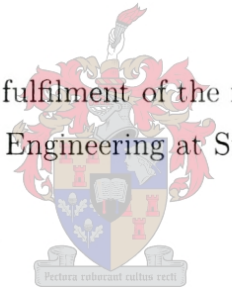
Handwritten Signature Verification Using Hidden Markov Models.

Colin Sindle

Thesis presented in partial fulfilment of the requirements for the degree of Master of Electronic Engineering at Stellenbosch University.



Supervisor: Prof. J.A. du Preez
Co-supervisor: Prof. B.M. Herbst

November, 2003

Declaration

I, the undersigned, do hereby declare that the work contained in this thesis is my own original work, except where otherwise indicated

Abstract

Handwritten signatures are provided extensively to verify identity for all types of transactions and documents. However, they are very rarely actually verified. This is because of the high cost of training and employing enough human operators (who are still fallible) to cope with the demand. They are a very well known, yet under-utilised biometric currently performing far below their potential. We present an on-line/dynamic handwritten signature verification system based on Hidden Markov Models, that far out performs human operators in both accuracy and speed. It uses only the local signature features—sampled from an electronic writing tablet—after some novel preprocessing steps, and is a fully automated system in that there are no parameters that need to be manually fine-tuned for different users. Novel verifiers are investigated which attain best equal error rates of between 2% and 5% for different types of high quality deliberate forgeries, and take a fraction of a second to accept or reject an identity claim on a 700 MHz computer.

Opsomming

Geskrewe handtekeninge word gereeld gebruik om die identiteit van dokumente en transaksies te bevestig. Aangesien dit duur is in terme van menslike hulpbronne, word die integriteit daarvan selde nagegaan. Om handtekeninge deur menslike operateurs te verifieër, is ook feilbaar—100% akkurate identifikasie is onrealisties. Handtekeninge is uiters akkurate en unieke identifikasie patrone wat in die praktyk nie naastenby tot hul volle potensiaal gebruik word nie. In hierdie navorsing gebruik ons verskuilde Markov modelle om dinamiese handtekeningherkenningstelsels te ontwikkel wat, in terme van spoed en akkuraatheid heelwat meer effektief as operateurs is. Die stelsel maak gebruik van slegs lokale handtekening eienskappe (en verwerkings daarvan) soos wat dit verkry word vanaf 'n elektroniese skryftablet. Die stelsel is ten volle outomaties en geen parameters hoef aangepas te word vir verskillende gebruikers nie. 'n Paar tipes nuwe handtekeningverifieërders word ondersoek en die resulterende gelykbreekpunt vir vals-aanvaardings- en vals-verwerpingsfoute lê tussen 2% en 5% vir verskillende tipes hoë kwaliteit vervalsde handtekeninge. Op 'n tipiese 700 MHz verwerker word die identiteit van 'n persoon in minder as $\frac{1}{5}$ sekonde bevestig.

Acknowledgements

I would like to thank:

- My supervisors, Prof.J.A. du Preez and Prof.B.M. Herbst, for their wisdom, guidance and encouragement in this project, as well as the long not-so-academic discussions.
- Dr. J.G.A Dolfing for making his signature database available to Stellenbosch University.
- Dirk 'Surfelis' Wagener for his translational expertise.
- André du Toit for his technical help.
- My family and friends for their support and assistance.
- Louis Cordier for his LaTeX assistance.
- All the other people in the DSP Labs (and Lude and G-J), especially those who kept me working, and those who kept me sane.
- The National Research Foundation for their financial assistance.

Keywords

Handwritten Signature Verification, Hidden Markov Models, Ferguson, Equivalent Ferguson Initialisation, Explicit Time Duration Modelling, Transition Smoothing, Verifiers, On-line, Dynamic, Footprint.

Contents

List of Figures

List of Tables

List of Initialisations, Acronyms, Abbreviations, and Terms

PDF – Probability Density Function

CDF – Cumulative Distribution Function

PMF – Probability Mass Function. The discrete version of a PDF.

HMM – Hidden Markov Model

CBKLT – Class Based Karhunen-Loève Transform

FAV – Frame-As-Vector (transform)

PCA – Principle Component Analysis

GMM – Gaussian Mixture Model

HSV – Handwritten Signature Verification

MLE – Maximum Likelihood Estimation

FA – False Acceptance (mistaking an impostor for a genuine user)

FR – False Rejection (mistaking a genuine user for an impostor)

EE – Equal Error. The probability at which the FA rate equals the FR rate.

USB – Universal Serial Bus.

3D, 5D, etc. – Three dimensional, five dimensional, etc.

pps – points per second

dpi – dots per inch

kB – kilobyte

ms – millisecond

FAR Curve – False Acceptances and False Rejections Curve

DET Curve – Detection Error Trade-off Curve

ROC Curve – Receiver Operating Characteristic Curve.

PatRecII – the in-house pattern recognition library.

User or Subject – An individual who is enrolled in a signature verification system. That is, he/she has submitted a number of signatures (usually 1–15) which have then been used to train a HMM model.

Digitising Tablet/Writing Pad – The electronic pad that samples the pen positions at a fixed rate while the user is signing, and sends this information to the computer.

Normalise – May refer to any preprocessing transformation (e.g. differentiation), as well as classic normalisation.

Self-loop – A HMM state transition link that has the same destination as origin. It is treated the same as any other transition and also has an associated transition probability.

NULL **State** – A pseudo state with no distribution functions occupying no time-steps in the HMM state sequence, but with associated state transition weights. It is very useful for modelling start, end, and branching states and simplifies programming.

TOP **State** – The top state in a duration state stack in duration emphasised HMMs. It is also the only state with a self-loop.

BASE **State** – The final state of a duration state stack. All the original state exiting transitions now go leave from this state.

Chapter 1

Introduction

In the western world, 2.8 million signatures are made every minute [23], most of which are used to verify identity or affirm agreements. Signatures are the accepted proof of identity for cheques, credit card transactions, legal documents and business contracts. Society has endorsed this form of verification and individuals usually have no qualms with providing a signature to authenticate contracts. Having said that these signatures are provided for verification, the fact is that they are not often actually verified. A very small percentage of signatures captured every day are analysed by professional document examiners and forensic scientists to verify their authenticity. For the most part, signatures are not inspected at all, let alone by trained personnel. In fact most South African banks do not even check the signatures at all on cheques of less than R5000. It is just not worth it to manually verify all signatures, and even well trained personnel will not detect all forgeries.

Nevertheless, signing to prove identity remains popular because of its convenience. We are not required to be in possession of tokens, like keys or magnetic/smart cards, or knowledge, like codes or passwords. It is also an active process, which is a prerequisite in legal situations where it is necessary to confirm a contract with a conscious action. Passive biometrics such as iris and face recognition do not enjoy this advantage.

Biometrics refers to the automatic recognition of a person based on his/her physiological or behavioural characteristics. While signature verification is unlikely ever to rival the extremely low error rates of iris and retina recognition because of the normal variations between an individual's signatures, it does have one huge advantage, that of wide accep-

tance in Western society[1]. Signature verification is also often chosen above fingerprint recognition because of the criminal connotations that this biometric evokes.

The terms identification and verification are often used interchangeably though they refer to two very different modes of user validation. Whereas identification, or recognition, attempts to select the most likely user from an enrolled group, verification must decide whether a presented user is who he/she claims to be. In identification trials, the data is matched to the enrolled models until certain criteria are fulfilled, or the highest score, or lowest distance measure is chosen.

Signatures are an integral and accepted part of our lifestyle and, as such, their use is not likely to decrease for some time. Their use is entrenched in our financial and legal systems, and indeed our culture. However, currently they are not the most reliable biometric, with biometrics like iris and retina recognition taking those places. The error rates of signatures are comparable to those of fingerprints, mainly because of the difficulty of capturing perfect fingerprints. Dynamic signature verification enjoys a clear advantage over other biometrics in this regard—signatures are captured almost perfectly, whereas obtaining a flawless fingerprint, iris image, or retina scan, is a formidable task.

In this thesis we present a dynamic Handwritten Signature Verification (HSV) system that captures signatures while they are being written. With the extra dynamic information available of exactly how the signature was created, we develop a system which attains an Equal Error (EE) rate (the error rate at which the false acceptance and false rejection plots intersect) of below 5% for even very good forgeries. Dynamic HSV has recently matured and commercial products are being installed at various multi-national companies (e.g. Ford [13], Nationwide Building Societies [23]) around the world. It looks set to become one of the more popular biometrics in the next few years.

1.1 Background

Dynamic or on-line signature verification requires a method of capturing a signature while it is being written. This is usually achieved with an electronic digitising tablet that time samples the position of the pen strokes as the user is signing and relays these data points back to a computer. In contrast, static or off-line signature verification normally uses a scanned-in image of a signature from a normal paper document to perform verification.

[1]There is less importance associated with signatures in the East where one's name is simply written out when a signature is required.

It is easier to forge or trace the spatial image of a signature than the underlying dynamics and, as such, an off-line system will not usually achieve as low error rates as an on-line system. Off-line systems have higher equal error rates of typically 10% and 23% for zero-effort and skilled forgeries respectively [8].

Dynamic systems usually make use of more than just the time sampled x and y-coordinates. They often have access to the pressure exerted by the pen on the writing surface, as well as one or two tilt angles that describe the orientation of the pen. Thus the data points or raw feature vectors can contain up to five dimensions (x-coordinate, y-coordinate, pressure, and one or two tilt angles). These sampled data points are often known as local features because they describe the properties of a signature at a specific time and place. In contrast, global features, as their name suggests, describe properties relating to the entire signature. These include size, distance of pen movement, and enclosed area. These features are not used in this research as they are usually used in off-line systems, though a fusion of the two systems could prove advantageous.

Handwritten signature verification (HSV) systems usually require the user to submit between one and fifteen signatures for enrolment. These are used as reference signatures or to train models that will provide confidence values for verification attempts. When an individual presents a signature and a claim of identity to the system, this confidence value determines whether his access request will be accepted or rejected depending on whether it is above or below a specified threshold.

In the last decade, Hidden Markov Models (HMMs) have been used in more and more varied applications and have recently entered the field of HSV. An HMM is a statistical model that models with discrete states, the phases a signal (in this case a signature) goes through as time progresses. It also describes the probabilities of changing from one state to another. At each time-step another probability is produced that shows how well the current feature vector fits a probability density function (PDF) associated with each state. The product of all these probabilities and the state transition probabilities tells how well the presented signature data matches the model. This probability is often given as a log likelihood score. A verifier then uses this score and possibly other *a priori* and *posteriori* information to arrive at a confidence value that states how confident it is that the presented signature belongs to the claimed identity. From this, the individual will either be accepted or rejected.

1.2 Objectives

The objectives of this study are to:

- optimise the structure and components of the users' HMM signature models.
- determine a set of preprocessing transformations to extract the most useful information from the raw signature feature vectors.
- investigate statistically sound methods of verification.
- keep the size (footprint) of a users' signature model as small as possible.
- implement a signature verification scheme using portable code that has an error rate comparable to the best systems in the world.

1.3 Contributions

The accomplishments of this study are as follows:

- A modified Ferguson duration emphasised Hidden Markov Model was designed that can be initialised to behave identically to the trained model it is expanded from. It also permits elegant statistical methods of transition probability smoothing using various one-dimensional probability density functions. This considerably reduces training data scarcity problems.
- Various methods were used to preprocess the signature data. Most successful of these included framing the data, then using a class based Karhunen-Loève transform to select the dimensions of greatest inter-class variance. In the end, the best results were obtained using a combination of four different types of preprocessing transformations.
- Four statistically tractable verifiers were investigated. The most suitable one for our purposes was the novel C-Norm verifier. This verifier can reach an accept/reject decision for an average signature in 160 ms on a 700MHz personal computer running Linux.
- The size of the complex model representing a person's signature was brought down from a text size of 50kB, to a more convenient size of under 4kB that can be transferred from a cheap smart card in less than half a second.

- All the statistical modelling components necessary for a high quality signature verification system have been incorporated into PatRecII[2], a fully portable, flexible library, from which a final application can be built. These experiments were also a good test of the library's pattern recognition robustness since many of the existing functions have only ever been used before in speech trials.

1.4 Thesis Review

This thesis describes the components necessary for a dynamic handwritten signature verification system that uses only the local features of a signature. The person requiring access to the HSV protected resource will need to make an identity claim, and then verify this with a signature on a writing pad connected to a computer. The electronic writing tablet produces 5-dimensional data points at a constant rate while the individual is signing. This dynamic signature information allows signatures to be verified much more accurately than is possible with traditional static spatial image comparisons.

The set of raw feature vectors that makes up one signature, often goes through a number of preprocessing filters before being used for training of the models or verification. These preprocessing filters ensure that the more useful signature characteristics are modelled, and that the numerical errors of digital computers are minimised. Many different types and combinations of preprocessors were investigated that can be divided into two categories: normalisers and transformers.

The normalisers guarantee that the signatures have a constant size, orientation, and position around the XY origin, as well as scaling the other dimensions to a unity variance to remove numerical range difficulties.

The preprocessing transformations include differentiators, that calculate velocities and accelerations from the position dimensions using various methods, as well as the non-linear Volterra expansion. This expansion produces a series of all the possible combinations of product terms of the five dimensions up to a specified order. Another two transforms that work particularly well together, are the frame-as-vector transform (that groups many adjacent data points to form a single higher dimensional feature vector) and the class based Karhunen-Loève transform (CBKLT). The CBKLT is a data decorrelating transform that determines the axes of maximum inter-class variance (see Section 3.6). This allows the

[2]PatRecII is being written by Professor J.du Preez and the postgraduate students in the Digital Signal Processing group at Stellenbosch University.

important correlations between neighbouring points to be modelled effectively, and results is a significant fall in error rates. The preprocessors are described in detail in Chapter 3.

All the enrolled users in an HSV system have a corresponding model to which signatures are applied to verify identity. We use hidden Markov models (HMMs) to perform this task. HMMs are a logical choice for modelling signatures because signatures can easily be divided into their constituent strokes which can be modelled well by HMM states. Of course, strict stroke boundaries are not enforced (hard segmentation), rather the training algorithms are allowed to determine these boundaries while optimising the model (soft segmentation).

The choice of HMM topology is vital to the success of the application. Many different topologies are possible because it is not normally required for each state to have a link to every other state. Allowing only certain transition probabilities to be non-zero, specifies the behaviour of the HMM. A left-to-right configuration was chosen to model signatures. This formation can be imagined as a number of states forming a linear chain. The state sequence is only allowed to move in one direction from the start ('leftmost') state to the finish ('rightmost') state, staying for as long as necessary in each state, before jumping 'right' one or more states, until the finish is reached. In fact, we determined that allowing only one or two forward state jumps (i.e. only to the next and next-but-one states) worked best.

The left-to-right topology needs some further expansion if it is to model signatures accurately. At every time-step, a transition is made, though the link destination may be the same as its origin if that state has a self-loop. This provides implicit time duration modelling for staying in each state (and therefore being associated with a specific state PDF) for more than one time-step. The problem with this is that the chance of staying in any one state is an exponential decay function as time progresses.

This is certainly not optimal when modelling signatures that have relatively constant characteristics for non-zero amounts of time. The solution to this problem is to add explicit duration modelling to the HMM by altering the structure. We achieve this by replacing each state with a self-loop by a twenty state duration formation called a duration stack. All these twenty new states still refer back to the one state PDF of the original state. We settled on the Ferguson method [5] of explicit duration modelling because of some useful characteristics.

Firstly though, we modify the classic Ferguson topology to allow the duration emphasised model to be initialised to be equivalent to the pre-trained left-to-right model it was

expanded from. The modified Ferguson HMM now behaves identically to the initial left-to-right model. This new HMM can then be re-estimated, and allowed to automatically make use of its new time duration modelling capabilities.

The most useful characteristic of the Ferguson model is the ease with which the duration modelling transition probabilities can be smoothed. They need to be smoothed because there is not enough training data (nor will be in most real-world situations) to ensure that the probabilities are estimated correctly and that all likely signature stroke lengths are catered for.

We determine that Gaussian Mixture Models (GMMs) are well suited to this smoothing task and suggest that three underlying Gaussians be used to allow specialisation, yet still smooth sufficiently.

Remaining in Chapter 4, we discuss three different unimodal Gaussian PDFs that can be used as HMM state PDFs. The full-covariance, diagonal, and circular Gaussian PDFs have decreasing modelling abilities, but have a decreasing number of parameters that need estimating. Combating data scarcity (or lack of training data) is a common thread running through this thesis.

Another important aspect that must be considered in commercial HSV systems is the digital size of each user's signature model. It is a relevant question because signature models will surely be used in applications where they must be saved on portable digital media, or transferred over limited bandwidth networks. Current technology uses smart, or magnetic stripe, cards to hold such information, and they have a limited storage capacity. Three-track magnetic stripe cards can store around 200 bytes, and smart cards 64kB, though this is increasing steadily. It is therefore of primary importance to keep the size (footprint) of the stored model down. After discarding the unnecessary information and compressing using the Lempel-Ziv algorithm [17] (as used in programs such as *gzip*, *zip* and *pkzip*), the size of the complex model representing a person's signature was brought down from a text size of 50kB, to a more convenient size of under 4kB. This can be transferred from a cheap smart card in less than half a second.

After all the model components have been discussed, we detail the process of verification. Four types of verifiers are discussed in Chapter 5. Two original verifiers are presented, with the C-Norm (see Section 5.4) proving to to be most accurate in the conducted signature experiments.

The accuracy of trials such as these is traditionally read from the False Accept and False Reject (FAR) curves, but these are shown to be inadequate for objectively compar-

ing verification results. A better understanding of the results can be gained using the relatively new Detection Error Trade-off curves. DET curves plot the false acceptances versus the false rejections on axes that are scaled according to a Gaussian cumulative distribution function (CDF). This ensures that results with Gaussian distributions (as often happens) appear as straight lines on the plot.

Many experiments were performed to determine which preprocessors and HMM configurations achieve the best verification results. These are presented in Chapter 6. Our final system achieves equal error rates of 1.96%, 3.92%, and 5.01% for the different high quality forgery categories, and 2.27% for zero-effort forgeries. An accept/reject decision is made in less than $\frac{1}{5}$ of a second on a 700MHz PC running Linux.

Chapter 2

Background

This chapter provides a review of HSV's state-of-technology, as well as describing some of the key components used in these systems. The first two sections give an overview of two HMM based HSV systems, along with some examples of commercial systems already in operation. Next, the electronic digitising tablet technology is explained and the properties of the database used in these experiments are discussed. This is followed by a short note on our system's implementation environment.

2.1 HMM Based Systems

The first use of Hidden Markov Models for signature verification dates back to Paulik & Mohankrishnan [20] in 1993. Subsequently Yang [27] and Yang, Widaja & Prasad [28], both in 1995, also used HMMs in their verification systems. HMMs have been used for speech recognition tasks since the mid eighties, but it is only relatively recently that their use has become more popular and spread to fields such as signature verification. This is primarily due to the increased computing power available, which now makes it possible to use models with the complexity required for worthwhile accuracies.

2.1.1 Dolfing's Research

Dolfing's [2] high quality verification system does not use the raw (or transformed versions of the raw) data as feature vectors, but rather segments the signature into a number of strokes from which feature vectors are extracted. An averaging filter with a width of 5 samples is used to smooth the raw data and it is then segmented wherever the velocity in the Y direction equals zero, that is $v_y = 0$. The basic six features extracted from each stroke are the velocities, v, and accelerations, a, of the stylus in the spatial domain

(only X and Y dimensions). This baseline feature vector is defined as $Dynamic(6) \equiv (v_{begin}, v_{end}, v_{max}, v_{avg}, a_{max}, a_{min})$, where the subscripts indicate the respective properties of a specific segment. After trying out many combinations of feature dimensions, the following features gave good results:

$$Dynamic(14) \equiv (\ Dynamic(6),\ P_{max},\ P_{min},\ \Delta P_{max},\ \Delta P_{min},\ \theta_{x_{max}},\ \theta_{x_{min}},\ \theta_{y_{max}},\ \theta_{y_{min}}). \tag{2.1}$$

where P is the pressure exerted by the pen on the writing surface, Δ is a differentiating operation, and θ_x and θ_y describe the tilt of the pen from the X and Y axes (see Section 2.3 for a full explanation and diagrams).

This was later replaced by another feature vector with the slightly lower equal error (EE) rates that are shown in Table 2.1:

$$\begin{aligned} Dynamic(13) \equiv (\ & v_{begin},\ v_{end},\ v_{max},\ v_{avg},\ v_{max} - v_{avg},\ a_{max},\ a_{min}, \\ & P_{max},\ P_{min},\ \Delta P_{max} - \Delta P_{min},\ 2 \times \theta_{x_{max}} - \theta_{x_{min}}, \\ & 2 \times \theta_{y_{max}} - \theta_{y_{min}},\ number\ of\ samples). \end{aligned} \tag{2.2}$$

Name	EE	EE-SH	EE-HI
Dynamic(6)	12	15.1	8.6
Dynamic(14)	6.0	7.9	3.9
Dynamic(13)	5.4	-	-
Combined(32) t_{at}	1.9	2.6	1.1

Table 2.1: *Dolfing's Equal Error (EE) rates for Over-the-shoulder (SH) and Home Improved (HI) forgeries using different feature vector combinations and the same database as used in this thesis.*

The *Combined*(32) t_{at} feature vector uses the features of *Dynamic*(13) along with 13 spatial features and six contextual (global features relating to the entire signature) features to make up the 32 dimensions. The t_{at} signifies that an adaptive threshold was used. That is, each individual had a different threshold based on the scores of a validation set of test signatures. This is similar to the C-Norm verifier presented in this research. A unique characteristic of these signature models is that each state in the system has more than one PDF associated with it to allow people with two or more distinct signature shapes to be accommodated. For example, this would be useful if one sometimes crosses

a 't' in mid signature, while at other times crossing it after the entire signature has been completed.

Thus, the *Combined*(32) t_{at} system performs the best with an EE rate of only 1.9% by using a fusion of local (as used in this thesis) and global (more commonly used in offline systems) features.

2.1.2 Le Riche's Research

Le Riche [16] presents a handwritten signature verification system using similar techniques as this research. The feature vectors used in this research are the raw signature data points or linear transformations thereof. That is, no segment based or contextual features were used.

Features used	Lowest Error Rates (lowest combined FA and FR)
$X, Y, P, \theta_x, \theta_y$	0.5%
X, Y, P	6.3%
X, Y	10.5%
P, θ_x, θ_y	7.3%

Table 2.2: *Le Riche's accuracy rates using the Dolfing database. The accept/reject threshold for each user has been chosen manually to best distinguish genuine signatures from forgeries.*

Table 2.2 shows the results obtained using the Dolfing database. Le Riche further reports the FR rate for a FA rate of 0%, to be 6.6%, and alternately, the FA rate for a FR rate of 0%, is 2.75%. These results are all achieved using personal thresholds tuned manually to minimise error rates. Though it is possible to tweak individual thresholds in real world situation, it is unlikely that example forgeries will be available to set this optimally. No mention is made here of whether separate training forgeries were used to determine the optimal threshold, or whether the same forgeries used to calculate the threshold were then re-used to test the system.

Further work was done with an in-house database of 500 signatures with an equal number of forgeries and genuine signatures. The important fact here, is that these signatures were collected over a 12 month period, which allows the effect of natural signature developments to be observed. The Dolfing database seems to have been collected over a short period of time, since there is less variance between Dolfing signatures than those in

le Riche's database. With this database, a maximum accuracy of 92.2% (an error rate of 7.8%) is achieved.

2.2 Commercial Systems

Signature verification systems have been deployed for a few years now. Some of the available systems are mentioned below. The respective companies are reluctant to divulge any information as to the inner workings of their software. The list is by no means exhaustive, but does mention some applications and shows the acceptance of signature verification.

2.2.1 The MotionTouch System

It was announced in January 2003 [23] that Nationwide Building Society has decided to use electronic signature pads rather than iris recognition for their 681 branches across the United Kingdom. After conducting a high profile trial of iris recognition technology, they concluded that "it was too expensive and couldn't provide a close enough tie between identification and transaction." Signature verification on the other hand requires the customers physically sign rather than just present themselves to a camera. They also rejected fingerprint recognition because of the unease customers felt about their finger impressions being stored in a database.

When this technology, supplied by the company MotionTouch [11], is combined with integrity testing, time stamping and unique device identities, the electronic signatures are considered legally binding under the European Digital Signature Directive and the UK Electronic Communications Bill. Their system uses the following loosely defined variables: speed through the letters, rhythm, direction, flow, pressure, and the graphical representation. Over 1400 of their Legapad (legally-compliant) electronic tablets will be deployed. These pads have a sampling rate of 200 Hz and a paper-feel surface to ensure that accurate natural signatures are recorded. If disputes arise, these signatures can be compared to pen on paper signature samples by forensic examiners in a court of law.

This project will be the largest biometrics installation for public use in the UK. The main incentive of this system is not to curb fraud, but rather to reduce paperwork and avoid chores such as faxing copies of signatures between branches. However, MotionTouch does state these electronic signatures provide a "greater assurance on the origin of the signature than established paper documents do" and that there were no false accepts or

rejects during trials.

2.2.2 The Cyber-Sign System

Cyber-Sign Biometric Signature Verification System [10] analyses the shape, speed, stroke order, pen-up motion, pen pressure and timing information captured during the act of signing. From this an electronic signature is created that declares the validity of the data it is appended to. They hope that this technology will be used for user authentication on electronic documents wherever a written signature would have been needed on normal documents. Information is not freely available concerning the internal algorithms.

2.2.3 The SoftPro System

SignPlus [12] has been performing visual signature verification since 1987, and released the SoftPro system that performs automatic verification in 1994. Their website states successful installations of the system in over 250 banks worldwide with more than 1.5 million cheques verified daily. They extract static verification parameters from the signature such as the enclosed area, inclination of strokes, and compactness. They then use these, the usual dynamic parameters, and many analysis techniques, which compare presented data with reference values, for a combined verification system, stating that "the static verification offers a lower false reject rate (FRR), where other biometric systems that rely only on dynamic parameters have substantial problems, and the dynamic verification minimises the false acceptance rate (FAR)".

2.2.4 The Valyd and Interlink Systems

These partners [13] provide a full range of security services from authenticating electronic documents and data, to securing transactions, networks and databases. This year they received a contract from Ford Credit, a subsidiary of Ford Motor Company, to supply thousands of dealerships across the USA with their ePad and eSign handwritten electronic signature verification system. It provides a method for customers to authenticate themselves around the USA, as well as being a legally binding agreement to the electronic lease contracts. This is said to reduce operations costs and lead to faster document processing times.

2.3 The Digitising Tablet

All the systems mentioned above require hardware to capture the signature along with its dynamic information. This essential device that makes dynamic handwritten signature recognition possible, is the electronic digitising tablet or pad, and pen or stylus. The tablet obtains the position of the pen on its surface and relays this information via the COM or USB ports to the computer. Depending on the nature of the system, the pen may contain electronic circuits or the pad may do all the sensing. These stylus circuits are either powered by an internal battery, or they have a small antenna that picks up low energy RF signals from the tablet, which are then used to power the device. The stylus is then responsible for measuring the pressure exerted by the nib on the pad and transmitting this data back to the tablet to convey to the computer. The pens either have plastic non-marking nibs, or replaceable ink nibs. The ink nibs are preferred for signature recognition, since not being able to see the progress of the signature while signing, has a disconcerting effect and increases the intra-signature variability of a user's signatures.

Figure 2.1: *A Wacom Intuos2 tablet with grip pen and mouse [15].*

These tablets are usually used by artists, graphic designers, and industrial designers, but are eminently suited to handwriting/signature capturing. They are widely available and come in a range of sizes and technologies. To be suitable for this task, the digitiser must be able to report the following five features at a sampling rate of at least 100 points per second:

X-coordinate (X) – The position of the pen nib on the tablet in the horizontal x-

dimension increasing from left to right.

Y-coordinate (Y) – The position of the pen nib on the tablet in the y-dimension increasing from bottom to top.

Pressure (P) – The measured pressure is the axial pen force directed along the barrel of the pen, as supposed to the normal pen force that is defined as the pressure the nib exerts in the direction perpendicular to the pad surface. In the Dolfing database, it is divided linearly into 64 levels over a range of 0 to 200 grams.

X-tilt (θ_x) – The angle from the x-axis to the projection of the pen on the XZ plane. The measured θ_x is divided linearly into 64 levels between $-90°$ and $+90°$.

Y-tilt (θ_y) – The angle from the y-axis to the projection of pen on the YZ plane. The measured θ_y is divided linearly into 64 levels between $-90°$ and $+90°$.

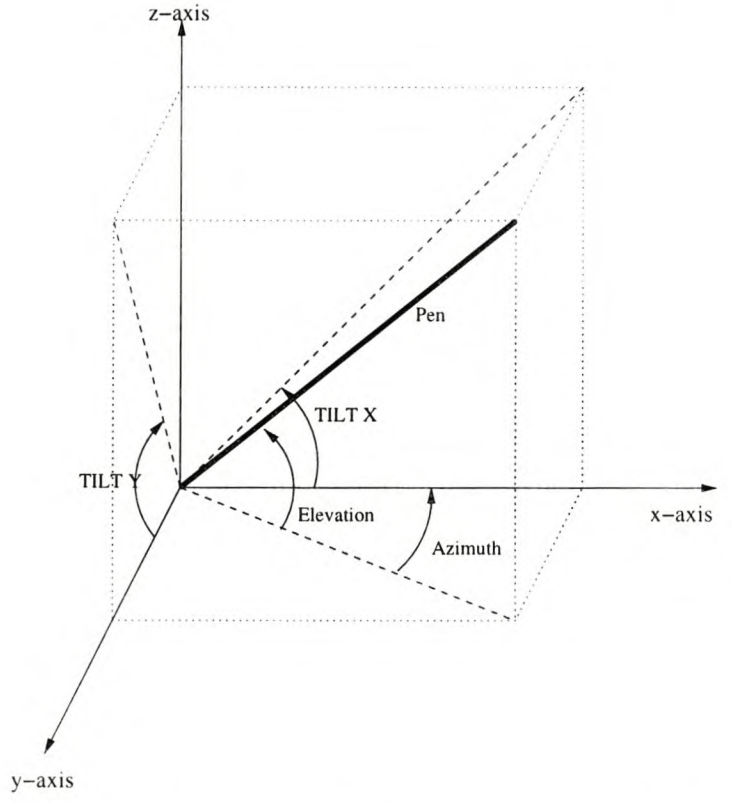


Figure 2.2: *Description of the different tilt angles of the pen.*

The tilt levels were not converted to degrees for the experiments, though they were for Figures 2.3 and 2.4. Some tablets rather report the different pen tilt dimensions of azimuth and elevation. Azimuth is the angle in the XY plane ('compass bearing') that

the pen is held at, and is measured in degrees from the x-axis, increasing in a counter-clockwise direction. Elevation is the Z angle of the stylus, increasing from the pad surface in the positive Z direction. This coordinate system has the disadvantage that there is a discontinuity in the azimuth between 0° and 360°, which needs corrective unwrapping software to rectify for certain signers. Otherwise, these dimensions can easily be converted to θ_x and θ_y with simple trigonometry and cylindrical to Cartesian coordinate system transformations. See Figure 2.2 for a description of the different angle positions.

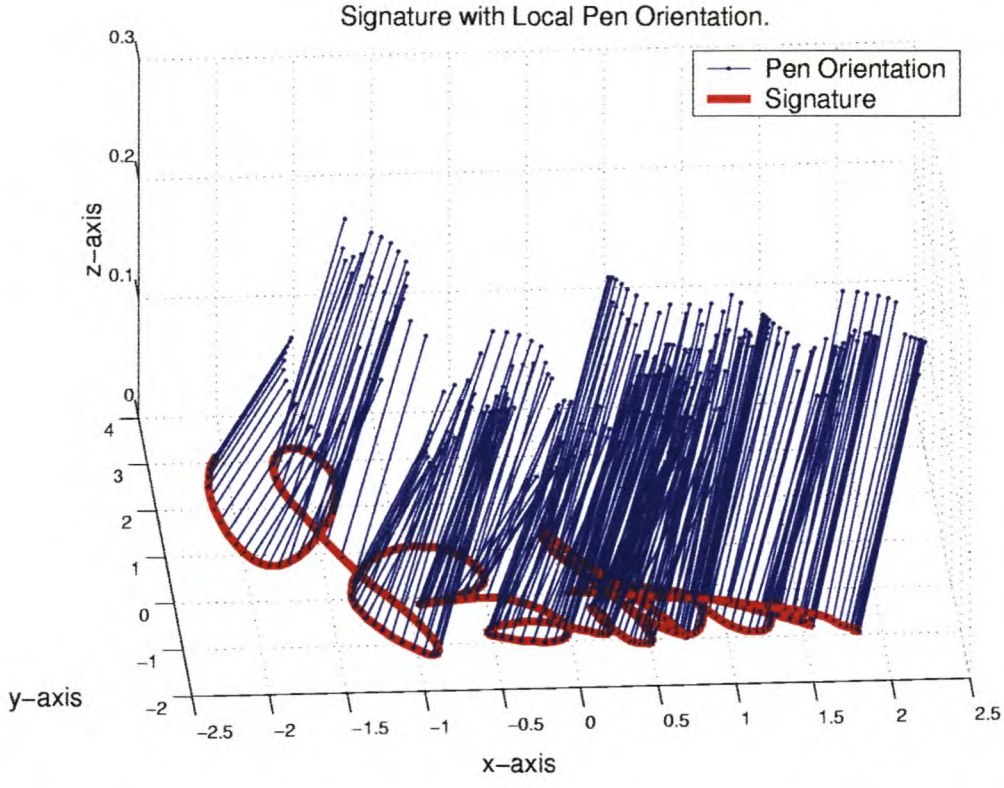


Figure 2.3: *The quiver lines show the orientation of the pen at each sampling point.*

Another subtly useful capability of most tablets is their ability to measure this data while the stylus is slightly above the pad's surface (up to $\approx$ 1cm). This means characteristic air-strokes (say going back to cross a 't' or dot an 'i'), that are invisible to forgers, can form part of the model. Currently, the most widely available tablet that meets (and greatly exceeds) these specifications is the Wacom Intuos2 A6 size pad whose specifications are noted in Table 2.3.

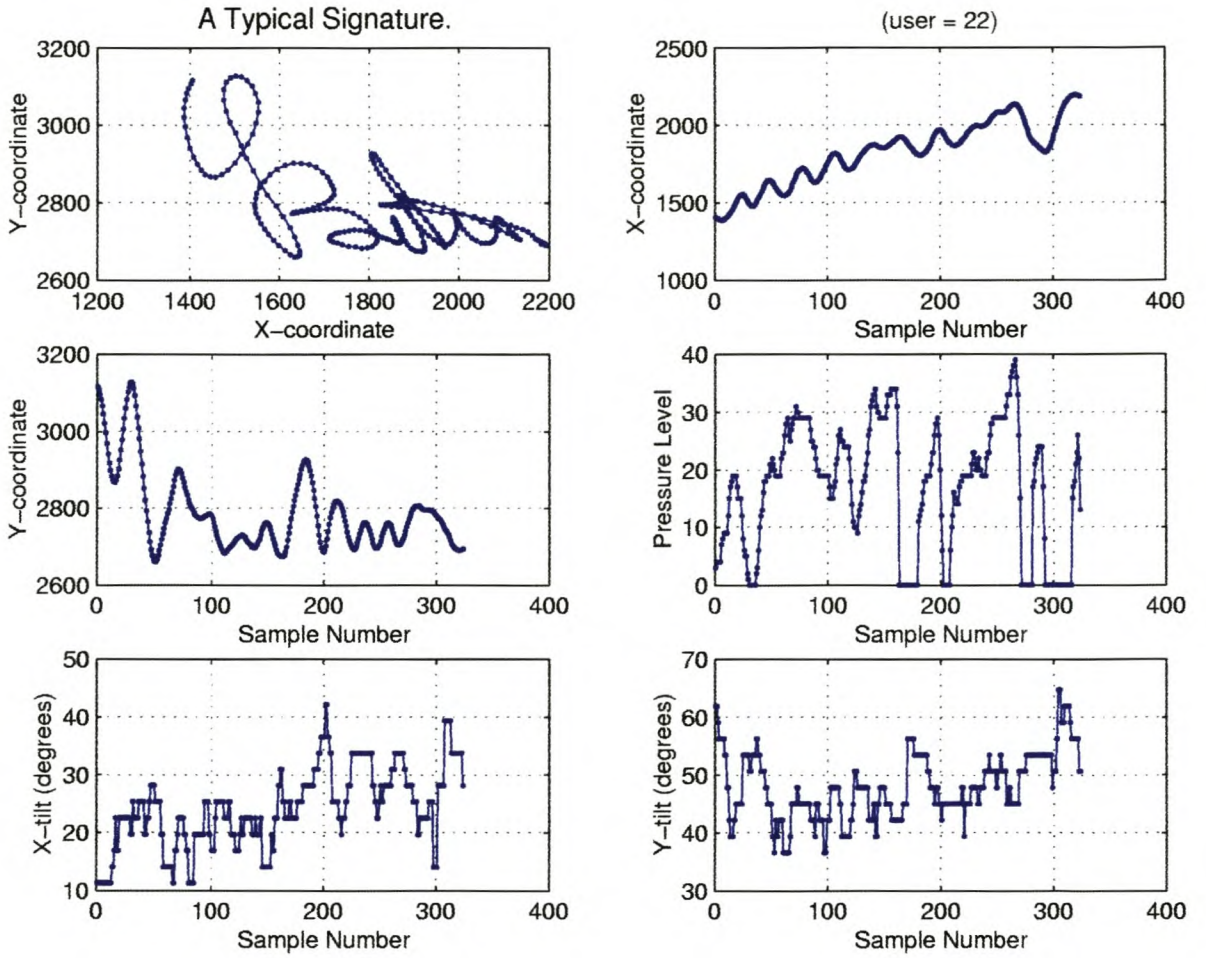


Figure 2.4: *Plots of the 5 dimensions of a typical signature. The X and Y tilts have been converted to degrees.*

	Intuos2	PAID
Active Surface	$127.0 \times 106.0mm$	
Size	$225.0 \times 210.0 \times 75.0mm$	
Weight	$520g$	
Resolution	$10\mu m$	$40\mu m$
Accuracy	$\pm 250\mu m$	$\pm 250\mu m$
Pressure Levels	1024	64
Max. Reading Height	$10mm$	$10mm$
Tilt Angle	$\pm 50°$	$\pm 90°$
Tilt Accuracy Up To 40°	$\pm 2°$	$\pm 2.8°$
Max. Report (Sampling) Rate	$200pps$	$200pps$
Cost	**R3073.00**	

Table 2.3: *Sample electronic digitiser pad specifications for the latest Wacom Intuos2 and the older Philips Advanced Interactive Display (PAID) used to collect Dolfing's signature database.*

2.4 The Dolfing Database

Other fields of research, such as speech recognition, have a few widely known databases (such as the popular TIMIT database) to which new approaches and techniques can be applied. The results can then be reported and realistically compared to other methods. In signature recognition however, there are no such 'standard' databases. This complicates objective comparisons. In addition, there is no standard nomenclature for describing the quality of forgeries. Terms such as casual, amateur, semi-skilled and skilled are often used, yet there is no agreement as to the quality the terms imply. For example, amateur forgeries in one database (so named because the practised forger is not a professional forgery expert) could be classed as as semi-skilled or even skilled forgeries in another (where amateur would mean the forger has never practised or seen the signature before).

One database does stand out, that of Dr. J.G.A. Dolfing [2] working at the Philips Research Laboratories in Eindhoven, Netherlands. It has a relatively large number of signatures, as well as three different forgery qualities, and represents a more real-world situation in that the signatures are very varied and even very bad examples have been included (see Figure 2.5). They have most likely been collected over a short time span though.

The Dolfing database was used exclusively in this research. It was captured in 1995 using the slightly different technology of the Philips Advanced Interactive Display (PAID) [25]. Here, the pen trajectory sensor overlays a real-time interactive 11 inch VGA (640 × 480 pixel) back-lit LCD that shows the pen strokes while signing directly on the screen (so called "electronic ink"). The signatures were sampled at 120 pps which current research shows is entirely adequate. Typically the magnitude of the Y dimension variations at 50 Hz are already 30 dB below those at 1 Hz [16].

The Dolfing database contains a number of different types of signatures for, at most, 51 users. These individuals (45 male and 6 female) are (or were) employees of Philips National Laboratories and Eindhoven University of Technology. The signatures are saved, one per file, in UniPen[1] text format. The composition of the Dolfing database is shown in Table 2.4.

Description	Subjects/users	Signatures	Total Examples
Training Signatures	51	15	765
Testing Signatures	51	15	765
Home-Improved Forgeries	51	30	1530
Over-the-shoulder Forgeries	49	30	1470
Professional Forgeries	13	10	130
TOTAL in database:			4660

Table 2.4: *Composition of the Dolfing database.*

All the signers (including genuine users) were given ample time to practise writing on the digitising tablet and become comfortable with it. The characteristics of the forgers and the different signature categories are described in detail below.

2.4.1 Training and Testing Signatures

The 30 genuine signatures per subject were randomly divided up into equal sized testing and training groups. Only the training signatures were ever used to train the HMMs. For some verifiers that require training, an additional 5 signatures were removed from each subject's testing set, to comprise a development or validation set for this purpose. In any

[1] A standard developed for handwriting sampling. It has header with numerous keywords describing the data, sampling rate, signer, handedness, etc.as well as having hierarchal capabilities that support the recording of single characters, words, sentences, paragraphs, and so on.

case, signatures that may have been used at any stage for training, were never used for testing.

2.4.2 Home Improved Forgeries

Here the forger is given paper copies of the genuine signatures and more than a day to practise signing them. Relatively good looking forgeries can be expected from this group, but note that only the static information is available for them to practise with.

2.4.3 Over-the-shoulder Forgeries

These practised forgers were also allowed to observe while the signers signed their own signatures. This access to more of the dynamic and hidden (pen angle, pressure, etc.) information allows the forgers to repeat the genuine signature much more accurately. This group generally thwarts the our HSV system most often.

2.4.4 Professional Forgeries

Then there are a number of forgeries produced by experts in handwriting analysis, from document examiners, to forensic scientists. These people are trained to discriminate between genuine and forged signatures, and, as such, are able to avoid the usual forger inaccuracies. It must be noted that although these individuals are able to detect high quality spatial forgeries, they have no more experience than anyone else in reproducing the genuine signatures with their associated dynamic properties.

2.4.5 Casual Forgeries

Casual or zero-effort forgeries are not really forgeries at all, as they are not deliberate attempts to reproduce genuine signatures. Instead, this term refers to arbitrary signatures, or segments of handwriting, that are applied to the claimed models to determine the models' robustness to random attacks. This trial has important implications for the banking sector as many cheques are 'forged' without the forger ever having seen the original signature, though often the required printed name is accessible. These are classed as typical casual forgeries.

Table 2.5 shows the average number of feature vectors for each of the above signature types. Note the vast differences in size between the genuine and the impostor signature types. The professionals, who can be expected to make the most accurate looking forgeries,

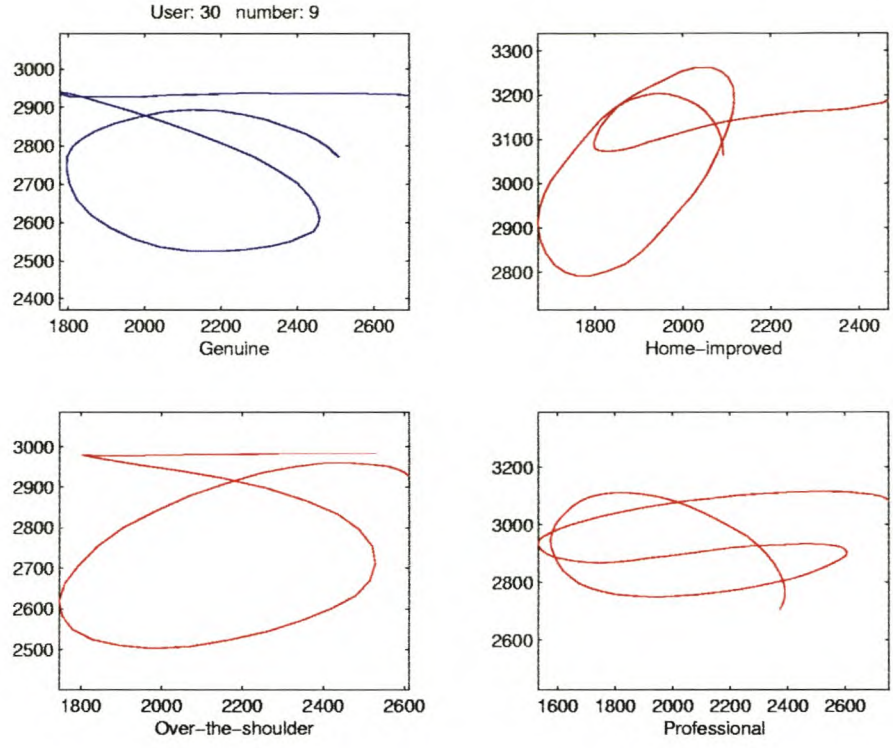


Figure 2.5: *A example of the least descriptive signature and its forgeries.*

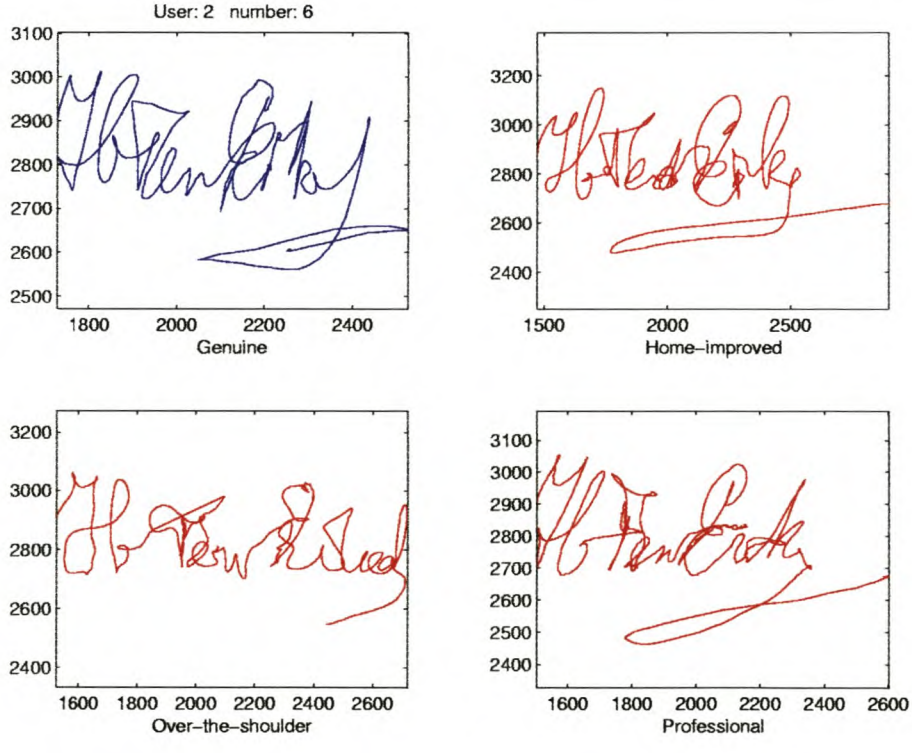


Figure 2.6: *A more representative example of the signatures, also showing the quality of the forgeries.*

Category	Mean Size	Standard Deviation
Training	348.5	150.0
Testing	345.4	147.6
Home Improved	549.2	370.6
Over-the-shoulder	435.7	212.8
Professional	627.0	425.0
All Genuine Signatures	**346.9**	**148.8**
All Forgeries	**504.6**	**322.1**

Table 2.5: *Mean number of feature vectors per signature for the different categories with the standard deviations.*

take almost twice as long as the genuine signers to complete their examples. This has important consequences when deciding on a model for the signatures, as long signatures (hesitant signers) are often a sign of impostors. Having said that, the signing time of both genuine and impostor signers is very varied and differs over a large range, so it is difficult to set rules to reject all signers who take too long.

2.5 The Software Library – PatRecII

PatRecII is the in-house pattern recognition library. It consists of a large number of C++ classes (≈500) with their associated procedures, and a few applications to perform the necessary pattern recognition tasks. There are currently over a quarter of a million lines of code in this library that is built on a modular easily extensible backbone incorporating powerful pattern recognition techniques. It has evolved primarily as an extremely capable and flexible HMM speech recognition facility, which now has also been equipped with a signature verification ability. All the theories presented in this thesis were implemented in C++ within the PatRecII framework.

2.6 Summary

Though it is difficult to compare different HSV systems directly because of the different forgery qualities, we expect an EE rate of between 4% and 8% for the various categories of forgeries when using only the local features of a captured signature. The potential commercial value of a system that can detect over 92% of forgeries is vast, which explains

the marked increase in the number of HSV systems deployed in business over the last two years.

All dynamic HSV systems need a method of capturing the instantaneous signature samples as a user signs. The most effective way to do this is to use a special pen and pad that closely duplicates the natural feel of a conventional pen on paper. The more advanced tablets return the X and Y coordinates, pen pressure and two pen tilt angles. These five dimensions are all useful for signature verification.

The database that we use was collected by Dolfing, and contains over 4000 signatures including genuine signatures and very high quality forgeries.

Chapter 3

Feature Preprocessing: Normalisations and Transformations

As mentioned in Section 2.3, the electronic tablet samples the subject's pen strokes and returns a five dimensional feature vector at least 100 times a second. This raw data can be used directly to train the HMMs, but invariably, better results can be obtained if it is first preprocessed. This is performed with a number of in-line normalisations and transformations which operate on the raw feature vectors to produce new feature vectors. It is these converted feature vectors that are applied to the signature models for training or testing. This concept is illustrated in Figure 3.1.

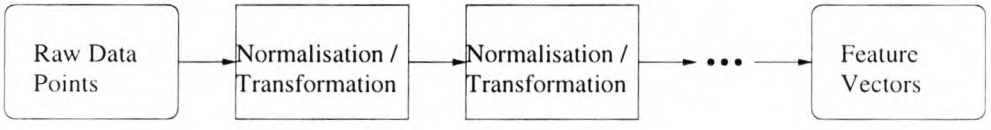


Figure 3.1: *The raw feature vectors or data points go through any number of normalisation/transformations before being to used for HMM training or testing. One entire signature is handled at a time.*

All experiments were performed with raw data signatures, and in-line normalisers or transformers, that process the signatures on the fly each time they are needed. The verification times mentioned later in this text are therefore overly pessimistic. Quicker speeds would be possible if the signature was only preprocessed once, then saved in this format. Nevertheless, in a test situation where this haste is not required, it is more practicable not to have thousands of saved processed signatures.

Some of the reasons that motivate preprocessing are mentioned below:

- Scale – Difficulties arise during training if the data is badly scaled. This can occur if there are major differences in the order of magnitudes of different dimensions, or even if the features contain some extremely high or low values. It is mainly a problem when matrix inversion calculations are performed. This occurs during training of the signature models and during some other preprocessing steps.
- Signature consistency – There is normally no guarantee that a signer's signature will be the same size and orientation each time. Normalisations that decrease a signer's intra-signature variability would be useful.
- Data correlations – Modelling the feature vectors individually may not be as favourable as grouping them in some way to permit their inter-relations to be portrayed more precisely.

The rest of this chapter is dedicated to describing the different signature preprocessors.

3.1 Spatial Normaliser

This normaliser was developed to ensure that the signature always

1. is centred around the XY origin,
2. has a constant orientation,
3. has a constant size.

The first task is achieved simply by subtracting the mean of the X and Y dimensions from all of the samples in those dimensions.

Three methods were tried to accomplish the second goal. Aligning the signature's principal component axis (using eigen value decomposition) with the x-axis was a failure because of the incredible variability of some users' signatures (Try and imagine a repeatable principal axis for the signatures in Figure 2.5). Next, surmising that people generally write from left to right, the signature was rotated so that the mean pen velocity was horizontal. It was found that people generally do not *sign* from left to right. Following that, the slightly different approach of rotating according to the *median* velocity was investigated which encountered the same problem. In the end, with this database of regularly orientated signatures, no rotation at all produced the best results. Real world applications would have a signature box to sign in and so may also find that no rotation produces the most consistent results.

Thirdly, the size is normalised by scaling each of X and Y dimensions to a standard deviation of one. This is done because people's signature tend to vary in size depending on the area available for signing.

Note that the above mentioned steps only apply to the spatial dimensions of X and Y; the other feature dimensions are not affected at all. This normaliser was always the first to be applied to the raw data and was used in all the trials, once its usefulness had been established.

3.2 Differentiators

A number of preprocessors were investigated that perform various differencing operations on neighbouring feature vector dimensions. Some of these differentiators also include smoothing techniques.

These transformations were used specifically on the X and Y dimensions in an attempt to lessen the positional dependence of the signature model. Normally, the probability density functions associated with each HMM state would have to model the absolute positions of each signature stroke. It is hoped that modelling the velocities will increase the system's immunity to forgeries. It should be more difficult for forgers to reproduce a signature's velocity information than its spatial information, which may just have been traced.

Modelling the velocities may also allow PDF codebooks to be used. The codebook PDFs would model a range of velocities from which a users' HMM state PDFs could be constructed. It is expected that less codebook entries would be needed to model signature stroke velocities and accelerations, than positions.

3.2.1 Basic Discrete Differentiation: Delta2

The most basic differentiator that performs straightforward discrete differentiation is the *delta*2. This differences the specified dimensions of consecutive incoming feature vectors and appends the results to this feature vector, as in (3.1).

$$Incoming\ Feature\ Vector \equiv (X, Y, P, \theta_X, \theta_Y) \overset{delta2([0,1])}{\rightarrow} (X, Y, P, \theta_X, \theta_Y, \delta X, \delta Y). \quad (3.1)$$

Here $delta2([0, 1])$ means only differentiate dimension indices 0 and 1 (i.e. X and Y).

There are two forms to this differentiator: the causal and anti-causal, that introduce either a $+0.5$ or -0.5 time-step mis-alignment between the data and its derivatives. This is not usually a problem, but to reduce the compounding affects during repeated applications of this transform, the causal and anti-causal forms should alternated. For example, this reduces feature mis-alignments when calculating accelerations from position.

$$\text{Causal:} \quad \delta x_{i+1} = x_{i+1} - x_i \qquad i \in \{0, 1, 2, \ldots, N-2\} \tag{3.2}$$

$$\text{Anti-causal:} \quad \delta x_i = x_{i+1} - x_i \qquad i \in \{0, 1, 2, \ldots, N-2\} \tag{3.3}$$

Where x can be any dimension that requires differentiation.

Original Dimension	**Anti-causal**		**Causal**	
Value	Equation	Value	Equation	Value
3	$5-3$	2	0	0
5	$9-5$	4	$5-3$	2
9	⋮	⋮	$9-5$	4
⋮	⋮	⋮	⋮	⋮
17	$33-17$	16	⋮	⋮
33	0	0	$33-17$	16

Table 3.1: *The* delta2 *transform simply differences the required dimensions (this table only shows one dimension) of consecutive feature vectors. The third and fifth columns show the dimensions that would be appended to the original feature vector (first column) for the anti-causal and causal cases.*

3.2.2 Alternative Discrete Differentiation: Delta3

Another derivative that was investigated was the *delta*3 transform. It is described by

$$\delta x_i = \frac{1}{2}x_{i+1} - \frac{1}{2}x_{i-1} \qquad i \in \{1, 2, 3, \ldots, N-2\}, \tag{3.4}$$

which does not suffer from the introduction of time shifts between dimensions and their derivatives.

3.2.3 Smoothed Discrete Differentiation: Delta5

The *delta*5 transform provides a smoothed discrete derivative of the required dimensions. The modest low-pass filtering may reduce quantisation and other noise that becomes more

significant when differentiating.

$$\delta x_i = \frac{1}{8}x_{i+2} + \frac{1}{4}x_{i+1} - \frac{1}{4}x_{i-1} - \frac{1}{8}x_{i-2} \qquad i \in \{2, 3, 4, \ldots, N-3\} \tag{3.5}$$

3.2.4 Time Skewing

When calculating higher order derivatives from repeated applications of time-skewing transforms such as the *delta*2, it is always advisable to alternate its causality. If this is not done, the time alignment between the calculated and the original dimensions will become skewed. In the case of signature verification, this may cause problems for the HMM training algorithms that need to segment the signature into strokes. This is because discontinuities may happen at skewed times in derived dimensions when compared with the original dimensions.

Equations (3.6) and (3.7) show the small time mis-alignments (in terms of time-steps) if the 'accelerations' are calculated incorrectly. Equation (3.8) shows the correct method. Causal, then causal $\Rightarrow$ Time shift = 1.0:

$$\delta\delta x_{i+2} = x_i - 2x_{i+1} + x_{i+2} \qquad i \in \{0, 1, 2, \ldots, N-3\}. \tag{3.6}$$

Anti-causal, then anti-causal $\Rightarrow$ Time shift = -1.0:

$$\delta\delta x_i = x_i - 2x_{i+1} + x_{i+2} \qquad i \in \{0, 1, 2, \ldots, N-3\}. \tag{3.7}$$

Causal, then anti-causal (or *vice versa*) $\Rightarrow$ Time shift = 0.0:

$$\delta\delta x_{i+1} = x_i - 2x_{i+1} + x_{i+2} \qquad i \in \{0, 1, 2, \ldots, N-3\}. \tag{3.8}$$

In all the differentiating transforms above, the beginning and ending feature vectors may be calculated slightly differently as the situation demands. This keeps the number of input and output feature vectors the same. These transforms proved very useful for improving the accuracy of HSV systems that use only the time sampled X and Y coordinates—as would be obtained from the newer cellular phones and palmtop computers. However, they met with limited success in trials (see Section 6.2.2) where the full five dimensions could be captured from the writing pad.

3.3 Selecting A Subset Of Features

Often it is necessary to remove certain dimensions or intermediate steps from the feature vector. The selector transform selects certain dimensions to retain, while the rest are discarded. It proves to be a useful tool for trials needing only the X and Y spatial information (discarding P, θ_x, and θ_y) or the certain acceleration dimensions (the intermediate velocity dimensions are discarded).

3.4 Scaling The Features

This normaliser is used primarily to ensure that there are no numerical range difficulties while training and testing the signature models. It scales each data dimension to have a standard deviation of one. This ensures that the occurrence of numerical inaccuracies (especially loss of significance), when manipulating numbers on finite machines (all computers), is lessened. Numerical inaccuracies could otherwise happen during operations on numbers with different ranges, especially when inverting matrices such as the state PDFs' covariance matrices and CBKLTs' (see Section 3.6) projection matrices. The standard deviation is estimated from *all* the training data at once.

3.5 Grouping Feature Vectors Into Frames

Thus far, all previous transformations have been applied to any or all of the dimensions X, Y, P, θ_x, and θ_y, taken from up to four feature vectors (e.g. $delta5$). This however, does not permit the relationships between nearby points to be exploited fully – the points after all make up a line segment, and are not just randomly distributed.

The frame-as-vector transformation groups M consecutive feature vectors of dimension N into a new $(M \times N)$-dimensional composite feature *vector*. This effectively allows line segments, instead of single points, to be modelled. In the example shown in Table 3.2, a frame length of 8 (8 consecutive feature vectors grouped together) with a frame shift of 1 (1 time-step difference between initial feature vectors in consecutive frames) is illustrated. Thus, in this case, there is an overlap of 7 between following composite feature vectors, and each original feature vector gets used 8 times.

From Table 3.2:

$$Frame_0 = (X_0, Y_0, P_0, \theta_{X_0}, \theta_{Y_0},\ X_1, Y_1, P_1, \theta_{X_1}, \theta_{Y_1},\ \ldots,\ X_7, Y_7, P_7, \theta_{X_7}, \theta_{Y_7}). \quad (3.9)$$

Frame Number:	Consists of these Feature Vectors:
0	0, 1, 2, 3, 4, 5, 6, 7
1	1, 2, 3, 4, 5, 6, 7, 8
2	2, 3, 4, 5, 6, 7, 8, 9
3	3, 4, 5, 6, 7, 8, 9, 10
⋮	⋮

Table 3.2: *Constructing composite versions of the original feature vectors. This example uses a frame size of 8, and a frame shift of 1.*

Modelling the line segments, as supposed to individual feature vectors, is motivated by the imperfect manner in which these feature vectors are normally represented by Gaussian HMM state PDFs. Figure 3.2 shows ten line segments from ten different versions of a user's signature. The segments are always taken from the corresponding locations in the signature. Beneath these lines is a 2D Gaussian PDF that has been estimated from the points making up the lines. There are 17 points per line segment[1].

The distribution of these points is not, however, Gaussian, as would be modelled by the HMM state PDFs. Instead, it seems to be more like a bounded uniform distribution with a hole in the middle. Nevertheless, though the feature vectors' X and Y dimensions are not Gaussian distributed, the *line segments* themselves may well be Gaussian distributed in a higher dimensional space. The frame-as-vector transformation allows us to test this hypothesis. In fact, this transformation to line segment feature space was used to great effect to produce the most successful verification results.

3.6 Feature Vector De-correlation: The Class Based Karhunen-Loève Transform

One of the more complex techniques used to transform the data, before passing it to the signature model, is based on the well known Karhunen-Loève Transform (KLT) or Principal Component Analysis (PCA)[2]. The KLT can be used to determine, and then

[1]The mean number of feature vectors per genuine signature in the Dolfing database is 346.9, and almost all of the conducted experiments use HMMs with 20 state PDFs. Therefore, the mean number of training vectors per PDF per signature is $346.9/20 = 17.345$.

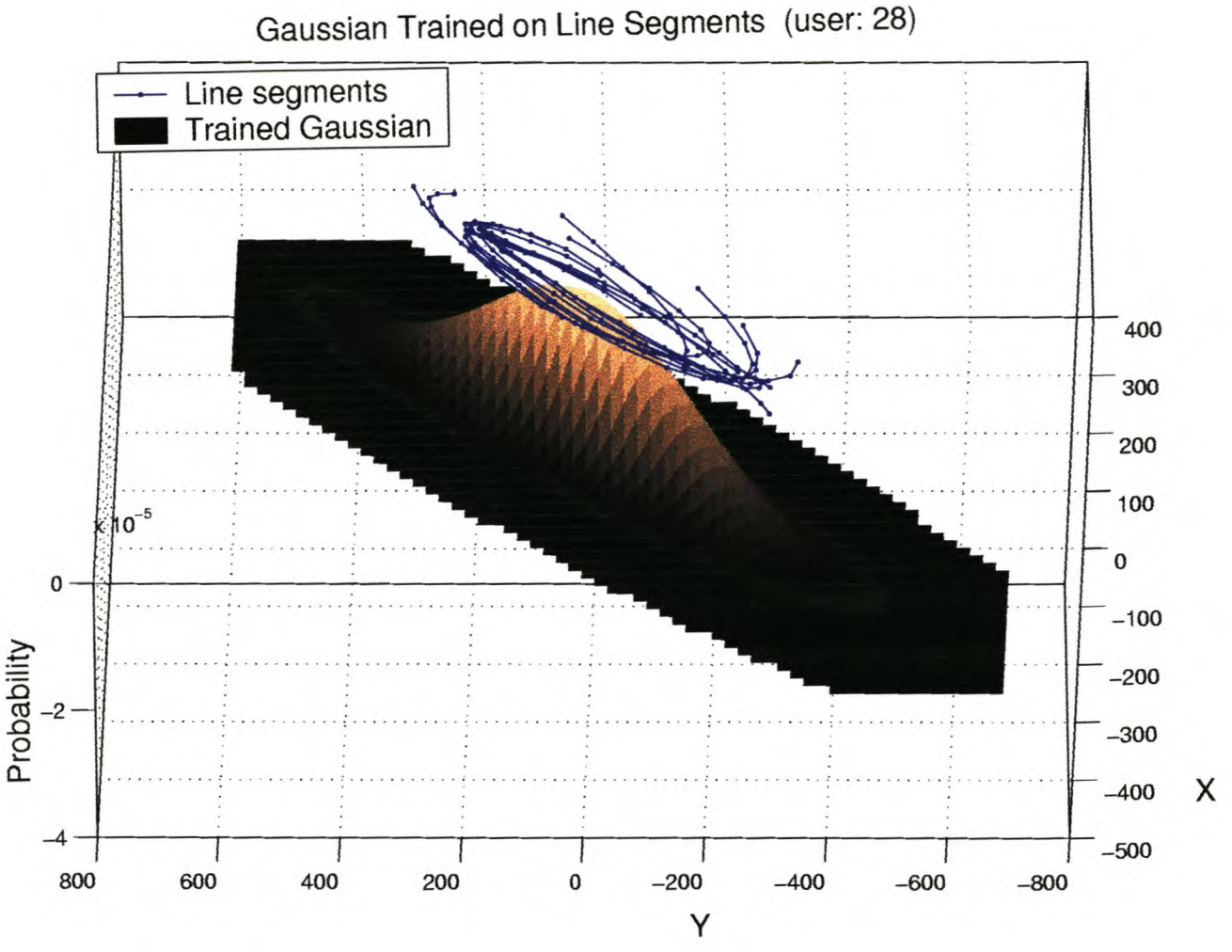


Figure 3.2: *Corresponding line segments from 10 versions of a user's signature. Beneath them is the estimated Gaussian probability density function as would be used in an HMM state. Note that the data points are* not *Gaussian distributed, but rather display a bounded uniform distribution with a central 'hole'. (The PDF's height is shifted down to a maximum of zero for clarity.)*

transform to, the data's underlying dimensionality. The principal axis returned from a KLT is the direction of maximum variance in the original data, with the next orthogonal axis being that of second greatest variance, and so on.

The aim of these feature preprocessors is to make it easier to distinguish individual's signatures from one another. The KLT clearly does not transform the signature feature vectors to a space that eases this task, since it will probably be no easier to separate the different classes here than in the original feature space. The Class Based Karhunen-Loève Transform (CBKLT) however, does do this. It produces a principal axis in the direction of maximum inter-class variance (with the next axis being the direction of second largest inter-class variance, and so on). This gives the exact directions along which the classes can be separated most effectively.

The CBKLT uses knowledge of which class every feature vector is in. For signature verification, this knowledge is obtained by pre-training a set of similar (non CBKLT) HMMs—one for each user—and then presenting the signatures to the HMMs. The HMM state PDF that each signature feature vector matches best, is noted. Each HMM state PDF for each user is then deemed a class for the CBKLT. Therefore, if there are 51 user HMMs, each with 20 PDFs, there are $51 \times 20 = 1020$ classes. The training data for each class is then known and statistical properties (such as the covariance matrices) can be calculated.

The next step in the CBKLT is to pre-whiten all the data together, according to the average of all the classes' covariance matrices. Each class's covariance matrix is weighted by the probability of that class's data vectors occurring from all the data vectors. After this step, the average covariance matrix will be the identity matrix. This pre-whitening does assume the class distributions are of a similar shape. Next, a plain KLT is applied using the class means (weighted by their occurrence probabilities) as the input data. The resulting axes represent the directions of maximum inter-class variance in decreasing order.

Figure 3.3 shows two 2D classes bounded by the ellipses. The plain KLT axis of maximum variance is shown by axis u_1. This direction does not allow any class distinctions to be made. If however, the data is whitened to have a unity covariance matrix (the two circles), and then a KLT is applied, the axis x_1 produced. This direction permits the classes to be distinguished optimally.

Le Riche states that reducing the number of dimensions of single raw 5-dimensional feature vectors with the KL transform lowers the accuracy by a massive 10%. Nevertheless,

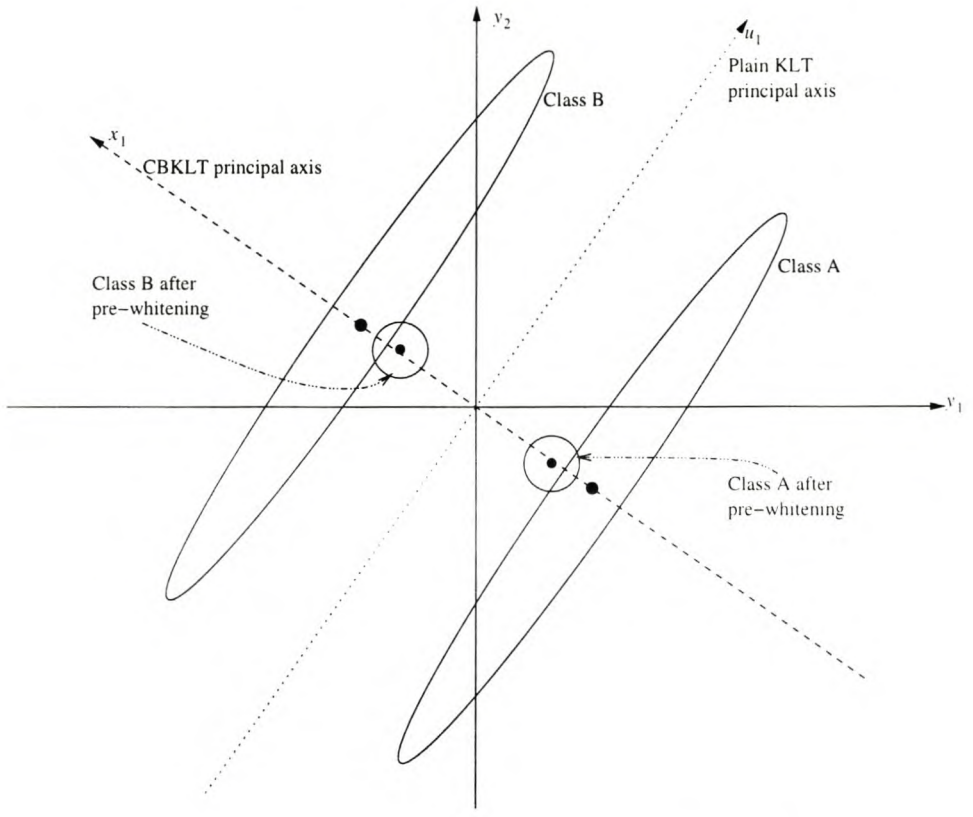


Figure 3.3: *Differences between the CBKLT and the KLT.*

we found the CBKLT an invaluable tool for dimension reduction on data frames containing multiple feature vectors.

3.7 The Non-linear Volterra Functional Series

The Volterra functional series or Volterra expansion is a non-linear transform that also operates on the incoming data. This means that some of the advantages of non-linear modelling can be realised with linear models (such as HMMs) by applying this transformation to the incoming data.

In essence, the Volterra expansion transforms the incoming feature vector to a vector consisting of all the mixing products of the original vector. That is, all the dimensions are cross-multiplied with all other dimensions for all the possible permutations (see (3.10) to (3.13) for four symbolic examples). It finds wide application in radio frequency circuit analysis where all frequencies are inter-modulated with all other frequencies to produce components at all original and all product frequencies.

$$(a, b) \overset{Volterra(2)}{\rightarrow} (a, b, aa, ba, bb). \tag{3.10}$$

$$(a, b) \overset{Volterra(3)}{\rightarrow} (a, b, aa, ba, bb, aaa, baa, bba, bbb). \tag{3.11}$$

$$(a, b, c) \overset{Volterra(2)}{\rightarrow} (a, b, c, aa, ba, bb, ca, cb, cc). \tag{3.12}$$

$$(a, b, c) \overset{Volterra(3)}{\rightarrow} (a, b, c, aa, ba, bb, ca, cb, cc, aaa, baa, bba, bbb, caa, cba, cbb, cca, ccb, ccc). \tag{3.13}$$

Where $Volterra(d)$ represents the Volterra expansion of order d.

It was envisioned that this transformation could be used with the frame-as-vector transformation to model multiple data points. If only the X and Y dimensions are considered, the $Volterra(3)$ series would provide all the polynomial terms ($X, Y, X^2, XY, Y^2, X^3, X^2Y, XY^2, Y^3$) necessary for a cubic interpolation between points. The training process could then choose the best polynomial fit through the given number of points. The problem is that the number of dimensions returned, from even some of the lower order expansions, exceeds personal computer capabilities. If only the second order expansion is taken for a 40D vector (eight 5D vectors after a frame-as-vector transformation), it results in the unmanageable total of 860 new dimensions. A CBKLT of over 1000 classes (as used later in Section 6.2.5) can not then be used to reduce the dimensions due to the huge memory requirements. It was however, possible to test this theory using only three vectors per frame, but this led to disappointing results. Table 3.3 shows how the number of mixing products explodes with increasing order and dimension.

	Input Dimension							
	1	2	3	4	5	10	15	20
Order 1	1	2	3	4	5	10	15	20
Order 2	2	5	9	14	20	65	135	230
Order 3	3	9	19	34	55	285	815	1770
Order 4	4	14	34	69	125	1000	3875	10625

Table 3.3: *The number of output dimensions increases sharply with increasing input dimensions or expansion orders.*

3.8 Summary

A number of different preprocessors have been described. They were all investigated, though not all of them (the differentiators and Volterra expansion) were used in our final system that produced the best results.

The preprocessors are presented above in the general order in which they would be applied to the data (though the scaling normaliser is used more than once).

After the signature data has been normalised and transformed to the correct format, it can be used to estimate a signature model for each user, or can be applied to an existing signature model to determine how well it matches. It is this signature model that we concern ourselves with next.

Chapter 4

Creating The Signature Model

One of the main goals of this research was to find a structure that represents the characteristics of a user's signature, especially those that make it different from other people's. There are a number of attributes that an ideal model should possess—it should be

- accurate—the number of false acceptances and false rejections must be acceptable for the application.
- small—it may have to be stored on media where space is a concern, or transferred across networks with limited bandwidth.
- secure—access to the model should not imply access to the signature verification protected system.
- adaptable—a person's signature changes over time, which means that the model representing it should also be able to.
- fast—again, depending on the application, an 'accept' or 'reject' answer may be required urgently or a more tardy response may be tolerated.
- flexible—the requirements of many different people with many signature idiosyncrasies will have to be accommodated.

Hidden Markov Models have been used for speech pattern recognition tasks since the 1960s, yet are still a new technique in the field of dynamic handwritten signature verification—a field to which they seem particularly well suited. The remainder of this chapter describes the HMM structures and components investigated in an attempt to find an appropriate model for real-world signatures.

4.1 Hidden Markov Models

Hidden Markov Models (HMMs) are used in many applications to model situations where feature vectors have characteristics that remain relatively constant for a certain amount of time, before changing quickly or slowly to different relatively constant characteristics. They have found wide application in areas of signal modelling such as speech processing.

In their classic article, Rabiner and Juang [22], introduce HMMs with the following definition:

> "An HMM is a doubly stochastic process with an underlying stochastic process that is *not* observable (it is hidden), but can only be observed though another set of stochastic processes that produce a sequence of observed symbols."

Their paper also explains the principles behind HMMs, how they are trained, as well as how a score is obtained that signifies the similarity been the applied data and an HMM. The basic operation and components of a HMM are explained below.

An HMM consists of a number of states with interconnecting transition links, each transition having its own associated probability. The states output a symbol for each time-step[1]. In our case, the symbol that is output is the continuous probability of the feature vector belonging to the PDF associated with the current state. It is because one is not sure of which state the HMM is in at any time, given a set of output symbols, that the model is 'hidden'. The basic operating principles are described below.

1. The model contains a finite number, N, of states, $Q = \{q_1, q_2, \ldots, q_N\}$, in which the signal contains some distinctive, measurable properties.

2. The initial state distribution is given by $\pi = \{\pi_j\}$, with $\pi_j = P(q_j \text{ at } t = 1)$ determining the first state entered at clock time $t = 1$.

3. At a clock time t, a new state is entered based upon the current state, and a transition probability distribution defined by $A = \{a_{ij}\}$, with $a_{ij} = P(q_j \text{ at } t + 1 |\ q_i \text{ at } t)$.

4. After each transition, an observation probability is produced from the PDF associated with the current state. The observation probability distribution is given by $B = \{b_j(\boldsymbol{x})\}$, with $b_j(\boldsymbol{x}) = P(\boldsymbol{x} \text{ at } t | q_j \text{ at } t)$, where $\boldsymbol{x}$ is the feature vector.

---

[1] In this text, all the feature vectors are sampled against time, and, as such, all future references refer to HMMs that specifically accept time-sampled features.

HMMs can be constructed in many different topologies with certain special abilities, by limiting the state transition matrix so that it is not always possible to go from any state to any other state. For example, an ergodic structure, where each state does have a transition to every other state, is useful for aural language recognition. Here, each state would represent a word, and any word could follow after any other word, though some sequences would certainly be more probable than others. In contrast, the letters or strokes of a signature are always in a set sequence that cannot be disarranged, so a topology that does not allow previous state re-visitation would be more appropriate.

4.1.1 Introduction to Left-to-right HMMs

A structure that is very useful for modelling signatures, is the left-to-right topology that is so named because it can be drawn as a sequence of states that proceed from start to finish with no looping back to previous states. That is, the transitions exiting from a state can either be self-loops (destination same as origin) or forward transitions, skipping out any number of intermediate states, which will never be returned to. Figure 4.1 shows a three state left-to-right HMM. Each state has a self-loop and the exiting transitions are allowed to go on to the next state as well skipping forward just one state. This process seems to represent the physical attributes of a signature very well; the strokes of a signature are always repeated in the same order, and overall the signature will look similar except for possibly a few stroke insertions or deletions.

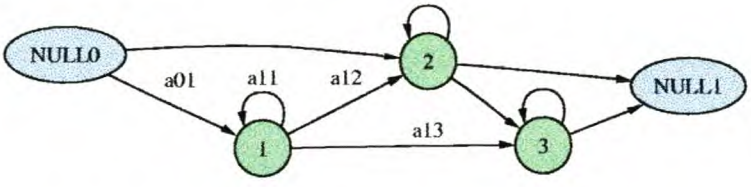


Figure 4.1: *A three state left-to-right HMM with initial and terminal* NULL *states.*

NULL States

Special mention should be made of the NULL states used in the implementation to simplify programming procedures. They do not have any associated density functions, and so do not emit an observation symbol, nor occupy any time-steps in the HMM's state sequence. They do possess regular state transitions however, that count towards the final score as usual. This has definite programming advantages. The state sequence will always start at state zero which is normally a NULL state with multiple 'initial' outgoing transitions that

replaces the initial state distribution vector π. This means that the programming can be implemented more homogeneously since the elements in π have been replaced by the first NULL state's output transition probabilities. That is $a_{0j} = \pi_j = P(q_j \text{ at } t = 1 | \ q_0 \text{ at } t = 0)$ where q_0 is the initial NULL state in which the state sequence always starts.

The left-to-right model, with certain modifications discussed below, was used to great effect for handwritten signature verification.

4.1.2 Introduction to Explicit Time Duration Modelling

Time duration modelling is required for HSV to statistically model how long each signature stroke stays associated with each HMM state PDF. It is implicit in all HMM states with self-loops. The chance of staying in one of these states decays exponentially with time. To illustrate this, consider a state that has a self-loop probability of 0.8. The probabilities of remaining in this state for another 1, 2, or 3 time steps, are 0.8^1, 0.8^2 and 0.8^3 respectively.

Unfortunately, this is not an accurate reflection of the duration properties of many signals, including signatures. For example, the corresponding pen strokes from one user's signatures will probably occur in similar amounts of time, which means that the duration spent in that state should not be modelled with an exponential decay function. More complex situations can also arise: users may sometimes lift their pen to cross a 't', while at other times draw the line connecting the previous signature position to cross the 't'. This means that two possible duration lengths have to be accommodated for the differing times it takes to cross the 't'. Of course the frequency of all these occurrences have to be taken into account for the final model.

4.1.3 Explicit Time Duration Modelling: Conventional Topology

It is clear that the implicit duration modelling of HMM states with self-loops is not going to be accurate enough to cope with signature representation. This is because matching signature strokes from one individual's signatures, all take similar non-zero amounts of time. Therefore, an exponentially decaying function is unsuitable.

Explicit duration modelling can be added to the left-to-right topology by replacing each state that has a self-loop with a duration stack of states. Figure 4.2 shows the resultant HMM when duration modelling of order three is added to the HMM shown previously in Figure 4.1. The order of the duration modelling determines the number of states used to

replace the original state. The states 1.0, 1.1, and 1.2 make up the duration stack that replaces the original state 1 in Figure 4.2. The TOP states (1.2, 2.2, and 3.2) still have implicit duration modelling provided by their respective self-loops. This topology has its origins in high-order HMMs as discussed in [4].

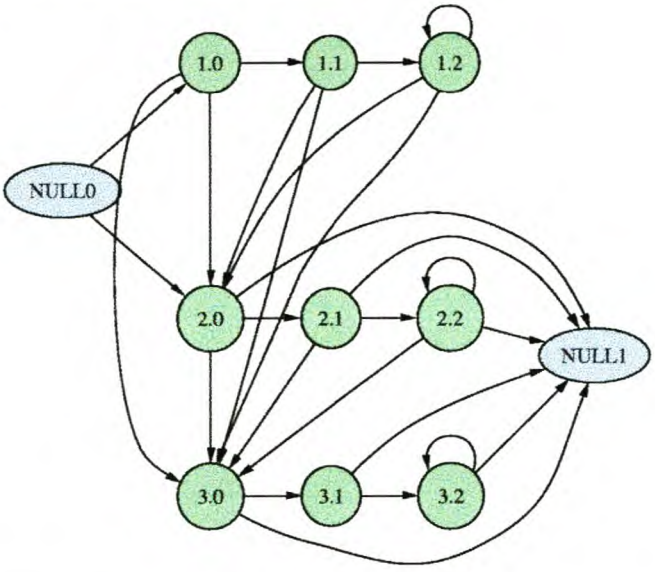


Figure 4.2: *The 3 state HMM with third order duration modelling added.*

Though there are now three times as many emitting (observation probability producing) states when compared with the original left-to-right model, there are still only three internal PDFs. This is because every state on the duration stack still refers to the one original PDF. This has the advantages of fewer PDFs to train (less training data is needed), and fewer entities to save (less space is needed), along with the accompanying speed increases.

This topology allows signal time durations to be modelled explicitly. For example, it is now possible for a signal to have a higher probability of staying associated with a certain PDF for *two* time-steps, than for *one*. This is not possible with normal exponentially decaying duration modelling. The disadvantage of explicit time duration modelling however, is that more training data is needed to accurately estimate all the extra transition probabilities.

For signature verification with the Dolfing database, which has an average signature length of 347 feature vectors, an HMM with 20 original states and 20^{th} order duration modelling was determined empirically to be the best configuration. This means that there are over 400 transition probabilities per HMM that need estimating. With only 15 training examples, it is unlikely that all the signature variations will be represented. This will result

in some of the transition links remaining at a probability of zero, while neighbouring links may be very popular. The solution is to smooth these granular transition probabilities as if they were a histogram, and so eliminate the holes and excessive peaks along the links to the duration stack. Le Riche [16] also reports that transition smoothing is essential for consistent, low error rates. This proves to be a rather complicated task when using the normal duration modelling topology of Figure 4.2 due to the multiple interdependencies of the duration links with themselves and other links. For example if we simply require the duration links from the states 1.0, 1.1, and 1.2 to 2.0 to be smoothed, we find there are numerous other links that then also have to adjusted to maintain a similar HMM behaviour. For this reason, a new topology was implemented that allows the duration transitions to be smoothed much more elegantly.

4.1.4 Explicit Time Duration Modelling: Modified Ferguson Topology

The duration modelling topology proposed by Ferguson [5] has the advantage that the duration fanout transitions (those leading to the duration stack) can easily be smoothed resulting in a more representative and robust HMM using less training data. We present a modified version of a Ferguson HMM that can be seen in Figure 4.3. It differs from a normal Ferguson HMM by the inclusion of the NULL states that allow an easier initialisation.

The duration fanout transition probabilities from the NULL states in this structure, can be smoothed much more simply. This is because they depend only on each other and the TOP state's self-loop, which essentially means that those probabilities can just be smoothed, then scaled to ensure the sum of the probabilities leaving each NULL state is one.

The structure shown in Figure 4.3 is the result of applying Ferguson duration modelling of order 3 to the original left-to-right HMM shown in Figure 4.1. Again, all the newly added states (except the NULL states) refer to the original PDF that the duration stack was expanded from. The inaugural probabilities of the transition links are also configured from those of the original left-to-right HMM as discussed in the following section.

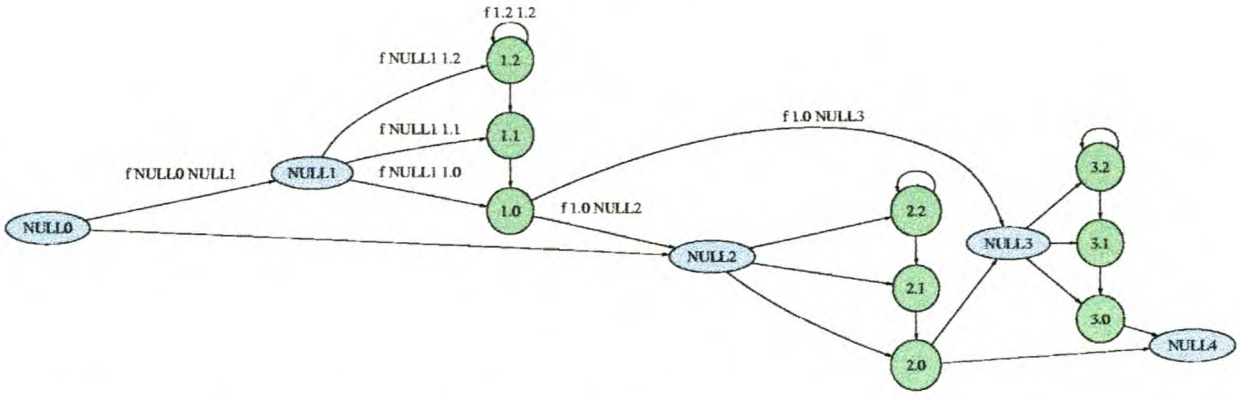


Figure 4.3: *The 3 state HMM with third order modified Ferguson duration modelling added.*

4.1.5 Initialisation of Modified Ferguson HMMs

A very important aspect of our revised Ferguson topology is that it can be initialised to have an identical behaviour to the left-to-right HMM it was expanded from. This is essential, as the left-to-right HMM has already been trained to represent the user's signature as best it can (See Section 4.4 for the HMM training process.). Also, there is probably not enough signature data to allow proper estimation of the duration modelling HMM if it was to be trained from scratch, so any contributions from previous models are useful. After this initialisation procedure, the duration model will be re-estimated to allow its improved time modelling abilities to be used.

The calculations that permit the initial modified Ferguson left-to-right, and simple left-to-right HMMs to behave identically, are shown below. The modified Ferguson transition probabilities (Figure 4.3) for one duration stack are determined in terms of the transition probabilities of the left-to-right HMM (Figure 4.1).

The link entering the original state is unchanged, but now terminates at a NULL state:

$$f_{NULL0\ NULL1} = a_{01}. \tag{4.1}$$

The duration fanout links that may require smoothing :

$$f_{NULL1\ 1.0} = (1 - a_{11}), \tag{4.2}$$

$$f_{NULL1\ 1.1} = a_{11}^{1}(1 - a_{11}), \tag{4.3}$$

$$f_{NULL1\ 1.2} = a_{11}^{2}(1 - a_{11}) + a_{11}^{3}. \tag{4.4}$$

The extra term in the last equation is because this link goes to the TOP state that has implicit duration modelling due to its self-loop.

The TOP state's self-loop and complement:

$$f_{1.2\ 1.2} = a_{11}, \tag{4.5}$$

$$f_{1.2\ 1.1} = (1 - a_{11}), \tag{4.6}$$

$$f_{1.1\ 1.0} = 1. \tag{4.7}$$

The links from the original state now only leave from the BASE state and are scaled because there is no self-loop on this state:

$$f_{1.0\ NULL2} = a_{12}/(1 - a_{11}), \tag{4.8}$$

$$f_{1.0\ NULL3} = a_{13}/(1 - a_{11}). \tag{4.9}$$

The HMM will then need to be re-estimated to allow it to use its extended duration modelling capacities to represent a user's signatures more accurately.

This example shows the calculations for expanding one self-loop state with multiple outputs, to a duration emphasised Ferguson topology of order three. It is trivial to broaden this example to any required duration order with any number of original states. Once the HMM has been initialised correctly using the above formulae, the links can be re-estimated with normal methods such as the Viterbi [22] or Baum-Welch algorithms. The Viterbi algorithm was used exclusively in this research.

4.2 Transition Link Smoothing

Ferguson duration emphasised HMMs were implemented to allow tractable transition smoothing so that a realistic number of signatures would be sufficient to train a robust model. There are many different ways to smooth the vector containing the duration fanout probabilities including averaging, low-pass filtering in the frequency domain, and representing them as various distribution functions. An example of this duration fanout vector is ($f_{NULL1\ 1.0}$, $f_{NULL1\ 1.1}$, $f_{NULL1\ 1.2}$) from Figure 4.3. There can also be indirect benefits of using transition smoothing such as a reduction in the number of model parameters with a consequent decrease in storage size. Before smoothing methods can be discussed further, the current method of transition link representation is described.

4.2.1 Relative Frequency Vectors

Transition link probabilities are normally saved as relative frequency vectors. This is simply a list of probabilities that matches up with another vector that describes the

destination state of each link probability. As mention above, if each model has over 400 links to train and there are only 15 training signatures each with about 347 feature vectors, it is probable that certain permutations of a user's signature will not be catered for. This phenomenon manifests itself as very granular distributions and 'holes' in the relative frequency vector profile. The zero probability of the 'holes' are then more a result of the lack of training data, than an accurate depiction of the users signature characteristics. Figure 4.4 shows that this is indeed the case. This motivates the following investigation of transition smoothing methods.

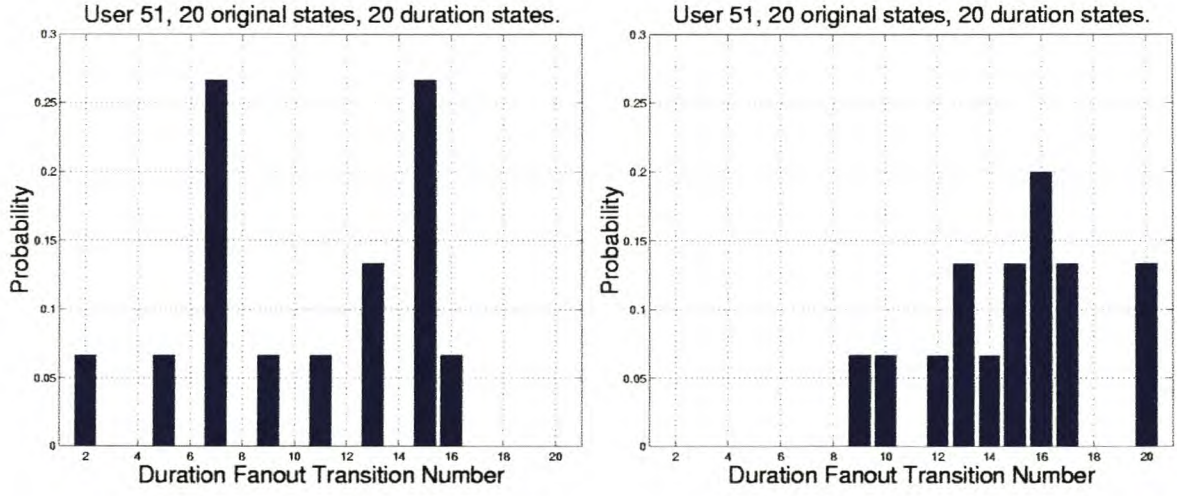


Figure 4.4: *The histograms of two arbitrary, yet typical examples of relative frequency vectors showing the abundance of 'holes' in the distribution. Number 1 on the x-axis corresponds with the transition to the* BASE *state, and 20 with that of the* TOP *state.*

4.2.2 Histogram Based Smoothing Techniques

These methods are non-parametric techniques that attempt to replace the 'holes' with floor values or otherwise estimate a more smooth histogram to represent the transition probabilities. Methods such as Witten-Bell [27] estimation fall into this category. They were not particularly successful because the zero probability 'holes' are not the only problem—the large discontinuities between adjacent link probabilities also have negative impacts on the accuracy. Furthermore, they provide no model size reduction and the issue of 'too many parameters, not enough training data' still holds. Therefore a new approach involving sampled continuous PDFs was investigated.

4.2.3 One-dimensional Gaussian Density Functions

This method models the duration fanout probabilities with a sampled, and therefore discrete, version of a one dimensional Gaussian probability density function. A 1D Gaussian only has two parameters, those being the mean and variance. A random variable, x, is Gaussian distributed if its density function has the form:

$$f_x(x) = \frac{1}{\sqrt{2\pi\sigma_X^2}} e^{-\frac{(x-a_X)^2}{2\sigma_X^2}}, \tag{4.10}$$

where:

$a_X = E[x]$,

$\sigma_X^2 = E[(x - E[x])^2]$,

and $E[x]$ is the expected value of x.

The Gaussian PDF is estimated during the training of the HMM from the relative frequency that each duration fanout position occurs. When the actual transition probabilities are required, the Gaussian PDF is sampled at the required position, normalised and returned as a probability. The normalisation step is required because the *area* under continuous density functions equals one, whereas the *sum* of the probabilities of a discrete density function should equal one.

The problem with a unimodal Gaussian PDF is that it smooths the duration fanout transition probabilities rather too well, and does not allow sufficient specialisation. Looking at the first plot in Figure 4.4 should convince the reader that in this case, a single Gaussian is not the optimum solution. Gaussian PDFs do however have advantages, such as the small number of defining parameters, which lead to Gaussian mixture models being investigated next.

4.2.4 Gaussian Mixture Models

Gaussian Mixture Models (GMMs) are density functions made up from the weighted sum of a number of underlying single Gaussian functions. This means that increasingly more complex distributions can be modelled by using more underlying Gaussians, which naturally means more parameters and less smoothing. The weightings assigned to each PDF are estimated from training, and together always sum to one.

After analysing many duration fanout probability distributions and error rates, we concluded that a GMM consisting of three underlying Gaussians was the best solution. This will model the time associated with each state PDF accurately, while smoothing the probabilities sufficiently to cope with the unseen yet possible stroke duration occurrences in the future. This GMM has nine parameters, three each of the means, variances, and PDF weightings, and was used in all further signature verification experiments.

4.3 HMM State Probability Distribution Functions

Each state of an HMM has a PDF associated with it that produces the observation symbol/vector at each time-step. This PDF must best represent the training data over a certain time period. It is difficult to intuitively choose a certain PDF because the dimension of the feature vectors is often more than three, which makes visualisation formidable. Evaluating the system performance in experimental trials with different state PDFs was used to determine the best type of PDF.

There is always the inherent trade-off between accurate modelling with more internal parameters requiring more training data, and decreased fitting ability needing less parameters and less training data. Three unimodal (having one peak) Gaussians with differing number of parameters are discussed in the following sections.

4.3.1 Full Covariance Gaussian PDFs

The full covariance Gaussian is so named because the covariance matrix is fully populated, which means a contour on the PDF can be a hyper-ellipse orientated in any direction. It is defined by the following equation:

$$f_x(\boldsymbol{x}) = \frac{1}{(2\pi)^{N/2}|\boldsymbol{C_{XX}}|^{1/2}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{a_X})^T \boldsymbol{C_{XX}}^{-1}(\boldsymbol{x}-\boldsymbol{a_X})}, \tag{4.11}$$

where:

$\boldsymbol{x}$ is an N-dimensional data vector,

$\boldsymbol{a_X}$ is the N-dimensional mean vector: $\boldsymbol{a_X} = E[\boldsymbol{x}]$,

$\boldsymbol{C_{XX}}$ is the $N \times N$ covariance matrix:

$$\boldsymbol{C_{XX}} = E[(\boldsymbol{x} - \boldsymbol{a_X})(\boldsymbol{x} - \boldsymbol{a_X})^T] = \begin{bmatrix} C_{11} & C_{12} & \dots & C_{1N} \\ C_{21} & C_{22} & \dots & C_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ C_{N1} & C_{N2} & \dots & C_{NN} \end{bmatrix}. \tag{4.12}$$

$|\boldsymbol{C_{XX}}|$ is the determinant of $\boldsymbol{C_{XX}}$,
and $(\cdot)^T$ indicates the vector transpose operation.

Since this PDF is determined only by the mean vector, $\boldsymbol{a_X}$, and the symmetric covariance matrix, $\boldsymbol{C_{XX}}$, there are $\frac{N(N+3)}{2}$ defining parameters.

4.3.2 Diagonal Covariance Gaussian PDFs

This PDF assumes the dimensions are statistically independent and does not model any correlations between different dimensions. Each dimension still has an associated variance that populates the main diagonal of the covariance matrix (hence the name), as in Equation 4.13. There are N means and N variances bring the total number of parameters to $2N$. A contour on this PDF can assume any hyper-ellipsoid shape, but can only be orientated along the standard axes of the space. That is, the variance along the axes can be adjusted, but the skewness from those axes can not.

$$[\mathrm{C_{XX}}] = \begin{bmatrix} C_{11} & 0 & \dots & 0 \\ 0 & C_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & C_{NN} \end{bmatrix}. \tag{4.13}$$

4.3.3 Circular Gaussian PDFs

This Gaussian PDF further limits the variances by setting them equal to each other and providing one global variance parameter. A contour on this PDF will be a circle with unconstrained radius and origin. This PDF has $N + 1$ parameters: the N-dimensional mean vector and the scalar variance. The covariance matrix for substitution into (4.12) is:

$$[\mathbf{C_{xx}}] = k \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix}, \tag{4.14}$$

where k is the global variance.

4.3.4 Discussion

All the Gaussian PDFs presented above were trained with maximum likelihood estimation (MLE) that determines the parameters that maximise the probability (likelihood) of the training data.

The advantage of using the PDFs with a limited number of parameters, becomes clear when we make an estimate as to the amount of data needed for training. Note, this is only an intuitive feel for the consequences of model parameter quantities.

On average there are 347 feature vectors per signature $\times$ 5 dimensions $\times$ 15 training signatures per user = 26025 items of training data per user.
In a 20 state HMM, each state PDF will be estimated from an average of $26025/20 \approx 1301$ data items.
If the state PDF is a 5D full covariance Gaussian, there are 20 parameters ($N(N+3)/2$ where N is the dimension) that need training.
This means $1301/20 \approx 65$ data items per parameter which should result in a good estimation.
If however the original 5D feature vectors are framed 8 at a time with no repetition (same number of data items), and the state PDF is a 40D full covariance Gaussian, there are 860 parameters that need training, which means $1301/860 \approx 1.5$ data items per parameter which is certainly too little.

4.4 Training The Signature Model

This section deals briefly with our process of creating a signature model for a user. Naturally, the first step is for the user to provide a number of signatures from which the model can be estimated. These should be collected under conditions as similar to the final environment of the HSV system as possible (i.e. will the user be sitting or standing, and

have to sign in a box, on a line, or unconstrained?). Our system uses fifteen signatures for training.

The state PDFs are the first parts of the model to be created. Each signature is segmented into 20 equal (or as close as possible if not exactly divisible by 20) sections that are used to train the corresponding PDFs using maximum likelihood estimation (MLE).

These 20 PDFs are then inserted into a 22 state (recall the initial and final NULL states) left-to-right HMM. The HMM is then re-estimated using a number of iterations of the Viterbi algorithm [22]. Thus, the previous hard segmentation (into 20 equal parts) was only a 'guideline' and this re-estimation will segment the signatures automatically at more appropriate locations.

The left-to-right HMM is then expanded to an identically behaved modified Ferguson HMM with explicit time duration modelling. Next the duration fanout transition smoothing scheme is initialised. The exact initialisation technique depends on the type of smoothing. For example, the GMMs are estimated from the duration fanout probabilities and positions using MLE.

Finally, the new model is trained once more with a number of iterations of the Viterbi algorithm, again using all the fifteen signatures. This results in a well-trained HMM that performs correctly.

4.5 Reducing The Footprint Of The Model

Thus far, no attention has been given to the storage size of the model that represents each user's signature. This is an important consideration since the signature model will surely be used in applications where the models must be saved on portable digital media or transferred over limited bandwidth networks. Current technology uses smart, or magnetic stripe cards to hold such information, and they have a limited storage capacity (200 bytes to 64kB). It is therefore of primary importance to keep the size (footprint) of the stored model down.

All the files handled in these experiments (models, data and experimental results) are saved either as human readable text files, or compressed versions thereof. In the PatRecII environment this is performed by using the Lempel-Ziv algorithm [17] for encoding (compressing) and decoding (decompressing). This is essentially the same method as used by programs such as *gzip*, *zip* and *pkzip*, and is fully integrated into the PatRecII file

input/output routines.

To demonstrate the compression ratios, the sizes of the raw data of an average signature in various file formats are considered. An average signature with a text format size of 12.0 kB is reduced to 3.0 kB by *gzip*. This same data saved as a Matlab binary file has a size of 5.4 kB. Clearly there is enough redundancy in the data for Lempel-Ziv encoding to out-perform the minimum raw data storage size.

The general PatRecII file protocol that users' models are currently saved in, also contains a large percentage of redundant information. This is because all the information necessary for reconstructing a huge variety of HMMs and other models is saved. Those HMMs and other models may also have a much higher information density than our HMMs presented above. For example, the structure of the Ferguson HMMs used throughout these experiments can be completely specified by 2 integers—one denoting the number of states in the original left-to-right HMM, and the other stating the order of duration modelling required. If the structure of the model is known beforehand, as would be the case in this signature verification system, only the unknown component parameters need be saved.

The size of a 20 state HMM with order 20 duration modelling, GMM smoothed transition links (which already reduces the number of parameters per duration fanout from 20 to 9) and 5D state PDFs, is 48 kB (zipped: 8.9 kB).

The link destinations are also fully specified by the Ferguson structure, so they can be removed to bring the size down to 27 kB (zipped: $4.3kB$).

All the unnecessary link probabilities, like those definite 1 probabilities down the duration stack, can then be removed leaving 14 kB (zipped: 3.9 kB).

The internal text tags describing the component objects, such as Gaussian_N for the state PDFs, and MixturePDF_1 for the smoothed transition links, can also be left out, which results in a final model size of 12kB (zipped: 3.4 kB). This size may vary slightly for different signatures.

This is a convenient size (comparable to that of the original raw signature data), and while not being able to fit on a magnetic stripe card, will easily fit on the new smart cards. If the verification is to be performed on an external computer[2], 3.4 kB can be uploaded from a smart card in just less than half a second using standard serial protocols.

---

[2]This will always be the case until more powerful processors become available on smart cards, which is scheduled to be within the next two years.

4.6 Summary

All the components of the signature model have been discussed and the HMM has demonstrated its suitability in some of the areas mentioned at the start of this chapter.

HMMs are

- accurate—low FA and FR rates can be achieved (far better than a human operator ever could.)
- small—they can easily be stored on smart cards and are about the size of the text version of a signature.
- secure—it would be extremely difficult for an impostor to learn how to forge a signature from its HMM representation.
- adaptable—HMMs can easily be adapted [4] as more signatures become available, though a investigation of the time-varying nature of signatures is beyond the scope of this research.
- fast—it takes between 120 and 390 milliseconds to reach an accept/reject decision with the system presented in this text on a 700 MHz processor running Linux.
- flexible—HMMs do not assume any special characteristics to be present in a signature. If a signature is exceedingly long or short though, it will become easier to forge. This is respectively because a state will either have to model too much information, or the model will not contain enough distinguishing information for accurate verification.

Chapter 5

Verification

The terms identification and verification are often used interchangeably though they refer to two very different modes of user validation. Whereas identification or recognition attempts to select the most likely user from an enrolled group, verification must decide whether a presented user is who he/she claims to be. In identification trials, the data is matched to the enrolled models until certain criteria are fulfilled, or the highest score, or lowest distance measure is chosen. On the other hand, verification has a little more information available since the presented data is accompanied by an identity claim. Verifiers can use this extra information to separate the target model from the impostor models, which allows the use of quite complex techniques to determine whether the presented data should be accepted or rejected. The term *impostor models* refers to all the models in the verifier besides the claimed model. These impostors are usually also enrolled users in the system, and they are used to get an idea of the impostor score distribution. This is equivalent to applying Casual forgeries to these impostor models.

The topic of verification has been largely neglected in this field. After all the complex signature modelling has been done, a simple verifier is usually hacked together and left to achieve the most important task of acceptance/rejection. Numerous verifiers have been invented for other pattern recognition fields, but that does not mean they are suitable for signature verification. The choice of verifier is very important. A verifier that performs brilliantly in speaker recognition tasks, where the impostors do not deliberately try to impersonate genuine users, will not be the correct choice for signature verification where these deliberate forgeries would foil the system.

Another challenge encountered while trying to select/create a suitable verifier, was the huge variance of the log likelihood scores obtained whenever impostor signatures were applied to the HMMs (see table 5.1). Certain verifiers attempt to model the impostor

scores with 1D Gaussian PDFs, which clearly is not going to work when the HMM log likelihood scores lie in a possible range of -10 to -10^8 and sometimes even down to -10^{150} and -10^{208}! The genuine log likelihood scores lie in the -10 to -1 range.

-1.4×10^2	-7.7×10^8	-6.3×10^8	-5.6×10^6	-1.3×10^9	-2.4×10^7
-1.0×10^2	-4.2×10^5	-2.0×10^8	-1.4×10^6	-9.2×10^1	-1.8×10^8
-1.5×10^7	-7.4×10^5	-8.2×10^1	-7.3×10^8	-7.9×10^1	-1.5×10^7
-1.3×10^7	-2.6×10^7	-1.1×10^2	-7.3×10^9	-7.7×10^1	-8.1×10^8
-5.5×10^8	-4.2×10^1	-3.3×10^8	-7.2×10^1	-4.2×10^8	-3.8×10^8
-7.2×10^7	-7.8×10^6	-3.4×10^7	-1.4×10^7	-4.6×10^7	-1.0×10^7
-7.3×10^0	-8.9×10^1	-2.9×10^8	-1.0×10^9	-2.2×10^9	-3.9×10^1

Table 5.1: *Example impostor scores rounded off to two significant digits. These are already in log likelihood format!*

The four different verifiers investigated do have certain common characteristics. They all

- contain a bank of trained models—one per user.
- are presented with target data and an accompanying identity claim. In this case, the data consists of all the feature vectors of one complete signature. These feature vectors could have already been transformed by any number of preprocessing transformations.
- return a single confidence value per signature. This confidence value is a number between 0 and 1 that signifies how sure the verifier is that the presented data belongs to the claimed model. Each verifier calculates these values differently, and, as such, different verifiers' confidence values can not be compared directly.

For investigation purposes, these confidence values are then compared to between 500 and 2000 thresholds between 0 and 1. Those below each threshold are rejected, while those equal or above, are accepted. From the number of counts of False Acceptances (FA), True Acceptances, False Rejections (FR) and True Rejections at each threshold value, the FA and FR probability curves can be plotted.

All these verification trials have been performed with global (user independent) threshold values. Some verifiers do use normalisation schemes to factor out individual variations, though it is expected that error rates would decrease if each users' accept/reject threshold value was calculated individually. Ideally, a number of deliberate forgeries would be

needed to train an impostor score PDF as well as a genuine score PDF. From these distributions, an optimal threshold could be chosen for each user that also best suits the application's inherent trade-off between FAs and FRs. This, however, is not a real-world feasible idea for all but the most stringent security applications.

The following four sections describe the operating principles and properties of the different verifiers investigated. Following that is a section on Detection Error Trade-off (DET) curves that simplifies the objective evaluation of different verifiers.

5.1 Ranking Verifier

This basic verifier probably has the most intuitive operation. The incoming signature is presented to the claimed model to obtain a target score. The signature is then presented to all the other users' models (50 in our case) to create a vector of impostor scores. The confidence value is calculated by counting the number of impostor scores below the target score, and then dividing by the total number of users (a total of 51 users). This is equivalent to ranking all the users' (impostors and target) scores in increasing order and returning the target's rank index (rank index $\in \{0, 1, 2, \ldots, N-1\}$ where N is the total number of classes) normalised by the total number of models. The highest possible confidence value is therefore $1 - 1/N$. The rationale for never allowing the confidence to equal 1 (i.e. being 100% sure that the data belongs to the claimed model) can be justified by thinking about the following problem: if the data was completely random, there would still be a $1/N$ probability that the claimed class was ranked highest. It would not be good to state that we are 100% positive that random data belongs to a specific user though.

The ranking verifier concept is illustrated in Figure 5.1. The confidence curve shows the confidence value that would be returned for target model scores of different values. This curve has a value of zero for all scores extending to the right of this graph. As can be seen, the returned confidences take on discrete values. This can be quite a disadvantage, especially when there are a small number of classes and the error rates are very low. This is the case with our 51 users and low error rates. The step size of the confidence values is $1/51 \approx 0.019607 \approx 2\%$ which makes it impossible to measure accurate EE rates as they are often less than 2%.

This verifier has the following advantages and disadvantages:

✓ No pre-training or training data needed.

✓ Can cope with any range of impostor scores (-10^{208} is below -10, just as -100 is).

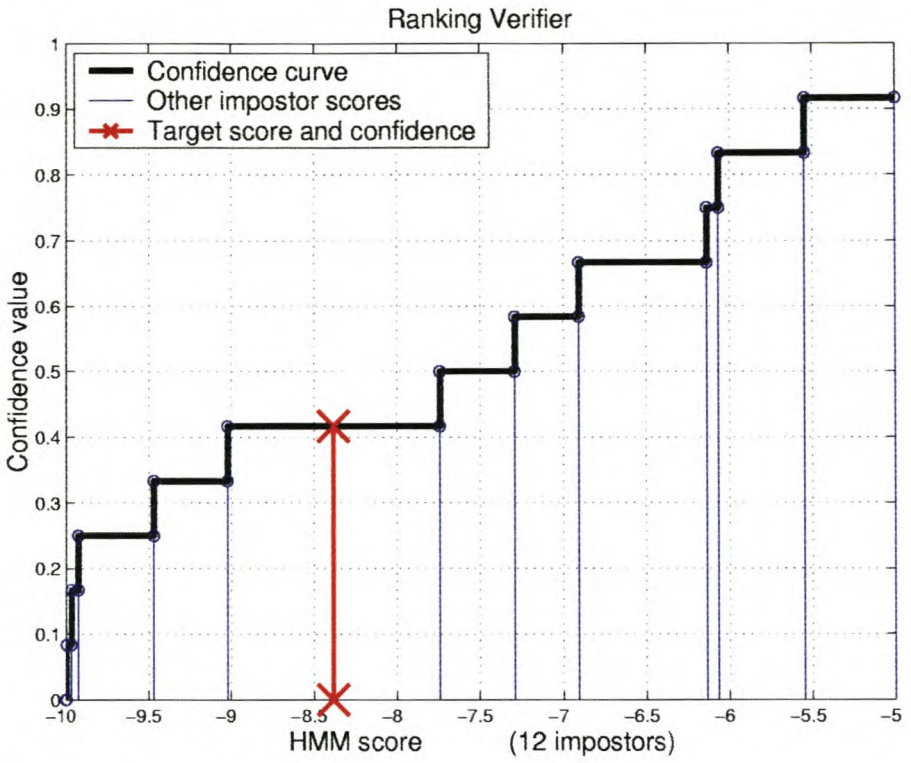


Figure 5.1: *Ranking verifier: The thicker line shows the confidence values that would be returned for different target scores.*

- Simple logical operation.
- No need for a background (world) model or cohort normalisation, as all models should be equally affected by global feature variations.

× Returns a discrete valued confidence.

× Impostor-centric verifier, therefore does not cope well with deliberate impostor attacks. This is because a deliberate (Over-the-shoulder, Home Improved or Professional) forgery is a skilled forgery to the target model, yet only a Casual forgery to all the other models. Naturally, then it will obtain a better score for the target model than for the others, and so be ranked very highly. The term impostor-centric is a bit of a misnomer since the forgery is only a real impostor to one model, i.e. the target model.

× Slow—the target data must be applied to all the models in the verifier to see where the target would be ranked.

5.2 Continuous Ranking Verifier

As mentioned above, the discrete confidence values are a major shortfall of the Ranking verifier. The novel Continuous Ranking verifier overcomes this problem by producing continuous valued confidences. It operates in much the same manner as the Ranking verifier by ordering all the impostor scores, and then determining where the target score would fit in. But now, instead of returning a confidence value equal to the number of lower scores divided by the total number of scores, the confidence value is linearly interpolated from the hypothetical confidence values of the impostor scores ranked just above and just below itself. This has the two-fold advantage of returning a more representative confidence value, and also smoothing the FA and FR curves and therefore the DET curve. This discrete-to-continuous transformation from Ranking to Continuous Ranking verifiers allows a much more accurate threshold value to be chosen.

If the target log likelihood score is ranked as the top score (as we hope happens for all genuine signatures), an exponential function is used to extrapolate a confidence value that is greater than $1 - 1/N$, yet never quite reaches 1 (or 100%). The procedure for determining this exponential function is described below.

First, the hypothetical confidences are calculated for the first and second highest impostor scores in the same manner as used in the Ranking verifier. Then the equation of the line passing through these points is determined:

$$y_1 = mx_1 + c, \tag{5.1}$$

$$m = \frac{C_U - C_L}{S_U - S_L}, \tag{5.2}$$

$$c = C_U, \tag{5.3}$$

where:

S_U is the upper (top) impostor log likelihood score,

S_L is the lower (second highest) impostor log likelihood score,

C_U is the confidence value of S_U,

C_L is the confidence value of S_L,

x_1 is the HMM score axis with the origin at S_U,

y_1 is the confidence value axis.

A function that asymptotically approaches a confidence of 1, and passes through (S_U, C_U) with the same gradient as the line between (S_L, C_L) and (S_U, C_U) is given by:

$$y_2 = 1 - ae^{-b(S_T - S_U)}, \tag{5.4}$$

where:

$a = 1 - c$,

$b = m/(1 - c)$,

S_T is target (claimed model) log likelihood score,

y_2 is the target confidence value.

This monotonically increasing extrapolation is exactly what is needed to produce higher confidence values than the $1 - 1/N$ of the ranking verifier, and yet never reach 1.

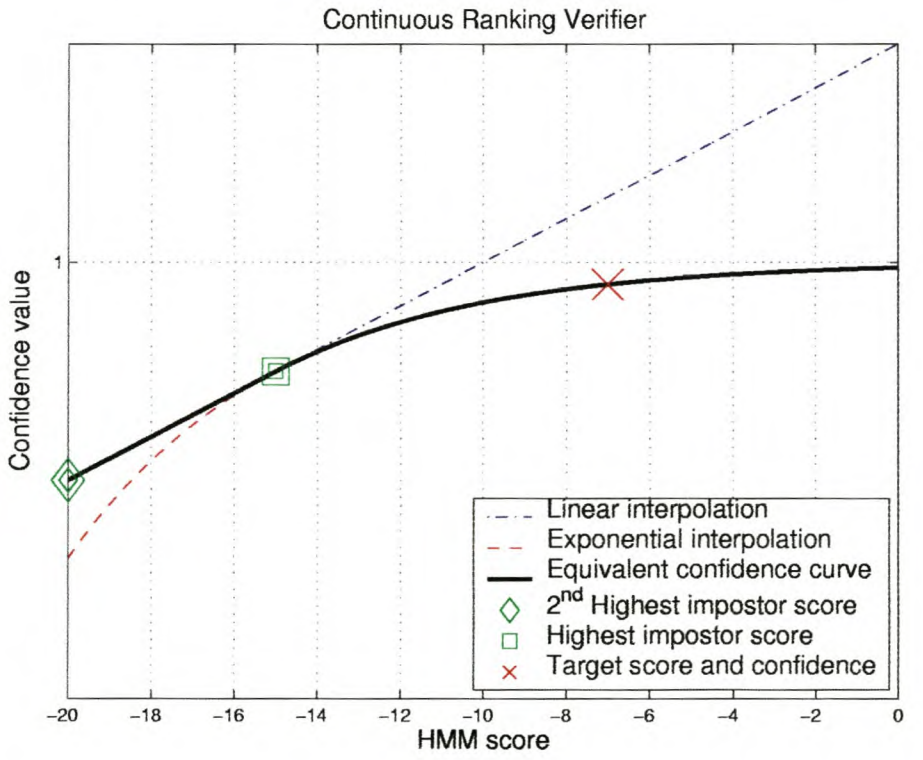


Figure 5.2: *Continuous Ranking verifier: The solid line shows the confidence values that would be returned for different target scores.*

The operation of this verifier is demonstrated in Figure 5.2. It shows the special case of the target score being the highest score. The confidence curve is shown for target scores from the second highest impostor score, to well above the highest impostor score.

This verifier has the following properties:

✓ Returns a continuous valued confidence.

✓ No pre-training or training data needed.

✓ Can cope with any range of impostor scores (-10^{208} is below -10, just as -100 is).

- Easy understandable operation.
- No need for a background (world) model or cohort normalisation as all models should be equally affected by global feature variations.

× Impostor-centric verifier, therefore does not cope well with deliberate impostor attacks. Deliberate forgeries are expected to be better than casual forgeries.

× Slow—the presented data must be applied to all the models to obtain scores that then have to be ranked.

5.3 Test Normalisation Verifier

The Test Normalisation or T-Norm verifier [1] is a relatively new verifier that performs very well in speaker recognition tasks. In essence, it is an impostor-centric verifier that estimates the mean and variance of the impostor scores' distribution. The presented signature is applied to the claimed model and to all the other models in the verifier. These impostor scores are then used to estimate a Gaussian PDF using MLE. The returned confidence is then the value of this Gaussian's cumulative distribution function (CDF) evaluated at the target score, as depicted in Figure 5.3.

Since the target score is normalised according to the impostors' performance on the actual test data (presented signature), this verifier is not too affected by data quality/quantity variations, since these should influence the impostor and claimed models together.

As mentioned in the beginning of this chapter, the impostor model scores (or model's Casual forgery scores) are very widely distributed, from -10^{208} to around -1, even though they are already in log likelihood format. A Gaussian PDF clearly will not perform well when estimated with training data in this range. Therefore, only the cohort (those very close to the target) impostors' scores were used for the PDF estimation. It was found that taking the top 10–14% (5–7 scores) of the 50 impostors' scores worked best.

The T-Norm verifier has the following properties:

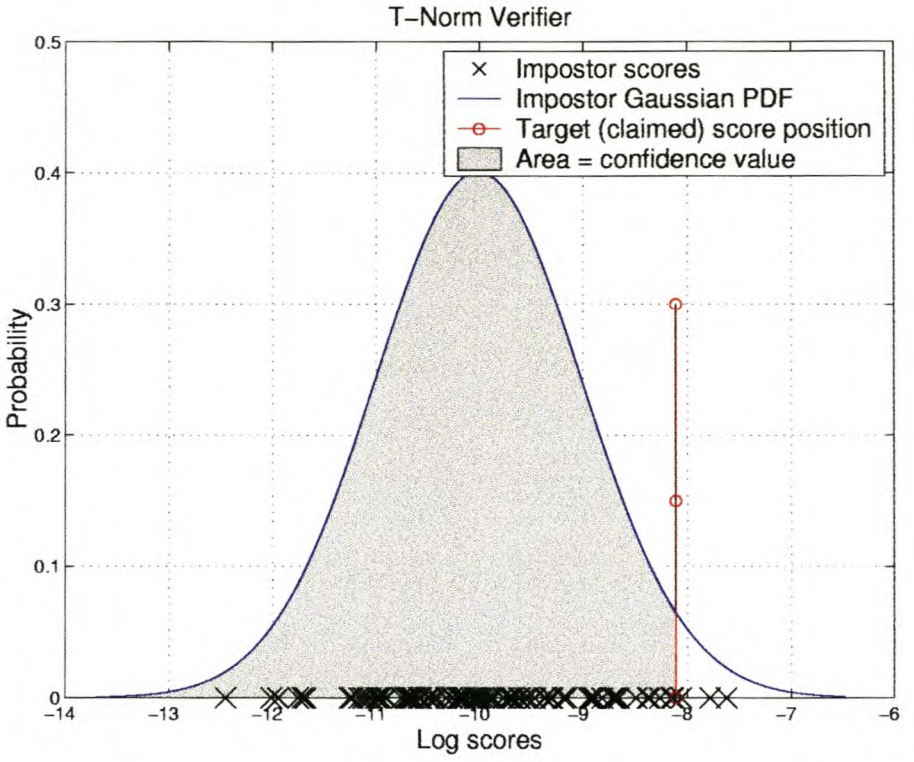


Figure 5.3: *T-Norm verifier: The area under the Gaussian to the left of the target score is the confidence value.*

✓ Returns a continuous valued confidence.

✓ No pre-training or training data needed.

✓ The internal Gaussian PDF is estimated from scores using the exact same data as the target score. Therefore, quality variations in the presented data should not affect the confidence value.

• Tractable operation with a solid statistical footing.

• Cohort normalisation is a good idea as all models should be equally affected by global feature variations.

× Very slow—the presented data must be applied to all the models to obtain the scores needed to train the impostor Gaussian PDF.

× Can not cope well with the huge range of impostor scores, even if just the cohort (those close to the genuine) scores are used to train the impostor PDF.

× Impostor-centric verifier, therefore does not perform well in deliberate 'impostor' attacks. Again, this is only because we are not using real impostors to estimate the internal Gaussian PDF, but rather applying a signature to other users' models get an approximation of an impostors behaviour.

5.4 C-Norm Verifier

Where the T-Norm verifier attempts to model the scores of a known set of impostors, the C-Norm attempts to model the scores of a known set of genuine signatures. The motivation for this approach is clear: these HMMs are generative models that are trained to recognise genuine data and become as close as possible to an entity that would generate a signature given random noise. They are not trained to discriminate between impostor and genuine signatures, nor to reject impostor signatures, rather, they are trained to recognise genuine signatures. Therefore it makes sense to base a verifier on the genuine score distribution rather than an impostor score distribution. This is especially important when verifying biometrics in which there may be deliberate impostor attacks or forgeries, since these will most probably achieve higher scores than most random impostors and will therefore be accepted. On the other hand, it is very unlikely that these forgery scores will be above most, if any, genuine scores, which should cause even these deliberate impostors to be rejected by a target-centric verifier. It is also unfeasible in all but the most important security applications, to collect deliberate forgeries to estimate a deliberate impostor PDF for each user.

To train the internal genuine one dimensional PDF, a development or validation set of data is needed. This data must be separate from the training data (used to train the individual component HMMs), as well as the testing data (used to evaluate the system's performance). Failure to adhere to this will respectively produce worse results (because of an overly optimistic genuine score PDF), or will just be cheating.

A number of development signatures are applied to their respective models and the resulting scores are used to train the internal PDF. After observing the genuine development signatures scores' distribution, a single Gaussian PDF was deemed to be the most fitting for this application. We empirically determined that a development set of 5 signatures per user was needed to estimate this PDF accurately enough.

There is also an option to normalise all the internal processes with a background model. A background model (trained on all users' training data) ensures global variations between databases are normalised out. However, with the impostors' models produc-

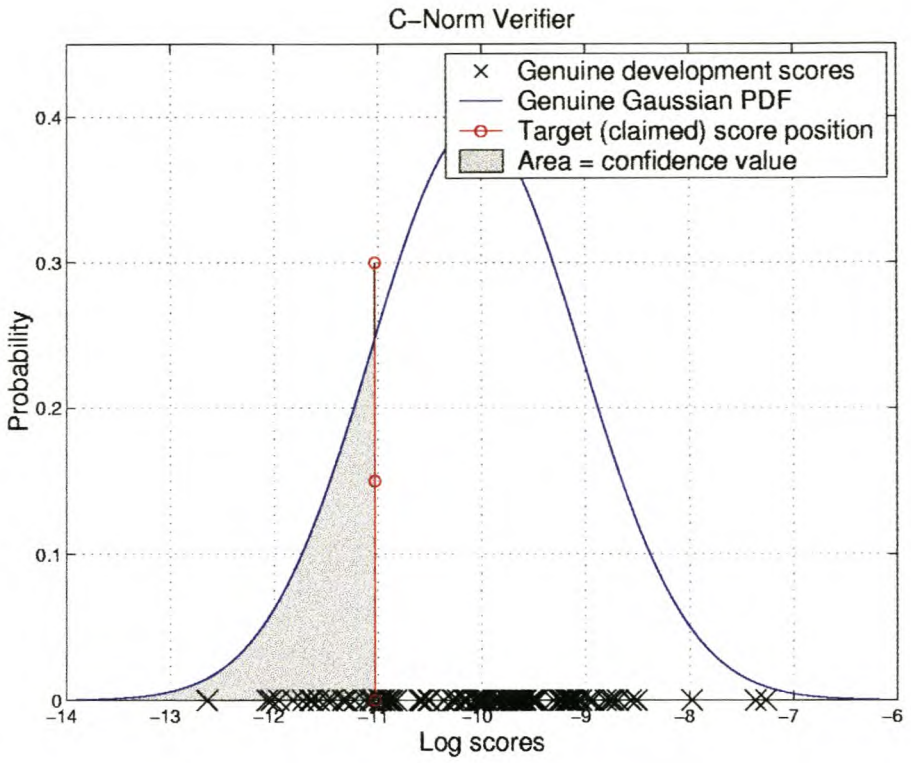


Figure 5.4: *C-Norm verifier: The area under the genuine score PDF (a Gaussian in this case) to the left of the target score is the confidence value.*

ing log likelihood scores spanning so many orders of magnitude (see Table 5.1) and the background model not faring much better, this normalisation leaves the above mentioned Gaussian PDF the task of representing scores over many orders of magnitude. Naturally this causes the PDF's variance to explode, which makes the verifier essentially useless. It can not discern impostor/genuine scores separated by a couple of orders of magnitude, while the PDF was trained on data spanning around nine orders of magnitude. Therefore, no background model was used.

The C-Norm verifier has the following properties:

- ✓ Returns a continuous valued confidence.
- ✓ Target-centric verifier, therefore performs well detecting deliberate forgeries/impostors.
- ✓ Fast—the presented data is applied to only one model. The total time taken to verify an average signature varies between 120 and 390 milliseconds depending on the preprocessing configuration.
- ✓ Different PDFs can be chosen depending on the development scores' distribution.

✓ Similar statistical footing as the widely accepted T-Norm verifier.

✓ The genuine scores' distribution from generative models (like HMMs) is much better behaved than that of the impostors. There are no wildly varying scores to try and represent.

- Background model normalisation is possible.
- Requires pre-training. The once-off internal PDF estimation time is negligible for each user.

× Requires separate set of development training data. This is a problem for limited sized databases.

× The verifier does not adjust for each instance of submitted data. The internal genuine score PDF is pre-trained and can not dynamically adapt according to the quality of the presented signature.

5.5 Detection Error Trade-off Curves

The normal method of displaying the results of verification trials are the False Acceptance and False Rejection curves. These curves plot the number of incorrect classifications divided by total number of classifications for a range of threshold values. There may be more than one FA curve per plot; for example, one each for all the falsely accepted Casual, Home Improved, Over-the-shoulder, and Professional forgeries.

Properties like the equal error (EE) rate (the error rate at which the FA and FR plots intersect) can be then easily be read off the graph. However, these plots make it rather difficult to evaluate the overall performances of the verifiers and models, or to make objective comparisons between different verifiers/models, since the FA and FR plots are often wildly different (see Figure 5.5). From another perspective, some verifiers have EE rates at thresholds near zero, other's EE rates (and surrounding high 'action' areas) are nearer a threshold value of one, which makes them difficult to collate.

These problems can be overcome by plotting the FA and FR rates against one another for corresponding thresholds, as opposed to against a range of thresholds. The resulting plots are called Detection Error Trade-off curves [19] [1] or DET curves. These curves are similar to Receiver Operating Characteristic (ROC) curves, but whereas ROC curves plot the False Acceptances (or False Alarms) versus the *Correct* Rejections (or Detections)

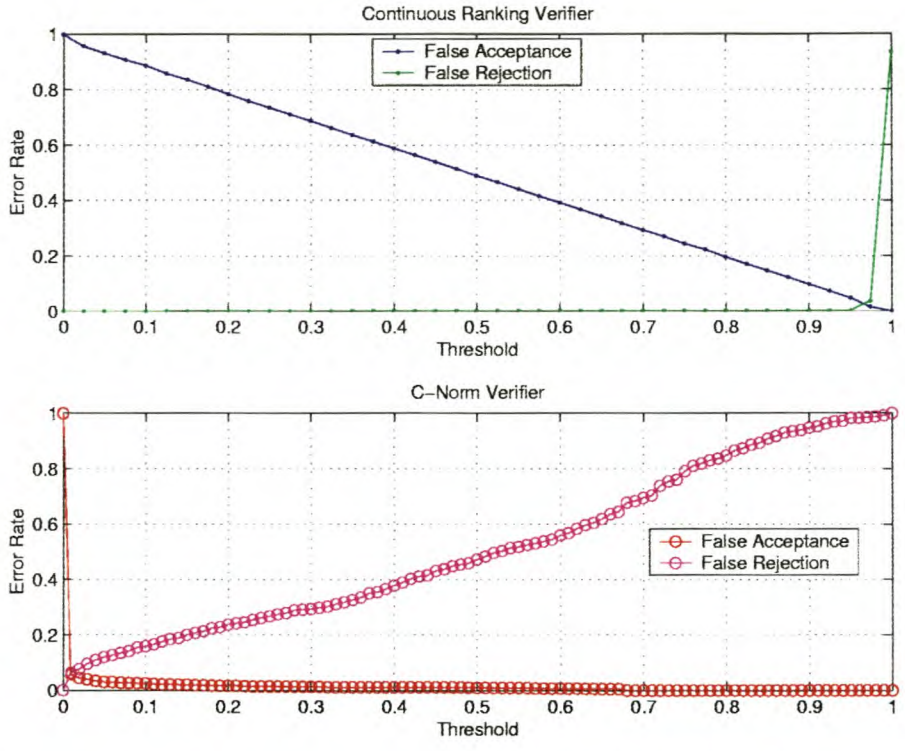


Figure 5.5: *FA and FR curves for a Continuous Ranking verifier and a C-Norm verifier. Note how difficult it is to evaluate their relative performance, even if they were plotted on the same axes.*

on linear axes, DET curves plot False Acceptances versus *False* Rejections on non-linear axes.

The axes of these plots are often scaled unconventionally to bring out useful characteristics in the verification score distributions. Scaling the axes according to a Gaussian CDF means that target or impostor score distributions will be plotted as straight lines if they themselves are Gaussian distributed. It also then has the property that a shifting of the mean of one of these distributions just shifts its DET curve closer or further from the origin, while a change in its variance rotates the DET curve. Figure 5.6 shows the DET curve version of the same results as shown in the FAR curves of Figure 5.5. Now, it much more obvious that the Continuous Ranking verifier performs better (has a lower error rate) than the C-Norm verifier in this case. The dotted diagonal line illustrates where the equal error rate occurs. DET curves are used extensively in presenting the results of this research.

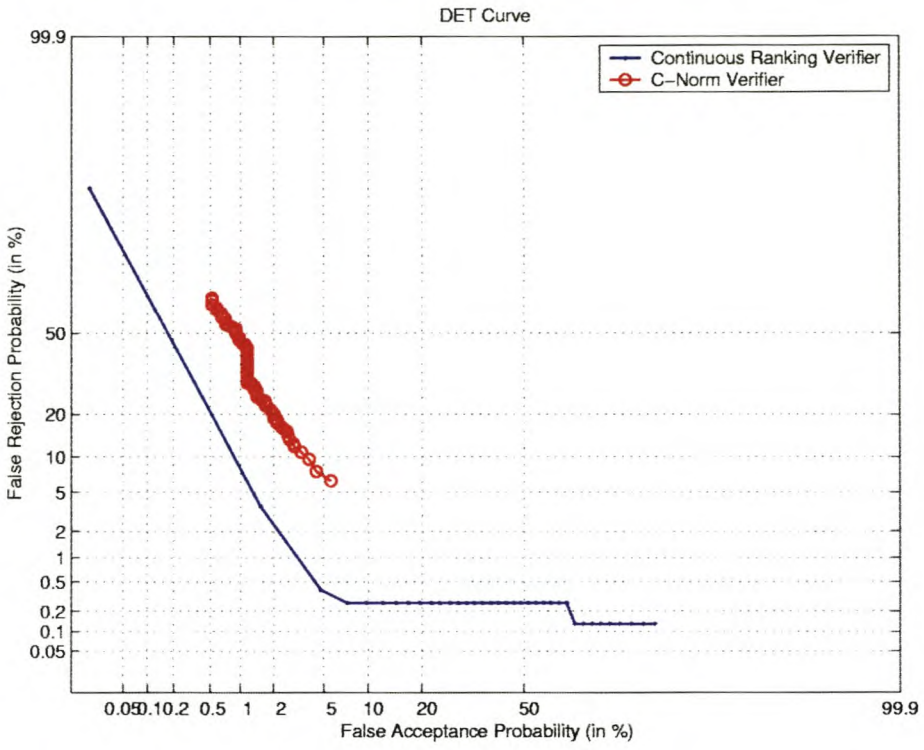


Figure 5.6: *DET curve display of the same FA and FR results. Now it is clear that the Continuous Ranking verifier has the lower error rate.*

5.6 Summary

The choice of verification method is extremely important in HSV as has a great influence on the system's error rates. Four verifiers were investigated, namely the Ranking, Continuous Ranking, T-Norm and C-Norm verifiers. The various pros and cons of each are presented. The C-Norm was designed specifically for our HSV task and is therefore the most accurate in our trials.

DET curves are a valuable technique for displaying the results of verification experiments, and are used later in preference to the traditional FAR curves.

In the next chapter, we move our focus to the experiment results that influenced the development of our HSV system.

Chapter 6

Experimental Investigations and Results

The results of the experimental investigations are presented in this chapter: firstly, the identification trial results are introduced, then the verification outcomes are covered. Where useful, supplementary tables and figures are included in Appendix A under matching numerical headings (i.e. Section 6.2.5 maps to Appendix A.2.5).

6.1 Identification Trials

Before signature verification was addressed, the less complicated challenge of signature identification was investigated. These trials were used initially to decide on certain model parameters such as the number of training signatures, number of original states, order of duration modelling, method of transition smoothing, and selection of preprocessing transformations.

Identification trials attempt to recognise the signers from presented signatures. These trials were not designed to simulate real-world situations, rather they provide a quick method for evaluating the performance of different model parameters. These tests were performed with a 765 strong set of genuine test signatures, 15 from each of the 51 users, that were individually presented to a bank of 51 trained user models. The model which returned the highest score was identified as the signature's creator.

After a number of design iterations, the success rate in these trials became so high ($\approx$100%) that statistically significant improvements could not be measured. This is entirely understandable since each signature should match its respective model well, and is equivalent to a casual forgery for all the other users' models. Despite the impracticalities

of these trials, they were a useful set of experiments to run, because they highlighted the strengths and weaknesses of various model and preprocessor configurations.

Table 6.1 shows the percentage of successful identifications using a Ferguson HMM with Gaussian Mixture Model transition smoothing with 3 underlying densities. A general observation from the identification trials is that there seems to be little increase in accuracy when more than 20 original states (in the left-to-right model) and 20^{th} order duration is used.

	Duration Order					
Original States	**10**	**15**	**20**	**25**	**30**	**35**
10	99.86	99.60	99.60	99.73	100	99.86
15	99.73	99.73	100	99.86	100	100
20	99.73	100	99.86	100	100	100
25	99.73	100	100	99.86	100	100

Table 6.1: *Success rates of the identification percentages using only the X and Y features with Delta3 preprocessing transformations. The first column shows the number of original states in the HMM.*

Only 2 feature vector dimensions were used to get these results, those of the *delta*3 (Equation 3.4 on page 27) transformed X and Y dimensions. The preprocessing normalisers that were used, in order, were:

1. the spatial normaliser (as always),
2. the *delta*3 (only on the X and Y dimensions),
3. the feature vector selector (to remove all of the original 5 dimensions, X, Y, P, θ_x, and θ_y, leaving just *delta*3(X) and *delta*3(Y)),
4. and then the scaler (to ensure the data is well scaled).

All the *delta* preprocessors performed exceptionally well in the identification trials, with the *delta*2s being the best. The reason for these successes, using only the spatial information, is clearly because no deliberate forgeries were considered. Thus, the X and Y positional information, and even more so with the velocity information, is all that is required to choose the correct identity very accurately. In contrast, when these features were used with deliberate forgeries in verification trials, the outcomes were much less favourable.

These trials provided the following answers to the decisions required in the first paragraph of this section.

- Number of training signatures: 15—this was also found to be optimal by le Riche [16] and Dolfing [2].
- Number of original states: 20.
- Order of duration modelling: 20.
- Method of transition smoothing: 1D Gaussian Mixture Models with three underlying density functions.
- Preprocessing transformations: the spatial normaliser, and as many scalers as necessary were always used.

6.2 Verification Trials

This section covers the experiments that demonstrate important signature verification findings. All these trials use the 3130 forgeries, as well as the either 510 genuine signatures, in the case of verifiers needing a development set of 5 signatures per user, or 765 genuine signatures otherwise.

Since it is impractical to test every possible parameter combination, the improvements are assumed to be independent, and each incremental improvement from varying a parameter is kept when moving on to vary the next parameter. The running of exhaustive tests for certain parameter combinations did seem to verify this believe. The term parameter is used to mean any model attribute or data preprocessing transformation.

All the following trials used Ferguson HMMs with 20 original states and 20 duration states. Furthermore, GMMs with 3 underlying Gaussian densities are used to smooth the duration fanout links, their viability being proved in the above identification trials. The data is always preprocessed though the spatial normaliser, as well as through as many scalers as necessary.

6.2.1 Impostor-centric Verifiers

The first verifiers that were investigated were the impostor centric verifiers, the T-Norm and the Continuous Ranking verifier. The normal discrete Ranking verifier was also considered, but due to the disadvantages noted in Section 5.1, it was discarded early on.

As discussed in Chapter 5, these verifiers do an excellent job of identifying the casual forgeries. However, as soon as certain impostors distinguish themselves above the rest, as in the case of deliberate forgeries, the verifiers break down.

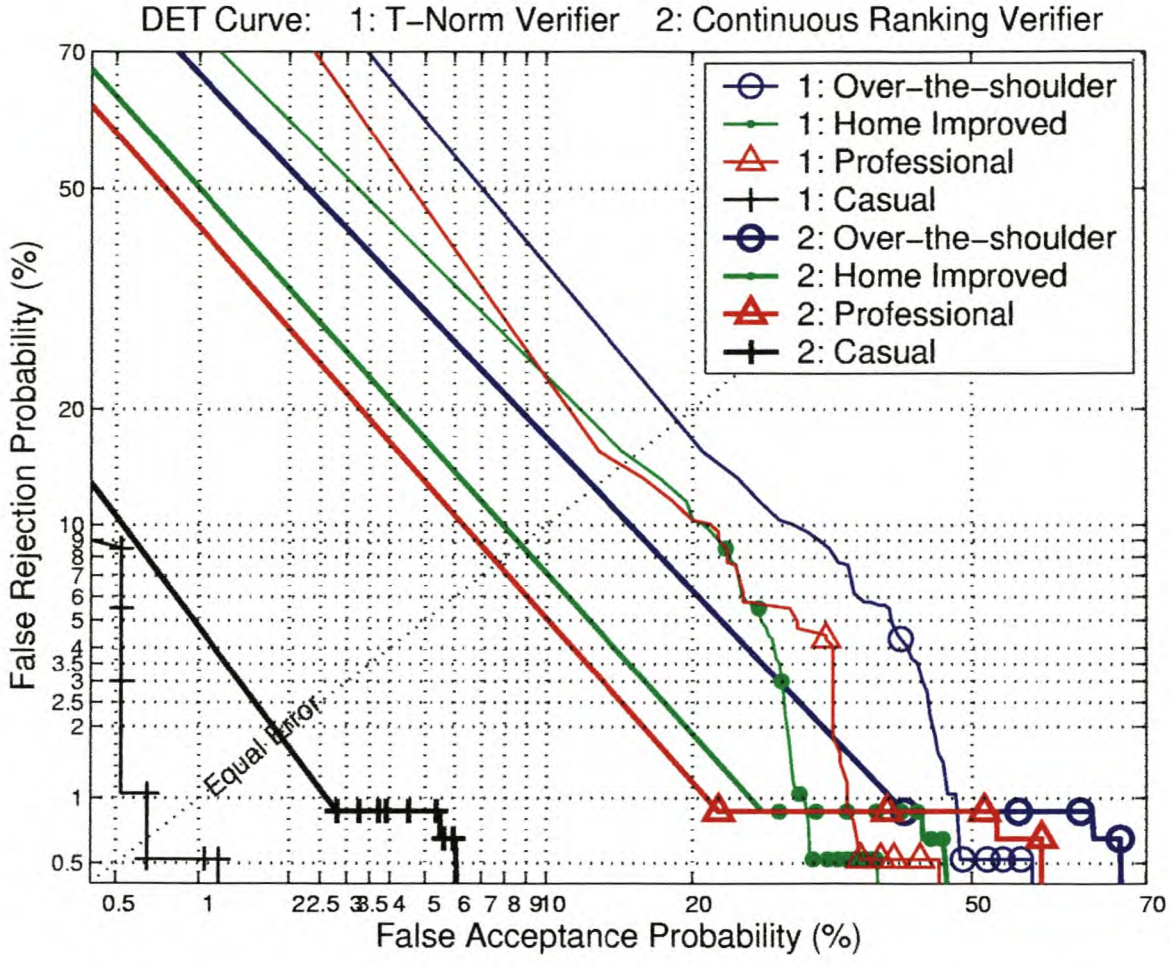


Figure 6.1: *DET curves of 2 impostor-centric verifiers: the T-Norm and the Continuous Ranking verifiers. The long straight line sections between data points of the DET curves are interpolated on non-linear axes, and may therefore be inaccurate.*

Most of the data points on these two graphs are in the region of high false acceptance and very low false rejection probabilities. Then the plots shoot to the $(FA, FR) = (0\%, 100\%)$ point. The lack of data around the EE rate in these specific plots makes it difficult to interpolate accurate EE values. The values in Table 6.2 are obtained from the FAR graphs which should be more accurate since the axes are linear. Note that the data points are not generally organised in a straight line, which, recalling from Section 5.5 on DET curves, means that the distributions of the different forgeries' scores are not actually Gaussian.

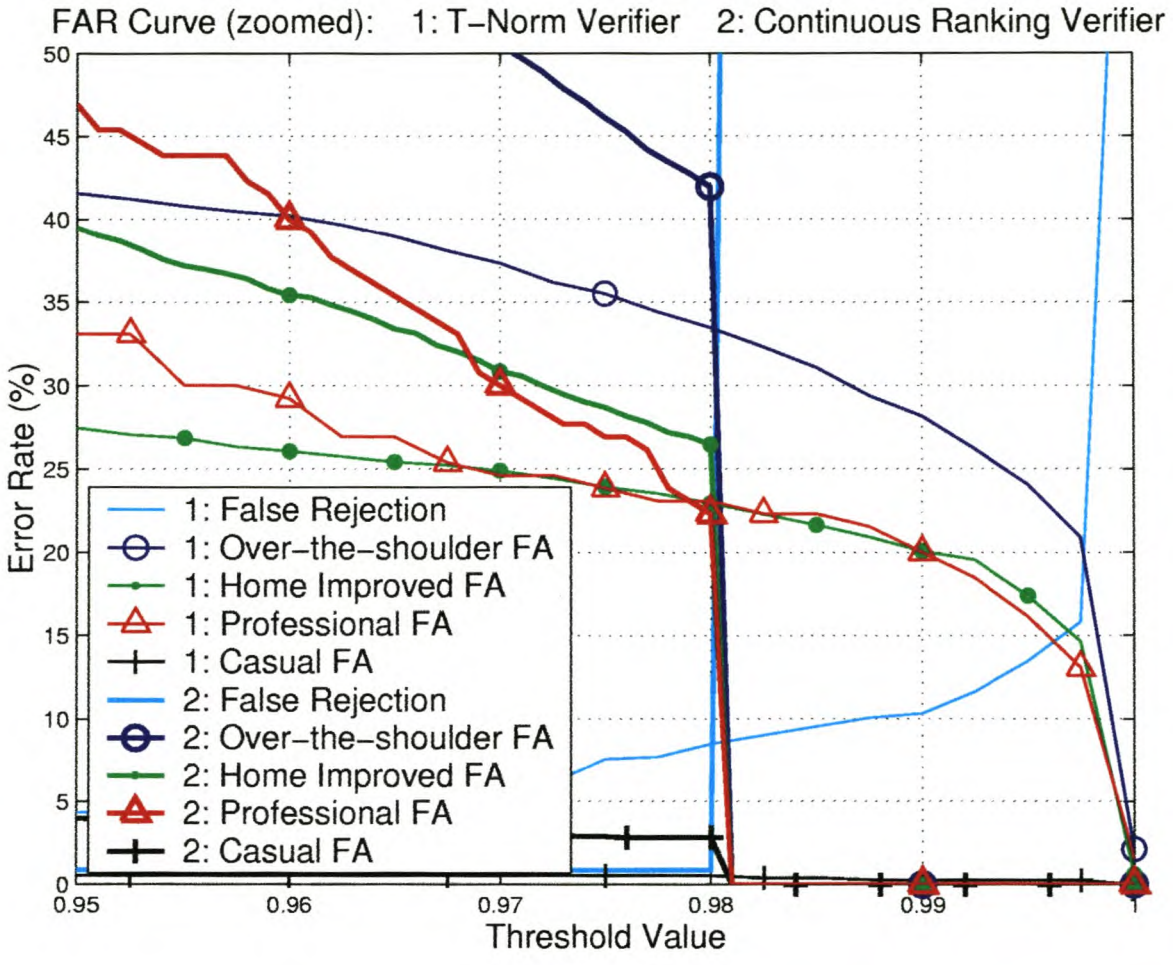


Figure 6.2: *Close up view of the FAR curves of the T-Norm and Continuous Ranking verifiers showing the equal error region.*

Forgery Type	T-Norm	C-Ranking
Over-the-shoulder	**19.7**	29.7
Home Improved	**15.3**	21.1
Professional	**14.6**	18.4
Casual	**0.7**	2.8

Table 6.2: *Equal Error percentages from the FAR plots for the T-Norm and Continuous Ranking verifiers. The bold font shows the lowest error rates for each forgery class.*

Discussion

The T-Norm verifier used here actually incorporates a form of cohort normalisation. This is used because of the difficulty of modelling the hugely varying impostor log likelihood scores, as mentioned in Chapter 5, and is achieved by estimating the internal Gaussian PDF on only the top 10% of the impostor scores. The value of 10% was determined empirically and corresponds to 5 scores from the 50 impostors available.

These verifiers, especially the T-Norm, achieve excellent results for casual forgeries, which shows why it is one of the premier verifiers used in speaker recognition tasks. In contrast, as soon as educated impostor signatures (deliberate forgeries) are used, the error rate climbs sharply. When the operating principles are recalled (Sections 5.2 and 5.3), it becomes clear why these verifiers should not be used in our test cases. These verifiers would most probably perform extremely well if *real* impostors with deliberate forgeries for each model were used. This is unfeasible, which is why Causal forgeries were used instead for estimating the "impostors' " behaviour.

It was deemed important to produce a verification system that can thwart deliberate attacks and for this reason, the impostor-centric (Ranking, Continuous Ranking, and Test Normalisation) verifiers were abandoned. The more important classes of forgeries are the Over-the-shoulder and Home Improved categories because we aim to foil deliberate attempts to beat the system, though the Casual forgeries are not forgotten. There are relatively few Professional forgeries and the system seems to have less of a problem detecting them compared to other types. For these reasons, the C-Norm verifier was used in all following trials.

6.2.2 Using different Feature Dimensions

These trials show the effects of using various preprocessing methods to transform the original feature vectors. The salient EE rates are presented in Table 6.3.

Discussion

Using just the spatial (X and Y) dimensions leads to the very poor results shown in the last column of Table 6.3. Notice the especially high error rate for the Professional forgery category when only the spatial (X and Y) dimensions are used. The verification system usually has the least difficulty in spotting Professional forgeries, yet here they manage to foil the system 89.5% of the time. This demonstrates the high visual quality of these forgeries.

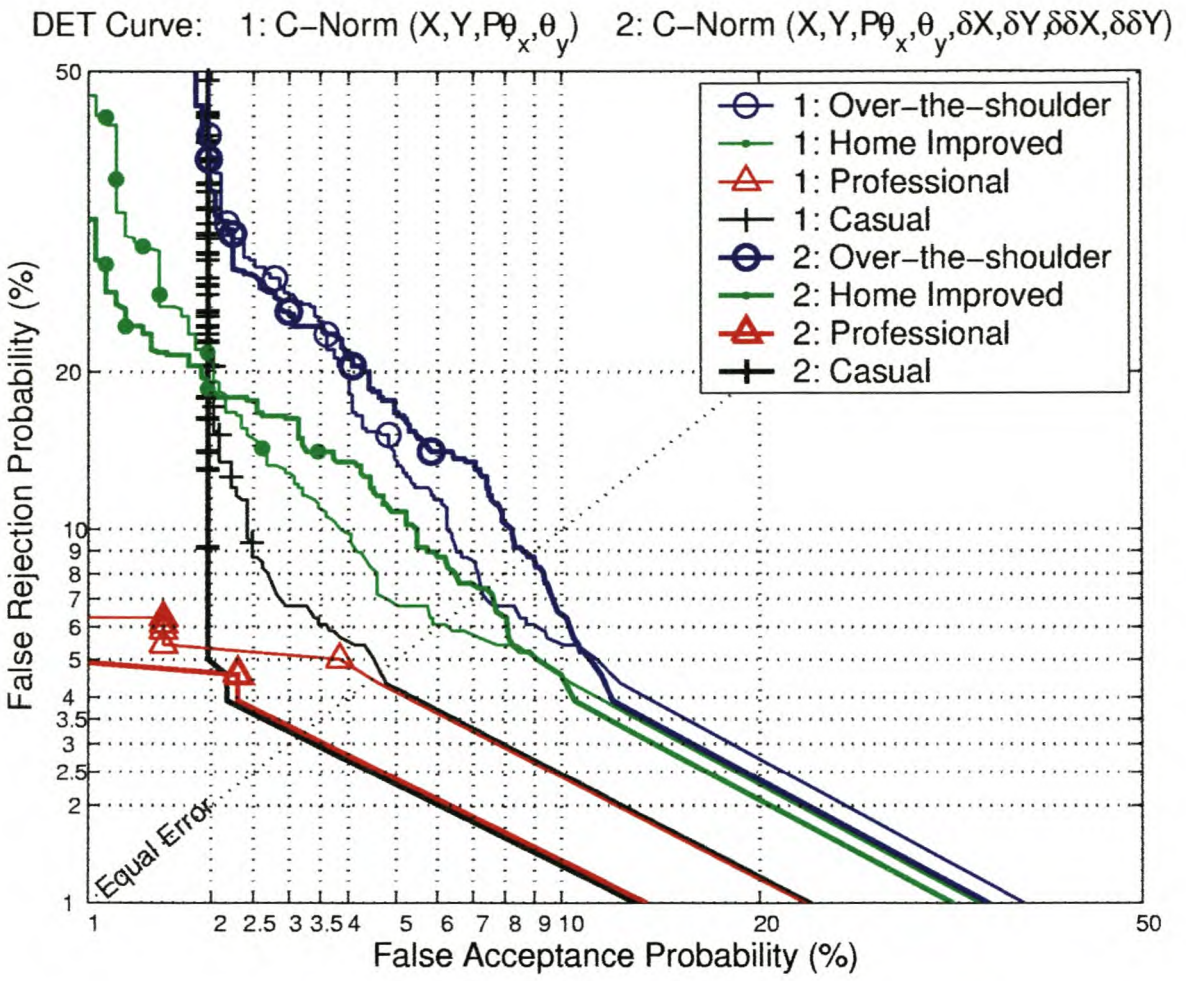


Figure 6.3: *The two best selections of features are those of the Normal 5 dimensions, and the Normal dimensions with the velocity and acceleration of both X and Y appended.*

Forgery Type	Normal	$X, Y, P, \theta_x, \theta_y, \delta X, \delta Y, \delta\delta X, \delta\delta Y$	$X, Y, \delta X, \delta Y$	$\delta X, \delta Y$	$\delta\delta X, \delta\delta Y$	X, Y
Over-the-shoulder	**7.27**	8.76	9.64	15.3	40.0	65.4
Home Improved	**6.10**	7.40	8.14	19.6	37.7	72.4
Professional	4.48	**3.86**	8.20	39.2	66.2	89.5
Casual	4.65	**3.85**	5.25	4.58	19.2	54.5

Table 6.3: *Equal Error percentages from the FAR plots for different preprocessing feature vector transformations.*

The acceleration features ($\delta\delta X$ and $\delta\delta Y$) of users' signatures were expected to be highly distinctive and yield good classification accuracies. The improvements are most evident in the Casual forgery class, though in general these features do not provide the anticipated benefits.

The velocity features ($\delta X \equiv delta2(X)$ and $\delta Y \equiv delta2(Y)$) of the spatial dimensions do provide real gains, though the Professionals are able to beat the system 39.2% of the time. Combining the $X, Y, \delta X, \delta Y$ features provides a very acceptable verification system that could be used with cheaper electronic pads that sample only the X and Y features from a person's signature. This type of digitiser is becoming much more prevalent—laptops now have touch screens and/or touch pads as mouse replacements, and personal digital assistants, palmtops and cell-phones often use styluses and touch screens for their human machine interfaces.

Using all the 'Normal' features of X, Y, P, θ_x, and θ_y along with the acceleration and velocity of the X and Y dimensions decreased the error rates further, though even with the added four dimensions, it does not beat just using the 'Normal' five dimensions (see Figure 6.3). In all further trials, only the original 5 'Normal' features were used.

6.2.3 Framing the Features

The characteristics of the individual data points are not independent of each other, but instead form line segments (the X and Y dimensions), or have variations that are limited by the agility and dexterity of the signers' hand. It therefore makes sense not just to model individual points, but rather model these short line segments that together make up a signature. To achieve this end, the feature vectors are grouped together using the frame-as-vector (FAV) transform of Section 3.5, to form a feature frame of the required number of feature vectors.

Additional experiments were performed using frame lengths of 2, 4, and 8 feature vectors, and the equal error rates are shown in Table 6.4.

Discussion

Framing the data proved to be a successful exercise and the error rates dropped almost 2% each when using a frame length of 2. Using larger frame lengths however, caused the error rates to increase dramatically for all classes. This counter-intuitive effect is explained by recalling from Section 4.3.1, that the number of parameters that define the full covariance Gaussian PDF in each HMM state, equals $\frac{1}{2}(N(N+3)$, where N is the

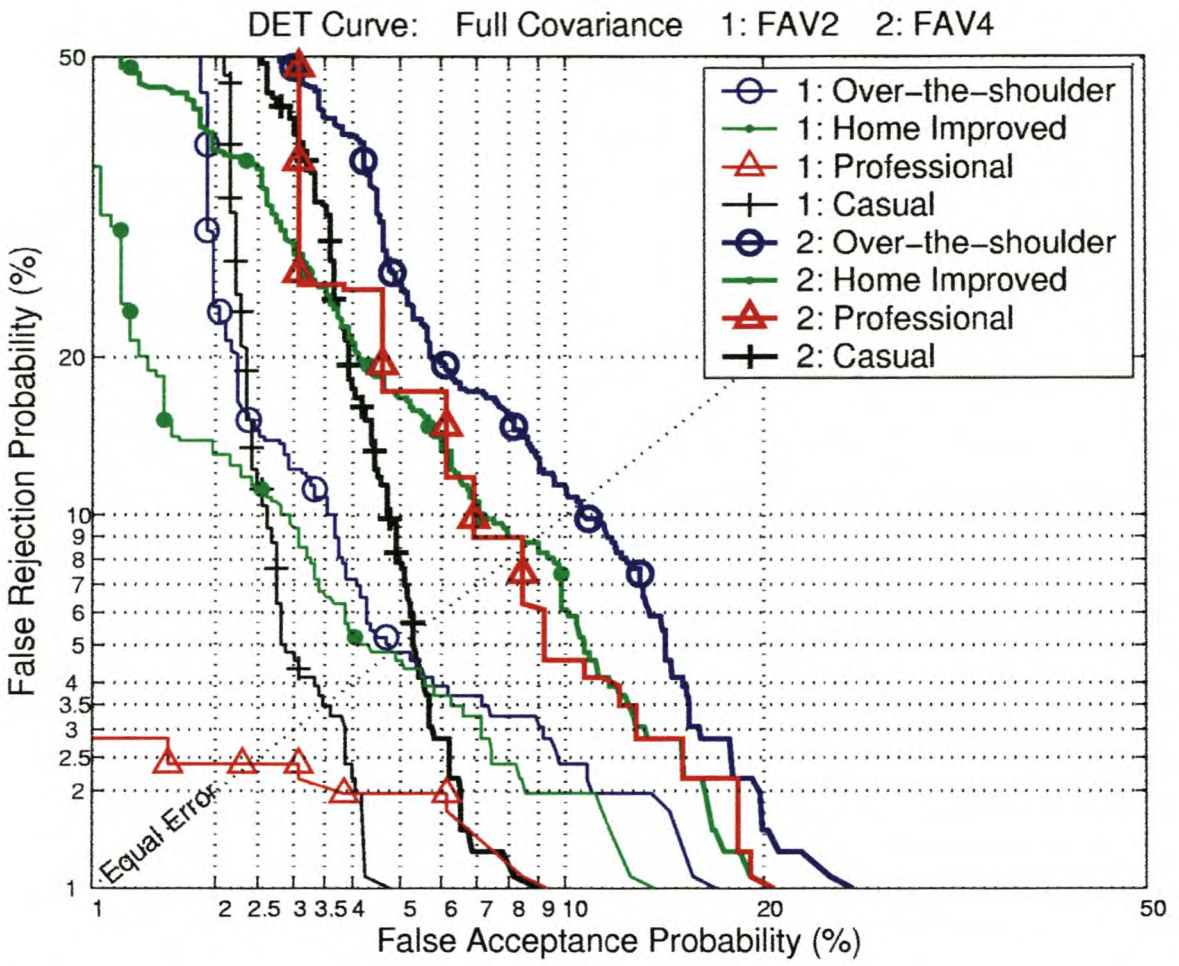


Figure 6.4: *DET: Lower error rates can be achieved by framing the feature vectors. Shown here are frame lengths of 2 and 4.*

	Frame Length			
Forgery Type	Normal: 1	2	4	8
Over-the-shoulder	7.27	**4.79**	10.6	15.0
Home Improved	6.10	**4.79**	8.71	14.2
Professional	4.48	**2.40**	8.46	23.1
Casual	4.65	**3.49**	5.29	8.82

Table 6.4: *Equal Error percentages from the FAR plots for different frame lengths.* ***Full covariance*** *Gaussians are used for the HMM state PDFs.*

PDF/feature vector dimension. That means that the 4 and 8 frame lengths use PDFs of 20 and 40 dimensions each having 230 and 860 determining parameters respectively. With a frame length of 2, there are only 65 parameters to estimate. It is surmised that there is simply not enough training data (nor will there be in real-world situations) to estimate PDFs of this complexity. Nonetheless, the technique of framing has proved valuable and it is investigated further below using less complex Gaussian state PDFs.

6.2.4 Using different Gaussian State PDFs

Firstly, the diagonal Gaussians, from Section 4.3.2, are investigated for the HMM state PDFs, followed by the circular Gaussians from Section 4.3.3. It is hoped that the reduced number of parameters needing to be estimated ($2N$ and $N + 1$ respectively) for these PDFs, will permit the model to be trained more completely for each user.

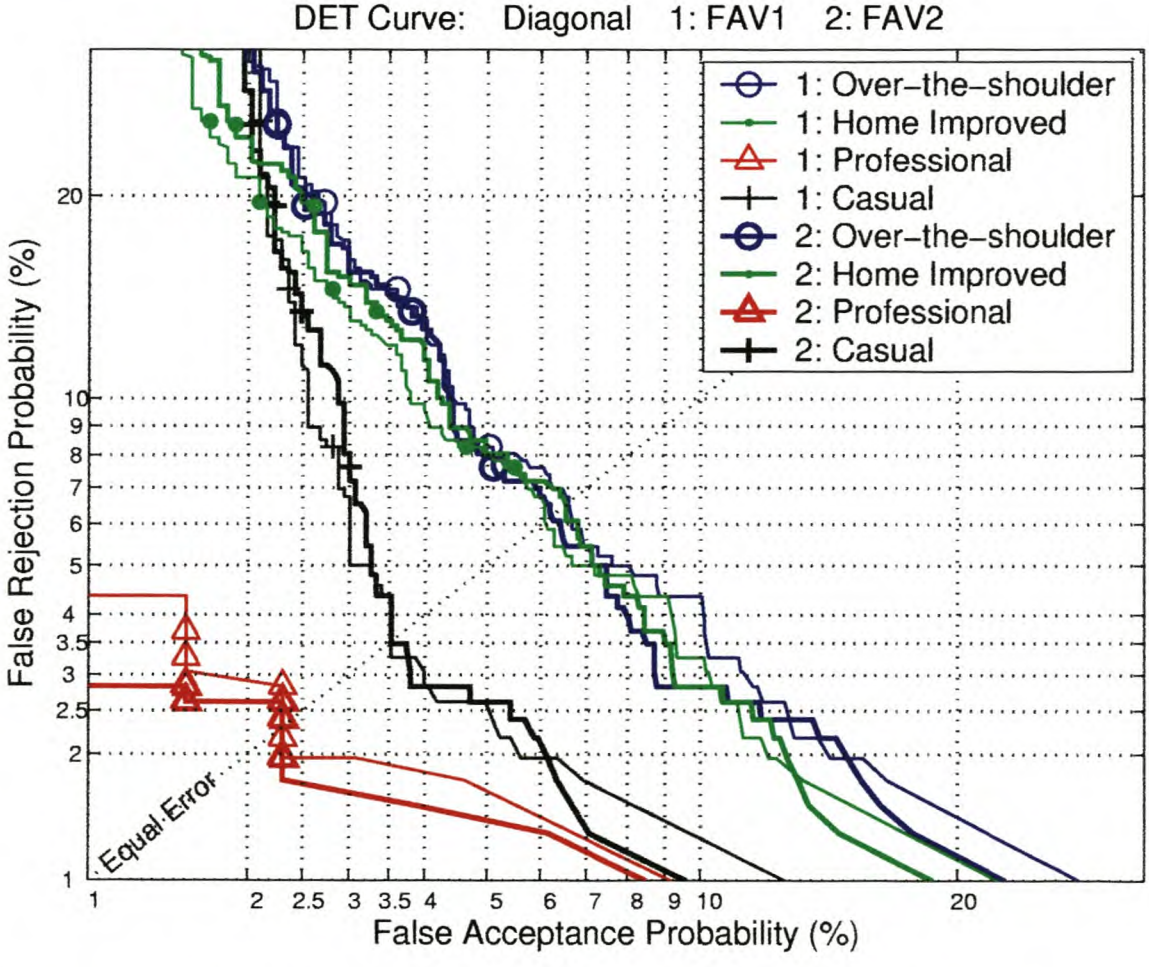


Figure 6.5: *DET: The best two* **diagonal** *Gaussian state PDF trials are shown here—those with frame lengths of 1 and 2*

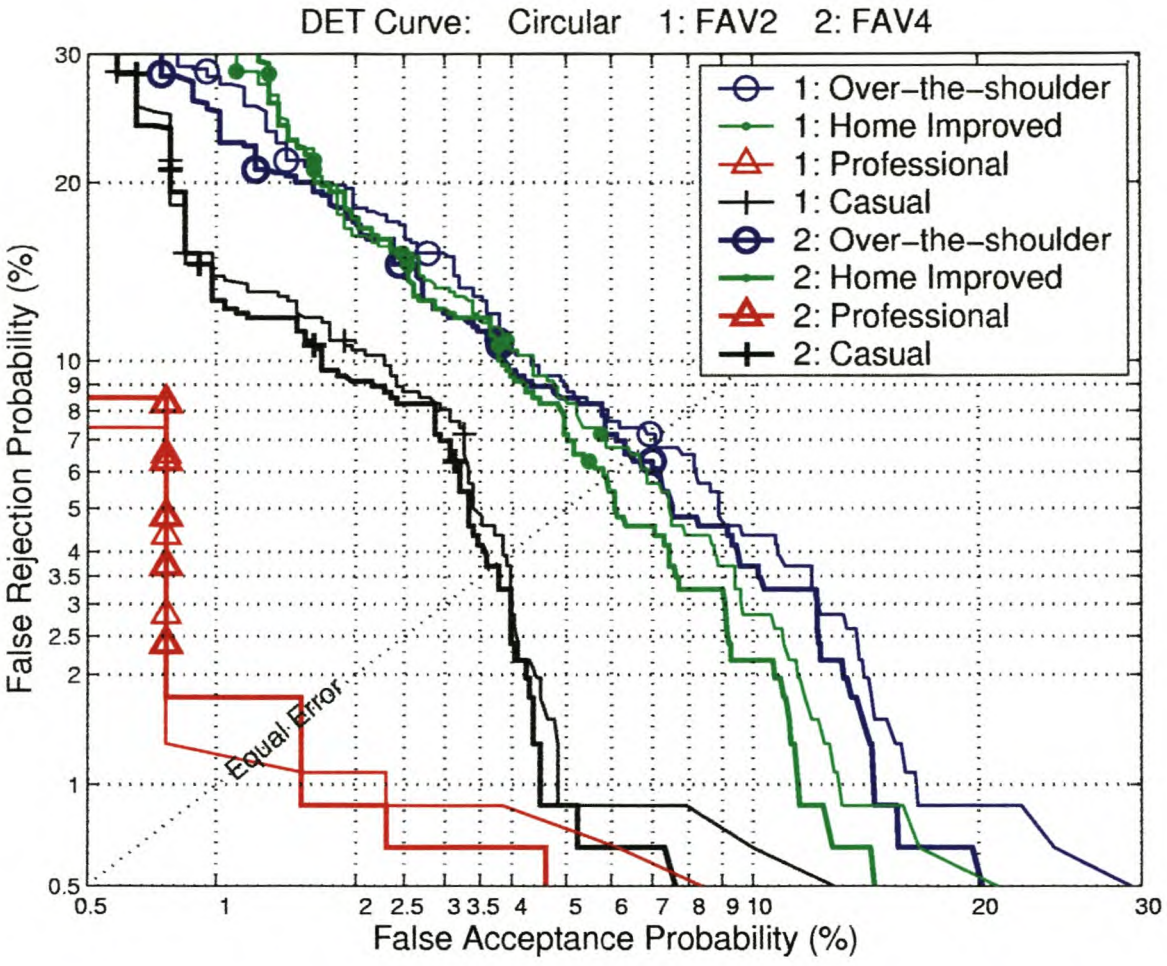


Figure 6.6: *DET: The best two* **circular** *Gaussian state PDF trials are shown here—those with frame lengths of 2 and 4*

	Frame Length			
Forgery Type	1	2	4	8
Over-the-shoulder	**6.07**	6.60	6.22	7.35
Home Improved	6.32	**6.08**	6.54	6.80
Professional	**2.30**	**2.30**	**2.30**	7.69
Casual	**3.32**	3.53	3.53	5.10

Table 6.5: *Equal Error percentages from the FAR plots for different frame lengths.* ***Diagonal*** *Gaussians are used for the HMM state PDFs.*

	Frame Length			
Forgery Type	1	2	4	8
Over-the-shoulder	6.75	7.07	**6.53**	7.41
Home Improved	6.54	6.54	**5.88**	6.67
Professional	1.54	**1.19**	1.54	1.54
Casual	3.86	3.92	**3.70**	4.36

Table 6.6: *Equal Error percentages from the FAR plots for different frame lengths.* ***Circular*** *Gaussians are used for the HMM state PDFs.*

Discussion

The diagonal and circular Gaussian PDFs in general produce more errors with the Over-the-shoulder and Home Improved forgeries, while beating the full covariance PDFs in the Casual (3.32%) and Professional (1.19%) categories.

The increased error rates of these two types over the full covariance Gaussians for the Over-the-shoulder and Home Improved forgeries, means that some useful information was contained in the covariance matrices, and that there were correlations between the different dimensions. It is not unexpected that different dimensions tend to vary together ('co-vary'), since they do come from neighbouring feature vectors and we would anticipate for example, that if X and Y were both increased over the previous two vectors, then they would also do so in the next feature vector. The minimum error rates for the circular Gaussians are found using a frame length of 4, which again shows that some useful information is gleaned by framing the data.

It seems there are conflicting requirements to improving the accuracy: those of needing full covariance Gaussians and feature vector framing, yet not having enough training data to train these PDFs sufficiently. A solution is addressed in the next section.

6.2.5 Class Based KL Transform

Intuitively, framing the feature vectors seems like a good idea, but the shortage of data needed to train high dimensional PDFs means that lower error rates are not realised. Yet the advantages of framing can still be attained without having PDFs of large dimensions, by reducing the number of feature frame dimensions using the Class Based Karhunen-Loève Transform (CBKLT) covered in Section 3.6. Here, eight of the original 5D feature vectors are framed to form a single 40D feature, which a CBKLT then reduces down.

using linear projection, to the required number of dimensions. This is then used to train HMMs with full covariance Gaussian state PDFs.

The classes in the CBKLT correspond to the HMM state PDFs of the users' signature models. If there are 20 original states per HMM, and 51 users in the system, there will be $20 \times 51 = 1020$ classes. To determine the linear projection matrix of the CBKLT, *all* the training data feature vectors are labelled with their users' model name and state PDF number. This is obtained from the Viterbi algorithm operating on previously trained HMMs. This means that a basic trained HMM is needed for each user to determine how the signature should be optimally segmented into the 20 state PDFs (or classes). It must be mentioned that the signature is never hard-segmented when estimating the HMM, rather the HMM is allowed to find the optimal positions for dividing up the signature for each state PDF, so as to maximise the final path probability through the model—so called soft-segmentation.

Table 6.7 shows the equal error rates when retaining a differing number of dimensions, and Table A.1 in Appendix A.2.5 gives the eigenvalues as well as the cumulative totals of all the summed eigenvalues. Approximately 99% of the variance is contained in the first nine dimensions.

		Dimensions Kept From CBKLT							
Forgery Type	Normal	6	7	8	9	10	11	14	16
Over-the-shoulder	7.27	5.44	5.23	**4.90**	5.01	5.30	5.45	5.37	5.86
Home Improved	6.10	4.36	3.70	**3.68**	3.92	4.79	4.58	5.16	5.66
Professional	4.48	2.89	**1.74**	2.30	1.96	2.18	2.31	2.83	2.72
Casual	6.65	3.27	2.7	2.61	2.27	2.16	2.16	**2.09**	2.18

Table 6.7: *Equal Error percentages from the FAR plots for different number of retained dimensions after a CBKLT.*

Discussion

The CBKLT seems not only to find a reduced dimensional hyperplane through the framed feature space, but also to maintain the more important inter-class variances. From Figure 6.7 we can see that keeping the first nine dimensions produces the lowest overall summed error rates, and in fact the best error rates found in this research.

If more dimensions are retained, the Casual forgery EE rates continue dropping to a minimum of 2.09% with 14 dimensions. These added dimensions clearly help refine the

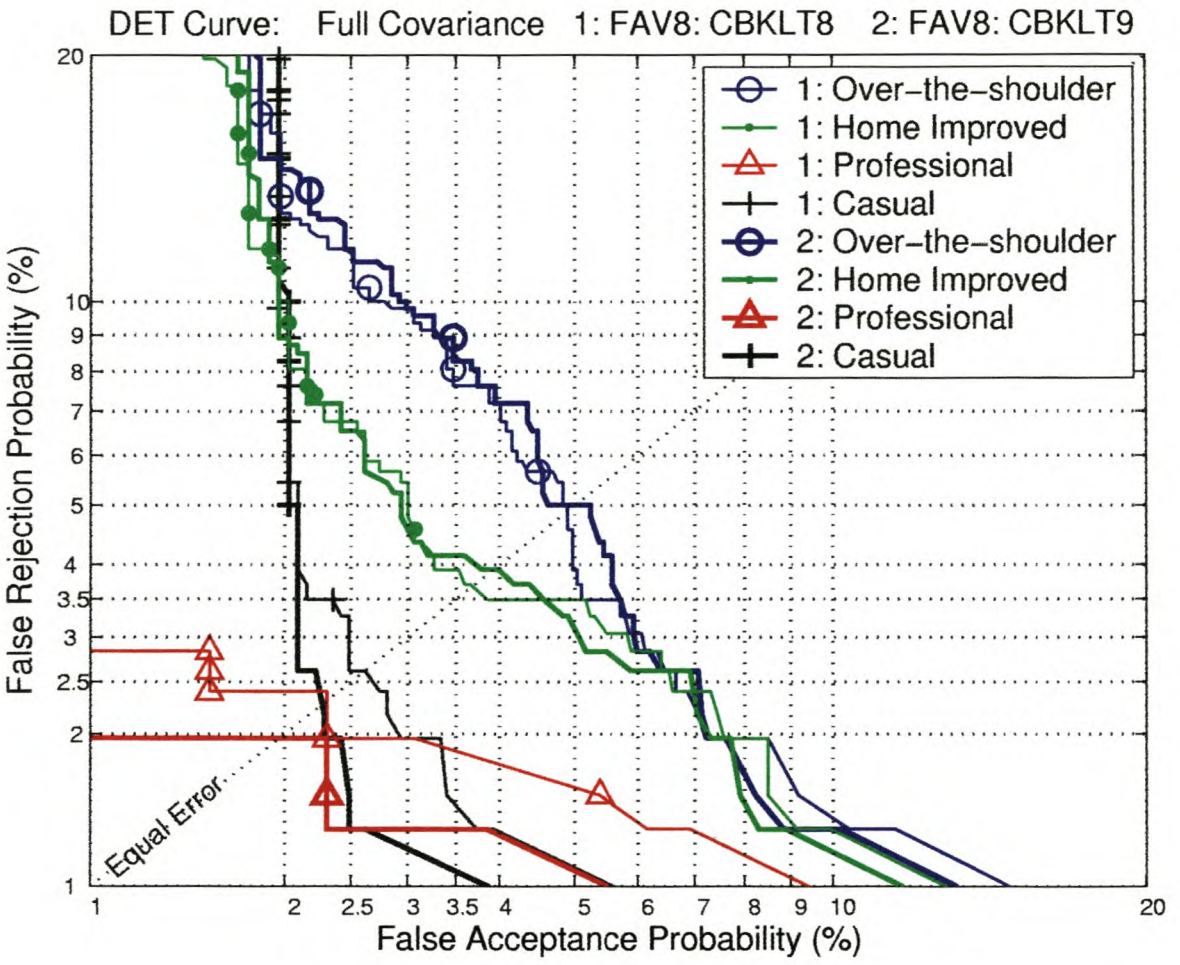


Figure 6.7: *DET: The best two CBKLT based trial results are shown here. The 40D feature frame is reduced to 8 and 9 dimension which produces these plots.*

model with respect to the the dissimilar shaped Casual forgeries, at the expense of the more precise forgeries' EE rates.

The total time taken to verify an average signature, using this optimal system configuration, is 160 milliseconds on a 700 MHz personal computer.

6.2.6 Volterra Series

The final experiments investigate the use of the Volterra series expansions to try and model possible non-linear characteristics in the signature signal. The motivation for this is that short line segments from the signature could be modelled as piecewise polynomials—a kind of spline interpolation with more degrees of freedom. The Volterra expansion would then allow the state PDFs to statistically model the coefficients of these polynomials to provide a more accurate model of the line segments. This concept could also apply to inter-relationships between the other feature dimensions.

Volterra expansions prove difficult to work with due to the very fast increase in dimensions with order and input dimensions. The 5D feature vectors might only be framed three at a time to produce a 15D feature frame, yet the Volterra expansion of order two will increased the number of dimensions to 135 (refer back to Table 3.3). This was the highest number of dimensions that could be handled by the CBKLT before running out of memory. The huge rise in memory requirements for further increases in input dimensions make it impractical to use frame lengths like those of the previous sections. A CBKLT is then used to reduce the number of dimensions for the HMM.

	Dimensions Kept From CBKLT	
Forgery Type	9	17
Over-the-shoulder	**9.27**	10.1
Home Improved	9.80	**9.03**
Professional	7.69	**7.63**
Casual	5.04	**4.71**

Table 6.8: *EE percentages for a 2^{nd} order Volterra expansion. A frame length of 3, full covariance Gaussian state PDFs, and CBKLTs to reduce the dimensionality to 9 and 17, were used.*

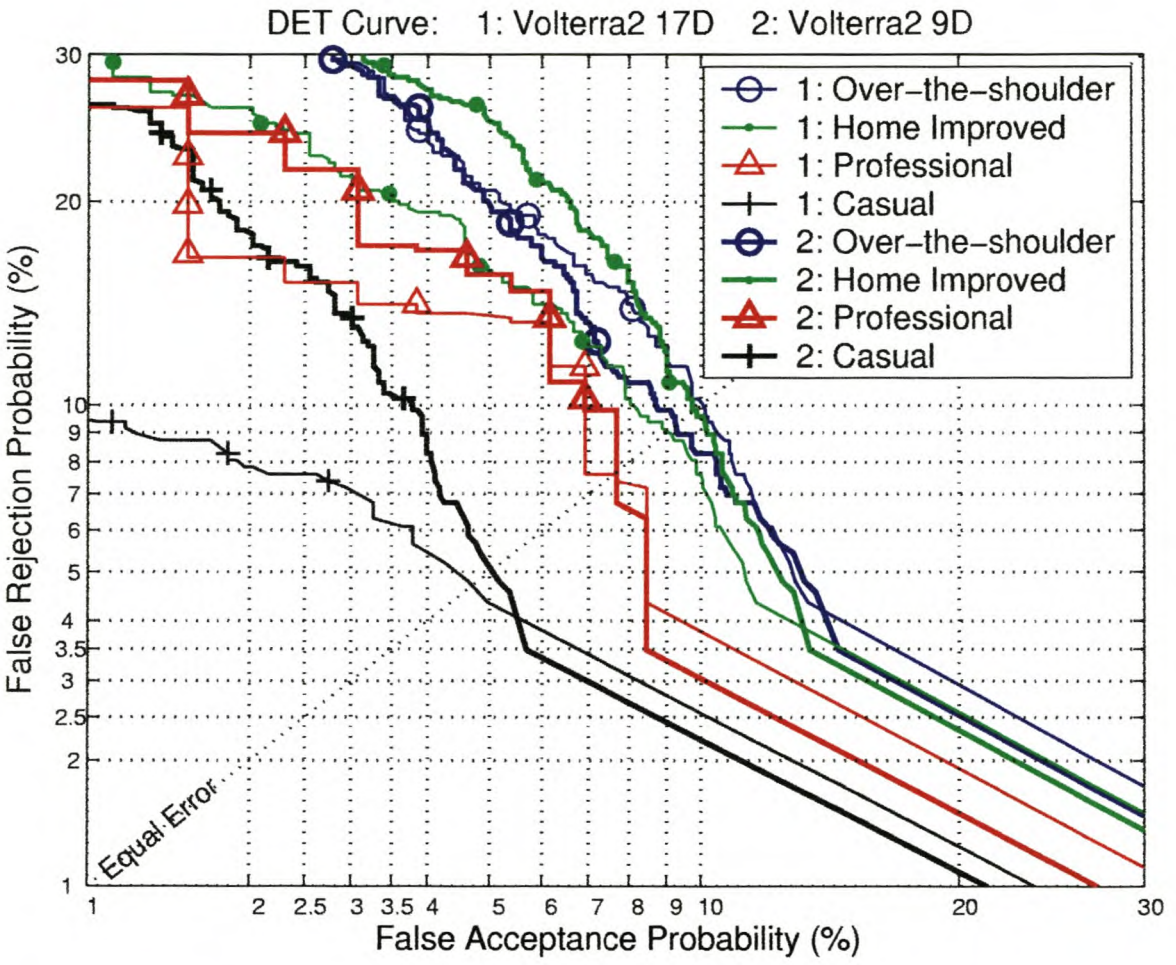


Figure 6.8: *DET: Second order Volterra expansion then using a CBKLT to reduce the number of dimensions from 135D to 9D and 17D.*

Discussion

Table 6.8 shows that the EE rates from the Volterra trials are higher than those of even the 'Normal' models with no framing and dimension reduction. Approximately 99% of the variance is contained in the first 17 dimensions from the CBKLT—the eigenvalues also decay much more slowly than in the simple framing situations discussed in the previous section. The mixing products from a Volterra expansion are clearly not the non-linear transformations needed to boost the accuracy.

The best results obtained are therefore those of the framed data with class based KL transform dimension reductions using full covariance Gaussian state PDFs.

6.3 Summary

Many design iterations were completed before arriving at our optimal modelling solution, which is a Ferguson duration emphasised HMM with 20 original states, 20^{th} order duration modelling, GMM smoothed duration fanout transition probabilities with 3 underlying 1D Gaussians, and 9D full covariance Gaussian state PDFs.

The raw data from the tablet is preprocessed first by the spatial normaliser, then a frame-as-vector transform is used with a frame length of 8 and a frame shift of 1. Next, each dimension is scaled to a standard deviation of one using the scaler normaliser. Then it is projected down from 40D space to 9D space using a class based Karhunen-Loève transform.

This results in the following equal error rates for the respective forgery categories:

Over-the-shoulder	—	**5.01%**
Home Improved	—	**3.92%**
Professional	—	**1.96%**
Casual	—	**2.27%**

Note that better results were achieved for each of the forgery categories separately, but these results are the best overall for the four forgery categories together.

Chapter 7

Conclusion

A handwritten signature verification system based on Hidden Markov Models was designed and implemented.

We achieve lower EE rates than Dolfing for comparable systems based only on local features. Our EE rates for Over-the-shoulder and Home Improved forgeries are 5.0% and 3.9% respectively. Dolfing system attains corresponding EEs of 7.9% and 3.9%. Banks do not divulge the percentages of forgeries that are falsely accepted by their systems, but it is likely to be many times these rates.

This thesis achieved all the objectives set out in Section 1.2 and the accomplishments are as follows:

- A modified Ferguson duration emphasised Hidden Markov Model was designed that can be initialised to behave identically to the trained model it is expanded from. It also permits elegant statistical methods of transition probability smoothing using various one-dimensional probability density functions.

- Various methods were used to preprocess the signature. The most successful of these involved framing the data, then using a class based Karhunen-Loève transform to select the dimensions of greatest inter-class variance.

- Four statistically tractable verifiers were investigated. The most suitable one for our purposes was the novel C-Norm verifier. This verifier can reach an accept/reject decision for an average signature in 160 ms.

- The size of the complex model representing a person's signature was brought down from a text size of 50kB, to a more convenient size of under 4kB that can be transferred from a cheap smart card in less than half a second.

- All the statistical modelling components necessary for a high quality signature verification system have been incorporated into PatRecII, a fully portable, flexible library, from which a final application can be built. These experiments were also a good test of PatRecII's pattern recognition robustness since many of the existing functions have only ever been used before in speech trials.

7.1 Future Work

The following areas could provide some useful accuracy gains and should be investigated before a commercial product is deployed:

1. The target-centric C-Norm verifier still has a higher error rate (1.98%) for Casual forgeries than that of the impostor-centric T-Norm (0.7%) verifier. We speculate that a verifier that models both the target scores (from a validation/development signature set) and the impostor scores (by applying the presented signature to all models instead of just the claimed model) will produce more accurate results.

2. A related issue is that of user specific thresholds. The C-Norm verifier does perform a similar function to an adaptive threshold in that the claimed data's score is normalised with respect to the genuine signatures distribution, then a global threshold is used. In essence the score is adapted, rather than the threshold. Dolfing improves the accuracy of his trials from his best EE rates of 5.4%, to 1.36% by moving from an adaptive threshold to a personal adaptive threshold. Le Riche also achieves a lowest error rate (not EE rate) of 0.5% by "manually choosing a decision boundary for each signer, in order to minimise the number of incorrect classifications." This would be difficult to achieve in practise where forgeries are not readily available. Nevertheless, an enquiry into this subject should prove beneficial.

3. This system achieves these error rates by modelling only the dynamic raw data (after suitable linear transformations) from the digitiser pad. There are hundreds of other static features that can help make a correct accept/reject decision. Some of these are: measuring the area enclosed by a signature, determining its compactness, jitter, moments of inertia, and obtaining maximum, minimum, average speeds and velocities (see [12]). A fusion of this dynamic system and a static verification system should produce even greater accuracies.

4. Since the amount of time taken to sign, and hence number of feature vectors produced (see Table 2.5), varies between individuals, it makes sense for the HMM to have only as many original and duration states as necessary. Shorter (in time) signatures would then be modelled with less states and a lower duration order than longer signatures. Conversely, it may be found that shorter signatures contain more information per unit time and may therefore need *more* original states, though obviously less duration states.

   The differing storage requirements of varying sized models on limited capacity media could present a problem if not bounded by a maximum allowed size.

5. The challenges of reducing the footprints of the models, and providing enough training data to train them sufficiently, may be overcome by using pre-trained codebooks from which the duration fanout weights and state densities can be assembled. Then the only parameters that would be needed to define a users' model are the indices of the required densities, and their weightings where mixture models are used. The footprint would then be in the order of 500 bytes.

6. Le Riche [16] has demonstrated that signatures do change over time, which means that the model that represents them should also be able to adjust. There are a variety of HMM adaption techniques [4] that can be used to modify a user's model once more training data becomes available. Possibly the model could be adapted every time a presented signature is accepted. There would certainly be issues that need to be addressed concerning the security of such methods, but it would provide a more user-friendly verification system.

Bibliography

[1] AUCKENTHALER, R., CAREY, M., and LLOYD-THOMAS, H., "Score Normalization for Text-Independent Speaker Verification Systems." *Digital Signal Processing*, January/April/July 2000, Vol. 10, No. 1–3, pp. 42–54.

[2] DEVIJVER, P. A. and KITTLER, J., *Pattern Recognition: A Statistical Approach*. First edition. Prentice-Hall, 1982.

[3] DOLFING, J. G. A., *Handwriting Recognition and Verification. A Hidden Markov Approach*. PhD thesis, Eindhoven, Netherlands, 1995.

[4] DU PREEZ, J. A., *Efficient High-order Hidden Markov Modelling*. PhD thesis, University of Stellenbosch, March 1998.

[5] FANNER, R., "Analysis and Implementation of Speaker Adaption Techniques: MAP, MLLR and MLED." Master's thesis, Stellenbosch University, December 2002.

[6] FERGUSON, J. D., "Variable Duration Models For Speech." *Proceedings for the Symposium on the Application of Hidden Markov Models to Text and Speech*. October 1980, pp. 143–179.

[7] GILLICK, L. and COX, S. J., "Some Statistical Issues in the Comparison of Speech Recognition Algorithms." *IEEE*, 1989.

[8] GRIESS, F. D., "On-line signature verification.." Project Report, Department of Computer Science and Engineering, Michigan State University, May 2000.

[9] HERBST, B. and COETZER, H., "On An Offline Signature Verification System." *PRASA Proceedings*, 1998. Department of Applied Mathematics, University of Stellenbosch.

[10] HTTP://WWW.BIOMETRICS.ORG/, "Biometric consortium." Internet.

[11] HTTP://WWW.CYBERSIGN.COM/, "Cyber-sign biometric signature verification." Internet.

[12] HTTP://WWW.MOTIONTOUCH.COM, "Motiontouch electronic handwritten signature capture." Internet. Weybridge, Surrey, United Kingdom.

[13] HTTP://WWW.SIGNPLUS.COM/, "Softpro visual signature verification system." Internet.

[14] HTTP://WWW.VALYD.COM/, "Valyd, inc. authentication and security technologies." Internet.

[15] HTTP://WWW.WACOM.COM/, "Wacom graphic tools homepage." Internet.

[16] KASHI, R. S., HU, J., NELSON, W. L., and TURIN, W., "On-line Handwritten Signature Verifications Using Hidden Markov Models Features." tech. rep., Bell Labs, Lucent Technologies, 600 Mountain Drive Ave, Murray Hill, NJ 07974.

[17] LE RICHE, P., "Handwritten Signature Verification: A Hidden Markov Model Approach." Master's thesis, Stellenbosch University, December 2000.

[18] LEMPEL, A. and ZIV, J., "A Universal Algorithm for Sequential Data Compression." *IEEE Transactions on Information Theory*, 1977, Vol. 23, pp. 337–343. No. 3.

[19] LEVINSON, S. E., "Continuously Variable Duration Hidden Markov Models for Automatic Speech Recognition." *Computer Speech and Language*, March 1986. Vol. 1, No. 1, pp. 29–45.

[20] MARTIN, A., DODDINGTON, G., KAMM, T., ORDOWSKI, M., and PRZYBOCKI, M., "The DET Curve In Assessment Of Detection Task Performance." National Institute of Standards and Technology, and Department of Defense, USA.

[21] PAULIK, M. and MOHANKRISHNAN, N., "A 1D, Sequence Decomposition Based, Autoregressive Hidden Markov Model for Dynamic Signature Identifiaction and Verification." *Proceedings of the Midwest Symposium on Circuits and Machines*, 1993, Vol. 1, pp. 138–141.

[22] PEEBLES, P. Z., *Probability, Random Variables, and Random Signal Principles*. Third edition. McGraw-Hill, 1993.

[23] RABINER, L. R. and JUANG, B. H., "An Introduction to Hidden Markov Models." *IEEE ASSP Magazine*, January 1986, pp. 4–16.

[24] UNKNOWN, "E-Signatures Win Over Iris Scans." *IEE Review*, January 2003, p. 14.

[25] VAN DER MERWE, R., "Variations on Statistical Phoneme Recognition - A Hybrid Approach." Master's thesis, Stellenbosch University, December 1997.

[26] VAN GELDEREN, T., JAMESON, A., and DUWAER, A., "Text Correction in Pen-based Computers: An Empirical Comparison of Methods." tech. rep., Philips Research Laboratories/Institute for Perception Research/Nijmegen Institute for Cognition and Information (NICI).

[27] WITTEN, I. H. and BELL, T. C., "The Zero Frequence Problem: Estimating the Probabilities of Novel Events in Adaptive Text Compression." *IEEE Trans. on Information Theory*, 1991, Vol. 37, No. 4, No. 4, pp. 1085–1093.

[28] YANG, L., *Processing and Recognition of Handwriting in Multimedia Environments.*. PhD thesis, Technische Universiteit Delft, 1995.

[29] YANG, L., WIDJAJA, B., and PRASAD, R., "Application of Hidden Markov Models for Signature Verification.." *Pattern Recognition*, 1995, Vol. 28, pp. 161–107.

Appendix A

Results

A.2 Extra Verification Results

A.2.1 Impostor-centric Verifiers

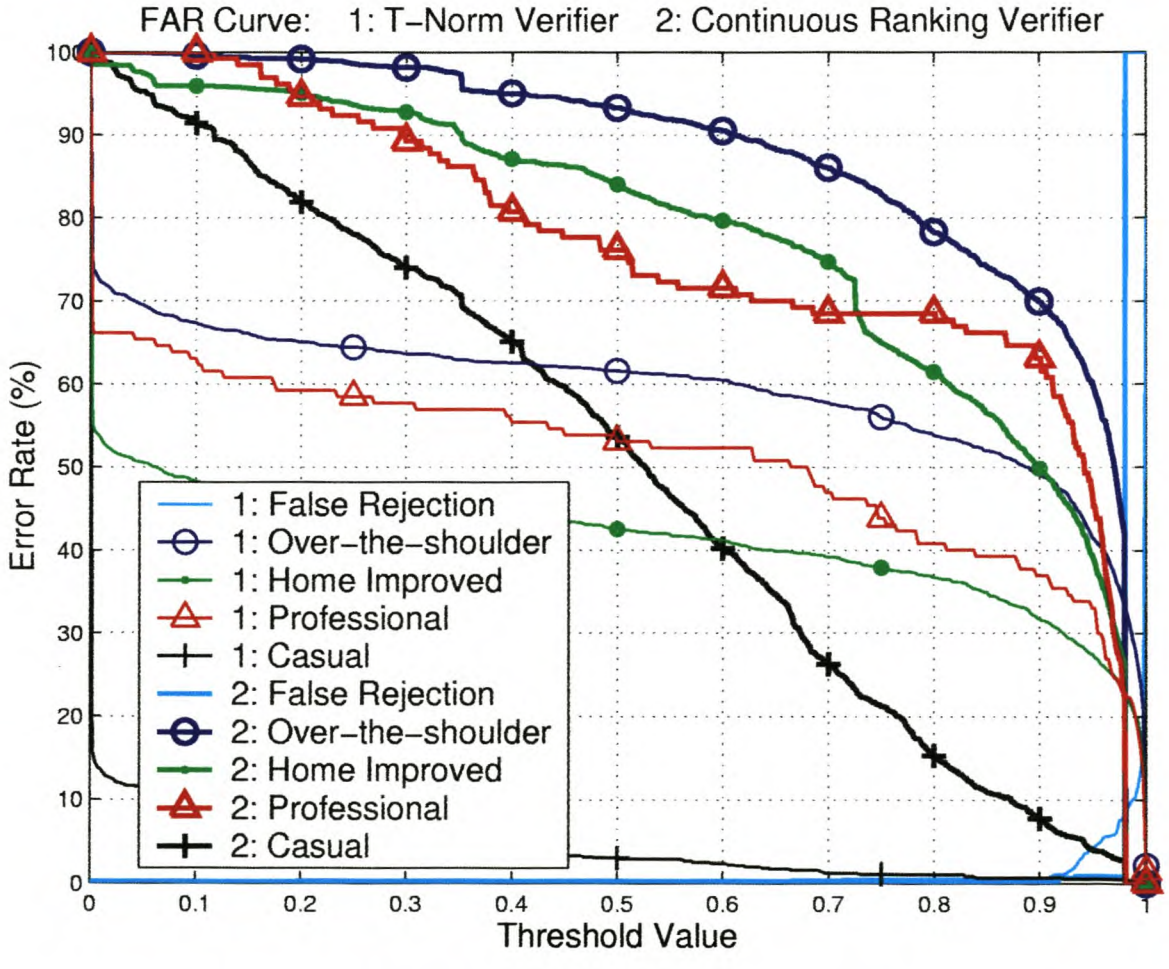


Figure A.1: *FAR curves of the T-Norm and Continuous Ranking verifiers (corresponds with Figure 6.1).*

A.2.2 Using Different Feature Dimensions

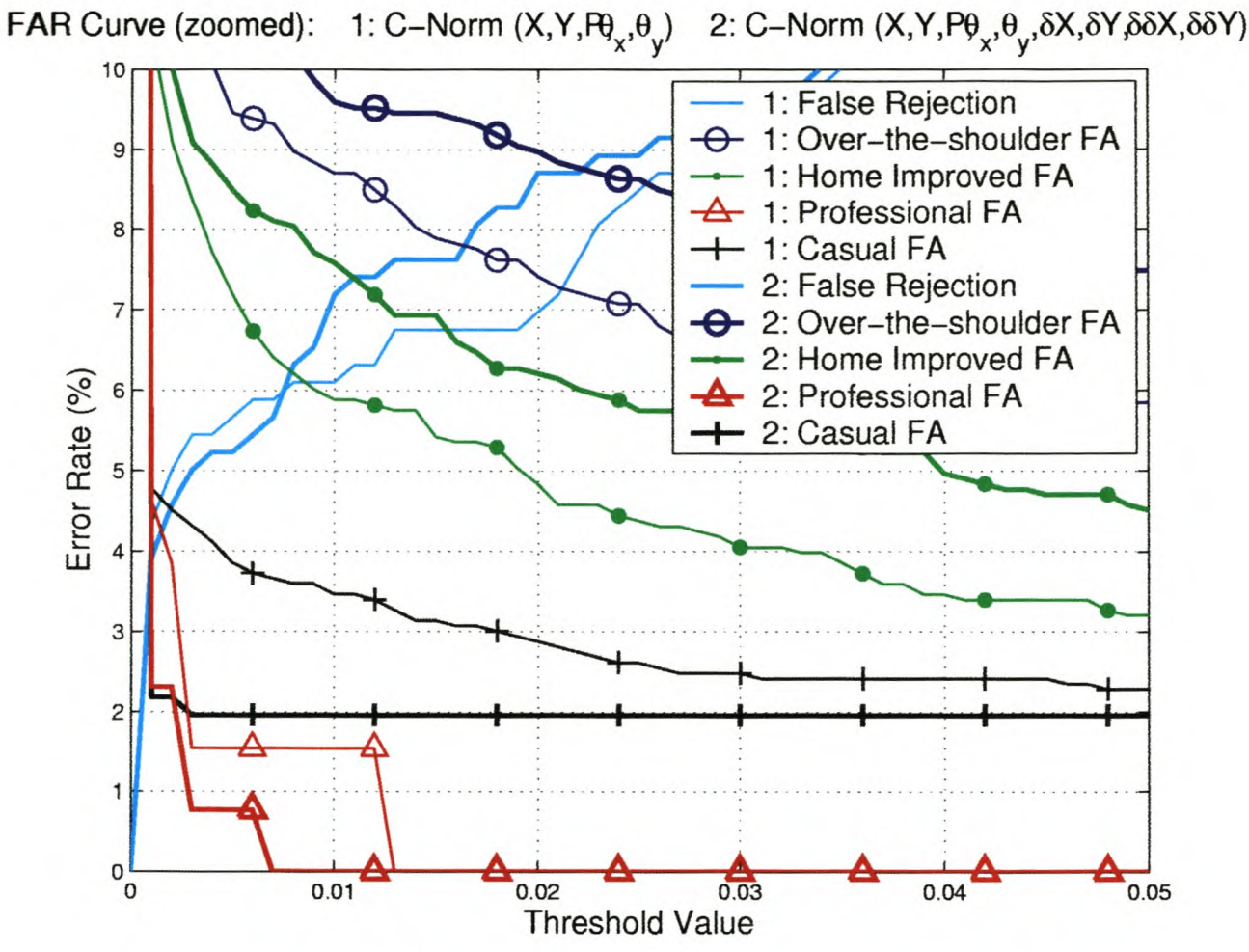


Figure A.2: *FAR zoomed: The effect of different preprocessing feature transformations (corresponds with Figure 6.3).*

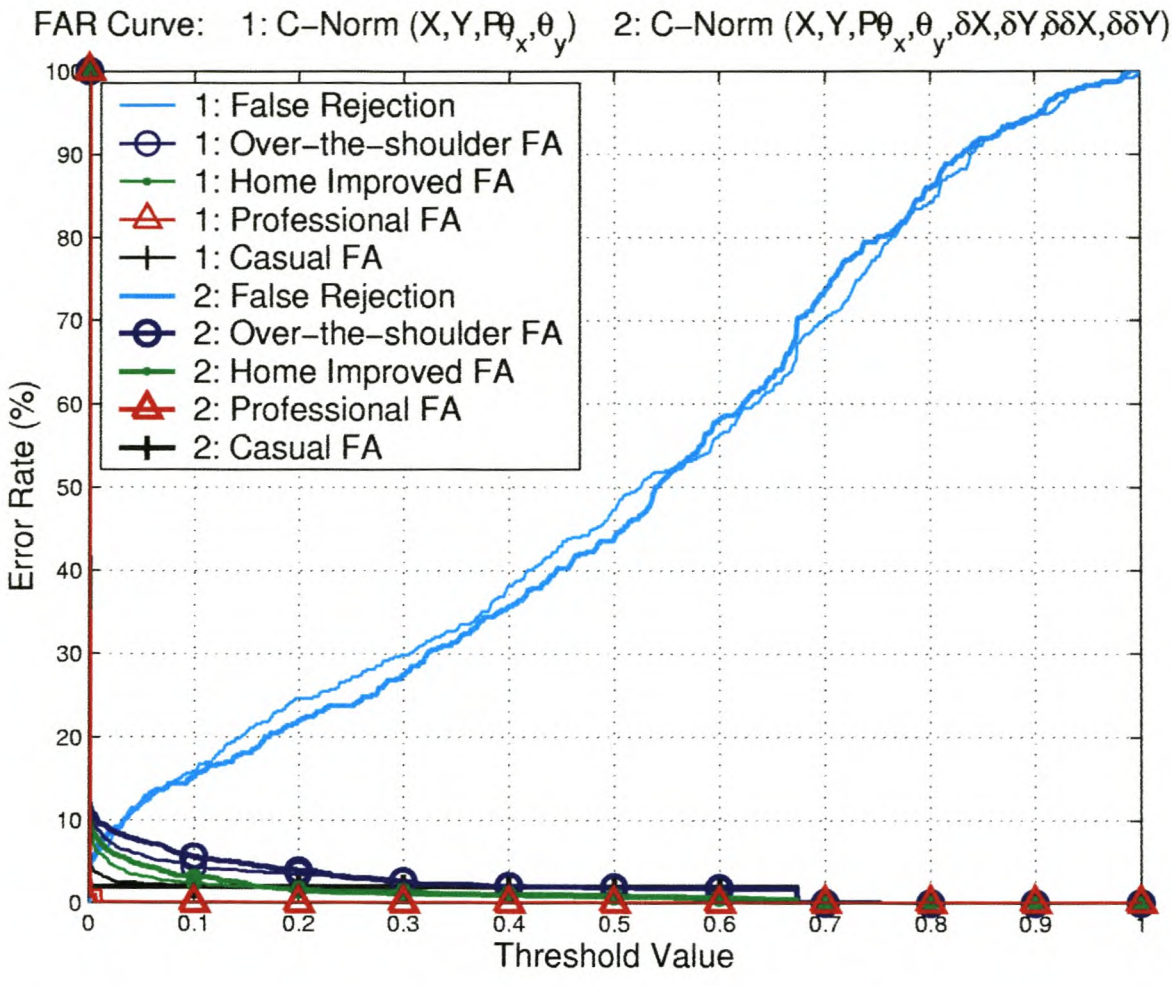


Figure A.3: *FAR: The effect of different preprocessing feature transformations. This plot is included to show how significantly the FAR curves differ from the impostor-centric verifiers' (corresponds with Figure 6.3).*

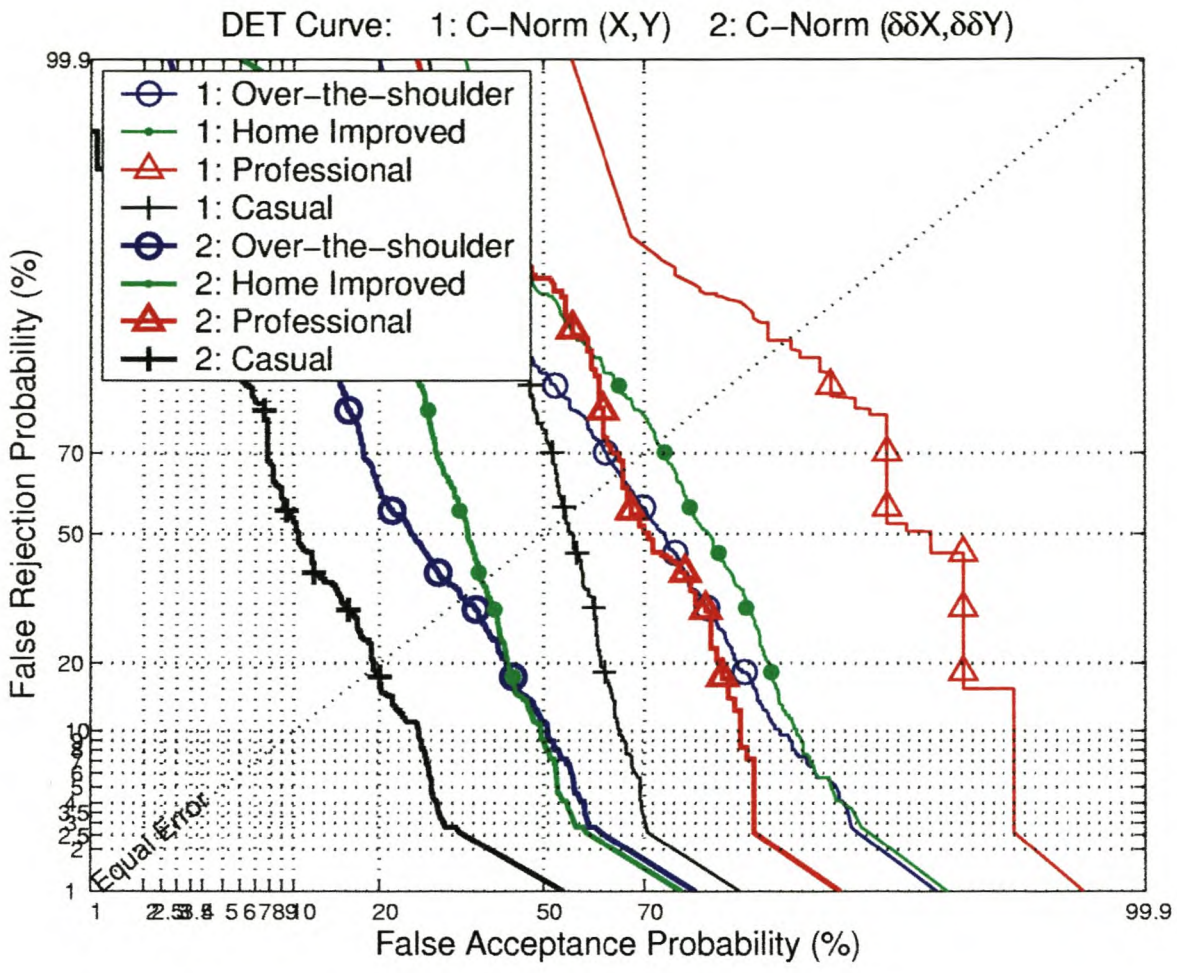


Figure A.4: *DET: The effect of different preprocessing feature transformations (corresponds with Table 6.3).*

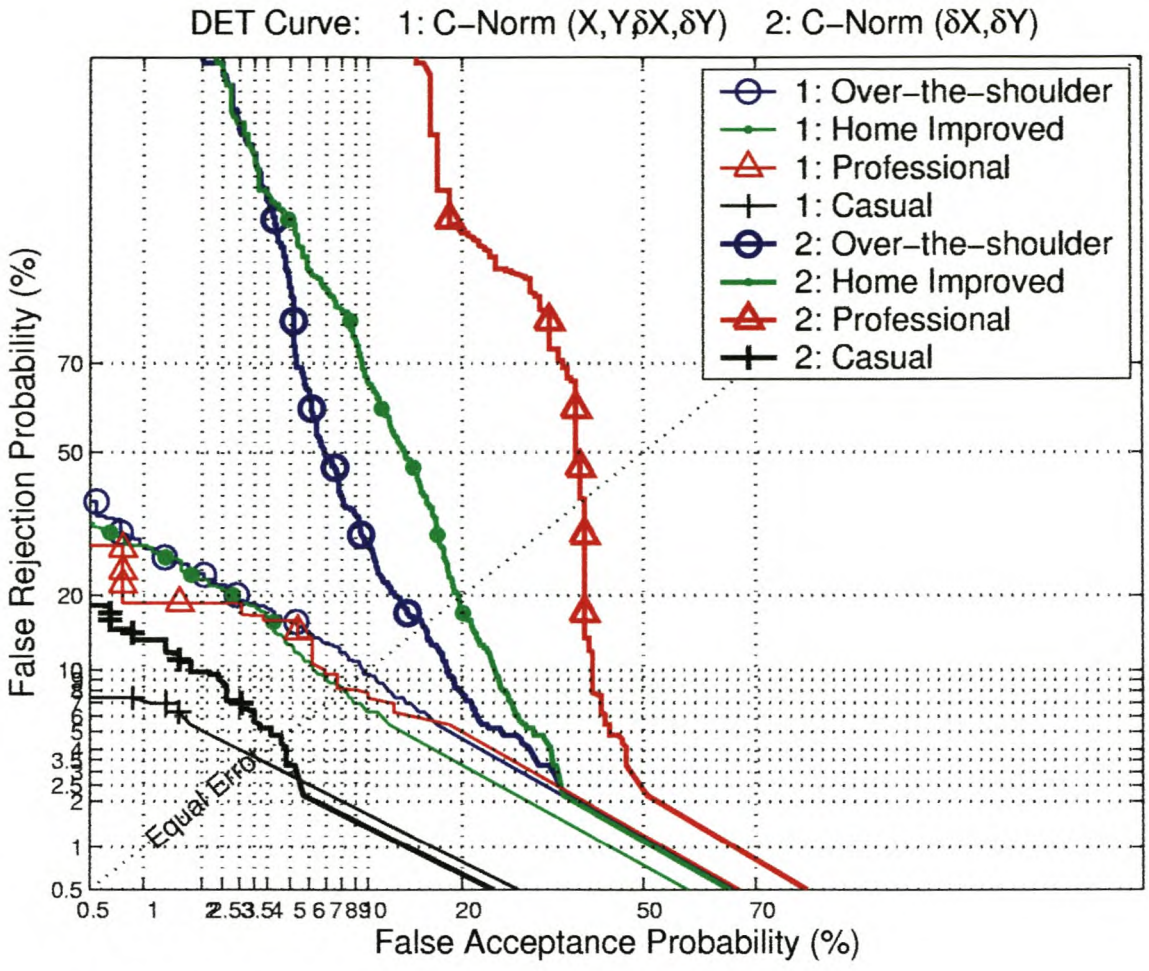


Figure A.5: *DET: The effect of different preprocessing feature transformations (corresponds with Table 6.3).*

A.2.3 Framing the Features

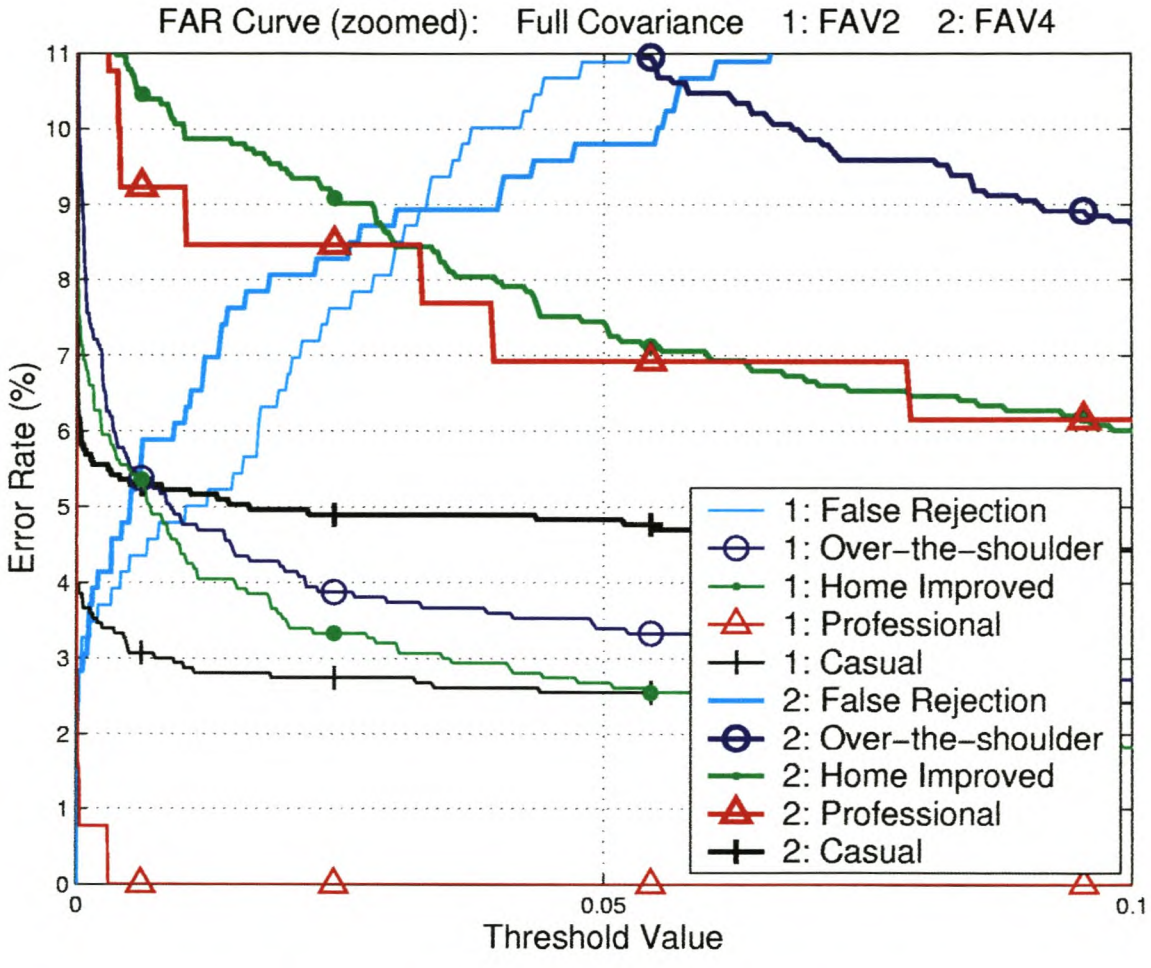


Figure A.6: *FAR zoomed: Lower error rates can be achieved by framing the feature vectors. Shown here are frame lengths of 2 and 4 (corresponds with Figure 6.4).*

A.2.4 Using Different Gaussian State PDFs

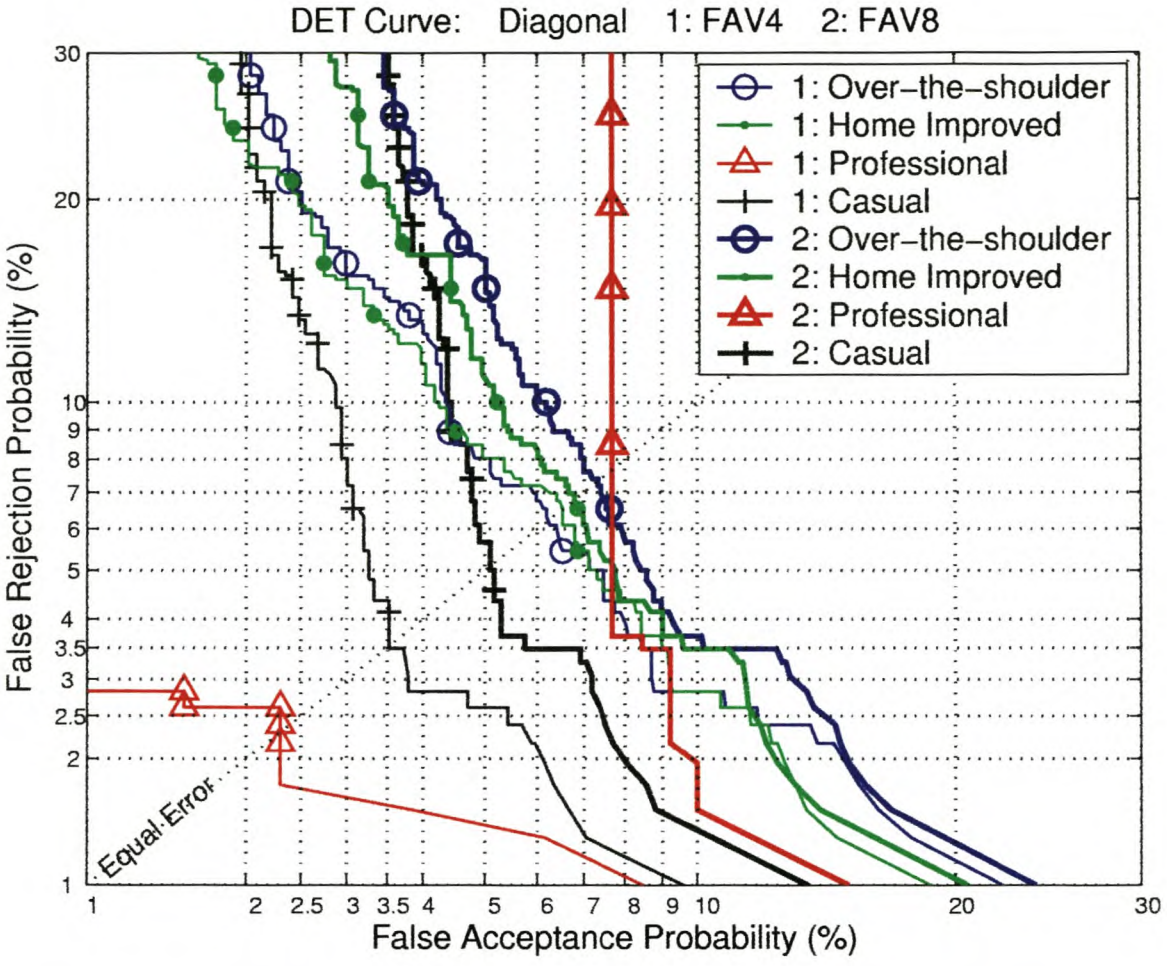


Figure A.7: *DET: The rest of the trial results:* **Diagonal** *Gaussians with frame lengths of 2 and 4 (corresponds with Table 6.5).*

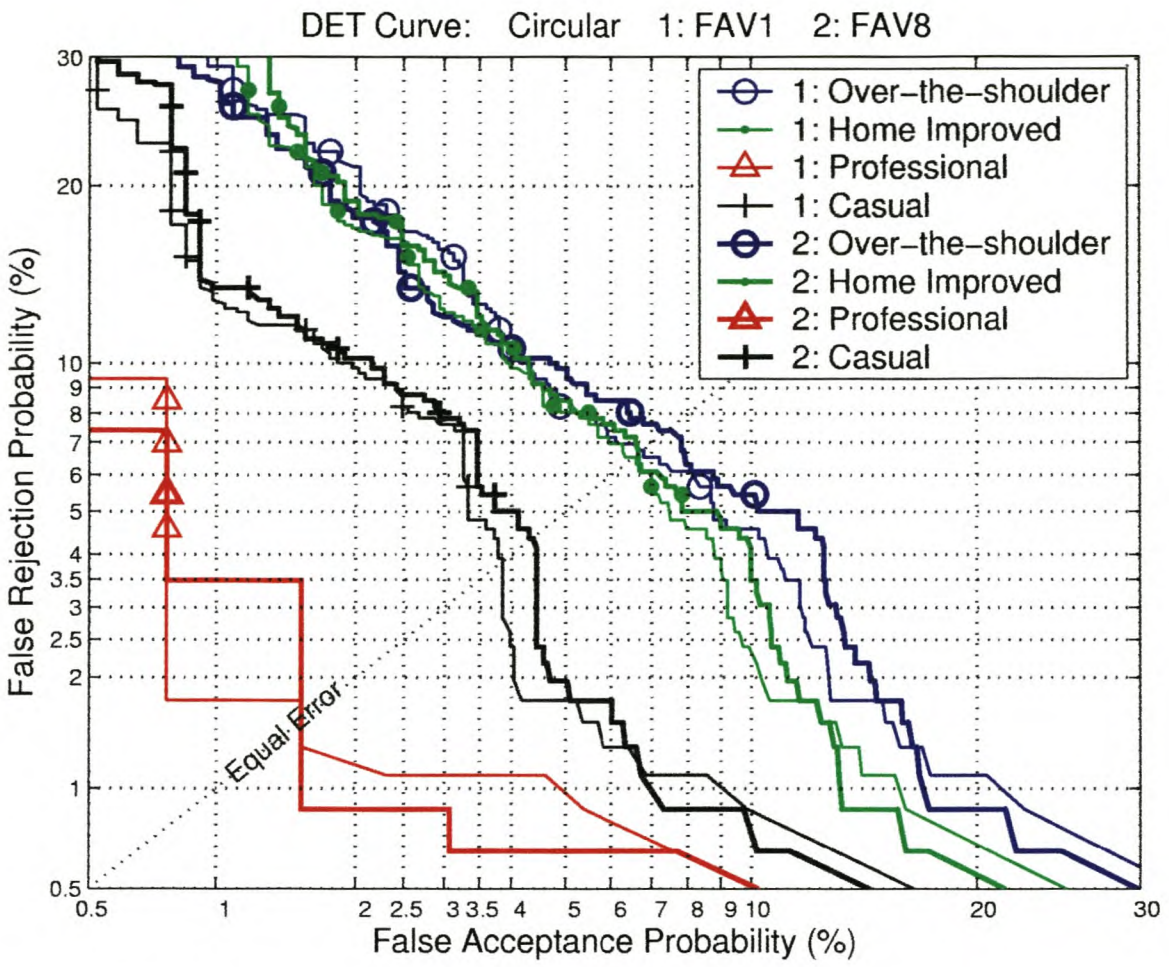


Figure A.8: *DET: The rest of the trial results:* **Circular** *Gaussians with frame lengths of 1 and 8 (corresponds with Table 6.6).*

A.2.5 Class Based KL Transform

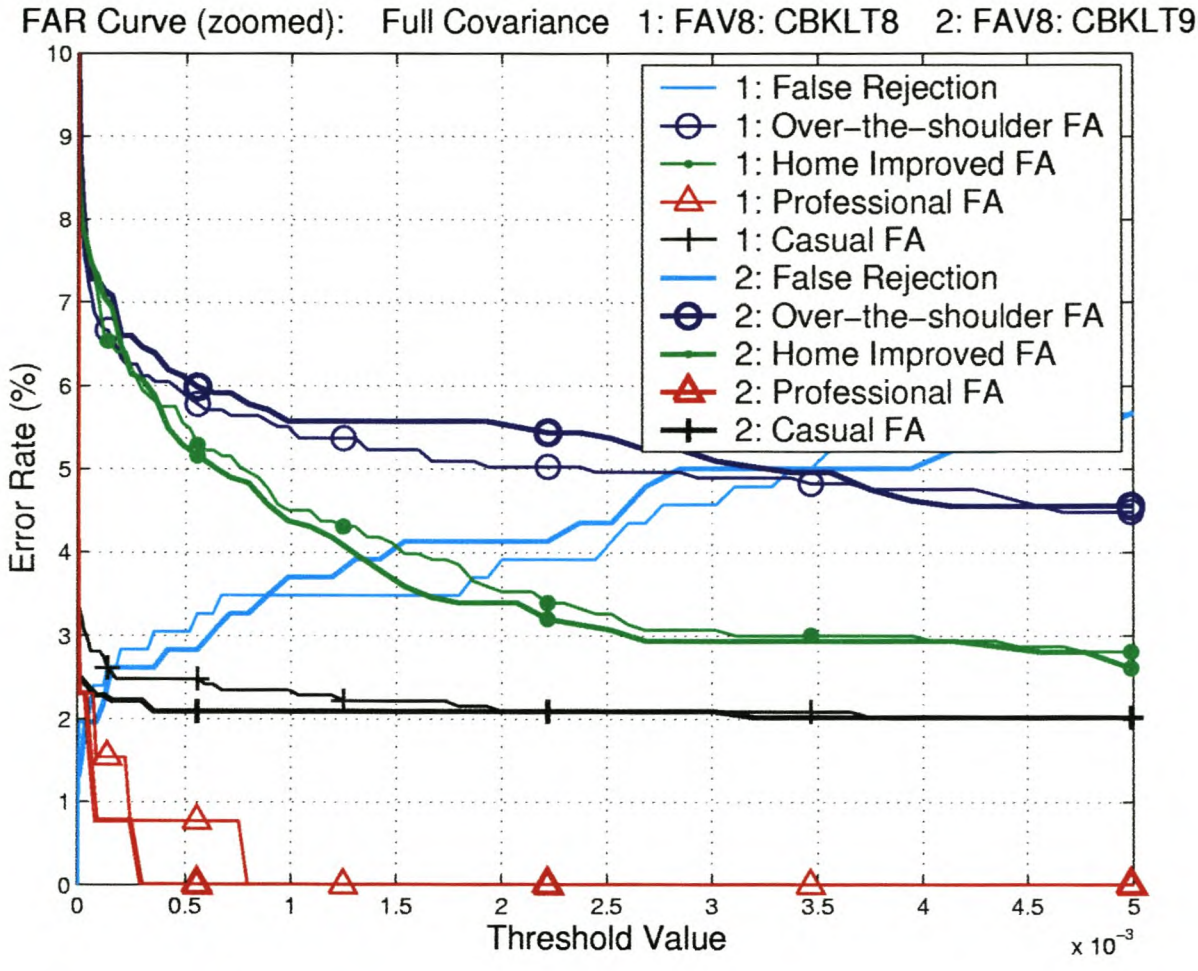


Figure A.9: *FAR: The best two CBKLT based trial results: projecting down to 8 and 9 dimensions (corresponds with Figure 6.7).*

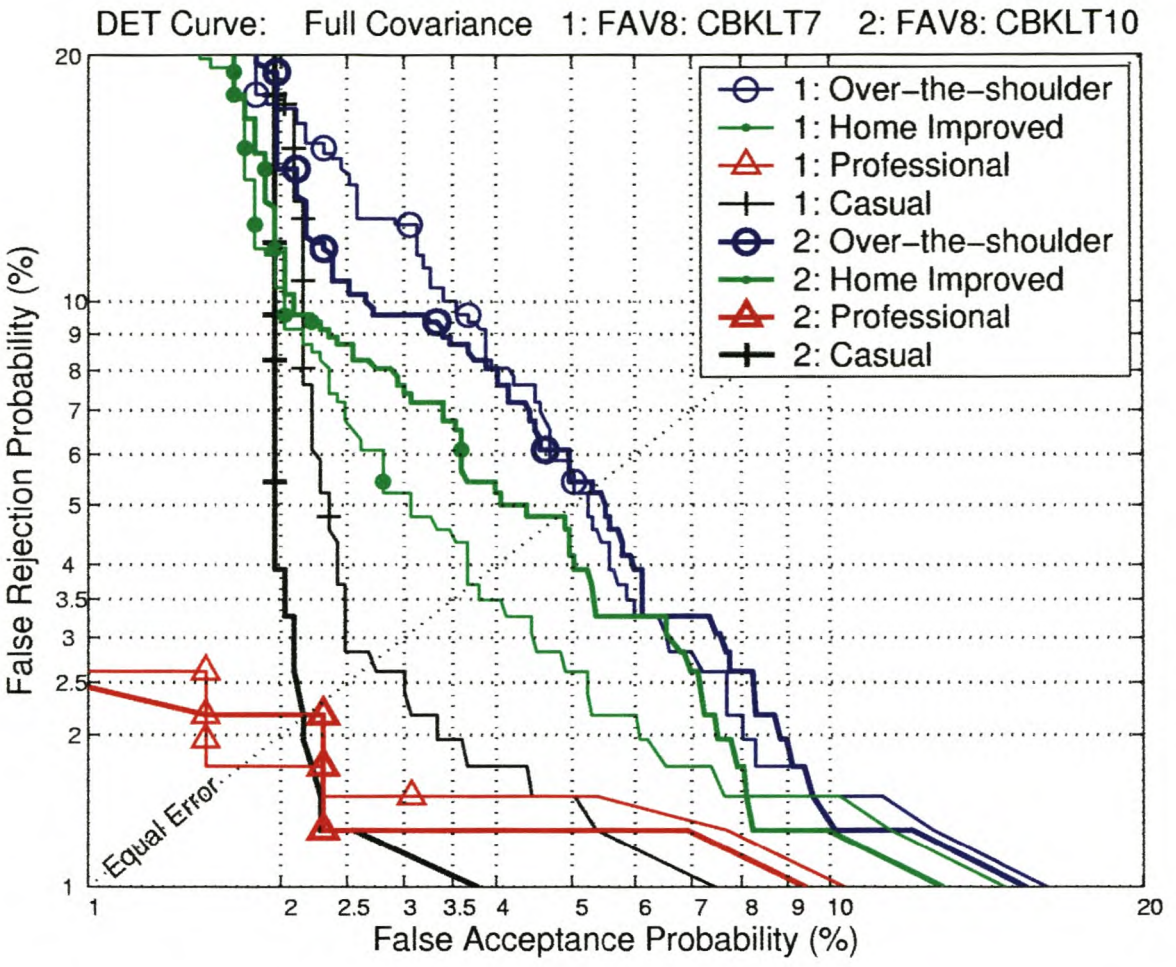


Figure A.10: *DET: CBKLT results keeping 7 and 10 dimensions (corresponds with Table 6.7).*

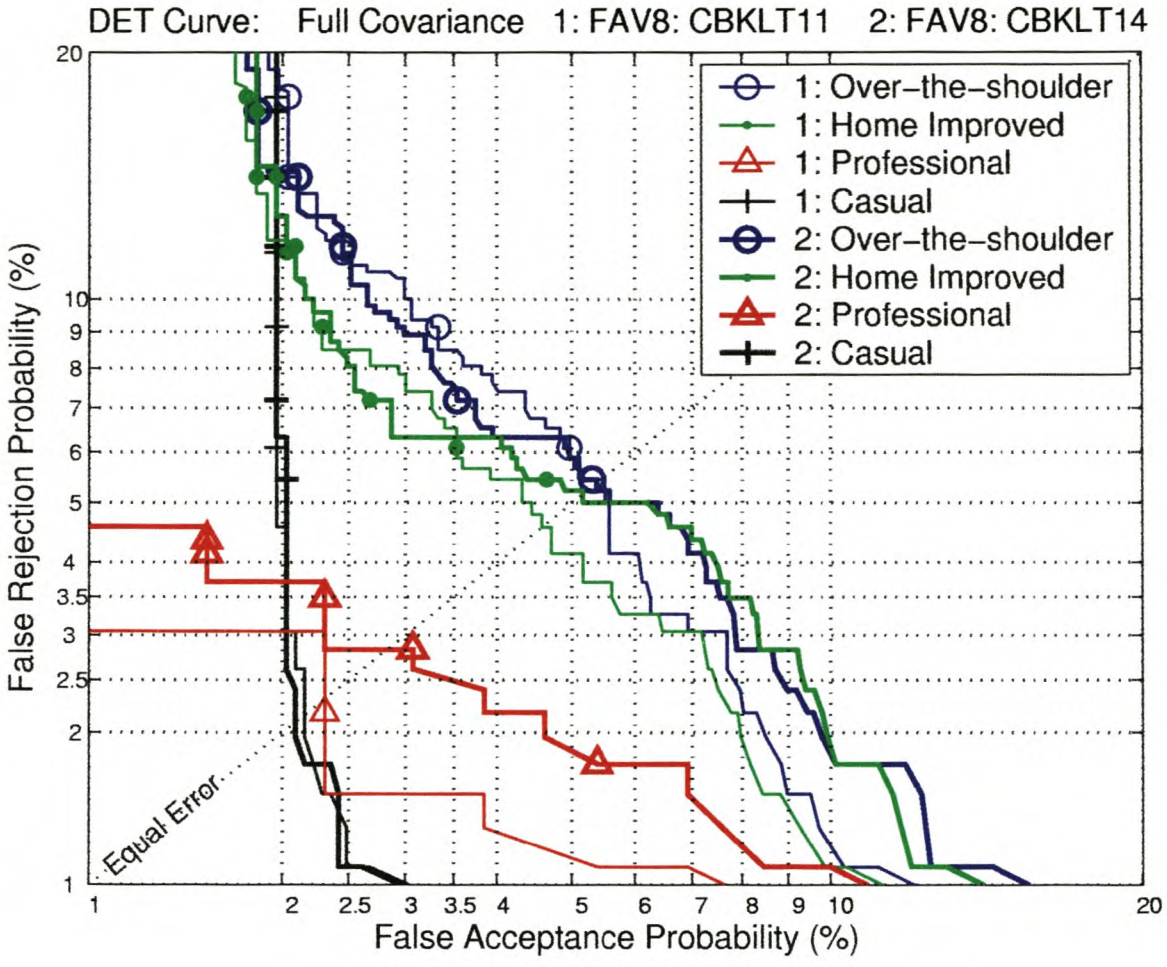


Figure A.11: *DET: CBKLT results keeping 11 and 14 dimensions (corresponds with Table 6.7).*

Dimension	Sorted Eigenvalues	Cumulative Sum (as % of total)
1	2.0904e+01	48.4628
2	1.2949e+01	78.4820
3	3.6255e+00	86.8872
4	3.3046e+00	94.5485
5	4.8816e-01	95.6802
6	4.5706e-01	96.7398
7	3.5537e-01	97.5637
8	3.0639e-01	98.2740
9	2.6563e-01	98.8898
10	2.1742e-01	99.3939
11	6.2446e-02	99.5386
12	5.6561e-02	99.6698
13	4.1880e-02	99.7669
14	3.1317e-02	99.8395
15	2.8118e-02	99.9047
16	1.1324e-02	99.9309
17	6.4981e-03	99.9460
18	5.0057e-03	99.9576
19	4.1303e-03	99.9672
20	2.8764e-03	99.9738
21	2.0318e-03	99.9785
22	1.6259e-03	99.9823
23	1.4429e-03	99.9856
24	1.0721e-03	99.9881
25	9.5908e-04	99.9904
26	8.2300e-04	99.9923
27	5.9925e-04	99.9937
28	4.3248e-04	99.9947
29	3.3675e-04	99.9954
30	3.1298e-04	99.9962
31	2.9533e-04	99.9968
32	2.7045e-04	99.9975
33	2.1922e-04	99.9980

Table A.1: *CBKLT Dimension Reduction: Typical values of the largest 33 eigen values.*

A.2.6 Volterra Series

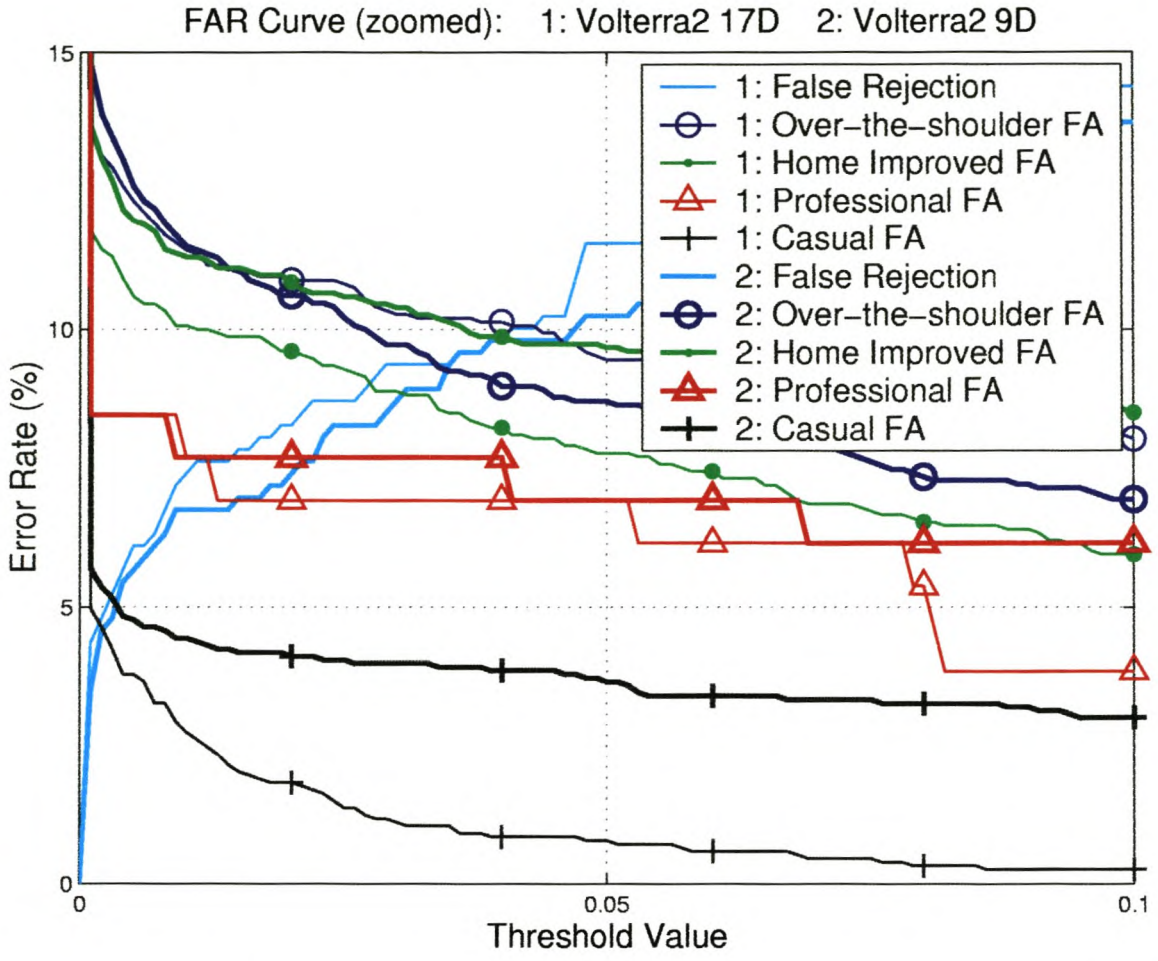


Figure A.12: *FAR zoomed: Second order Volterra expansion retaining 9 and 17 CBKLT dimensions (corresponds with Table 6.8).*

Appendix B

PatRecII Implementation: Creating a User's Signature Model

This appendix gives the details on how to create a signature model for a user in the PatRecII environment. It is included to show the general operating methods, help future researchers, and give an idea of the number of options that can be chosen.
The basic steps to create a model are:

1. make a state PDF
2. put a number of PDFs together into a PDF-Bank
3. train them with maximum likelihood estimation
4. make the PDF-Bank into a left-to-right HMM
5. train the HMM with the Viterbi algorithm
6. add Ferguson duration modelling and transition smoothing to the HMM
7. train again with the Viterbi algorithm and save the user's signature model.

Below is an example of a 'feed' file shell script used to create and train one user's signature model. The model can be made by choosing options interactively, or alternatively, the answers to all the options can be piped to the programs from the shell. The intermediate models are saved to disk at the end of each command line. This example creates a Ferguson duration emphasised HMM with 20 original states, 20^{th} order duration modelling, GMM smoothed duration fanout transition probabilities with 3 underlying 1D Gaussians, and 16D full covariance Gaussian state PDFs.

```
#!/bin/bash
##
##
## Colin Sindle, 28 August 2002.

## Info that should be set in the TOP level script:

  export TESTDIR="/home/export/csindle/devel/testing"
  export STATE="20"                          # Number of original states.
  export DUR="20"                            # Duration order.
  export PREFIX="$STATE.$DUR.TeStInG"
  mkdir  $TESTDIR/hmm/$PREFIX

## Make a BankOfSimilarities (BOS) containing all the HMM state PDFs.

## With full covariance (Gaussian_N:30199) PDFs ------>|<--dimension of PDFs:
echo "1 14922 9917 - $STATE 2 . . 13284 arrPDF . 30199 16 1 1 24960 24088
  bos.ini.gz 0"   |   objfun

## With diagonal (DiagGaussian_N:14335) PDFs ---------->|<--dimension of PDFs:
#echo "1 14922 9917 - $STATE 2 . . 13284 arrPDF . 14335 16 0 1 1 24960 24088
  bos.ini.gz 0"   |   objfun

## With circular (CircGaussian_N:15962) PDFs ---------->|<--dimension of PDFs:
#echo "1 14922 9917 - $STATE 2 . . 13284 arrPDF . 15962 16 n 0 1 1 24960 24088
  bos.ini.gz 0"   |   objfun

## Train the BOS with signatures segmented into equal
##  parts for each soon-to-be state PDF.
echo "8 . 5881 -1 2 bos.ini.gz bos.trn.gz cnfg g 0"   |   txnfun


## Make a left-to-right HMM from the BOS.
echo "1 8025 25279 user $(($STATE+2)) bos.trn.gz 2 0.2 user.ini.hmm.gz
  0"   |   objfun
```

```
## Train the left-to-right HMM.
echo "2 . 12374 -1 22 . 31369 . 25318 user.ini.hmm.gz user.trn.hmm.gz cnfg g
  0"   |   txnfun

## Add duration modelling to HMM with GMM smoothed duration fanouts probabilities.
##  The mean and variances of the GMM's underlying PDFs are set here, though they
##  will be overwritten because the GMM is now initialised from the original
##  duration stack relative frequency vector.
export VAR="$(($DUR/3))"
echo "1 user.trn.hmm.gz 8025 . 14701 $DUR . 11059 $DUR . 15908 . 7116 17924 3 .
  1534 $((1*$DUR/6)) $VAR . 1534 $((3*$DUR/6)) $VAR . 1534 $((5*$DUR/6)) $VAR e
  . 8233 2 e . 8233 2 e user.$PREFIX.ini.hmm.gz 0"   |   txnfun

## Train the Ferguson duration emphasised HMM:
echo "2 . 12374 -1 22 . 31369 . 25318 user.$PREFIX.ini.hmm.gz
 $TESTDIR/hmm/$PREFIX/user.$PREFIX.trn.hmm.gz cnfg g 0"   |   txnfun
```