule([f4(:,1) f4(:,4)]);
[P4_2_3,L4_2_3]=rule([f4(:,2) f4(:,3)]);
[P4_2_4,L4_2_4]=rule([f4(:,2) f4(:,4)]);
[P4_3_4,L4_3_4]=rule([f4(:,3) f4(:,4)]);
```

```
[P5_1_2,L5_1_2]=rule([f5(:,1) f5(:,2)]);
[P5_1_3,L5_1_3]=rule([f5(:,1) f5(:,3)]);
[P5_1_4,L5_1_4]=rule([f5(:,1) f5(:,4)]);
[P5_2_3,L5_2_3]=rule([f5(:,2) f5(:,3)]);
[P5_2_4,L5_2_4]=rule([f5(:,2) f5(:,4)]);
[P5_3_4,L5_3_4]=rule([f5(:,3) f5(:,4)]);

[P6_1_2,L6_1_2]=rule([f6(:,1) f6(:,2)]);
[P6_1_3,L6_1_3]=rule([f6(:,1) f6(:,3)]);
[P6_1_4,L6_1_4]=rule([f6(:,1) f6(:,4)]);
[P6_2_3,L6_2_3]=rule([f6(:,2) f6(:,3)]);
[P6_2_4,L6_2_4]=rule([f6(:,2) f6(:,4)]);
[P6_3_4,L6_3_4]=rule([f6(:,3) f6(:,4)]);

C=zeros(6);

% class 1

for i=1:100
  s1_1=passrule([f1(i,1) f1(i,2)],L1_1_2);
  s1_2=passrule([f1(i,1) f1(i,3)],L1_1_3);
  s1_3=passrule([f1(i,1) f1(i,4)],L1_1_4);
  s1_4=passrule([f1(i,2) f1(i,3)],L1_2_3);
  s1_5=passrule([f1(i,2) f1(i,4)],L1_2_4);
  s1_6=passrule([f1(i,3) f1(i,4)],L1_3_4);
%  s1_4=0; s1_5=0;
  S1=s1_1+s1_2+s1_3+s1_4+s1_5+s1_6;

  s2_1=passrule([f1(i,1) f1(i,2)],L2_1_2);
  s2_2=passrule([f1(i,1) f1(i,3)],L2_1_3);
  s2_3=passrule([f1(i,1) f1(i,4)],L2_1_4);
  s2_4=passrule([f1(i,2) f1(i,3)],L2_2_3);
  s2_5=passrule([f1(i,2) f1(i,4)],L2_2_4);
  s2_6=passrule([f1(i,3) f1(i,4)],L2_3_4);
%  s2_4=0; s2_5=0;
  S2=s2_1+s2_2+s2_3+s2_4+s2_5+s2_6;

  s3_1=passrule([f1(i,1) f1(i,2)],L3_1_2);
  s3_2=passrule([f1(i,1) f1(i,3)],L3_1_3);
  s3_3=passrule([f1(i,1) f1(i,4)],L3_1_4);
  s3_4=passrule([f1(i,2) f1(i,3)],L3_2_3);
  s3_5=passrule([f1(i,2) f1(i,4)],L3_2_4);
  s3_6=passrule([f1(i,3) f1(i,4)],L3_3_4);
```

```
%  s3_4=0; s3_5=0;
  S3=s3_1+s3_2+s3_3+s3_4+s3_5+s3_6;

  s4_1=passrule([f1(i,1) f1(i,2)],L4_1_2);
  s4_2=passrule([f1(i,1) f1(i,3)],L4_1_3);
  s4_3=passrule([f1(i,1) f1(i,4)],L4_1_4);
  s4_4=passrule([f1(i,2) f1(i,3)],L4_2_3);
  s4_5=passrule([f1(i,2) f1(i,4)],L4_2_4);
  s4_6=passrule([f1(i,3) f1(i,4)],L4_3_4);
%  s4_4=0; s4_5=0;
  S4=s4_1+s4_2+s4_3+s4_4+s4_5+s4_6;

  s5_1=passrule([f1(i,1) f1(i,2)],L5_1_2);
  s5_2=passrule([f1(i,1) f1(i,3)],L5_1_3);
  s5_3=passrule([f1(i,1) f1(i,4)],L5_1_4);
  s5_4=passrule([f1(i,2) f1(i,3)],L5_2_3);
  s5_5=passrule([f1(i,2) f1(i,4)],L5_2_4);
  s5_6=passrule([f1(i,3) f1(i,4)],L5_3_4);
%  s5_4=0; s5_5=0;
  S5=s5_1+s5_2+s5_3+s5_4+s5_5+s5_6;

  s6_1=passrule([f1(i,1) f1(i,2)],L6_1_2);
  s6_2=passrule([f1(i,1) f1(i,3)],L6_1_3);
  s6_3=passrule([f1(i,1) f1(i,4)],L6_1_4);
  s6_4=passrule([f1(i,2) f1(i,3)],L6_2_3);
  s6_5=passrule([f1(i,2) f1(i,4)],L6_2_4);
  s6_6=passrule([f1(i,3) f1(i,4)],L6_3_4);
%  s6_4=0; s6_5=0;
  S6=s6_1+s6_2+s6_3+s6_4+s6_5+s6_6;

  S=[S1;S2;S3;S4;S5;S6];
  [Y I]=sort(S);
  if I(6)==1
    C(1,1)=C(1,1)+1;
  elseif I(6)==2
    C(1,2)=C(1,2)+1;
  elseif I(6)==3
    C(1,3)=C(1,3)+1;
  elseif I(6)==4
    C(1,4)=C(1,4)+1;
  elseif I(6)==5
    C(1,5)=C(1,5)+1;
  elseif I(6)==6
```

```
    C(1,6)=C(1,6)+1;
  end
end

% class 2

for i=1:100
  s1_1=passrule([f2(i,1) f2(i,2)],L1_1_2);
  s1_2=passrule([f2(i,1) f2(i,3)],L1_1_3);
  s1_3=passrule([f2(i,1) f2(i,4)],L1_1_4);
  s1_4=passrule([f2(i,2) f2(i,3)],L1_2_3);
  s1_5=passrule([f2(i,2) f2(i,4)],L1_2_4);
  s1_6=passrule([f2(i,3) f2(i,4)],L1_3_4);
%  s1_4=0; s1_5=0;
  S1=s1_1+s1_2+s1_3+s1_4+s1_5+s1_6;

  s2_1=passrule([f2(i,1) f2(i,2)],L2_1_2);
  s2_2=passrule([f2(i,1) f2(i,3)],L2_1_3);
  s2_3=passrule([f2(i,1) f2(i,4)],L2_1_4);
  s2_4=passrule([f2(i,2) f2(i,3)],L2_2_3);
  s2_5=passrule([f2(i,2) f2(i,4)],L2_2_4);
  s2_6=passrule([f2(i,3) f2(i,4)],L2_3_4);
%  s2_4=0; s2_5=0;
  S2=s2_1+s2_2+s2_3+s2_4+s2_5+s2_6;

  s3_1=passrule([f2(i,1) f2(i,2)],L3_1_2);
  s3_2=passrule([f2(i,1) f2(i,3)],L3_1_3);
  s3_3=passrule([f2(i,1) f2(i,4)],L3_1_4);
  s3_4=passrule([f2(i,2) f2(i,3)],L3_2_3);
  s3_5=passrule([f2(i,2) f2(i,4)],L3_2_4);
  s3_6=passrule([f2(i,3) f2(i,4)],L3_3_4);
%  s3_4=0; s3_5=0;
  S3=s3_1+s3_2+s3_3+s3_4+s3_5+s3_6;

  s4_1=passrule([f2(i,1) f2(i,2)],L4_1_2);
  s4_2=passrule([f2(i,1) f2(i,3)],L4_1_3);
  s4_3=passrule([f2(i,1) f2(i,4)],L4_1_4);
  s4_4=passrule([f2(i,2) f2(i,3)],L4_2_3);
  s4_5=passrule([f2(i,2) f2(i,4)],L4_2_4);
  s4_6=passrule([f2(i,3) f2(i,4)],L4_3_4);
%  s4_4=0; s4_5=0;
  S4=s4_1+s4_2+s4_3+s4_4+s4_5+s4_6;
```

```
  s5_1=passrule([f2(i,1) f2(i,2)],L5_1_2);
  s5_2=passrule([f2(i,1) f2(i,3)],L5_1_3);
  s5_3=passrule([f2(i,1) f2(i,4)],L5_1_4);
  s5_4=passrule([f2(i,2) f2(i,3)],L5_2_3);
  s5_5=passrule([f2(i,2) f2(i,4)],L5_2_4);
  s5_6=passrule([f2(i,3) f2(i,4)],L5_3_4);
%  s5_4=0; s5_5=0;
  S5=s5_1+s5_2+s5_3+s5_4+s5_5+s5_6;

  s6_1=passrule([f2(i,1) f2(i,2)],L6_1_2);
  s6_2=passrule([f2(i,1) f2(i,3)],L6_1_3);
  s6_3=passrule([f2(i,1) f2(i,4)],L6_1_4);
  s6_4=passrule([f2(i,2) f2(i,3)],L6_2_3);
  s6_5=passrule([f2(i,2) f2(i,4)],L6_2_4);
  s6_6=passrule([f2(i,3) f2(i,4)],L6_3_4);
%  s6_4=0; s6_5=0;
  S6=s6_1+s6_2+s6_3+s6_4+s6_5+s6_6;

  S=[S1;S2;S3;S4;S5;S6];
  [Y I]=sort(S);
  if I(6)==1
    C(2,1)=C(2,1)+1;
  elseif I(6)==2
    C(2,2)=C(2,2)+1;
  elseif I(6)==3
    C(2,3)=C(2,3)+1;
  elseif I(6)==4
    C(2,4)=C(2,4)+1;
  elseif I(6)==5
    C(2,5)=C(2,5)+1;
  elseif I(6)==6
    C(2,6)=C(2,6)+1;
  end
end

% class 3

for i=1:100
  s1_1=passrule([f3(i,1) f3(i,2)],L1_1_2);
  s1_2=passrule([f3(i,1) f3(i,3)],L1_1_3);
  s1_3=passrule([f3(i,1) f3(i,4)],L1_1_4);
  s1_4=passrule([f3(i,2) f3(i,3)],L1_2_3);
  s1_5=passrule([f3(i,2) f3(i,4)],L1_2_4);
```

```
  s1_6=passrule([f3(i,3) f3(i,4)],L1_3_4);
%  s1_4=0; s1_5=0;
  S1=s1_1+s1_2+s1_3+s1_4+s1_5+s1_6;

  s2_1=passrule([f3(i,1) f3(i,2)],L2_1_2);
  s2_2=passrule([f3(i,1) f3(i,3)],L2_1_3);
  s2_3=passrule([f3(i,1) f3(i,4)],L2_1_4);
  s2_4=passrule([f3(i,2) f3(i,3)],L2_2_3);
  s2_5=passrule([f3(i,2) f3(i,4)],L2_2_4);
  s2_6=passrule([f3(i,3) f3(i,4)],L2_3_4);
%  s2_4=0; s2_5=0;
  S2=s2_1+s2_2+s2_3+s2_4+s2_5+s2_6;

  s3_1=passrule([f3(i,1) f3(i,2)],L3_1_2);
  s3_2=passrule([f3(i,1) f3(i,3)],L3_1_3);
  s3_3=passrule([f3(i,1) f3(i,4)],L3_1_4);
  s3_4=passrule([f3(i,2) f3(i,3)],L3_2_3);
  s3_5=passrule([f3(i,2) f3(i,4)],L3_2_4);
  s3_6=passrule([f3(i,3) f3(i,4)],L3_3_4);
%  s3_4=0; s3_5=0;
  S3=s3_1+s3_2+s3_3+s3_4+s3_5+s3_6;

  s4_1=passrule([f3(i,1) f3(i,2)],L4_1_2);
  s4_2=passrule([f3(i,1) f3(i,3)],L4_1_3);
  s4_3=passrule([f3(i,1) f3(i,4)],L4_1_4);
  s4_4=passrule([f3(i,2) f3(i,3)],L4_2_3);
  s4_5=passrule([f3(i,2) f3(i,4)],L4_2_4);
  s4_6=passrule([f3(i,3) f4(i,4)],L4_3_4);
%  s4_4=0; s4_5=0;
  S4=s4_1+s4_2+s4_3+s4_4+s4_5+s4_6;

  s5_1=passrule([f3(i,1) f3(i,2)],L5_1_2);
  s5_2=passrule([f3(i,1) f3(i,3)],L5_1_3);
  s5_3=passrule([f3(i,1) f3(i,4)],L5_1_4);
  s5_4=passrule([f3(i,2) f3(i,3)],L5_2_3);
  s5_5=passrule([f3(i,2) f3(i,4)],L5_2_4);
  s5_6=passrule([f3(i,3) f3(i,4)],L5_3_4);
%  s5_4=0; s5_5=0;
  S5=s5_1+s5_2+s5_3+s5_4+s5_5+s5_6;

  s6_1=passrule([f3(i,1) f3(i,2)],L6_1_2);
  s6_2=passrule([f3(i,1) f3(i,3)],L6_1_3);
  s6_3=passrule([f3(i,1) f3(i,4)],L6_1_4);
```

```
  s6_4=passrule([f3(i,2) f3(i,3)],L6_2_3);
  s6_5=passrule([f3(i,2) f3(i,4)],L6_2_4);
  s6_6=passrule([f3(i,3) f3(i,4)],L6_3_4);
%  s6_4=0; s6_5=0;
  S6=s6_1+s6_2+s6_3+s6_4+s6_5+s6_6;

  S=[S1;S2;S3;S4;S5;S6];
  [Y I]=sort(S);
  if I(6)==1
    C(3,1)=C(3,1)+1;
  elseif I(6)==2
    C(3,2)=C(3,2)+1;
  elseif I(6)==3
    C(3,3)=C(3,3)+1;
  elseif I(6)==4
    C(3,4)=C(3,4)+1;
  elseif I(6)==5
    C(3,5)=C(3,5)+1;
  elseif I(6)==6
    C(3,6)=C(3,6)+1;
  end
end

% class 4

for i=1:100
  s1_1=passrule([f4(i,1) f4(i,2)],L1_1_2);
  s1_2=passrule([f4(i,1) f4(i,3)],L1_1_3);
  s1_3=passrule([f4(i,1) f4(i,4)],L1_1_4);
  s1_4=passrule([f4(i,2) f4(i,3)],L1_2_3);
  s1_5=passrule([f4(i,2) f4(i,4)],L1_2_4);
  s1_6=passrule([f4(i,3) f4(i,4)],L1_3_4);
%  s1_4=0; s1_5=0;
  S1=s1_1+s1_2+s1_3+s1_4+s1_5+s1_6;

  s2_1=passrule([f4(i,1) f4(i,2)],L2_1_2);
  s2_2=passrule([f4(i,1) f4(i,3)],L2_1_3);
  s2_3=passrule([f4(i,1) f4(i,4)],L2_1_4);
  s2_4=passrule([f4(i,2) f4(i,3)],L2_2_3);
  s2_5=passrule([f4(i,2) f4(i,4)],L2_2_4);
  s2_6=passrule([f4(i,3) f4(i,4)],L2_3_4);
%  s2_4=0; s2_5=0;
  S2=s2_1+s2_2+s2_3+s2_4+s2_5+s2_6;
```

```
  s3_1=passrule([f4(i,1) f4(i,2)],L3_1_2);
  s3_2=passrule([f4(i,1) f4(i,3)],L3_1_3);
  s3_3=passrule([f4(i,1) f4(i,4)],L3_1_4);
  s3_4=passrule([f4(i,2) f4(i,3)],L3_2_3);
  s3_5=passrule([f4(i,2) f4(i,4)],L3_2_4);
  s3_6=passrule([f4(i,3) f4(i,4)],L3_3_4);
%  s3_4=0; s3_5=0;
  S3=s3_1+s3_2+s3_3+s3_4+s3_5+s3_6;

  s4_1=passrule([f4(i,1) f4(i,2)],L4_1_2);
  s4_2=passrule([f4(i,1) f4(i,3)],L4_1_3);
  s4_3=passrule([f4(i,1) f4(i,4)],L4_1_4);
  s4_4=passrule([f4(i,2) f4(i,3)],L4_2_3);
  s4_5=passrule([f4(i,2) f4(i,4)],L4_2_4);
  s4_6=passrule([f4(i,3) f4(i,4)],L4_3_4);
%  s4_4=0; s4_5=0;
  S4=s4_1+s4_2+s4_3+s4_4+s4_5+s4_6;

  s5_1=passrule([f4(i,1) f4(i,2)],L5_1_2);
  s5_2=passrule([f4(i,1) f4(i,3)],L5_1_3);
  s5_3=passrule([f4(i,1) f4(i,4)],L5_1_4);
  s5_4=passrule([f4(i,2) f4(i,3)],L5_2_3);
  s5_5=passrule([f4(i,2) f4(i,4)],L5_2_4);
  s5_6=passrule([f4(i,3) f4(i,4)],L5_3_4);
%  s5_4=0; s5_5=0;
  S5=s5_1+s5_2+s5_3+s5_4+s5_5+s5_6;

  s6_1=passrule([f4(i,1) f4(i,2)],L6_1_2);
  s6_2=passrule([f4(i,1) f4(i,3)],L6_1_3);
  s6_3=passrule([f4(i,1) f4(i,4)],L6_1_4);
  s6_4=passrule([f4(i,2) f4(i,3)],L6_2_3);
  s6_5=passrule([f4(i,2) f4(i,4)],L6_2_4);
  s6_6=passrule([f4(i,3) f4(i,4)],L6_3_4);
%  s6_4=0; s6_5=0;
  S6=s6_1+s6_2+s6_3+s6_4+s6_5+s6_6;

  S=[S1;S2;S3;S4;S5;S6];
  [Y I]=sort(S);
  if I(6)==1
    C(4,1)=C(4,1)+1;
  elseif I(6)==2
    C(4,2)=C(4,2)+1;
```

```
  elseif I(6)==3
    C(4,3)=C(4,3)+1;
  elseif I(6)==4
    C(4,4)=C(4,4)+1;
  elseif I(6)==5
    C(4,5)=C(4,5)+1;
  elseif I(6)==6
    C(4,6)=C(4,6)+1;
  end
end

% class 5

for i=1:100
  s1_1=passrule([f5(i,1) f5(i,2)],L1_1_2);
  s1_2=passrule([f5(i,1) f5(i,3)],L1_1_3);
  s1_3=passrule([f5(i,1) f5(i,4)],L1_1_4);
  s1_4=passrule([f5(i,2) f5(i,3)],L1_2_3);
  s1_5=passrule([f5(i,2) f5(i,4)],L1_2_4);
  s1_6=passrule([f5(i,3) f5(i,4)],L1_3_4);
%  s1_4=0; s1_5=0;
  S1=s1_1+s1_2+s1_3+s1_4+s1_5+s1_6;

  s2_1=passrule([f5(i,1) f5(i,2)],L2_1_2);
  s2_2=passrule([f5(i,1) f5(i,3)],L2_1_3);
  s2_3=passrule([f5(i,1) f5(i,4)],L2_1_4);
  s2_4=passrule([f5(i,2) f5(i,3)],L2_2_3);
  s2_5=passrule([f5(i,2) f5(i,4)],L2_2_4);
  s2_6=passrule([f5(i,3) f5(i,4)],L2_3_4);
%  s2_4=0; s2_5=0;
  S2=s2_1+s2_2+s2_3+s2_4+s2_5+s2_6;

  s3_1=passrule([f5(i,1) f5(i,2)],L3_1_2);
  s3_2=passrule([f5(i,1) f5(i,3)],L3_1_3);
  s3_3=passrule([f5(i,1) f5(i,4)],L3_1_4);
  s3_4=passrule([f5(i,2) f5(i,3)],L3_2_3);
  s3_5=passrule([f5(i,2) f5(i,4)],L3_2_4);
  s3_6=passrule([f5(i,3) f5(i,4)],L3_3_4);
%  s3_4=0; s3_5=0;
  S3=s3_1+s3_2+s3_3+s3_4+s3_5+s3_6;

  s4_1=passrule([f5(i,1) f5(i,2)],L4_1_2);
  s4_2=passrule([f5(i,1) f5(i,3)],L4_1_3);
```

```
  s4_3=passrule([f5(i,1) f5(i,4)],L4_1_4);
  s4_4=passrule([f5(i,2) f5(i,3)],L4_2_3);
  s4_5=passrule([f5(i,2) f5(i,4)],L4_2_4);
  s4_6=passrule([f5(i,3) f5(i,4)],L4_3_4);
%  s4_4=0; s4_5=0;
  S4=s4_1+s4_2+s4_3+s4_4+s4_5+s4_6;

  s5_1=passrule([f5(i,1) f5(i,2)],L5_1_2);
  s5_2=passrule([f5(i,1) f5(i,3)],L5_1_3);
  s5_3=passrule([f5(i,1) f5(i,4)],L5_1_4);
  s5_4=passrule([f5(i,2) f5(i,3)],L5_2_3);
  s5_5=passrule([f5(i,2) f5(i,4)],L5_2_4);
  s5_6=passrule([f5(i,3) f5(i,4)],L5_3_4);
%  s5_4=0; s5_5=0;
  S5=s5_1+s5_2+s5_3+s5_4+s5_5+s5_6;

  s6_1=passrule([f5(i,1) f5(i,2)],L6_1_2);
  s6_2=passrule([f5(i,1) f5(i,3)],L6_1_3);
  s6_3=passrule([f5(i,1) f5(i,4)],L6_1_4);
  s6_4=passrule([f5(i,2) f5(i,3)],L6_2_3);
  s6_5=passrule([f5(i,2) f5(i,4)],L6_2_4);
  s6_6=passrule([f5(i,3) f5(i,4)],L6_3_4);
%  s6_4=0; s6_5=0;
  S6=s6_1+s6_2+s6_3+s6_4+s6_5+s6_6;

  S=[S1;S2;S3;S4;S5;S6];
  [Y I]=sort(S);
  if I(6)==1
    C(5,1)=C(5,1)+1;
  elseif I(6)==2
    C(5,2)=C(5,2)+1;
  elseif I(6)==3
    C(5,3)=C(5,3)+1;
  elseif I(6)==4
    C(5,4)=C(5,4)+1;
  elseif I(6)==5
    C(5,5)=C(5,5)+1;
  elseif I(6)==6
    C(5,6)=C(5,6)+1;
  end
end

% class 6
```

```
for i=1:100
  s1_1=passrule([f6(i,1) f6(i,2)],L1_1_2);
  s1_2=passrule([f6(i,1) f6(i,3)],L1_1_3);
  s1_3=passrule([f6(i,1) f6(i,4)],L1_1_4);
  s1_4=passrule([f6(i,2) f6(i,3)],L1_2_3);
  s1_5=passrule([f6(i,2) f6(i,4)],L1_2_4);
  s1_6=passrule([f6(i,3) f6(i,4)],L1_3_4);
%  s1_4=0; s1_5=0;
  S1=s1_1+s1_2+s1_3+s1_4+s1_5+s1_6;

  s2_1=passrule([f6(i,1) f6(i,2)],L2_1_2);
  s2_2=passrule([f6(i,1) f6(i,3)],L2_1_3);
  s2_3=passrule([f6(i,1) f6(i,4)],L2_1_4);
  s2_4=passrule([f6(i,2) f6(i,3)],L2_2_3);
  s2_5=passrule([f6(i,2) f6(i,4)],L2_2_4);
  s2_6=passrule([f6(i,3) f6(i,4)],L2_3_4);
%  s2_4=0; s2_5=0;
  S2=s2_1+s2_2+s2_3+s2_4+s2_5+s2_6;

  s3_1=passrule([f6(i,1) f6(i,2)],L3_1_2);
  s3_2=passrule([f6(i,1) f6(i,3)],L3_1_3);
  s3_3=passrule([f6(i,1) f6(i,4)],L3_1_4);
  s3_4=passrule([f6(i,2) f6(i,3)],L3_2_3);
  s3_5=passrule([f6(i,2) f6(i,4)],L3_2_4);
  s3_6=passrule([f6(i,3) f6(i,4)],L3_3_4);
%  s3_4=0; s3_5=0;
  S3=s3_1+s3_2+s3_3+s3_4+s3_5+s3_6;

  s4_1=passrule([f6(i,1) f6(i,2)],L4_1_2);
  s4_2=passrule([f6(i,1) f6(i,3)],L4_1_3);
  s4_3=passrule([f6(i,1) f6(i,4)],L4_1_4);
  s4_4=passrule([f6(i,2) f6(i,3)],L4_2_3);
  s4_5=passrule([f6(i,2) f6(i,4)],L4_2_4);
  s4_6=passrule([f6(i,3) f6(i,4)],L4_3_4);
%  s4_4=0; s4_5=0;
  S4=s4_1+s4_2+s4_3+s4_4+s4_5+s4_6;

  s5_1=passrule([f6(i,1) f6(i,2)],L5_1_2);
  s5_2=passrule([f6(i,1) f6(i,3)],L5_1_3);
  s5_3=passrule([f6(i,1) f6(i,4)],L5_1_4);
  s5_4=passrule([f6(i,2) f6(i,3)],L5_2_3);
  s5_5=passrule([f6(i,2) f6(i,4)],L5_2_4);
  s5_6=passrule([f6(i,3) f6(i,4)],L5_3_4);
```

```
%  s5_4=0; s5_5=0;
   S5=s5_1+s5_2+s5_3+s5_4+s5_5+s5_6;

   s6_1=passrule([f6(i,1) f6(i,2)],L6_1_2);
   s6_2=passrule([f6(i,1) f6(i,3)],L6_1_3);
   s6_3=passrule([f6(i,1) f6(i,4)],L6_1_4);
   s6_4=passrule([f6(i,2) f6(i,3)],L6_2_3);
   s6_5=passrule([f6(i,2) f6(i,4)],L6_2_4);
   s6_6=passrule([f6(i,3) f6(i,4)],L6_3_4);
%  s6_4=0; s6_5=0;
   S6=s6_1+s6_2+s6_3+s6_4+s6_5+s6_6;

   S=[S1;S2;S3;S4;S5;S6];
   [Y I]=sort(S);
   if I(6)==1
     C(6,1)=C(6,1)+1;
   elseif I(6)==2
     C(6,2)=C(6,2)+1;
   elseif I(6)==3
     C(6,3)=C(6,3)+1;
   elseif I(6)==4
     C(6,4)=C(6,4)+1;
   elseif I(6)==5
     C(6,5)=C(6,5)+1;
   elseif I(6)==6
     C(6,6)=C(6,6)+1;
   end
end

% Display the confusion matrix
fprintf('\n Confusion Matrix \n\n')
fprintf('%d\t%d\t%d\t%d\t%d\t%d\n',C(1,1),C(1,2),C(1,3),C(1,4),C(1,5),C(1,6));
fprintf('%d\t%d\t%d\t%d\t%d\t%d\n',C(2,1),C(2,2),C(2,3),C(2,4),C(2,5),C(2,6));
fprintf('%d\t%d\t%d\t%d\t%d\t%d\n',C(3,1),C(3,2),C(3,3),C(3,4),C(3,5),C(3,6));
fprintf('%d\t%d\t%d\t%d\t%d\t%d\n',C(4,1),C(4,2),C(4,3),C(4,4),C(4,5),C(4,6));
fprintf('%d\t%d\t%d\t%d\t%d\t%d\n',C(5,1),C(5,2),C(5,3),C(5,4),C(5,5),C(5,6));
fprintf('%d\t%d\t%d\t%d\t%d\t%d\n',C(6,1),C(6,2),C(6,3),C(6,4),C(6,5),C(6,6));
```

NNEIGH.M

```
function C=nneigh(f1,f2,f3,f4,f5,f6)
% Script which returns the min 3 distances from unknown to f1, f2, f3,
% f4, f5 and f6.
% R. Hale

C=zeros(6);

% set number of neighbours
N=1;

% For class 1
for j=1:length(f1)
  c=0;
  for i=1:length(f1)
    if i ~= j
      c=c+1;
      distf1_f1(c)=norm(f1(j,:)-f1(i,:));
    end
  end
  tmp=sort(distf1_f1);
  distf1_f1=tmp(1:N);

  c=0;
  for i=1:length(f2)
    if i ~= j
      c=c+1;
      distf1_f2(c)=norm(f1(j,:)-f2(i,:));
    end
  end
  tmp=sort(distf1_f2);
  distf1_f2=tmp(1:N);

  c=0;
  for i=1:length(f3)
    if i ~= j
      c=c+1;
      distf1_f3(c)=norm(f1(j,:)-f3(i,:));
    end
  end
  tmp=sort(distf1_f3);
  distf1_f3=tmp(1:N);
```

```
c=0;
for i=1:length(f4)
  if i ~= j
    c=c+1;
    distf1_f4(c)=norm(f1(j,:)-f4(i,:));
  end
end
tmp=sort(distf1_f4);
distf1_f4=tmp(1:N);

c=0;
for i=1:length(f5)
  if i ~= j
    c=c+1;
    distf1_f5(c)=norm(f1(j,:)-f5(i,:));
  end
end
tmp=sort(distf1_f5);
distf1_f5=tmp(1:N);

c=0;
for i=1:length(f6)
  if i ~= j
    c=c+1;
    distf1_f6(c)=norm(f1(j,:)-f6(i,:));
  end
end
tmp=sort(distf1_f6);
distf1_f6=tmp(1:N);

D=[distf1_f1';distf1_f2';distf1_f3';distf1_f4';distf1_f5';distf1_f6'];
[M I]=sort(D);
if N==1
  if length(find(I(1:N)==1)) == 1
    C(1,1)=C(1,1)+1;
  elseif length(find(I(1:N)==2)) == 1
    C(1,2)=C(1,2)+1;
  elseif length(find(I(1:N)==3)) == 1
    C(1,3)=C(1,3)+1;
  elseif length(find(I(1:N)==4)) == 1
    C(1,4)=C(1,4)+1;
  elseif length(find(I(1:N)==5)) == 1
```

```
      C(1,5)=C(1,5)+1;
    elseif length(find(I(1:N)==6)) == 1
      C(1,6)=C(1,6)+1;
    end
  elseif N==3
    if length([find(I(1:N)==1) find(I(1:N)==2) find(I(1:N)==3)]) >= 2
      C(1,1)=C(1,1)+1;
    elseif length([find(I(1:N)==4) find(I(1:N)==5) find(I(1:N)==6)]) >= 2
      C(1,2)=C(1,2)+1;
    elseif length([find(I(1:N)==7) find(I(1:N)==8) find(I(1:N)==9)]) >= 2
      C(1,3)=C(1,3)+1;
    elseif length([find(I(1:N)==10) find(I(1:N)==11) find(I(1:N)==12)]) >= 2
      C(1,4)=C(1,4)+1;
    elseif length([find(I(1:N)==13) find(I(1:N)==14) find(I(1:N)==15)]) >= 2
      C(1,5)=C(1,5)+1;
    elseif length([find(I(1:N)==16) find(I(1:N)==17) find(I(1:N)==18)]) >= 2
      C(1,6)=C(1,6)+1;
    end
  elseif N==5
    if length([find(I(1:N)==1) find(I(1:N)==2) find(I(1:N)==3) find(I(1:N)==4)
find(I(1:N)==5)]) >= 3
      C(1,1)=C(1,1)+1;
    elseif length([find(I(1:N)==6) find(I(1:N)==7) find(I(1:N)==8) find(I(1:N)==9)
find(I(1:N)==10)]) >= 3
      C(1,2)=C(1,2)+1;
    elseif length([find(I(1:N)==11) find(I(1:N)==12) find(I(1:N)==13) find(I(1:N)==14)
find(I(1:N)==15)]) >= 3
      C(1,3)=C(1,3)+1;
    elseif length([find(I(1:N)==16) find(I(1:N)==17) find(I(1:N)==18) find(I(1:N)==19)
find(I(1:N)==20)]) >= 3
      C(1,4)=C(1,4)+1;
    elseif length([find(I(1:N)==21) find(I(1:N)==22) find(I(1:N)==23) find(I(1:N)==24)
find(I(1:N)==25)]) >= 3
      C(1,5)=C(1,5)+1;
    elseif length([find(I(1:N)==26) find(I(1:N)==27) find(I(1:N)==28) find(I(1:N)==29)
find(I(1:N)==30)]) >= 3
      C(1,6)=C(1,6)+1;
    end
  end
end

% For class 2
for j=1:length(f2)
```

```
c=0;
for i=1:length(f1)
  if i ~= j
    c=c+1;
    distf2_f1(c)=norm(f2(j,:)-f1(i,:));
  end
end
tmp=sort(distf2_f1);
distf2_f1=tmp(1:N);

c=0;
for i=1:length(f2)
  if i ~= j
    c=c+1;
    distf2_f2(c)=norm(f2(j,:)-f2(i,:));
  end
end
tmp=sort(distf2_f2);
distf2_f2=tmp(1:N);

c=0;
for i=1:length(f3)
  if i ~= j
    c=c+1;
    distf2_f3(c)=norm(f2(j,:)-f3(i,:));
  end
end
tmp=sort(distf2_f3);
distf2_f3=tmp(1:N);

c=0;
for i=1:length(f4)
  if i ~= j
    c=c+1;
    distf2_f4(c)=norm(f2(j,:)-f4(i,:));
  end
end
tmp=sort(distf2_f4);
distf2_f4=tmp(1:N);

c=0;
for i=1:length(f5)
  if i ~= j
```

```
    c=c+1;
    distf2_f5(c)=norm(f2(j,:)-f5(i,:));
  end
end
tmp=sort(distf2_f5);
distf2_f5=tmp(1:N);

c=0;
for i=1:length(f6)
  if i ~= j
    c=c+1;
    distf2_f6(c)=norm(f2(j,:)-f6(i,:));
  end
end
tmp=sort(distf2_f6);
distf2_f6=tmp(1:N);

D=[distf2_f1';distf2_f2';distf2_f3';distf2_f4';distf2_f5';distf2_f6'];
[M I]=sort(D);
if N==1
  if length(find(I(1:N)==1)) == 1
    C(2,1)=C(2,1)+1;
  elseif length(find(I(1:N)==2)) == 1
    C(2,2)=C(2,2)+1;
  elseif length(find(I(1:N)==3)) == 1
    C(2,3)=C(2,3)+1;
  elseif length(find(I(1:N)==4)) == 1
    C(2,4)=C(2,4)+1;
  elseif length(find(I(1:N)==5)) == 1
    C(2,5)=C(2,5)+1;
  elseif length(find(I(1:N)==6)) == 1
    C(2,6)=C(2,6)+1;
  end
elseif N==3
  if length([find(I(1:N)==1) find(I(1:N)==2) find(I(1:N)==3)]) >= 2
    C(2,1)=C(2,1)+1;
  elseif length([find(I(1:N)==4) find(I(1:N)==5) find(I(1:N)==6)]) >= 2
    C(2,2)=C(2,2)+1;
  elseif length([find(I(1:N)==7) find(I(1:N)==8) find(I(1:N)==9)]) >= 2
    C(2,3)=C(2,3)+1;
  elseif length([find(I(1:N)==10) find(I(1:N)==11) find(I(1:N)==12)]) >= 2
    C(2,4)=C(2,4)+1;
  elseif length([find(I(1:N)==13) find(I(1:N)==14) find(I(1:N)==15)]) >= 2
```

```
      C(2,5)=C(2,5)+1;
    elseif length([find(I(1:N)==16) find(I(1:N)==17) find(I(1:N)==18)]) >= 2
      C(2,6)=C(2,6)+1;
    end
  elseif N==5
    if length([find(I(1:N)==1) find(I(1:N)==2) find(I(1:N)==3) find(I(1:N)==4)
find(I(1:N)==5)]) >= 3
      C(2,1)=C(2,1)+1;
    elseif length([find(I(1:N)==6) find(I(1:N)==7) find(I(1:N)==8) find(I(1:N)==9)
find(I(1:N)==10)]) >= 3
      C(2,2)=C(2,2)+1;
    elseif length([find(I(1:N)==11) find(I(1:N)==12) find(I(1:N)==13) find(I(1:N)==14)
find(I(1:N)==15)]) >= 3
      C(2,3)=C(2,3)+1;
    elseif length([find(I(1:N)==16) find(I(1:N)==17) find(I(1:N)==18) find(I(1:N)==19)
find(I(1:N)==20)]) >= 3
      C(2,4)=C(2,4)+1;
    elseif length([find(I(1:N)==21) find(I(1:N)==22) find(I(1:N)==23) find(I(1:N)==24)
find(I(1:N)==25)]) >= 3
      C(2,5)=C(2,5)+1;
    elseif length([find(I(1:N)==26) find(I(1:N)==27) find(I(1:N)==28) find(I(1:N)==29)
find(I(1:N)==30)]) >= 3
      C(2,6)=C(2,6)+1;
    end
  end
end

% For class 3
for j=1:length(f3)
  c=0;
  for i=1:length(f1)
    if i ~= j
      c=c+1;
      distf3_f1(c)=norm(f3(j,:)-f1(i,:));
    end
  end
  tmp=sort(distf3_f1);
  distf3_f1=tmp(1:N);

  c=0;
  for i=1:length(f2)
    if i ~= j
      c=c+1;
```

```
    distf3_f2(c)=norm(f3(j,:)-f2(i,:));
  end
end
tmp=sort(distf3_f2);
distf3_f2=tmp(1:N);

c=0;
for i=1:length(f3)
  if i ~= j
    c=c+1;
    distf3_f3(c)=norm(f3(j,:)-f3(i,:));
  end
end
tmp=sort(distf3_f3);
distf3_f3=tmp(1:N);

c=0;
for i=1:length(f4)
  if i ~= j
    c=c+1;
    distf3_f4(c)=norm(f3(j,:)-f4(i,:));
  end
end
tmp=sort(distf3_f4);
distf3_f4=tmp(1:N);

c=0;
for i=1:length(f5)
  if i ~= j
    c=c+1;
    distf3_f5(c)=norm(f3(j,:)-f5(i,:));
  end
end
tmp=sort(distf3_f5);
distf3_f5=tmp(1:N);

c=0;
for i=1:length(f6)
  if i ~= j
    c=c+1;
    distf3_f6(c)=norm(f3(j,:)-f6(i,:));
  end
end
```

```
tmp=sort(distf3_f6);
distf3_f6=tmp(1:N);

D=[distf3_f1';distf3_f2';distf3_f3';distf3_f4';distf3_f5';distf3_f6'];
[M I]=sort(D);
if N==1
  if length(find(I(1:N)==1)) == 1
    C(3,1)=C(3,1)+1;
  elseif length(find(I(1:N)==2)) == 1
    C(3,2)=C(3,2)+1;
  elseif length(find(I(1:N)==3)) == 1
    C(3,3)=C(3,3)+1;
  elseif length(find(I(1:N)==4)) == 1
    C(3,4)=C(3,4)+1;
  elseif length(find(I(1:N)==5)) == 1
    C(3,5)=C(3,5)+1;
  elseif length(find(I(1:N)==6)) == 1
    C(3,6)=C(3,6)+1;
  end
elseif N==3
  if length([find(I(1:N)==1) find(I(1:N)==2) find(I(1:N)==3)]) >= 2
    C(3,1)=C(3,1)+1;
  elseif length([find(I(1:N)==4) find(I(1:N)==5) find(I(1:N)==6)]) >= 2
    C(3,2)=C(3,2)+1;
  elseif length([find(I(1:N)==7) find(I(1:N)==8) find(I(1:N)==9)]) >= 2
    C(3,3)=C(3,3)+1;
  elseif length([find(I(1:N)==10) find(I(1:N)==11) find(I(1:N)==12)]) >= 2
    C(3,4)=C(3,4)+1;
  elseif length([find(I(1:N)==13) find(I(1:N)==14) find(I(1:N)==15)]) >= 2
    C(3,5)=C(3,5)+1;
  elseif length([find(I(1:N)==16) find(I(1:N)==17) find(I(1:N)==18)]) >= 2
    C(3,6)=C(3,6)+1;
  end
elseif N==5
  if length([find(I(1:N)==1) find(I(1:N)==2) find(I(1:N)==3) find(I(1:N)==4)
find(I(1:N)==5)]) >= 3
    C(3,1)=C(3,1)+1;
  elseif length([find(I(1:N)==6) find(I(1:N)==7) find(I(1:N)==8) find(I(1:N)==9)
find(I(1:N)==10)]) >= 3
    C(3,2)=C(3,2)+1;
  elseif length([find(I(1:N)==11) find(I(1:N)==12) find(I(1:N)==13) find(I(1:N)==14)
find(I(1:N)==15)]) >= 3
    C(3,3)=C(3,3)+1;
```

```
    elseif length([find(I(1:N)==16) find(I(1:N)==17) find(I(1:N)==18) find(I(1:N)==19)
find(I(1:N)==20)]) >= 3
      C(3,4)=C(3,4)+1;
    elseif length([find(I(1:N)==21) find(I(1:N)==22) find(I(1:N)==23) find(I(1:N)==24)
find(I(1:N)==25)]) >= 3
      C(3,5)=C(3,5)+1;
    elseif length([find(I(1:N)==26) find(I(1:N)==27) find(I(1:N)==28) find(I(1:N)==29)
find(I(1:N)==30)]) >= 3
      C(3,6)=C(3,6)+1;
    end
  end
end

% For class 4
for j=1:length(f4)
  c=0;
  for i=1:length(f1)
    if i ~= j
      c=c+1;
      distf4_f1(c)=norm(f4(j,:)-f1(i,:));
    end
  end
  tmp=sort(distf4_f1);
  distf4_f1=tmp(1:N);

  c=0;
  for i=1:length(f2)
    if i ~= j
      c=c+1;
      distf4_f2(c)=norm(f4(j,:)-f2(i,:));
    end
  end
  tmp=sort(distf4_f2);
  distf4_f2=tmp(1:N);

  c=0;
  for i=1:length(f3)
    if i ~= j
      c=c+1;
      distf4_f3(c)=norm(f4(j,:)-f3(i,:));
    end
  end
  tmp=sort(distf4_f3);
```

```
distf4_f3=tmp(1:N);

c=0;
for i=1:length(f4)
  if i ~= j
    c=c+1;
    distf4_f4(c)=norm(f4(j,:)-f4(i,:));
  end
end
tmp=sort(distf4_f4);
distf4_f4=tmp(1:N);

c=0;
for i=1:length(f5)
  if i ~= j
    c=c+1;
    distf4_f5(c)=norm(f4(j,:)-f5(i,:));
  end
end
tmp=sort(distf4_f5);
distf4_f5=tmp(1:N);

c=0;
for i=1:length(f6)
  if i ~= j
    c=c+1;
    distf4_f6(c)=norm(f4(j,:)-f6(i,:));
  end
end
tmp=sort(distf4_f6);
distf4_f6=tmp(1:N);

D=[distf4_f1';distf4_f2';distf4_f3';distf4_f4';distf4_f5';distf4_f6'];
[M I]=sort(D);
if N==1
  if length(find(I(1:N)==1)) == 1
    C(4,1)=C(4,1)+1;
  elseif length(find(I(1:N)==2)) == 1
    C(4,2)=C(4,2)+1;
  elseif length(find(I(1:N)==3)) == 1
    C(4,3)=C(4,3)+1;
  elseif length(find(I(1:N)==4)) == 1
    C(4,4)=C(4,4)+1;
```

```
    elseif length(find(I(1:N)==5)) == 1
      C(4,5)=C(4,5)+1;
    elseif length(find(I(1:N)==6)) == 1
      C(4,6)=C(4,6)+1;
    end
  elseif N==3
    if length([find(I(1:N)==1) find(I(1:N)==2) find(I(1:N)==3)]) >= 2
      C(4,1)=C(4,1)+1;
    elseif length([find(I(1:N)==4) find(I(1:N)==5) find(I(1:N)==6)]) >= 2
      C(4,2)=C(4,2)+1;
    elseif length([find(I(1:N)==7) find(I(1:N)==8) find(I(1:N)==9)]) >= 2
      C(4,3)=C(4,3)+1;
    elseif length([find(I(1:N)==10) find(I(1:N)==11) find(I(1:N)==12)]) >= 2
      C(4,4)=C(4,4)+1;
    elseif length([find(I(1:N)==13) find(I(1:N)==14) find(I(1:N)==15)]) >= 2
      C(4,5)=C(4,5)+1;
    elseif length([find(I(1:N)==16) find(I(1:N)==17) find(I(1:N)==18)]) >= 2
      C(4,6)=C(4,6)+1;
    end
  elseif N==5
    if length([find(I(1:N)==1) find(I(1:N)==2) find(I(1:N)==3) find(I(1:N)==4)
find(I(1:N)==5)]) >= 3
      C(4,1)=C(4,1)+1;
    elseif length([find(I(1:N)==6) find(I(1:N)==7) find(I(1:N)==8) find(I(1:N)==9)
find(I(1:N)==10)]) >= 3
      C(4,2)=C(4,2)+1;
    elseif length([find(I(1:N)==11) find(I(1:N)==12) find(I(1:N)==13) find(I(1:N)==14)
find(I(1:N)==15)]) >= 3
      C(4,3)=C(4,3)+1;
    elseif length([find(I(1:N)==16) find(I(1:N)==17) find(I(1:N)==18) find(I(1:N)==19)
find(I(1:N)==20)]) >= 3
      C(4,4)=C(4,4)+1;
    elseif length([find(I(1:N)==21) find(I(1:N)==22) find(I(1:N)==23) find(I(1:N)==24)
find(I(1:N)==25)]) >= 3
      C(4,5)=C(4,5)+1;
    elseif length([find(I(1:N)==26) find(I(1:N)==27) find(I(1:N)==28) find(I(1:N)==29)
find(I(1:N)==30)]) >= 3
      C(4,6)=C(4,6)+1;
    end
  end
end

% For class 5
```

```
for j=1:length(f5)
  c=0;
  for i=1:length(f1)
    if i ~= j
      c=c+1;
      distf5_f1(c)=norm(f5(j,:)-f1(i,:));
    end
  end
  tmp=sort(distf5_f1);
  distf5_f1=tmp(1:N);

  c=0;
  for i=1:length(f2)
    if i ~= j
      c=c+1;
      distf5_f2(c)=norm(f5(j,:)-f2(i,:));
    end
  end
  tmp=sort(distf5_f2);
  distf5_f2=tmp(1:N);

  c=0;
  for i=1:length(f3)
    if i ~= j
      c=c+1;
      distf5_f3(c)=norm(f5(j,:)-f3(i,:));
    end
  end
  tmp=sort(distf5_f3);
  distf5_f3=tmp(1:N);

  c=0;
  for i=1:length(f4)
    if i ~= j
      c=c+1;
      distf5_f4(c)=norm(f5(j,:)-f4(i,:));
    end
  end
  tmp=sort(distf5_f4);
  distf5_f4=tmp(1:N);

  c=0;
  for i=1:length(f5)
```

```
  if i ~= j
    c=c+1;
    distf5_f5(c)=norm(f5(j,:)-f5(i,:));
  end
end
tmp=sort(distf5_f5);
distf5_f5=tmp(1:N);

c=0;
for i=1:length(f6)
  if i ~= j
    c=c+1;
    distf5_f6(c)=norm(f5(j,:)-f6(i,:));
  end
end
tmp=sort(distf5_f6);
distf5_f6=tmp(1:N);

D=[distf5_f1';distf5_f2';distf5_f3';distf5_f4';distf5_f5';distf5_f6'];
[M I]=sort(D);
if N==1
  if length(find(I(1:N)==1)) == 1
    C(5,1)=C(5,1)+1;
  elseif length(find(I(1:N)==2)) == 1
    C(5,2)=C(5,2)+1;
  elseif length(find(I(1:N)==3)) == 1
    C(5,3)=C(5,3)+1;
  elseif length(find(I(1:N)==4)) == 1
    C(5,4)=C(5,4)+1;
  elseif length(find(I(1:N)==5)) == 1
    C(5,5)=C(5,5)+1;
  elseif length(find(I(1:N)==6)) == 1
    C(5,6)=C(5,6)+1;
  end
elseif N==3
  if length([find(I(1:N)==1) find(I(1:N)==2) find(I(1:N)==3)]) >= 2
    C(5,1)=C(5,1)+1;
  elseif length([find(I(1:N)==4) find(I(1:N)==5) find(I(1:N)==6)]) >= 2
    C(5,2)=C(5,2)+1;
  elseif length([find(I(1:N)==7) find(I(1:N)==8) find(I(1:N)==9)]) >= 2
    C(5,3)=C(5,3)+1;
  elseif length([find(I(1:N)==10) find(I(1:N)==11) find(I(1:N)==12)]) >= 2
    C(5,4)=C(5,4)+1;
```

```
    elseif length([find(I(1:N)==13) find(I(1:N)==14) find(I(1:N)==15)]) >= 2
      C(5,5)=C(5,5)+1;
    elseif length([find(I(1:N)==16) find(I(1:N)==17) find(I(1:N)==18)]) >= 2
      C(5,6)=C(5,6)+1;
    end
  elseif N==5
    if length([find(I(1:N)==1) find(I(1:N)==2) find(I(1:N)==3) find(I(1:N)==4)
find(I(1:N)==5)]) >= 3
      C(5,1)=C(5,1)+1;
    elseif length([find(I(1:N)==6) find(I(1:N)==7) find(I(1:N)==8) find(I(1:N)==9)
find(I(1:N)==10)]) >= 3
      C(5,2)=C(5,2)+1;
    elseif length([find(I(1:N)==11) find(I(1:N)==12) find(I(1:N)==13) find(I(1:N)==14)
find(I(1:N)==15)]) >= 3
      C(5,3)=C(5,3)+1;
    elseif length([find(I(1:N)==16) find(I(1:N)==17) find(I(1:N)==18) find(I(1:N)==19)
find(I(1:N)==20)]) >= 3
      C(5,4)=C(5,4)+1;
    elseif length([find(I(1:N)==21) find(I(1:N)==22) find(I(1:N)==23) find(I(1:N)==24)
find(I(1:N)==25)]) >= 3
      C(5,5)=C(5,5)+1;
    elseif length([find(I(1:N)==26) find(I(1:N)==27) find(I(1:N)==28) find(I(1:N)==29)
find(I(1:N)==30)]) >= 3
      C(5,6)=C(5,6)+1;
    end
  end
end

% For class 6
for j=1:length(f6)
  c=0;
  for i=1:length(f1)
    if i ~= j
      c=c+1;
      distf6_f1(c)=norm(f6(j,:)-f1(i,:));
    end
  end
  tmp=sort(distf6_f1);
  distf6_f1=tmp(1:N);

  c=0;
  for i=1:length(f2)
    if i ~= j
```

```
    c=c+1;
    distf6_f2(c)=norm(f6(j,:)-f2(i,:));
  end
end
tmp=sort(distf6_f2);
distf6_f2=tmp(1:N);

c=0;
for i=1:length(f3)
  if i ~= j
    c=c+1;
    distf6_f3(c)=norm(f6(j,:)-f3(i,:));
  end
end
tmp=sort(distf6_f3);
distf6_f3=tmp(1:N);

c=0;
for i=1:length(f4)
  if i ~= j
    c=c+1;
    distf6_f4(c)=norm(f6(j,:)-f4(i,:));
  end
end
tmp=sort(distf6_f4);
distf6_f4=tmp(1:N);

c=0;
for i=1:length(f5)
  if i ~= j
    c=c+1;
    distf6_f5(c)=norm(f6(j,:)-f5(i,:));
  end
end
tmp=sort(distf6_f5);
distf6_f5=tmp(1:N);

c=0;
for i=1:length(f6)
  if i ~= j
    c=c+1;
    distf6_f6(c)=norm(f6(j,:)-f6(i,:));
  end
```

```
  end
  tmp=sort(distf6_f6);
  distf6_f6=tmp(1:N);

  D=[distf6_f1';distf6_f2';distf6_f3';distf6_f4';distf6_f5';distf6_f6'];
  [M I]=sort(D);
  if N==1
    if length(find(I(1:N)==1)) == 1
      C(6,1)=C(6,1)+1;
    elseif length(find(I(1:N)==2)) == 1
      C(6,2)=C(6,2)+1;
    elseif length(find(I(1:N)==3)) == 1
      C(6,3)=C(6,3)+1;
    elseif length(find(I(1:N)==4)) == 1
      C(6,4)=C(6,4)+1;
    elseif length(find(I(1:N)==5)) == 1
      C(6,5)=C(6,5)+1;
    elseif length(find(I(1:N)==6)) == 1
      C(6,6)=C(6,6)+1;
    end
  elseif N==3
    if length([find(I(1:N)==1) find(I(1:N)==2) find(I(1:N)==3)]) >= 2
      C(6,1)=C(6,1)+1;
    elseif length([find(I(1:N)==4) find(I(1:N)==5) find(I(1:N)==6)]) >= 2
      C(6,2)=C(6,2)+1;
    elseif length([find(I(1:N)==7) find(I(1:N)==8) find(I(1:N)==9)]) >= 2
      C(6,3)=C(6,3)+1;
    elseif length([find(I(1:N)==10) find(I(1:N)==11) find(I(1:N)==12)]) >= 2
      C(6,4)=C(6,4)+1;
    elseif length([find(I(1:N)==13) find(I(1:N)==14) find(I(1:N)==15)]) >= 2
      C(6,5)=C(6,5)+1;
    elseif length([find(I(1:N)==16) find(I(1:N)==17) find(I(1:N)==18)]) >= 2
      C(6,6)=C(6,6)+1;
    end
  elseif N==5
    if length([find(I(1:N)==1) find(I(1:N)==2) find(I(1:N)==3) find(I(1:N)==4)
find(I(1:N)==5)]) >= 3
      C(6,1)=C(6,1)+1;
    elseif length([find(I(1:N)==6) find(I(1:N)==7) find(I(1:N)==8) find(I(1:N)==9)
find(I(1:N)==10)]) >= 3
      C(6,2)=C(6,2)+1;
    elseif length([find(I(1:N)==11) find(I(1:N)==12) find(I(1:N)==13) find(I(1:N)==14)
find(I(1:N)==15)]) >= 3
```

```
      C(6,3)=C(6,3)+1;
    elseif length([find(I(1:N)==16) find(I(1:N)==17) find(I(1:N)==18) find(I(1:N)==19)
find(I(1:N)==20)]) >= 3
      C(6,4)=C(6,4)+1;
    elseif length([find(I(1:N)==21) find(I(1:N)==22) find(I(1:N)==23) find(I(1:N)==24)
find(I(1:N)==25)]) >= 3
      C(6,5)=C(6,5)+1;
    elseif length([find(I(1:N)==26) find(I(1:N)==27) find(I(1:N)==28) find(I(1:N)==29)
find(I(1:N)==30)]) >= 3
      C(6,6)=C(6,6)+1;
    end
  end
end

% Display the confusion matrix
fprintf('\n Confusion Matrix \n\n')
fprintf('%d\t%d\t%d\t%d\t%d\t%d\n',C(1,1),C(1,2),C(1,3),C(1,4),C(1,5),C(1,6));
fprintf('%d\t%d\t%d\t%d\t%d\t%d\n',C(2,1),C(2,2),C(2,3),C(2,4),C(2,5),C(2,6));
fprintf('%d\t%d\t%d\t%d\t%d\t%d\n',C(3,1),C(3,2),C(3,3),C(3,4),C(3,5),C(3,6));
fprintf('%d\t%d\t%d\t%d\t%d\t%d\n',C(4,1),C(4,2),C(4,3),C(4,4),C(4,5),C(4,6));
fprintf('%d\t%d\t%d\t%d\t%d\t%d\n',C(5,1),C(5,2),C(5,3),C(5,4),C(5,5),C(5,6));
fprintf('%d\t%d\t%d\t%d\t%d\t%d\n',C(6,1),C(6,2),C(6,3),C(6,4),C(6,5),C(6,6));
```

MAL_DIST.M

```
function C=mal_dist(f1,f2,f3,f4,f5,f6)
% Script which returns the min 3 distances from unknown to f1, f2, f3,
% f4, f5 and f6.
% R. Hale

% Calculate class covariance matrices
K1=cov(f1);
K2=cov(f2);
K3=cov(f3);
K4=cov(f4);
K5=cov(f5);
K6=cov(f6);

% Inverse covariance matrices
K1_inv=inv(K1);
K2_inv=inv(K2);
K3_inv=inv(K3);
```

```
K4_inv=inv(K4);
K5_inv=inv(K5);
K6_inv=inv(K6);

% Calculate class means
m1=mean(f1);
m2=mean(f2);
m3=mean(f3);
m4=mean(f4);
m5=mean(f5);
m6=mean(f6);

C=zeros(6);

% For class 1
% Find distance from all other classes
for j=1:length(f1)
  distf1_f1=(f1(j,:)-m1)*K1_inv*(f1(j,:)-m1)';

  distf1_f2=(f1(j,:)-m2)*K2_inv*(f1(j,:)-m2)';

  distf1_f3=(f1(j,:)-m3)*K3_inv*(f1(j,:)-m3)';

  distf1_f4=(f1(j,:)-m4)*K4_inv*(f1(j,:)-m4)';

  distf1_f5=(f1(j,:)-m5)*K5_inv*(f1(j,:)-m5)';

  distf1_f6=(f1(j,:)-m6)*K6_inv*(f1(j,:)-m6)';

  D=[distf1_f1;distf1_f2;distf1_f3;distf1_f4;distf1_f5;distf1_f6];
  [M I]=sort(D);

  % Classify as class with minimum distance
  if I(1)==1
    C(1,1)=C(1,1)+1;
  elseif I(1)==2
    C(1,2)=C(1,2)+1;
  elseif I(1)==3
    C(1,3)=C(1,3)+1;
  elseif I(1)==4
    C(1,4)=C(1,4)+1;
  elseif I(1)==5
    C(1,5)=C(1,5)+1;
```

```
  elseif I(1)==6
    C(1,6)=C(1,6)+1;
  end
end

% For class 2
% Find distance from all other classes
for j=1:length(f2)
  distf2_f1=(f2(j,:)-m1)*K1_inv*(f2(j,:)-m1)';

  distf2_f2=(f2(j,:)-m2)*K2_inv*(f2(j,:)-m2)';

  distf2_f3=(f2(j,:)-m3)*K3_inv*(f2(j,:)-m3)';

  distf2_f4=(f2(j,:)-m4)*K4_inv*(f2(j,:)-m4)';

  distf2_f5=(f2(j,:)-m5)*K5_inv*(f2(j,:)-m5)';

  distf2_f6=(f2(j,:)-m6)*K6_inv*(f2(j,:)-m6)';

  D=[distf2_f1;distf2_f2;distf2_f3;distf2_f4;distf2_f5;distf2_f6];
  [M I]=sort(D);

  % Classify as class with minimum distance
  if I(1)==1
    C(2,1)=C(2,1)+1;
  elseif I(1)==2
    C(2,2)=C(2,2)+1;
  elseif I(1)==3
    C(2,3)=C(2,3)+1;
  elseif I(1)==4
    C(2,4)=C(2,4)+1;
  elseif I(1)==5
    C(2,5)=C(2,5)+1;
  elseif I(1)==6
    C(2,6)=C(2,6)+1;
  end
end

% For class 3
% Find distance from all other classes
for j=1:length(f3)
  distf3_f1=(f3(j,:)-m1)*K1_inv*(f3(j,:)-m1)';
```

```
  distf3_f2=(f3(j,:)-m2)*K2_inv*(f3(j,:)-m2)';

  distf3_f3=(f3(j,:)-m3)*K3_inv*(f3(j,:)-m3)';

  distf3_f4=(f3(j,:)-m4)*K4_inv*(f3(j,:)-m4)';

  distf3_f5=(f3(j,:)-m5)*K5_inv*(f3(j,:)-m5)';

  distf3_f6=(f3(j,:)-m6)*K6_inv*(f3(j,:)-m6)';

  D=[distf3_f1;distf3_f2;distf3_f3;distf3_f4;distf3_f5;distf3_f6];
  [M I]=sort(D);

  % Classify as class with minimum distance
  if I(1)==1
    C(3,1)=C(3,1)+1;
  elseif I(1)==2
    C(3,2)=C(3,2)+1;
  elseif I(1)==3
    C(3,3)=C(3,3)+1;
  elseif I(1)==4
    C(3,4)=C(3,4)+1;
  elseif I(1)==5
    C(3,5)=C(3,5)+1;
  elseif I(1)==6
    C(3,6)=C(3,6)+1;
  end
end

% For class 4
% Find distance from all other classes
for j=1:length(f4)
  distf4_f1=(f4(j,:)-m1)*K1_inv*(f4(j,:)-m1)';

  distf4_f2=(f4(j,:)-m2)*K2_inv*(f4(j,:)-m2)';

  distf4_f3=(f4(j,:)-m3)*K3_inv*(f4(j,:)-m3)';

  distf4_f4=(f4(j,:)-m4)*K4_inv*(f4(j,:)-m4)';

  distf4_f5=(f4(j,:)-m5)*K5_inv*(f4(j,:)-m5)';
```

```
  distf4_f6=(f4(j,:)-m6)*K6_inv*(f4(j,:)-m6)';

  D=[distf4_f1;distf4_f2;distf4_f3;distf4_f4;distf4_f5;distf4_f6];
  [M I]=sort(D);

  % Classify as class with minimum distance
  if I(1)==1
    C(4,1)=C(4,1)+1;
  elseif I(1)==2
    C(4,2)=C(4,2)+1;
  elseif I(1)==3
    C(4,3)=C(4,3)+1;
  elseif I(1)==4
    C(4,4)=C(4,4)+1;
  elseif I(1)==5
    C(4,5)=C(4,5)+1;
  elseif I(1)==6
    C(4,6)=C(4,6)+1;
  end
end

% For class 5
% Find distance from all other classes
for j=1:length(f5)
  distf5_f1=(f5(j,:)-m1)*K1_inv*(f5(j,:)-m1)';

  distf5_f2=(f5(j,:)-m2)*K2_inv*(f5(j,:)-m2)';

  distf5_f3=(f5(j,:)-m3)*K3_inv*(f5(j,:)-m3)';

  distf5_f4=(f5(j,:)-m4)*K4_inv*(f5(j,:)-m4)';

  distf5_f5=(f5(j,:)-m5)*K5_inv*(f5(j,:)-m5)';

  distf5_f6=(f5(j,:)-m6)*K6_inv*(f5(j,:)-m6)';

  D=[distf5_f1;distf5_f2;distf5_f3;distf5_f4;distf5_f5;distf5_f6];
  [M I]=sort(D);

  % Classify as class with minimum distance
  if I(1)==1
    C(5,1)=C(5,1)+1;
  elseif I(1)==2
```

```
    C(5,2)=C(5,2)+1;
  elseif I(1)==3
    C(5,3)=C(5,3)+1;
  elseif I(1)==4
    C(5,4)=C(5,4)+1;
  elseif I(1)==5
    C(5,5)=C(5,5)+1;
  elseif I(1)==6
    C(5,6)=C(5,6)+1;
  end
end

% For class 6
% Find distance from all other classes
for j=1:length(f6)
  distf6_f1=(f6(j,:)-m1)*K1_inv*(f6(j,:)-m1)';

  distf6_f2=(f6(j,:)-m2)*K2_inv*(f6(j,:)-m2)';

  distf6_f3=(f6(j,:)-m3)*K3_inv*(f6(j,:)-m3)';

  distf6_f4=(f6(j,:)-m4)*K4_inv*(f6(j,:)-m4)';

  distf6_f5=(f6(j,:)-m5)*K5_inv*(f6(j,:)-m5)';

  distf6_f6=(f6(j,:)-m6)*K6_inv*(f6(j,:)-m6)';

  D=[distf6_f1;distf6_f2;distf6_f3;distf6_f4;distf6_f5;distf6_f6];
  [M I]=sort(D);

  % Classify as class with minimum distance
  if I(1)==1
    C(6,1)=C(6,1)+1;
  elseif I(1)==2
    C(6,2)=C(6,2)+1;
  elseif I(1)==3
    C(6,3)=C(6,3)+1;
  elseif I(1)==4
    C(6,4)=C(6,4)+1;
  elseif I(1)==5
    C(6,5)=C(6,5)+1;
  elseif I(1)==6
    C(6,6)=C(6,6)+1;
```

```
  end
end

% Display the confusion matrix
fprintf('\n Confusion Matrix \n\n')
fprintf('%d\t%d\t%d\t%d\t%d\t%d\n',C(1,1),C(1,2),C(1,3),C(1,4),C(1,5),C(1,6));
fprintf('%d\t%d\t%d\t%d\t%d\t%d\n',C(2,1),C(2,2),C(2,3),C(2,4),C(2,5),C(2,6));
fprintf('%d\t%d\t%d\t%d\t%d\t%d\n',C(3,1),C(3,2),C(3,3),C(3,4),C(3,5),C(3,6));
fprintf('%d\t%d\t%d\t%d\t%d\t%d\n',C(4,1),C(4,2),C(4,3),C(4,4),C(4,5),C(4,6));
fprintf('%d\t%d\t%d\t%d\t%d\t%d\n',C(5,1),C(5,2),C(5,3),C(5,4),C(5,5),C(5,6));
fprintf('%d\t%d\t%d\t%d\t%d\t%d\n',C(6,1),C(6,2),C(6,3),C(6,4),C(6,5),C(6,6));
```

SEG_WIN.M

```
% Script to take segmented data from and pass to window.m function

% for seg1 & gest1
load seg1;load gest1;
for i=1:length(seg1)-1
   eval(['f4f1_' num2str(i) '=window(gest1(seg1(i):seg1(i+1),:));'])
   eval(['save f4f1_' num2str(i) ' f4f1_' num2str(i) ' -ascii;'])
end

% for seg2 & gest2
load seg2;load gest2;
for i=1:length(seg2)-1
   eval(['f4f2_' num2str(i) '=window(gest2(seg2(i):seg2(i+1),:));'])
   eval(['save f4f2_' num2str(i) ' f4f2_' num2str(i) ' -ascii;'])
end

% for seg3 & gest3
load seg3;load gest3;
for i=1:length(seg3)-1
   eval(['f4f3_' num2str(i) '=window(gest3(seg3(i):seg3(i+1),:));'])
   eval(['save f4f3_' num2str(i) ' f4f3_' num2str(i) ' -ascii;'])
end

% for seg4 & gest4
load seg4;load gest4;
for i=1:length(seg4)-1
   eval(['f4f4_' num2str(i) '=window(gest4(seg4(i):seg4(i+1),:));'])
   eval(['save f4f4_' num2str(i) ' f4f4_' num2str(i) ' -ascii;'])
end
```

```
% for seg5 & gest5
load seg5;load gest5;
for i=1:length(seg5)-1
   eval(['f4f5_' num2str(i) '=window(gest5(seg5(i):seg5(i+1),:));'])
   eval(['save f4f5_' num2str(i) ' f4f5_' num2str(i) ' -ascii;'])
end

% for seg6 & gest6
load seg6;load gest6;
for i=1:length(seg6)-1
   eval(['f4f6_' num2str(i) '=window(gest6(seg6(i):seg6(i+1),:));'])
   eval(['save f4f6_' num2str(i) ' f4f6_' num2str(i) ' -ascii;'])
end
```

F_O.M

```
% f4f1_1 -> f4f1_100
for w=1:100
   eval(['load f4f1_' num2str(w) ';'])
   eval(['FS=[FS;f4f1_' num2str(w) '];'])
end
maxFS=max(FS);
for w=1:100
   for k=1:4 % this matches the number of features
      eval(['f4f1_' num2str(w) '(:,' num2str(k) ')=f4f1_' num2str(w) ...
      '(:,' num2str(k) ')/maxFS(' num2str(k) ');'])
   end
   eval(['feat=f4f1_' num2str(w) ';'])

   if w ==1
     vector;
   else
     vector2;
   end

   eval(['O4f1_' num2str(w) '=seq(S1,S2,S3,S4,S5,S6,S7,S8);'])
   eval(['save O4f1_' num2str(w) ' O4f1_' num2str(w) ' -ascii;'])
   clear D D_tot M N S1 S2 S3 S4 S5 S6 S7 S8 d b d_tot e epsilon error j k l m mindist n
numcols numrows pos temp x i
end
clear

% f4f2_1 -> f4f2_100
```

```
for w=1:100
   eval(['load f4f2_' num2str(w) ';'])
   eval(['FS=[FS;f4f2_' num2str(w) '];'])
end
maxFS=max(FS);
for w=1:100
   for k=1:4 % this matches the number of features
      eval(['f4f2_' num2str(w) '(:,' num2str(k) ')=f4f2_' num2str(w) ...
      '(:,' num2str(k) ')/maxFS(' num2str(k) ');'])
   end
   eval(['feat=f4f2_' num2str(w) ';'])

   if w ==1
     vector;
   else
     vector2;
   end

   eval(['O4f2_' num2str(w) '=seq(S1,S2,S3,S4,S5,S6,S7,S8);'])
   eval(['save O4f2_' num2str(w) ' O4f2_' num2str(w) ' -ascii;'])
   clear D D_tot M N S1 S2 S3 S4 S5 S6 S7 S8 d b d_tot e epsilon error j k l m mindist n
numcols numrows pos temp x i
end
clear

% f4f3_1 -> f4f3_100
for w=1:100
   eval(['load f4f3_' num2str(w) ';'])
   eval(['FS=[FS;f4f3_' num2str(w) '];'])
end
maxFS=max(FS);
for w=1:100
   for k=1:4 % this matches the number of features
      eval(['f4f3_' num2str(w) '(:,' num2str(k) ')=f4f3_' num2str(w) ...
      '(:,' num2str(k) ')/maxFS(' num2str(k) ');'])
   end
   eval(['feat=f4f3_' num2str(w) ';'])

   if w ==1
     vector;
   else
     vector2;
   end
```

```
   eval(['O4f3_' num2str(w) '=seq(S1,S2,S3,S4,S5,S6,S7,S8);'])
   eval(['save O4f3_' num2str(w) ' O4f3_' num2str(w) ' -ascii;'])
   clear D D_tot M N S1 S2 S3 S4 S5 S6 S7 S8 d b d_tot e epsilon error j k l m mindist n
numcols numrows pos temp x i
end
clear

% f4f4_1 -> f4f4_100
for w=1:100
   eval(['load f4f4_' num2str(w) ';'])
   eval(['FS=[FS;f4f4_' num2str(w) '];'])
end
maxFS=max(FS);
for w=1:100
   for k=1:4 % this matches the number of features
      eval(['f4f4_' num2str(w) '(:,' num2str(k) ')=f4f4_' num2str(w) ...
      '(:,' num2str(k) ')/maxFS(' num2str(k) ');'])
   end
   eval(['feat=f4f4_' num2str(w) ';'])

   if w ==1
     vector;
   else
     vector2;
   end

   eval(['O4f4_' num2str(w) '=seq(S1,S2,S3,S4,S5,S6,S7,S8);'])
   eval(['save O4f4_' num2str(w) ' O4f4_' num2str(w) ' -ascii;'])
   clear D D_tot M N S1 S2 S3 S4 S5 S6 S7 S8 d b d_tot e epsilon error j k l m mindist n
numcols numrows pos temp x i
end
clear

% f4f5_1 -> f4f5_100
for w=1:100
   eval(['load f4f5_' num2str(w) ';'])
   eval(['FS=[FS;f4f5_' num2str(w) '];'])
end
maxFS=max(FS);
for w=1:100
   for k=1:4 % this matches the number of features
      eval(['f4f5_' num2str(w) '(:,' num2str(k) ')=f4f5_' num2str(w) ...
```

```
      '(:,' num2str(k) ')/maxFS(' num2str(k) ');'])
   end
   eval(['feat=f4f5_' num2str(w) ';'])

   if w ==1
     vector;
   else
     vector2;
   end

   eval(['O4f5_' num2str(w) '=seq(S1,S2,S3,S4,S5,S6,S7,S8);'])
   eval(['save O4f5_' num2str(w) ' O4f5_' num2str(w) ' -ascii;'])
   clear D D_tot M N S1 S2 S3 S4 S5 S6 S7 S8 d b d_tot e epsilon error j k l m mindist n
numcols numrows pos temp x i
end
clear

% f4f6_1 -> f4f6_100
for w=1:100
   eval(['load f4f6_' num2str(w) ';'])
   eval(['FS=[FS;f4f6_' num2str(w) '];'])
end
maxFS=max(FS);
for w=1:100
   for k=1:4 % this matches the number of features
      eval(['f4f6_' num2str(w) '(:,' num2str(k) ')=f4f6_' num2str(w) ...
      '(:,' num2str(k) ')/maxFS(' num2str(k) ');'])
   end
   eval(['feat=f4f6_' num2str(w) ';'])

   if w ==1
     vector;
   else
     vector2;
   end

   eval(['O4f6_' num2str(w) '=seq(S1,S2,S3,S4,S5,S6,S7,S8);'])
   eval(['save O4f6_' num2str(w) ' O4f6_' num2str(w) ' -ascii;'])
   clear D D_tot M N S1 S2 S3 S4 S5 S6 S7 S8 d b d_tot e epsilon error j k l m mindist n
numcols numrows pos temp x i
end
clear
```

SEQ.M

```
function O=seq(S1,S2,S3,S4,S5,S6,S7,S8)
  % Function to convert S1, S2, S3 & S4 output from vector.m
  % into output sequence O.  Edit when other than 4 S's input.
  for i=1:length(S1)
    O(S1(i))=1;
  end
  for i=1:length(S2)
    O(S2(i))=2;
  end
  for i=1:length(S3)
    O(S3(i))=3;
  end
  for i=1:length(S4)
    O(S4(i))=4;
  end
  for i=1:length(S5)
    O(S5(i))=5;
  end
  for i=1:length(S6)
    O(S6(i))=6;
  end
  for i=1:length(S7)
    O(S7(i))=7;
  end
  for i=1:length(S8)
    O(S8(i))=8;
  end
```

VECTOR.M

```
echo off;
% Vector quantization based; number of levels must be updated in
% initialization code below; number of dimensions or features must
% updated in initialization code below.
% Rodney Hale July 1996

tic

% Initialization
N=8;  % N classes; N levels
k=4;  % k features; k dimensional
```

```
e=0.001; % error
D(1)=9.99e62; % set distortion to infinity

% Define training vectors
%load feat.dat;
x=feat;

[numrows numcols]=size(x);
n=numrows; % number of training vectors in training set x

% Initial reproduction alphabet by splitting
M=1;
for j=1:k % vector if length k; k features
   A(1,j)=mean(x(:,j)); % one-level quantizer; centroid of the training sequence
end

% "Split" using epsilon=10% of the initial centroid vector
epsilon=A/10; % fixed for the splitting process
temp=A;
A(1,:)=temp + epsilon;
A(2,:)=temp - epsilon;

M=2*M;

m=1; % initial step 1
error = 9.99e62; % set error initially to large value

% continue until error less than e
while error > e
   % Reset each iteration
   d_tot=[];

   % Calculate distortion matrix; minimum value in each row represents
   % the partition, S, location of that vector
   for j=1:n % loop through for n vectors
      for l=1:M % loop through N levels of quantizer
         d_tot(j,l)=abs(x(j,:)-A(l,:))*abs(x(j,:)-A(l,:))'; % squared error distortion
      end
   end

   % Determine minimum distrotion for each vector and record the
   % location of the minimum in pos(n)
   for j=1:n % loop through for n vectors, find minimum distortion and
```

```
        % put in mindist matrix; find position of minimum distrotion
        % and put in pos matrix
    [mindist(j,1) pos(j,1)]=min(d_tot(j,:));
  end

  % compute error
  D_tot(m)=0;
  D_tot(m)=sum(mindist)/n;

  if m > 1  % do not have to calculate error on first iteration
    error = (D_tot(m-1)-D_tot(m))/D_tot(m);
  end

  % find the optimal reproduction alphabet; update A

  for i=1:M
    eval(['S' num2str(i) '=find(pos==i);'])  % determine partitions; S's

    if size(eval(['S', num2str(i)]),1)>1
      eval(['A(i,:)=sum(x(S' num2str(i) ',:))/length(S' num2str(i) ');'])
    else
      eval(['A(i,:)=x(S' num2str(i) ',:)/length(S' num2str(i) ');'])
    end

  end


  m=m+1; % move to next interation

  if m > 15 % end loop after 15 iterations; error does not converge
    error=0;
    echo on
    %
    % After 15 iterations error did not converge!!!
    %
    echo off
  end
end

%  Repeat splitting process until M=N
while M < N  % halt when M=N
  % Repeat algorithm to produce a good reproduction alphabet for
  % M dimensional quantizer
```

```
temp=A;
for b=1:M  % Produces a M-dimensional reproduction alphabet
   A(2*b-1,:) = temp(b,:) + epsilon;
   A(2*b,:) = temp(b,:) - epsilon;
end

M=2*M;
m=1; % initial step 1
error = 9.99e62; % set error initially to large value

% continue until error less than e
while error > e
   d=[];   % Reset each iteration
   d_tot=[];

   % Calculate distortion matrix; minimum value in each row represents
   % the partition, S, location of that vector
   for j=1:n % loop through for n vectors
      for l=1:M % loop through N levels of quantizer
         d_tot(j,l)=abs(x(j,:)-A(l,:))*abs(x(j,:)-A(l,:))'; % squared error distortion
      end
   end

   % Determine minimum distrotion for each vector and record the
   % location of the minimum in pos(n)
   for j=1:n  % loop through for n vectors, find minimum distortion and
         % put in mindist matrix; find position of minimum distrotion
         % and put in pos matrix
      [mindist(j,1) pos(j,1)]=min(d_tot(j,:));
   end

   % compute error
   D_tot(m)=0;
   D_tot(m)=sum(mindist)/n;

   if m > 1  % do not have to calculate error on first iteration
      error = (D_tot(m-1)-D_tot(m))/D_tot(m);
   end

   % find the optimal reproduction alphabet; update A

   for i=1:M
      eval(['S' num2str(i) '=find(pos==i);'])  % determine partitions; S's
```

```
        if size(eval(['S', num2str(i)]),1)>1
           eval(['A(i,:)=sum(x(S' num2str(i) ',:))/length(S' num2str(i) ');'])
        elseif length(eval(['S' num2str(i)]))==0
           A(i,:)=A(i,:);
        else
           eval(['A(i,:)=x(S' num2str(i) ',:)/length(S' num2str(i) ');'])
        end
     end


     m=m+1; % move to next interation

     if m > 15 % end loop after 15 iterations; error does not converge
        error=0;
        echo on
        %
        % After 15 iterations error did not converge!!!
        %
        echo off
     end
  end

end

toc
```

VECTOR2.M

```
echo off;
% Vector quantization based; number of levels must be updated in
% initialization code below; number of dimensions or features must
% updated in initialization code below.
% Rodney Hale July 1996

tic

% Initialization
N=8;  % N classes; N levels
k=4;  % k features; k dimensional
e=0.001;  % error
D(1)=9.99e62;  % set distortion to infinity

% Define training vectors
```

```
%load feat.dat;
x=feat;

[numrows numcols]=size(x);
n=numrows;  % number of training vectors in training set x

M=2;

m=1; % initial step 1
error = 9.99e62; % set error initially to large value

% continue until error less than e
while error > e
  % Reset each iteration
  d_tot=[];

  % Calculate distortion matrix; minimum value in each row represents
  % the partition, S, location of that vector
  for j=1:n % loop through for n vectors
    for l=1:M % loop through N levels of quantizer
      d_tot(j,l)=abs(x(j,:)-A(l,:))*abs(x(j,:)-A(l,:))'; % squared error distortion
    end
  end

  % Determine minimum distrotion for each vector and record the
  % location of the minimum in pos(n)
  for j=1:n  % loop through for n vectors, find minimum distortion and
    % put in mindist matrix; find position of minimum distrotion
    % and put in pos matrix
    [mindist(j,1) pos(j,1)]=min(d_tot(j,:));
  end

  % compute error
  D_tot(m)=0;
  D_tot(m)=sum(mindist)/n;

  if m > 1  % do not have to calculate error on first iteration
    error = (D_tot(m-1)-D_tot(m))/D_tot(m);
  end

  % find the optimal reproduction alphabet; update A

  for i=1:M
```

```
    eval(['S' num2str(i) '=find(pos==i);']) % determine partitions; S's
  end


  m=m+1; % move to next interation

  if m > 15 % end loop after 15 iterations; error does not converge
    error=0;
    echo on
    %
    % After 15 iterations error did not converge!!!
    %
    echo off
  end
end

% Repeat splitting process until M=N
while M < N % halt when M=N
  % Repeat algorithm to produce a good reproduction alphabet for
  % M dimensional quantizer

  M=2*M;
  m=1; % initial step 1
  error = 9.99e62; % set error initially to large value

  % continue until error less than e
  while error > e
    d=[];   % Reset each iteration
    d_tot=[];

    % Calculate distortion matrix; minimum value in each row represents
    % the partition, S, location of that vector
    for j=1:n % loop through for n vectors
      for l=1:M % loop through N levels of quantizer
        d_tot(j,l)=abs(x(j,:)-A(l,:))*abs(x(j,:)-A(l,:))'; % squared error distortion
      end
    end

    % Determine minimum distrotion for each vector and record the
    % location of the minimum in pos(n)
    for j=1:n  % loop through for n vectors, find minimum distortion and
      % put in mindist matrix; find position of minimum distrotion
      % and put in pos matrix
```

```
    [mindist(j,1) pos(j,1)]=min(d_tot(j,:));
  end

  % compute error
  D_tot(m)=0;
  D_tot(m)=sum(mindist)/n;

  if m > 1  % do not have to calculate error on first iteration
    error = (D_tot(m-1)-D_tot(m))/D_tot(m);
  end

  for i=1:M
    eval(['S' num2str(i) '=find(pos==i);'])  % determine partitions; S's
  end

  m=m+1; % move to next interation

  if m > 15 % end loop after 15 iterations; error does not converge
    error=0;
    echo on
    %
    % After 15 iterations error did not converge!!!
    %
    echo off
  end
 end

end

toc
```

HMMREEST.M

```
function [A_num,A_den,B_num,B_den,pi_n]=hmmreest(O,N,Q)
% HMM training
% R. Hale

%N=8; % Must match the number in tr1hmm.m & recoghmm.m
%Q=8; % Must match the number of possible observations from vector.m
T=length(O); % Length of observation vector O

% Randomize A
```

```
for i=1:N
  for j=1:N
    A(i,j)=rand(1);
%     if j < i
%       A(i,j)=0;
%     elseif i==j
%       A(i,j)=1;
%     end
  end
  A(i,:)=A(i,:)/sum(A(i,:));
end

% Randomize B

for i=1:N
  for j=1:Q
    B(i,j)=rand(1);
  end
  B(i,:)=B(i,:)/sum(B(i,:));
end

% Randomize pi
%pi=[pi1   Probability of using urn 1 initially
%   pi2   Probability of using urn 2 initially
%   pi3]; Probability of using urn 3 initially

for i=1:N
  pi(i,1)=rand(1);
end
pi=pi/sum(pi);  % normalize pi

% for a left-to-right model
%pi=zeros(N,1);
%pi(1)=1;

%-----------------------------

for i=1:N
  alpha1(i,1)=pi(i)*B(i,O(1));
end
C1=1/sum(alpha1);
alpha1(:,1)=alpha1(:,1).*C1;
```

```
% Find alpha2 -> alphaT
for t=2:T
  for j=1:N
    Atrans=A(:,j)';
    eval(['alpha' num2str(t) '(j,1)=(Atrans*alpha' num2str(t-1) ')*B(j,O(t));'])
    clear Atrans,
  end
  eval(['C' num2str(t) '=1/sum(alpha' num2str(t) ');'])
  eval(['alpha' num2str(t) '(:,1)=alpha' num2str(t) '(:,1).*C' num2str(t) ';'])
end

for t=1:T
  eval(['CC(' num2str(t) ')=C' num2str(t) ';'])
end

for i=1:N
  eval(['beta' num2str(T) '(i,1)=0;'])
end
eval(['beta' num2str(T) '(N,1)=1;'])

eval(['beta' num2str(T) '(:,1)=beta' num2str(T) '(:,1).*C' num2str(T) ';'])

% find betaT-1 -> beta1
for t=T-1:-1:1
  for i=1:N
    Btrans=B(:,O(t+1))';
    beta_trans=eval(['beta' num2str(t+1)])';
    eval(['beta' num2str(t) '(i,1)=sum(A(i,:).*Btrans.*beta_trans);'])
    clear Btrans beta_trans,
  end
  eval(['beta' num2str(t) '(:,1)=beta' num2str(t) '(:,1).*C' num2str(t) ';'])
end

% calculate probability
P=inv(prod(CC));

for t=1:T
  eval(['ALPHA(:,' num2str(t) ')=alpha' num2str(t) ';'])
end

for t=1:T
  eval(['BETA(:,' num2str(t) ')=beta' num2str(t) ';'])
end
```

```
A_num=zeros(N);
A_den=zeros(N);
for i=1:N
  for j=1:N
    for t=1:T-1
      A_num(i,j)=A_num(i,j)+ALPHA(i,t)*A(i,j)*B(j,O(t+1))*BETA(j,t+1);
      A_den(i,j)=A_den(i,j)+ALPHA(i,t)*BETA(i,t);
    end
    A_num(i,j)=A_num(i,j)*inv(P);
    A_den(i,j)=A_den(i,j)*inv(P);
  end
end

for i=1:N
  for j=1:Q
    if length(find(O==j)) >= 1
      B_num(i,j)=sum(ALPHA(i,find(O(1:T)==j)).*BETA(i,find(O(1:T)==j)));
    else
      B_num(i,j) = .0001;
    end
    B_den(i,j)=sum(ALPHA(i,1:T).*BETA(i,1:T));
    B_num(i,j)=B_num(i,j)*inv(P);
    B_den(i,j)=B_den(i,j)*inv(P);
  end
end

tmp=0;
for i=1:N
  for j=1:N
    tmp=tmp+ALPHA(i,1)*A(i,j)*B(j,O(2))*BETA(j,2);
  end
end

eta1=zeros(N);
for i=1:N
  for j=1:N
    eta1(i,j)=(ALPHA(i,1)*A(i,j)*B(j,O(2))*BETA(j,2))/tmp;
  end
end

for i=1:N
  pi_n(i)=sum(eta1(i,:));
end
```

```
pi_n=pi_n';
```

HMMTRAIN.M

```
function
[A1,B1,pi1,A2,B2,pi2,A3,B3,pi3,A4,B4,pi4,A5,B5,pi5,A6,B6,pi6]=hmmtrain(N,Q)
%N must match N in hmmreest.m

% training for gesture 1
A1_N=zeros(N);
A1_D=zeros(N);
B1_N=zeros(N,Q);
B1_D=zeros(N,Q);
for i=1:100
  fprintf('g1\t%d\n',i)
  eval(['load o4f1_' num2str(i) ';'])
  eval(['[A_n A_d B_n B_d pi_n]=hmmreest(o4f1_' num2str(i) ',N,Q);'])
  A1_N=A1_N+A_n;
  A1_D=A1_D+A_d;
  B1_N=B1_N+B_n;
  B1_D=B1_D+B_d;
  PI1(:,i)=pi_n;
end
A1=A1_N./A1_D;
B1=B1_N./B1_D;
for i=1:N
  pi1(i,1)=sum(PI1(1,:));
end
pi1=pi1/sum(pi1);
% training for gesture 2
A2_N=zeros(N);
A2_D=zeros(N);
B2_N=zeros(N,Q);
B2_D=zeros(N,Q);
for i=1:100
  fprintf('g2\t%d\n',i)
  eval(['load o4f2_' num2str(i) ';'])
  eval(['[A_n A_d B_n B_d pi_n]=hmmreest(o4f2_' num2str(i) ',N,Q);'])
  A2_N=A2_N+A_n;
  A2_D=A2_D+A_d;
  B2_N=B2_N+B_n;
  B2_D=B2_D+B_d;
  PI2(:,i)=pi_n;
```

```
end
for i=1:N
  pi2(i,1)=sum(PI2(1,:));
end
pi2=pi2/sum(pi2);
A2=A2_N./A2_D;
B2=B2_N./B2_D;

% training for gesture 3
A3_N=zeros(N);
A3_D=zeros(N);
B3_N=zeros(N,Q);
B3_D=zeros(N,Q);
for i=1:100
  fprintf('g3\t%d\n',i)
  eval(['load o4f3_' num2str(i) ';'])
  eval(['[A_n A_d B_n B_d pi_n]=hmmreest(o4f3_' num2str(i) ',N,Q);'])
  A3_N=A3_N+A_n;
  A3_D=A3_D+A_d;
  B3_N=B3_N+B_n;
  B3_D=B3_D+B_d;
  PI3(:,i)=pi_n;
end
for i=1:N
  pi3(i,1)=sum(PI3(1,:));
end
pi3=pi3/sum(pi3);
A3=A3_N./A3_D;
B3=B3_N./B3_D;

% training for gesture 4
A4_N=zeros(N);
A4_D=zeros(N);
B4_N=zeros(N,Q);
B4_D=zeros(N,Q);
for i=1:100
  fprintf('g4\t%d\n',i)
  eval(['load o4f4_' num2str(i) ';'])
  eval(['[A_n A_d B_n B_d pi_n]=hmmreest(o4f4_' num2str(i) ',N,Q);'])
  A4_N=A4_N+A_n;
  A4_D=A4_D+A_d;
  B4_N=B4_N+B_n;
  B4_D=B4_D+B_d;
```

```
  PI4(:,i)=pi_n;
end
for i=1:N
  pi4(i,1)=sum(PI4(1,:));
end
pi4=pi4/sum(pi4);
A4=A4_N./A4_D;
B4=B4_N./B4_D;

% training for gesture 5
A5_N=zeros(N);
A5_D=zeros(N);
B5_N=zeros(N,Q);
B5_D=zeros(N,Q);
for i=1:100
  fprintf('g5\t%d\n',i)
  eval(['load o4f5_' num2str(i) ';'])
  eval(['[A_n A_d B_n B_d pi_n]=hmmreest(o4f5_' num2str(i) ',N,Q);'])
  A5_N=A5_N+A_n;
  A5_D=A5_D+A_d;
  B5_N=B5_N+B_n;
  B5_D=B5_D+B_d;
  PI5(:,i)=pi_n;
end
for i=1:N
  pi5(i,1)=sum(PI5(1,:));
end
pi5=pi5/sum(pi5);
A5=A5_N./A5_D;
B5=B5_N./B5_D;

% training for gesture6
A6_N=zeros(N);
A6_D=zeros(N);
B6_N=zeros(N,Q);
B6_D=zeros(N,Q);
for i=1:100
  fprintf('g6\t%d\n',i)
  eval(['load o4f6_' num2str(i) ';'])
  eval(['[A_n A_d B_n B_d pi_n]=hmmreest(o4f6_' num2str(i) ',N,Q);'])
  A6_N=A6_N+A_n;
  A6_D=A6_D+A_d;
  B6_N=B6_N+B_n;
```

```
  B6_D=B6_D+B_d;
  PI6(:,i)=pi_n;
end
for i=1:N
  pi6(i,1)=sum(PI6(1,:));
end
pi6=pi6/sum(pi6);
A6=A6_N./A6_D;
B6=B6_N./B6_D;
```

HMMRECOG.M

```
function
C=hmmrecog(A1,B1,pi1,A2,B2,pi2,A3,B3,pi3,A4,B4,pi4,A5,B5,pi5,A6,B6,pi6)

C=zeros(6);
% Recognition for gest1
for i=1:100
  eval(['load o4f1_' num2str(i) ';'])
  eval(['P1=hmmrec(o4f1_' num2str(i) ',A1, B1, pi1);'])
  eval(['P2=hmmrec(o4f1_' num2str(i) ',A2, B2, pi2);'])
  eval(['P3=hmmrec(o4f1_' num2str(i) ',A3, B3, pi3);'])
  eval(['P4=hmmrec(o4f1_' num2str(i) ',A4, B4, pi4);'])
  eval(['P5=hmmrec(o4f1_' num2str(i) ',A5, B5, pi5);'])
  eval(['P6=hmmrec(o4f1_' num2str(i) ',A6, B6, pi6);'])

  S=[P1;P2;P3;P4;P5;P6];
  [Y I]=sort(S);
  if I(6)==1
    C(1,1)=C(1,1)+1;
  elseif I(6)==2
    C(1,2)=C(1,2)+1;
  elseif I(6)==3
    C(1,3)=C(1,3)+1;
  elseif I(6)==4
    C(1,4)=C(1,4)+1;
  elseif I(6)==5
    C(1,5)=C(1,5)+1;
  elseif I(6)==6
    C(1,6)=C(1,6)+1;
  end
end
```

```
% Recognition for gest2
for i=1:100
  eval(['load o4f2_' num2str(i) ';'])
  eval(['P1=hmmrec(o4f2_' num2str(i) ',A1, B1, pi1);'])
  eval(['P2=hmmrec(o4f2_' num2str(i) ',A2, B2, pi2);'])
  eval(['P3=hmmrec(o4f2_' num2str(i) ',A3, B3, pi3);'])
  eval(['P4=hmmrec(o4f2_' num2str(i) ',A4, B4, pi4);'])
  eval(['P5=hmmrec(o4f2_' num2str(i) ',A5, B5, pi5);'])
  eval(['P6=hmmrec(o4f2_' num2str(i) ',A6, B6, pi6);'])

  S=[P1;P2;P3;P4;P5;P6];
  [Y I]=sort(S);
  if I(6)==1
    C(2,1)=C(2,1)+1;
  elseif I(6)==2
    C(2,2)=C(2,2)+1;
  elseif I(6)==3
    C(2,3)=C(2,3)+1;
  elseif I(6)==4
    C(2,4)=C(2,4)+1;
  elseif I(6)==5
    C(2,5)=C(2,5)+1;
  elseif I(6)==6
    C(2,6)=C(2,6)+1;
  end
end

% Recognition for gest3
for i=1:100
  eval(['load o4f3_' num2str(i) ';'])
  eval(['P1=hmmrec(o4f3_' num2str(i) ',A1, B1, pi1);'])
  eval(['P2=hmmrec(o4f3_' num2str(i) ',A2, B2, pi2);'])
  eval(['P3=hmmrec(o4f3_' num2str(i) ',A3, B3, pi3);'])
  eval(['P4=hmmrec(o4f3_' num2str(i) ',A4, B4, pi4);'])
  eval(['P5=hmmrec(o4f3_' num2str(i) ',A5, B5, pi5);'])
  eval(['P6=hmmrec(o4f3_' num2str(i) ',A6, B6, pi6);'])

  S=[P1;P2;P3;P4;P5;P6];
  [Y I]=sort(S);
  if I(6)==1
    C(3,1)=C(3,1)+1;
  elseif I(6)==2
    C(3,2)=C(3,2)+1;
```

```
  elseif I(6)==3
    C(3,3)=C(3,3)+1;
  elseif I(6)==4
    C(3,4)=C(3,4)+1;
  elseif I(6)==5
    C(3,5)=C(3,5)+1;
  elseif I(6)==6
    C(3,6)=C(3,6)+1;
  end
end

% Recognition for gest4
for i=1:100
  eval(['load o4f4_' num2str(i) ';'])
  eval(['P1=hmmrec(o4f4_' num2str(i) ',A1, B1, pi1);'])
  eval(['P2=hmmrec(o4f4_' num2str(i) ',A2, B2, pi2);'])
  eval(['P3=hmmrec(o4f4_' num2str(i) ',A3, B3, pi3);'])
  eval(['P4=hmmrec(o4f4_' num2str(i) ',A4, B4, pi4);'])
  eval(['P5=hmmrec(o4f4_' num2str(i) ',A5, B5, pi5);'])
  eval(['P6=hmmrec(o4f4_' num2str(i) ',A6, B6, pi6);'])

  S=[P1;P2;P3;P4;P5;P6];
  [Y I]=sort(S);
  if I(6)==1
    C(4,1)=C(4,1)+1;
  elseif I(6)==2
    C(4,2)=C(4,2)+1;
  elseif I(6)==3
    C(4,3)=C(4,3)+1;
  elseif I(6)==4
    C(4,4)=C(4,4)+1;
  elseif I(6)==5
    C(4,5)=C(4,5)+1;
  elseif I(6)==6
    C(4,6)=C(4,6)+1;
  end
end

% Recognition for gest5
for i=1:100
  eval(['load o4f5_' num2str(i) ';'])
  eval(['P1=hmmrec(o4f5_' num2str(i) ',A1, B1, pi1);'])
  eval(['P2=hmmrec(o4f5_' num2str(i) ',A2, B2, pi2);'])
```

```
  eval(['P3=hmmrec(o4f5_' num2str(i) ',A3, B3, pi3);'])
  eval(['P4=hmmrec(o4f5_' num2str(i) ',A4, B4, pi4);'])
  eval(['P5=hmmrec(o4f5_' num2str(i) ',A5, B5, pi5);'])
  eval(['P6=hmmrec(o4f5_' num2str(i) ',A6, B6, pi6);'])

  S=[P1;P2;P3;P4;P5;P6];
  [Y I]=sort(S);
  if I(6)==1
    C(5,1)=C(5,1)+1;
  elseif I(6)==2
    C(5,2)=C(5,2)+1;
  elseif I(6)==3
    C(5,3)=C(5,3)+1;
  elseif I(6)==4
    C(5,4)=C(5,4)+1;
  elseif I(6)==5
    C(5,5)=C(5,5)+1;
  elseif I(6)==6
    C(5,6)=C(5,6)+1;
  end
end

% Recognition for gest6
for i=1:100
  eval(['load o4f6_' num2str(i) ';'])
  eval(['P1=hmmrec(o4f6_' num2str(i) ',A1, B1, pi1);'])
  eval(['P2=hmmrec(o4f6_' num2str(i) ',A2, B2, pi2);'])
  eval(['P3=hmmrec(o4f6_' num2str(i) ',A3, B3, pi3);'])
  eval(['P4=hmmrec(o4f6_' num2str(i) ',A4, B4, pi4);'])
  eval(['P5=hmmrec(o4f6_' num2str(i) ',A5, B5, pi5);'])
  eval(['P6=hmmrec(o4f6_' num2str(i) ',A6, B6, pi6);'])

  S=[P1;P2;P3;P4;P5;P6];
  [Y I]=sort(S);
  if I(6)==1
    C(6,1)=C(6,1)+1;
  elseif I(6)==2
    C(6,2)=C(6,2)+1;
  elseif I(6)==3
    C(6,3)=C(6,3)+1;
  elseif I(6)==4
    C(6,4)=C(6,4)+1;
  elseif I(6)==5
```

```
    C(6,5)=C(6,5)+1;
  elseif I(6)==6
    C(6,6)=C(6,6)+1;
  end
end


% Display the confusion matrix
fprintf('\n Confusion Matrix \n\n')
fprintf('%d\t%d\t%d\t%d\t%d\t%d\n',C(1,1),C(1,2),C(1,3),C(1,4),C(1,5),C(1,6));
fprintf('%d\t%d\t%d\t%d\t%d\t%d\n',C(2,1),C(2,2),C(2,3),C(2,4),C(2,5),C(2,6));
fprintf('%d\t%d\t%d\t%d\t%d\t%d\n',C(3,1),C(3,2),C(3,3),C(3,4),C(3,5),C(3,6));
fprintf('%d\t%d\t%d\t%d\t%d\t%d\n',C(4,1),C(4,2),C(4,3),C(4,4),C(4,5),C(4,6));
fprintf('%d\t%d\t%d\t%d\t%d\t%d\n',C(5,1),C(5,2),C(5,3),C(5,4),C(5,5),C(5,6));
fprintf('%d\t%d\t%d\t%d\t%d\t%d\n',C(6,1),C(6,2),C(6,3),C(6,4),C(6,5),C(6,6));
```

HMMREC.M

```
function [P]=hmmrec(O, A, B, pi)

[N Q]=size(B);
T=length(O); % Length of observation vector O


% Based on the Model M1 find P(O|M1)

for i=1:N
  alpha1(i,1)=pi(i)*B(i,O(1));
end
C1=1/sum(alpha1);
alpha1(:,1)=alpha1(:,1).*C1;

% Find alpha2 -> alphaT
for t=2:T
  for j=1:N
    Atrans=A(:,j)';
    eval(['alpha' num2str(t) '(j,1)=(Atrans*alpha' num2str(t-1) ')*B(j,O(t));'])
    clear Atrans,
  end
  eval(['C' num2str(t) '=1/sum(alpha' num2str(t) ');'])
  eval(['alpha' num2str(t) '(:,1)=alpha' num2str(t) '(:,1).*C' num2str(t) ';'])
end
```

```
for t=1:T
  eval(['CC(' num2str(t) ')=C' num2str(t) ';'])
end

% calculate probability
P=inv(prod(CC));
```

WINDOW.M

```
function pass=window(file)
   % Function which extracts features from a file by windowing;
   % data is not necessarily segmented. The resulting file
   % contains a stream of features which can then be segmented or
   % processed by a vector quantizer. This function calls the feature
   % extraction function feat.m.

   [rows,columns]=size(file);
   N=fix(rows/8);  % Window size
%   N=rows;  % For feature extraction of whole gestures
   count1=0;  % count the number of windows
   for i=1: N: rows-N  % loop through for all data N at a time
      count1=count1+1;
      pass(count1,:)=feat(file(i:i+N,:));
   end
```

Bibliography

Bengio, Y., LeCun, Y., Nohl, C. and Burges, C. (1995). "A NN/HMM Hybrid for On-Line Handwriting Recohnition", *Neural Computation*, Vol. 7, no. 5.

Cho, T-H., Connors, R.W. and Araman, P.A. (1991). "A Comparison of Rule-Based, K-Nearest Neighbor, and Neural Network Classifiers for Automated Industrial Inspection", *IEEE/ACM International Conference on Development and Managing Expert System Programs*, Washington, DC., pp. 202-209.

Cox, S.J., "Hidden Markov Models for Automatic Speech Recognition: Theory and Application", *The Bell Systems Technical Journal*, Vol. 6, no. 2, pp. 105-115.

Fukunaga, K. (1990). *Introduction to Statistical Pattern Recognition*, Academic Press, Inc., San Diego, CA., 591 p.

Gersho, A. (1982). "On the Structure of Vector Quantizers", *IEEE Transactions in Information Theory*, Vol. IT-28, no. 2, pp. 157-166.

Kharin, Y. (1992). *Robustness in Statistical Pattern Recognition*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 302 p.

Linde, Y., Buzo, A. and Gray, R.M. (1980). "An Algorithm for Vector Quantizer Design", *IEEE Transactions on Communications*, Vol. COM-28, no. 1, pp. 84-95.

Malaviya, A., Leja, C. and Peters, L. (1996). "Multi-Script Handwritten Recognition with FOHDEL", *Proceedings of the Biennial Conference of North American Information Processing Society (NAFIPS'96)*, Berkeley, pp. 147-151.

Nadler, M. and Smith, E.P. (1993). *Pattern Recognition Engineering*, John Wiley & Sons, Inc., New York, NY., 588 p.

Rabiner, L.R. and Juang, B.H. (1986). "An Introduction to Hidden Markov Models", *IEEE ASSP Magazine*, pp. 4-16.

Raudys, S. and Jain, A.K. (1991). "Small Sample Size Effects in Statistical Pattern Recognition: Recommendations for Practitioners" *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 13, no. 3, pp.252-264.

Ripley, B.D. (1996). *Pattern Recognition and Neural Networks*, Press Syndicate of the University of Cambridge, Cambridge, 403 p.

Schalkoff, R. (1992). *Pattern Recognition: Statistical, Structural and Neural*

Approaches, John Wiley & Sons, Inc., New York, NY., 364 p.

White, D.J. (1993). *Markov Decision Processes*, John Wiley & Sons, Inc., New York, NY., 224 p.

Xu, Y. and Yang, J. (1995). "Towards Human-Robot Coordination: Skill Modeling and Transferring via Hidden Markov Model", *IEEE International Conference on Robotics and Automation*, Vol. 6, pp. 1906-1911.

Yan, H. (1994). "A Comparison of Multi-Layer Neural Networks and Optimized Nearest Neighbor Classifiers for Handwritten Digit Recognition", *International Symposium on Speech, Image Processing and Neural Networks*, Hong Kong, pp. 312-315.

Yang, J., Xu, Y. and Chen, C.S. (1994). "Hidden Markov Model Approach to Skill Learning and Its Application to Telerobotics", *IEEE Transactions on Robotics and Automation*, Vol. 10, no. 5, pp. 621-631.