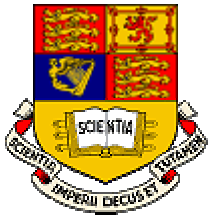
IMPERIAL COLLEGE
OF SCIENCE, TECHNOLOGY AND MEDICINE
UNIVERSITY OF LONDON
DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

A GRAPHICAL FRAMEWORK FOR THE EVALUATION OF SPEAKER VERIFICATION SYSTEMS

NIKOLAOS MITIANOUDIS



SUPERVISOR: DR. P. NAYLOR

This report is submitted in partial fulfillment of the requirement of the Degree of Master of Science (M.Sc.) and the Diploma of Imperial College (D.I.C.)

SEPTEMBER 2000

ACKNOWLEDGEMENTS

I would like to express my gratitude to Dr. P. A. Naylor and Mr. M. Brookes for their guidance, help and support given throughout the course of the project.

I would also like to express my greatest gratitude to Dr. T. Stathaki for her help, guidance and support throughout this year at Imperial College.

Most of all, I would like to thank my family for their great support during this year at Imperial College and throughout my five years of undergraduate study at the Aristotle University of Thessaloniki, Greece.

ABSTRACT

Biometric identification methods are trying to replace traditional identification methods such as PIN numbers, identity cards etc. One of the common biometrics that can be used to identify a person's identity is *speech*. *Speaker verification systems* accept or reject the identity claim of a speaker by comparing a set of measurements of his speech with a reference set of measurements of the speech of the person whose identity is claimed. Many speaker verification systems were proposed and developed in the last decade with good performance.

The basic aim of this thesis will be the analysis and development of a modular configurable speaker verification system, employing a graphical interface that will guide the user through enrolment and verification. In this framework, we will evaluate the performance of a speaker verification system using *Gaussian Mixture Models*, with many different configurations

TABLE OF CONTENTS

CHAPTER 5: GAUSSIAN MIXTURE MODELS

CHAPTER 6:GRAPHICAL USER INTERFACE DESIGN FOR A SPEAKER VERIFICATION SYSTEM

CHAPTER 7: IMPLEMENTATION AND TEST RESULTS

CHAPTER 1

INTRODUCTION

The process of authentication of a person in order to gain access to a specific system or service has obtained a great increase in interest over the last years. In order to enhance the security level obtained by identification cards and personal identification numbers (PIN), a number of biometric patterns are being used, including fingerprint, hand geometry, iris, retina, and voice. Different levels of accuracy can be achieved with these techniques, but maybe the one inducing greater problems is the human voice, due to its inherent variability [28].

However, if we are able to cope with this variability using the appropriate modeling, updating and thresholding techniques, speech is one of the best ways of introducing biometric patterns into security systems because of the simplicity of the speech acquisition systems. Additionally, for remote applications, such as telephone banking etc, speech is the more appropriate way of performing secure transactions. In this project, we are to explore the development of a *Speaker Verification* system, i.e. a system that uses speech to verify the identity of a person.

The main objectives of this project are to gain a good knowledge of current speaker verification techniques and develop a software framework for testing and evaluating speaker verification algorithms. The project involves the development of a software library that will perform all the essential speech processing tasks, as well as a Graphical User Interface that will enhance and facilitate the evaluation procedure. The whole design should give emphasis on the modularity of the system, so that it can be reconfigured. Finally, a baseline speaker verification technique using *Gaussian Mixture Models* will be implemented and evaluated within the software framework.

The structure of this report is as follows:

In *Chapter 2*, the importance of biometrics in modern identification methods will be analysed. Moreover, the design of a modular speaker verification system will be discussed in detail.

In *Chapter 3*, all the essential preprocessing techniques that are used in speaker verification are presented.

In *Chapter 4*, basic speech feature extraction techniques are presented. Some of the most commonly used feature sets in speaker verification are analysed, such as LPC, reflection, complex cepstrum, mel-frequency cepstral, delta and delta^2 mel-frequency cepstral and PLP coefficients.

In *Chapter 5*, the current speaker modelling algorithms will be presented. Emphasis will be given on the Gaussian Mixture Modelling technique.

In *Chapter 6*, the design of a configurable *Graphical User Interface (GUI)* for evaluating and testing speaker verification algorithms will be analysed

In *Chapter 7*, the implementation of a GMM speaker verification system within the testing framework is tested and the results for various configurations are presented and analysed.

In *Chapter 8*, some of the conclusions of this study will be presented, as well as some proposals for future work.

CHAPTER 2

SPEAKER VERIFICATION SYSTEM DESIGN

2.1 Introduction to biometrics

There are many every day applications, where somebody needs to 'authenticate' himself, in other words to prove his identity, in order to gain access to certain services, i.e. his email, his bank account, his office etc. Hitherto, men have used many different ways for authentication. Recently, the immense technological evolution has introduced many new methods for identification, such as identity cards, passwords, passports or security methods. However, identity cards can be lost, forged, stolen and passwords can be forgotten or compromised.

Over the last years, we have witnessed the emergence of biometrics into the identification procedure. A *biometric system* is a pattern recognition system that establishes the authenticity of a specific physiological or behavioural characteristic possessed by a user [9].

Fingerprint recognition is one of the first and best-known biometric technologies. Automatic fingerprint-based identification systems have made their first commercial appearance in the early 1960s. These systems were primarily used in forensic applications for investigating criminals. However, biometric technology has now become a feasible alternative to traditional identification systems in many application domains.

In addition to fingerprint recognition technology, other biometric technologies are beginning to emerge. New biometric applications include face (both optical and infrared), hand, finger, iris, retina, signature, and voice recognition. Investigations of other characteristics, like ear, odour, keystroke entry pattern, and gait are into research.

2.2 Performance Criteria for Biometric systems

However, despite the great demand for personal identification applications and the inefficiency of the conventional means for personal identification, biometrics technology is not as widespread as many of us might expect. One of the primary reasons is *performance*. Issues affecting the performance of biometric systems include *accuracy*, *cost*, *integrity*, and *ease of use*.

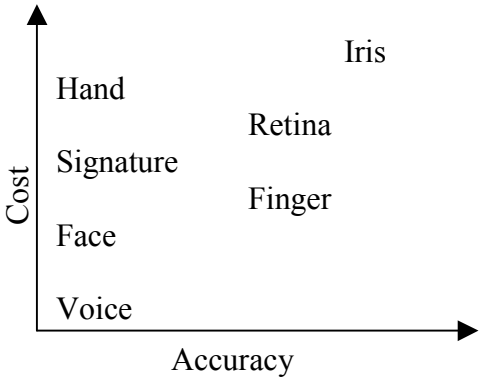


Figure 2.1: This figure illustrates the close relation between cost and accuracy of biometric identification systems [9]

- *Accuracy*

Presenting a correct password in a password-based authentication system always correctly results in the acceptance of an identity claim. In a biometric-based authentication system, even if a legitimate biometric characteristic is presented, correct authentication cannot be guaranteed. This could be due to sensor noise, limitations of the processing methods, and, more importantly, the variability in both the biometric characteristic as well as its presentation. There is also the possibility that an impostor could be incorrectly authenticated. Furthermore, the accuracy of a given biometric implementation is sensitive to the target population. To successfully apply a biometric technology to a personal identification application, it is important to understand and realistically evaluate the technology in the context of the target application and target population.

- *Cost*

Cost is greatly related to accuracy, as Figure 2.1 illustrates. Many applications are sensitive to the additional cost of including biometric technology, such logging to a PC. Given the increasing availability of inexpensive processing power, it will become possible in the near future to make biometrics accessible to new personal identification appli-

cations. The increased demand for biometrics may lower their prices, thus making biometric identification systems affortable.

- *Integrity*

Authentication is of no use if the system cannot provide assurance that the legitimate owner indeed presented the characteristic. Data from multiple, independent biometric characteristics can serve to reinforce the identity of a subject. Multiple biometrics can alleviate several other practical problems in biometrics-based personal identification. For example, a fraction of the target population may either not actually possess a particular biometric identifier or may present a characteristic that does not tender any usable information. Furthermore, certain biometrics may not be acceptable to segments of the target population.

- *Ease of use*

A biometric identification system that can not be easily operated by average people is very unlikely to present any commercial and practical success. Moreover, long and tiring training procedures or the usage of characteristics that may be considered intrusive are bound to undermine the practical application of such systems.

- *Privacy*

Despite its obvious strengths, there are a few negative preconceptions about biometrics, concerning the possible usage of biometrics in order to track people and therefore secretly violating their right to privacy. On the other hand, it is dangerous to avoid certain technologies for fear that they will be used unfairly.

All these aspects should be taken into consideration, so that biometrics identification systems can really be incorporated into every day applications.

2.3 Properties of Biometric Identification systems

In order to measure the real-life performance of biometric systems and consequently understand their strengths and weaknesses better, we must understand the elements that comprise an ideal biometric system [8]. In an ideal system

- all members of the population possess the characteristic that the biometric identifies, like irises or fingerprints;
- each biometric signature differs from all others in the controlled population;
- the biometric signatures don't vary under the conditions in which they are collected
- the system resists countermeasures.

There are two kinds of biometric systems: *identification* and *verificatio*n.

In *identification systems*, a biometric signature of an unknown person is presented to a system. The system compares the new biometric signature with a database of biometric signatures of known individuals. On the basis of the comparison, the system reports (or estimates) the identity of the unknown person from this database. Systems that rely on identification include those that the police use to identify people from fingerprints and mug shots. Civilian applications include those that check for multiple applications by the same person for welfare benefits and driver's licenses.

In *verification systems*, a user presents a biometric signature and a claim that a particular identity belongs to the biometric signature. The algorithm either accepts or rejects the claim. Alternatively, the algorithm can return a confidence measurement of the claim's validity. Verification applications include those that authenticate identity during point-of-sale transactions or that control access to computers or secure buildings. Performance statistics for verification applications differ substantially from those for identification systems. The main performance measure for identification systems is the system's ability to identify a biometric signature's owner. More specifically, the performance measure equals the percentage of queries in which the correct answer can be found in the top few matches.

The performance of a verification system, on the other hand, is traditionally characterised by two error statistics: *false-reject rate* and *false-alarm rate*. These error rates come in pairs; for each false-reject rate there is a corresponding false alarm. A false reject occurs when a system rejects a valid identity, whereas a false alarm occurs when a system incorrectly accepts an identity (an impostors).

In a perfect biometric system, both error rates would be zero. Unfortunately, biometric systems aren't perfect, so you must determine what trade-offs you're willing to make. If you deny access to every-one, the false-reject rate will be one and the false-alarm rate will be zero. At the other extreme, if you grant everyone access, the false-reject rate will be zero and the false-alarm rate will be one. Clearly, systems operate between the two extremes. For most applications, you adjust a system parameter (a threshold) to achieve a desired false-alarm rate, which results in a corresponding false-reject rate. There is also an operation point where the FA and FR rates are equal. This is *called equal-error rate (EER)*. The next figure illustrates this concept.

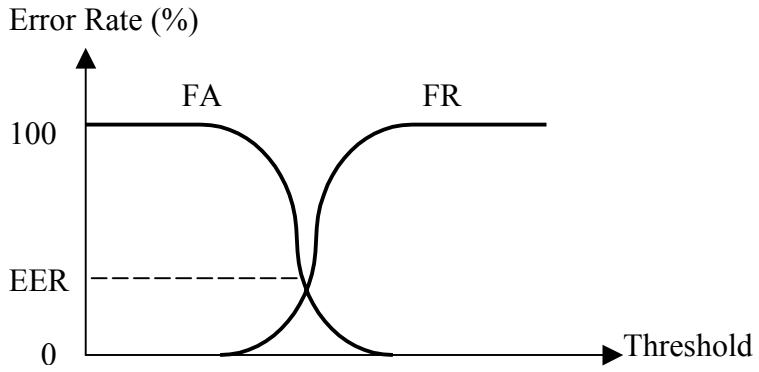


Figure 2.2: False-Alarm and False-Reject curves

2.4 Speaker Verification Systems

Among the other biometrics methods, identity verification based on a person's voice has special advantages for practical deployment. Speech is our most natural mean of communication and therefore the public acceptance of the system can be very high. Moreover, the added advantage of being less intrusive to the user's privacy, compared to other biometrics systems, such as retinal scanning and fingerprint verification is also well

appreciated [25]. Furthermore, research on speech has been extensive over recent years offering a significant background for the development of high performance verification systems. We will shortly look at some of the human speech properties.

2.4.1 Human Speech properties

Speech is produced by co-ordinated movements of the articulators (tongue, jaw, vocal folds etc) in a sequence of configurations, and by simultaneous application of pressure via the diaphragm to air in the lungs [1]. Chosen amounts of air exiting through a chosen shape of vocal tract produce the desired sequence of sounds. We can generally model speech as a baseband signal, limited to a bandwidth of 7-8 KHz. The spectral characteristics of the speech wave are time-varying, since the physical system changes rapidly over time. As a result, speech can be divided into sound segments that possess similar acoustic properties over short periods of time, i.e. the vowel o in the word 'shop'. Nonetheless, speech is not quite a string of discrete well-formed sounds, but rather a series of steady-state sounds with intermediate transitions. The preceding and/or succeeding sound in a string can affect whether a target is reached completely, how long it is held and other finer details of the sound. This interplay is generally called *coarticulation*. The inability of the speech production system to change instantaneously is due to requirement of finite movement of the speech system articulators (i.e. muscles) to produce the sound.

Furthermore, if we look at the spectrum of a steady-state segment, we can observe several frequency components. The exact position of these frequency components is much dependent on the actual shape, size and position of several cavities that are formed in the vocal tract. Each vocal tract shape is characterised by a set of resonant frequencies. Since those resonances tend to 'form' the overall spectrum, we refer to them as *formants* or *formant frequencies*. In principle, there is an infinite number of formants in a given sound, but in practice, we usually find 3-5 in the Nyquist band after sampling. Formant frequencies usually appear as peaks in the spectrum.

A special category of segments that speech is usually divided into are the so-called *phonemes*. More specifically, phonemes are the basic theoretical units for describing how speech conveys linguistic mean. Each phoneme can be considered to be a code that con-

sists of a unique set of articulatory gestures. From an acoustical point of view, we can say that the phoneme represents a class of sounds that convey the same meaning. For example, in the word 'shade', we can meet the phonemes 'sh' and 'aa'. Phonemes are usually divided into two categories: *voiced* and *unvoiced*. Voiced phonemes (and sounds generally) are produced by forcing air through the glottis or an opening between the vocal folds. A phoneme can be regarded as unvoiced, if it is generated by forming a constriction at some point along the vocal tract, and forcing air through the constriction to produce turbulence. However, there are some sounds that can be simultaneously voiced and unvoiced and they are called mixed phonemes.

Moreover, there are some more metrics that are used to describe phonemes. One of these is the *fundamental period* T_o, which is actually the time between successive vocal fold openings, while the rate of vibration is called the *fundamental frequency* $F_o=1/T_o$. More frequently, we will meet the term *pitch* basically implying the fundamental frequency F_o. Pitch can take values from 50 to 200 Hz and shouldn't be confused with formant frequencies 250 to 5000 Hz. In speech signals, pitch doesn't remain constant, but it fluctuates according to the stress the speaker poses to his phrasing.

2.4.2 Speaker Verification Systems architecture

Speaker verification systems fall into two basic types: *text-dependent* and *text-independent* [6,8].

In *text-dependent verification*, the speaker says a predetermined phrase. This technique inherently enhances verification performance, but requires a cooperative user. In such systems, with adequate time alignment, one can make precise and reliable comparisons between two utterances of the same text, using extended phonetic classification of recorded utterances.

In *text-independent recognition*, the speaker need not say a predetermined phrase and need not co-operate or even be aware of the recognition system. These systems often use long-term statistics of speech to extract speaker specific data. Normally, these systems require longer speech samples for training and verification.

A Speaker verification task is normally divided into two procedures:

- *Enrolment procedure*

During this procedure the user needs to register himself to the system. In other words, the user may provide the system with a set of utterances so that it can build his speech model and use it as a reference later. The procedure builds an initial reference template for a speaker by capturing the verification utterance. It's important to obtain a good enrolment template for a high performance system.

The whole procedure can be modelled by the following three modules. The first stage is the *Preprocessing* module, which prepares the training speech samples for the next modules. The next stage is the *Feature Extraction* module that aims to extract the information about the user conveyed in the training speech samples. The final stage, the *Reference Template* module, aims to arrange and store the extracted feature sets, so that it can be then used for reference, when the user requires authorisation.

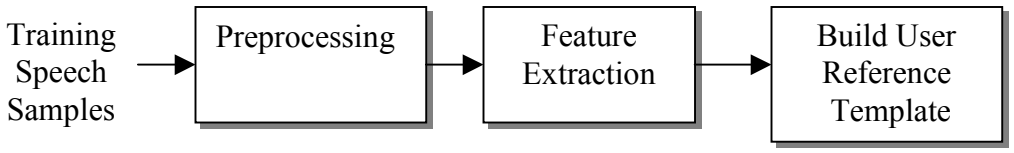


Figure 2.3: Speaker Enrolment Procedure

- *Verification procedure*

During this procedure the user provides a set of utterances, so as to authenticate himself to the system, so to gain access to certain resources. The system tries to compare the presented speech samples to the already recorded user's speech model and decide whether to accept or reject the user.

The whole procedure can be modelled by the following block diagram. The first two modules are identical to the ones used in the enrolment procedure. Hence, these two modules extract a set of features vectors from the user's recorded utterance that are compared with the user's reference template. Finally, a decision rule grants or declines authorisation to the user.

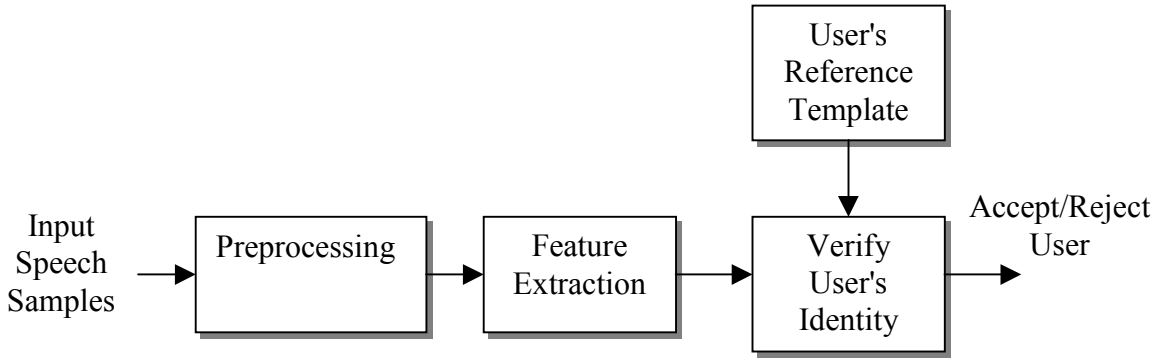


Figure 2.4: Speaker Verification Procedure

The basic aim of this project is to build a modular speaker verification front-end with a graphical user interface. In other words, we aim to build a system, where the administrator can change the configuration of the speaker verification system by adding, removing stages in each module or even modify its parameters. This made it absolutely necessary to maintain the modularity of the system in the MATLAB programmes developed as well as in the Graphical User Interface. The following figures describe the proposed system's operation in detail. The next chapters follow the same order in the analysis of the system.

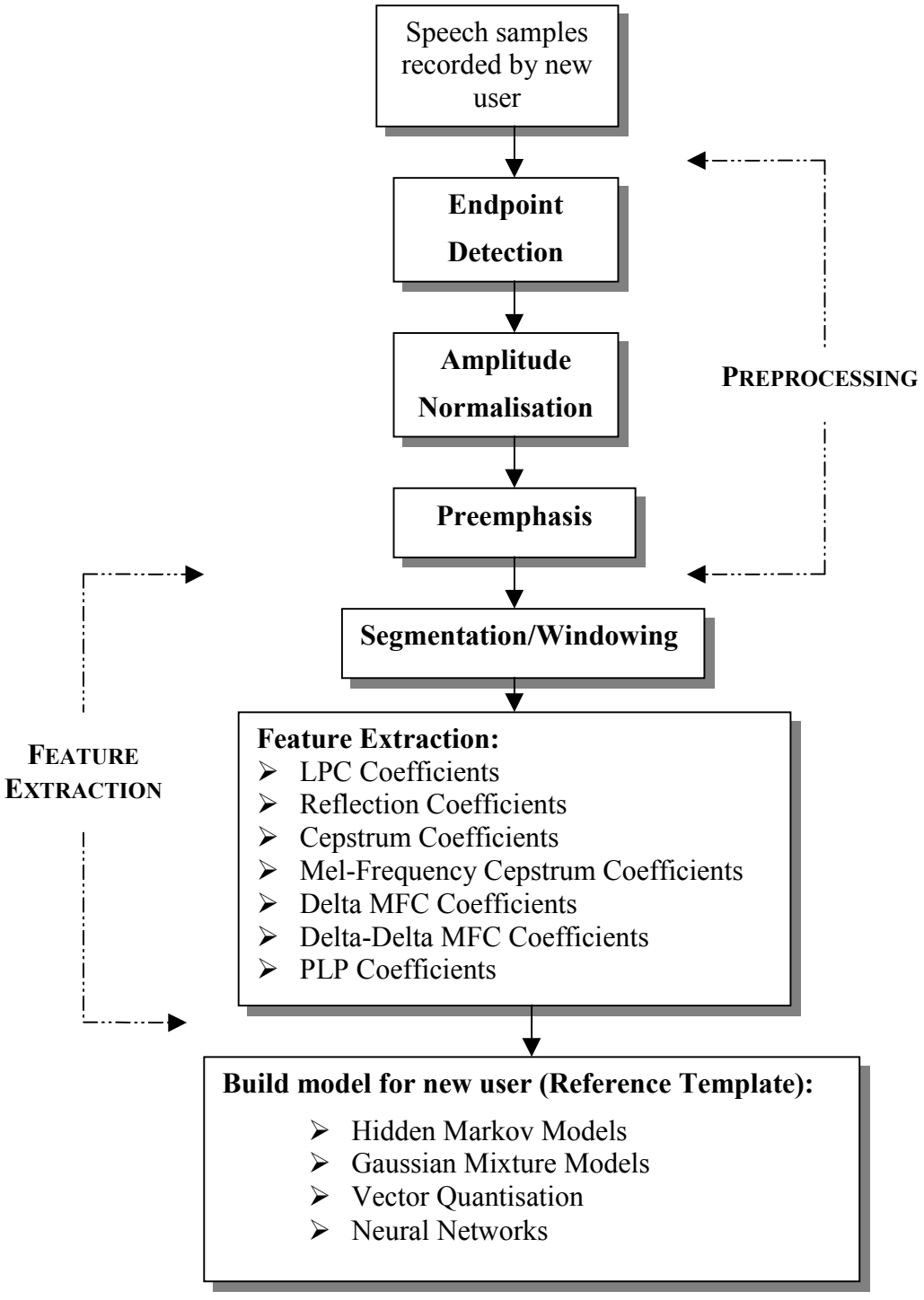


Figure 2.5: Speaker Enrolment Procedure Overview

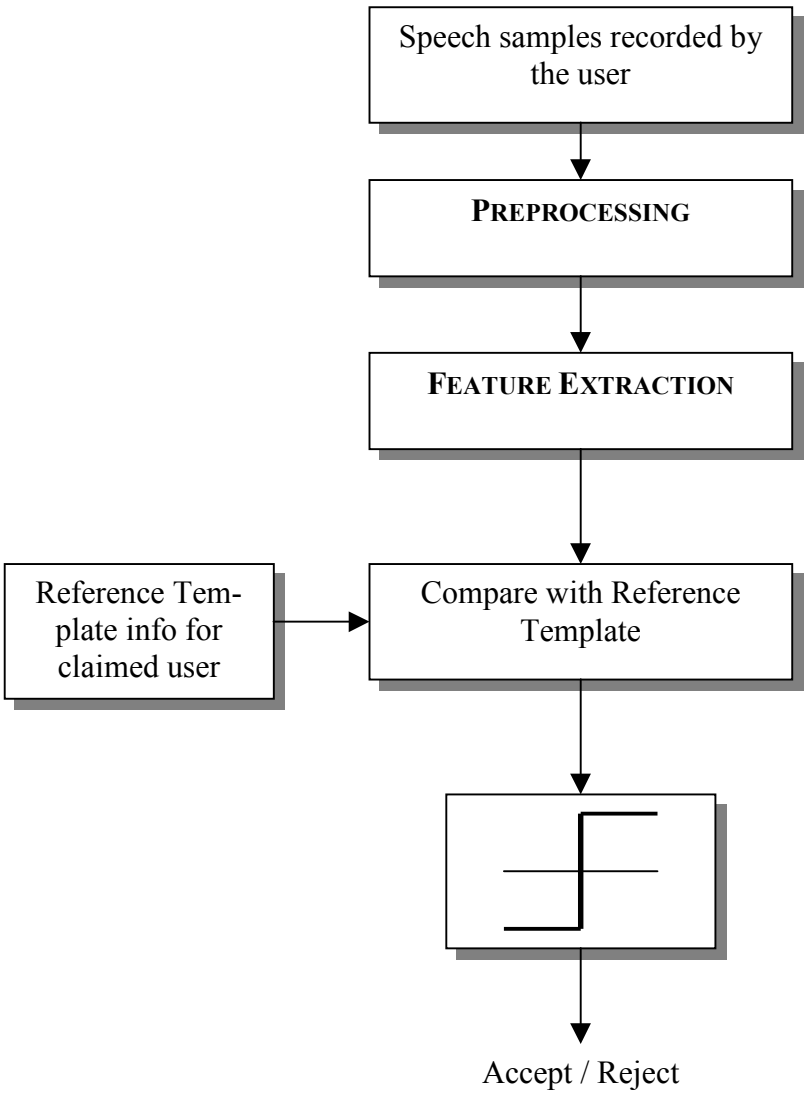


Figure 2.6: Speaker Verification Overview

CHAPTER 3

PREPROCESSING

3.1 Introduction

The first module in a speaker verification front-end will be the preprocessing of the input speech data. In this module, the input speech data used either for training the system or for the verification procedure are imposed to certain signal processing algorithms, so that the *feature extraction,* performed in the next module can be more accurate.

In our preprocessing module, we will include *endpoint detection*, *amplitude normalisation* and *preemphasis*. It is not absolutely necessary to use all these algorithms. It usually depends on the recording quality of the speech samples.

3.2 Endpoint Detection

In a speaker verification system, it is a fundamental task to detect the endpoints of speech signals. In other words, the system must detect and remove the non-voiced parts in the user's recorded utterance. The main motivation behind *endpoint detection* is that the processing of these non-voiced parts is bound to undermine the performance of our system. The features extracted from these segments can not characterise the speaker's identity and can mislead the recogniser. Moreover, we reduce the total processing time of the system by removing these unwanted parts.

The detection of the edges of a spoken word is a very difficult procedure. One particular class of problems is those attributed to the speaker and to the manner of producing speech [2]. For example, during articulation, the talker often produces sound artifacts, including lip smacks, heavy breathing and mouth clicks and pops. Some of them can be

separated quite easily, other not. For example, a heavy breathing noise in the beginning of the utterance can not be separated, whereas a mouth click can.

A second factor that makes reliable speech endpoint detection difficult is the environmental conditions in which speech is produced [2]. The ideal recording environment is a quiet room with no acoustic noise or signal generators other than that produced by the speaker. Such an ideal environment is not always practical. Therefore, an endpoint detection algorithm should consider noise produced by other common sources (fans, road noise, etc). Sometimes non-stationary sounds may occur during the recording, such as a door slam, a car horn, or even speech interference by a radio, TV or background conversation.

The procedure that's followed in most endpoint detection algorithms is basically the following: the speech waveform is firstly segmented into overlapping frames. The choice of the frame length is extremely important. Usually, speech is segmented into 25.6ms frames with 50% overlapping.

There are certain metrics that are normally used to identify a spoken utterance from the background noise present in a recorded utterance. The first metrics is the *logarithm of the frame's energy* and the other one is the number of zero-crossings.

- **Log-Energy**

This metric calculates the log energy of each frame, as it is obvious by the next equation:

$$LE_i = \log\left(\sum_{n=1}^{N} x_i[n]^2\right) \qquad (3.1)$$

The logarithm function makes a non-linear compression to the amplitude of the signal and the weak portions of the signal have the opportunity to reveal their details sufficiently [12]. Sometimes, we can normalise the energy LE_i by the maximum frame energy in the speech segment.

It's very logical to say that the utterance will be represented by that contiguous set of frames whose energy is above a set threshold ITU. However, research has shown that using only this energy criterion for endpoint detection can omit unvoiced phonemes that

usually appear in the beginning or at the end of a word. This can be compensated by using the *zero-crossings* rate.

- **Zero Crossings**

This metric calculates the number of times our signal crosses zero level (zero-crossings rate) in each frame. The next equation gives the mathematical definition:

$$CX_i = \sum_{n=1}^{N} | \operatorname{sgn}(x[n]) - \operatorname{sgn}(x[n-1]) | \qquad (3.2)$$

Sometimes, we can normalise the zero-crossings rate CX_i by the maximum frame rate in the speech segment.

Actually, we use zero-crossings rate to correct the endpoints defined by the energy criterion and correctly depict the unvoiced phonemes. The zero-crossings rate can be an efficient tool for that operation, as we know that unvoiced phonemes feature greater high-frequency terms.

We are going to present two algorithms that employ the above metrics to define the endpoints N_1, N_2 of an utterance. They are actually based on the same concept, The choice for the ITU and IZCT thresholds is done by using successive tests. It depends much on the noise level in the input speech waveforms.

- **Algorithm 1** [10]

1. Segment the waveform into overlapping frames and calculate the log-energy and the number of zero-crossings for each frame.
2. Start from the frame with the great log-energy and move left, until you find a frame, whose energy is below the set-threshold ITU. This sets the left endpoint N1. Perform the same procedure by moving right and define the right endpoint N2.
3. Perform a correction to both endpoints, by using zero-crossings. Move up to 25 from N1 and calculate the number of frames, whose number of zero-crossings exceeds the threshold IZCT. If they are less than three, no correction is performed, otherwise N1 moves to the first frame that exceeds the threshold IZCT. The same procedure is followed to update the right endpoint N2.

The following figures illustrate the application of the above endpoint detection algorithm. We can see that this algorithm performs accurate endpoint detection, preserving the unvoiced parts in the beginning and at the end of the utterance.

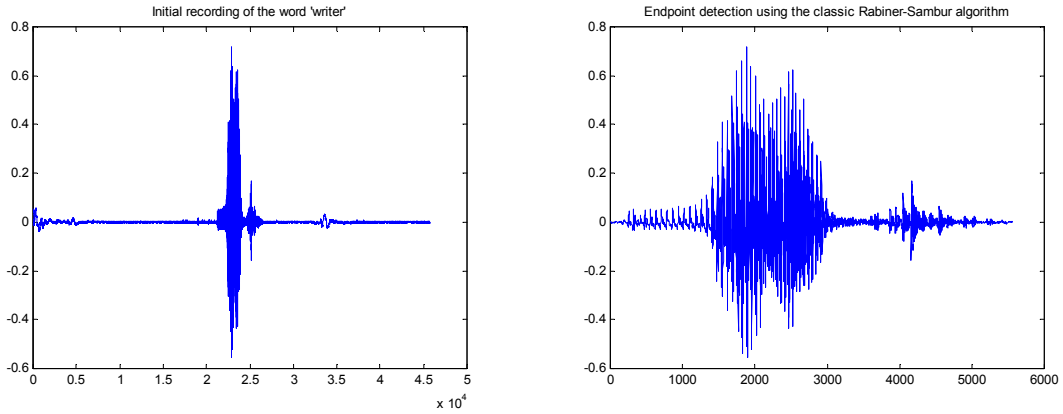


Figure 3.1: Endpoint Detection using Algorithm 1

- **Algorithm 2** [11]

This algorithm progresses in the same way as the previous one. Nonetheless, it uses different criteria for endpoint definition. The next steps describe the left limit detection procedure. Equally, we easily adapt this algorithm for right limit detection.

Left limit detection using Energy

1. Detect the frame containing the maximum energy (ba).
2. Set bi = ba and bcl = ba +2
3. if the log energy of the ba frame is greater than 0.4 then decrease ba by 1 and go to step 3.
4. if bcl-ba ≥ 4 then set bcl=ba
5. Decrease ba by 1
6. if bi – ba ≥ 12 then go to step 8.
7. if log energy of ba is less than 0.7 then go to step 5, else set bcl=ba, decrease ba by 1 and go to step 3.

8. Left limit is ba

Left limit correction using zero-crossing rate

1. Suppose ba represents the previous left limit estimate
2. if the number of zero crossings in the ba frame is greater than the threshold IZCT, then decrease ba by one and go to step 2.
3. Set bi = ba
4. Set bcpl = bi and NCX=0
5. Decrease ba by one, if the number of zero crossings in the ba frame is greater than the threshold IZCT, increase NCX by 1, otherwise go to step 7.
6. if NCX ≥ 2, set bi = ba
7. if bcpl –ba <4, go to step 5
8. if NCX ≥ 2, go to step 4
9. Corrected left limit is ba.

The following figures illustrate the application of the above endpoint detection algorithm. We can see that this algorithm performs relatively accurate endpoint detection, leaving some extra silent parts.

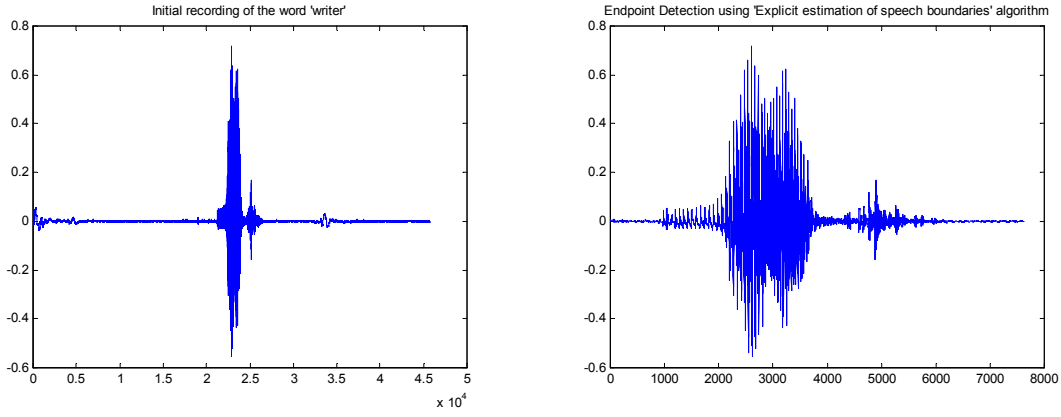


Figure 3.2: Endpoint Detection using Algorithm 2

Some other algorithms employ a combination of zero-crossings with an energy criterion, named *band crossings,* in order to capture the unvoiced segments more efficiently.

We can also improve the performance of an endpoint detection algorithm by preemphasising (chapter 3.4) the input signal before log-energy calculation [12]. This is related to the physiology property of human hearing. Human ear is insensitive to low frequency signals, but is good at capturing the active properties of speech signal. In the following figure, we can see the validity of this statement on Algorithm 1.

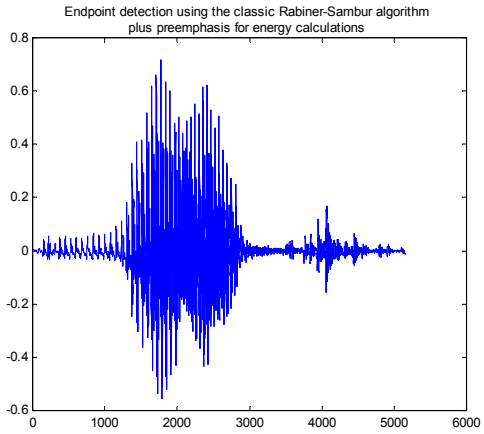


Figure 3.3: Improvement on Endpoint Detection Algorithm 1

However, we can see that the endpoint detection algorithms are mainly empirical in the way they define endpoints and their performance is mainly based on the quality of the input speech waveform.

3.3 Amplitude Normalisation

The amplitude normalisation is a very common technique used in many signal processing applications. When the application has to deal with signals coming from different sources with different output signal levels, it would be very beneficial to have a programme that will amplify or attenuate the input signal to a constant output level. The whole concept matches that of an *Automatic Gain Control (AGC)* device.

Suppose we have the following discrete signal x[n] that has resolution of N bits and therefore take values from 0 to 2^N-1.

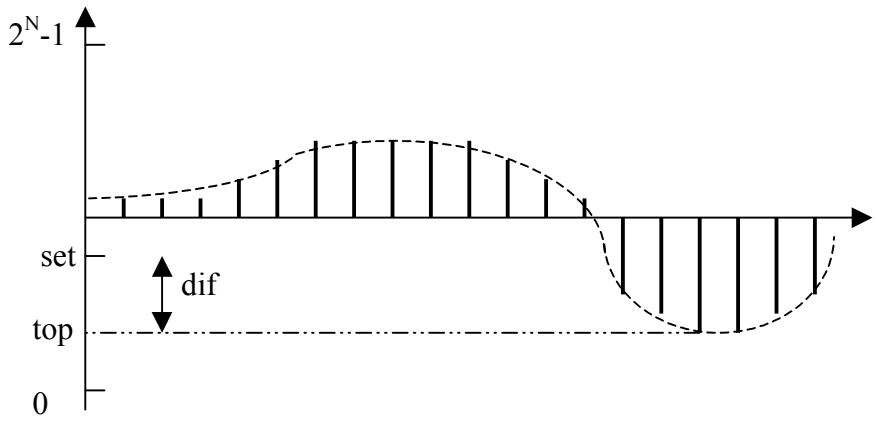


Figure 3.4: Amplitude Normalisation

Suppose we want to normalise the signal to a% of the maximum resolution. The first step will be to trace the maximum absolute value x[n] takes in this segment. Then, we calculate the maximum amplification (or attenuation) that will be added to the signal (for those x[n]=top). The amplification get linearly smaller as x[n] values decrease in absolute value towards the 2^{N-1}-1 level.

The whole normalisation algorithm can be summed in the next simple steps:

1. Find the maximum absolute value of x[n] in the corresponding speech segment (top)
2. Calculate set = (1-a) (2^{N-1}-1) , (suppose $0 \le a \le 1$)
3. Calculate dif = top-set
4. For every sample x[n] of the segment do the following:

   If x[n] ≥ 0 then

$$x[n] = x[n] + dif * \frac{x[n] - (2^{N-1} - 1)}{(2^{N-1} - 1) - top} \tag{3.3}$$

   if x[n]< 2^{N-1}-1 then x[n] = 2^{N-1}-1

   else

$$x[n] = x[n] - dif * \frac{(2^{N-1} - 1) - x[n]}{(2^{N-1} - 1) - top} \tag{3.4}$$

   if x[n]> 2^{N-1}-1 then x[n] = 2^{N-1}-1

The next figures show an example of amplitude normalisation.

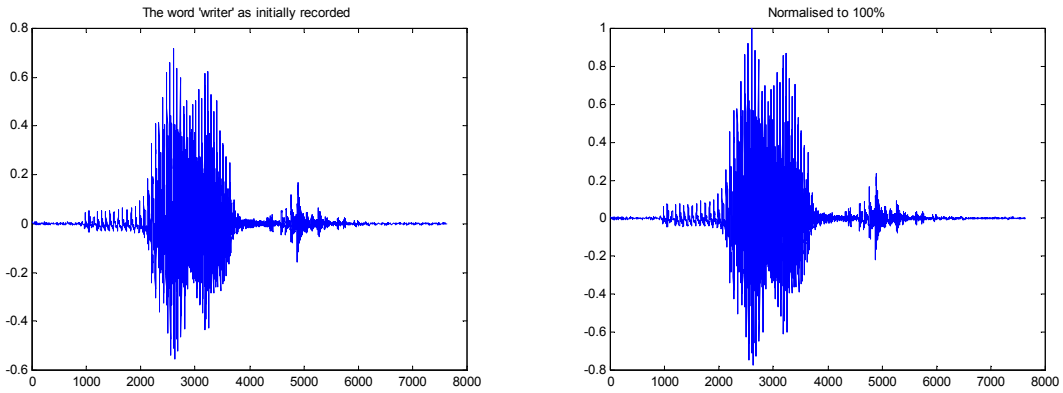


Figure 3.5: Amplitude Normalisation example

3.4 Preemphasis

In practical speech analysis applications, we will notice that speech waveforms are usually preemphasised prior to extracting speech feature sets. That is to say that we apply a filter that increases the relative energy of the high-frequency spectrum, i.e. a high pass filter. We use a FIR HP filter, described by the following transfer function:

$$H_{preemphasis}(z) = 1 - az^{-1} \tag{3.5}$$

Normally, a takes values from 0.9 to 1.0. We can see the frequency response of this HP filter for a=0.98 in the following figure:

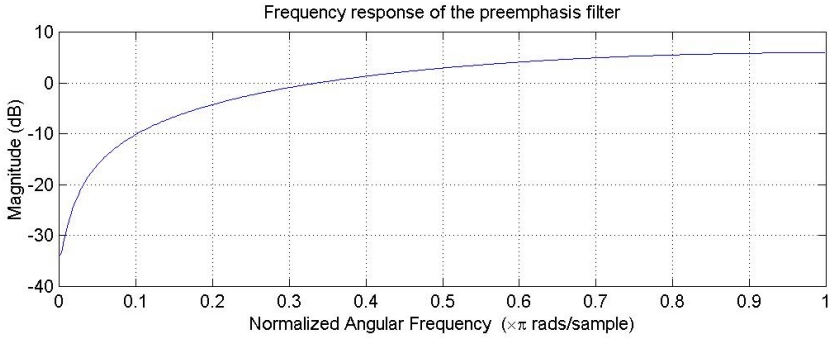


Figure 3.6: Preemphasis Filter Frequency Response

The preemphasis filter actually models the lip radiation characteristics, and introduces a zero near w=0, and a 6-dB per octave shift on the speech spectrum. There are several reasons for employing a preemphasis filter. First of all, the introduction of this filter tends to cancel the glottal or lip radiation effects on speech production and we are more accurate when we represent the whole speech production procedure with the vocal tract model filter [1].

Another reason for pre-emphasis is to prevent numerical instability. If the speech signal is dominated by low frequencies, then this may result to an autocorrelation matrix that its inversion will cause numerical instability. This can be prevented with a preemphasis filter.

Moreover, some phonemes feature peaks in high frequencies, which can not be detected by the LP algorithm (see also 4.5), because they contain low-frequency components. A little boost in high-frequency components can facilitate the whole procedure. This can have great application in case of voiced phonemes. We also note that such preemphasis filters also raise frequencies above 5 KHz, a region in which the auditory system becomes increasingly less sensitive. However, frequencies 5 KHz above are naturally attenuated by the speech production system and normally are assigned a significantly smaller weight in a typical speech recognition system [29].

CHAPTER 4

FEATURE EXTRACTION

4.1 Introduction

After performing all the essential preprocessing in the user's input speech samples, the next module in a speaker verification front-end will be the *Feature Extraction* module. The basic role of this module is to use the input speech samples, so as to calculate certain parameters (feature sets) that will be used to register the identity of the user. These feature sets should be able to depict certain metrics in human speech that will be effective in the user's identification.

In this chapter, we will examine the methodology of extraction of some the most commonly-used feature sets. We will also look into the segmentation and windowing of the input speech samples.

4.2 Segmentation and Windowing

The *Segmentation* and *Windowing* of speech signals is a very common part in many speech processing applications. According to this technique, the input signal is segmented into frames of constant length M that overlap each other. Each of these frames is then multiplied by a *hamming window*. This is much more preferable than using a rectangular window, because the hamming window helps to smooth the abrupt discontinuity at the frame boundaries [1]. Figure 4.1 illustrates the segmentation and overlapping procedure.

The fact that can justify the segmentation of the speech signal is that during speech production the vocal tract constantly changes. Hence, vocal tract modelling would only

have meaning in certain speech segments where its parameters stay unchanged. Therefore, the choice of the frame length is very important.

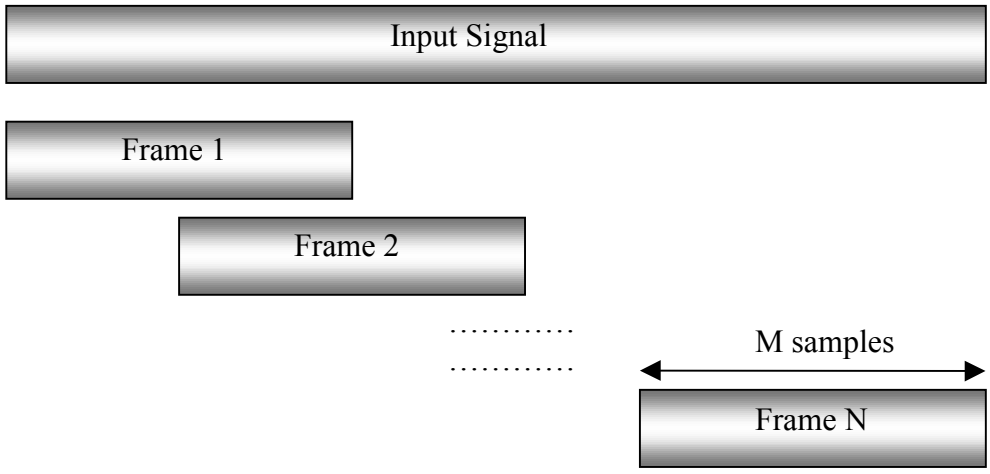


Figure 4.1: Segmentation and Overlapping

A long speech frame would normally cause errors in the estimation of feature sets, as the vocal tract parameters will not remain the same. A short speech frame may not provide sufficient data for the feature extraction algorithms, leading to errors.

The overlapping of speech frames is used in order to increase the redundancy of the input signal, so as to provide more speech data to the feature extraction algorithms. Moreover, we can capture the changes in the vocal tract more accurately. A common choice for overlapping is 50%, however, one could choose a smaller overlapping ratio.

After segmenting the input signal, the feature extraction algorithms manipulate each frame in order to produce the corresponding feature sets.

Normally, this procedure is a part of the preprocessing module. Nonetheless, in our front-end each feature extraction algorithms performs a separate segmentation and windowing of the input signal. This is done mainly to facilitate experimentation, so that each feature extraction algorithm can choose separate window length and overlapping ratio.

4.3 Properties of a Speech Feature Set

Defining a set of speech parameters that can identify a person's identity is a very difficult task. The speech signal is a complex function of the speaker's certain physical characteristics, such as the vocal source/tract dimensions, emotional state and recording environment. Speech data collected from the same speaker at different times generally show great variability. This illustrates the importance of measuring a proper acoustic feature set in our speaker verification system.

Some basic properties, a speech feature set should possess are stated below [6]:

- Discriminate between speakers while being tolerant of intra-speaker variabilities.
- Be easily measurable from the speech signal
- Be stable over time
- Be less susceptible to impostors.

Classic speech signal metrics such as *voice pitch*, *formant frequencies* etc tend to become less useful in a speaker verification application, as they are too difficult to measure and vary significantly especially in noisy environments and by the user's sentimental mood. We are going to look into some important feature sets such as the *linear predictive, mel-frequency cepstrum, and perceptual linear predictive* coefficients.

4.4 Log – Energy

One of the first measurements (features) that can be extracted from a speech segment is its energy. The energy spectrum of a speech signal describes the frequency content of the signal over time. In many speaker verification systems, we take the logarithm of the segment's energy. Suppose we have a speech segment x[n] of length L. Then, the log energy can be calculated from the following set:

$$E = \log\left(\sum_{n=1}^{L} | X(e^{j\omega}) |^2 \right) \tag{4.1}$$

, where $X(e^{j\omega})$ is the DFT of the speech segment

4.5 Linear Predictive Coefficients

One of the most powerful speech analysis techniques is the method of *linear predictive analysis*. This method has been one of the basic speech processing techniques over the past years, used to estimate the basic speech parameters, e.g., pitch formants, spectra, vocal tract area functions, as well in basic low bit-rate transmission algorithms. The importance of this method lies basically on the accurate estimates of speech parameters and its relative small processing time.

The basic concept behind linear predictive analysis is that a speech sample can be approximated by a linear combination of past speech samples [1,2,3,14]. In order words, we can write that:

$$\hat{x}[n] \approx a_1 x[n-1] + a_2 x[n-2] + ... + a_p x[n-p] = \sum_{k=1}^{p} a_k x[n-k] \qquad (4.2)$$

, where a_i are the *LP coefficients*, which are considered to be constant over the speech frame. This estimator is actually a linear predictor, or in other words a FIR filter. We can convert the above equation to an equality by including an excitation term Gu[n]. Therefore,

$$x[n] = \sum_{k=1}^{p} a_k x[n-k] + Gu[n] \qquad (4.3)$$

Taking the z-transform of the above equation we get that:

$$X(z) = GU(z) + X(z)\sum_{k=1}^{p} a_k z^{-k} \Rightarrow GU(z) = X(z)(1 - \sum_{k=1}^{p} a_k z^{-k}) \Leftrightarrow$$

$$H(z) = \frac{X(z)}{U(z)} = \frac{G}{1 - \sum_{k=1}^{p} a_k z^{-k}} = \frac{G}{A(z)} \qquad (4.4)$$

As we can see the speech signal is created after the excitation signal passes through an all-pole filter H(z).

In many books, the whole human speech-production procedure is modelled by the schematic in Figure 4.2.

The actual excitation function for speech u[n] is either a quasi-periodic pulse train (for voiced speech sounds) or a random noise source (for unvoiced sounds).We can see that the normalised excitation signal u[n] is chosen by a switch whose position is controlled by the voiced-unvoiced character of the speech. Then, the appropriate gain G of the source is estimated from the speech signal and finally the scaled input signal produces the

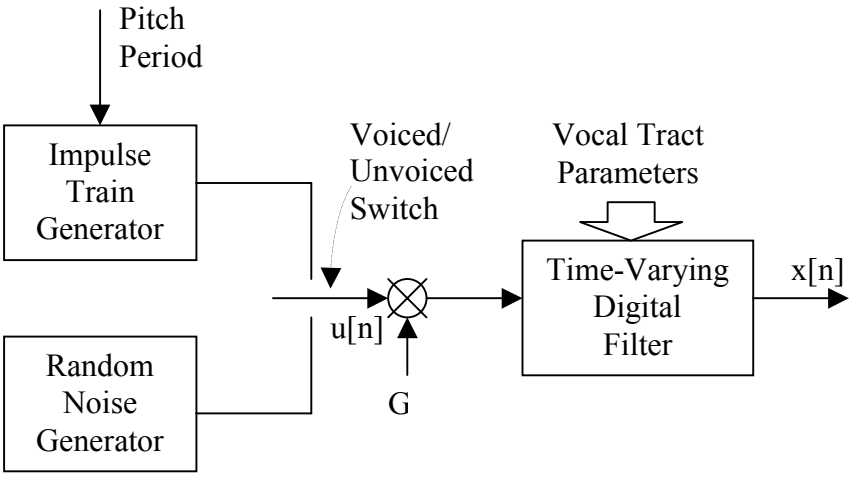


Figure 4.2: Simplified Model for speech production

signal x[n] after passing through a time-varying digital filter (all-pole filter) H(z) controlled by the human vocal tract parameters. We can see that the actual parameters of this model are the voiced/unvoiced classification, pitch period for voiced sounds, the gain parameter and the coefficients of the digital filter. The characteristic of all these parameters is that they vary slowly with time.

The basic characteristic of the linear prediction coefficients a_i is that the frequency response of the *vocal tract filter* H(z) outlines the actual frequency response of the speech signal over that segment. Moreover, the peaks that appear in the frequency response of the *vocal tract filter* H(z) depict the formant frequencies, in the case that the specified speech segment contains a phoneme.

A very important issue is the choice of the linear prediction coefficients order p. It's very logical that for greater values of p, the frequency response of the vocal tract model filter will outline the frequency response of the speech segment more accurately. On the other hand, a big choice for the p will definitely increase the required processing time. A common choice is p=12.

The basic problem of linear prediction analysis is to determine the set of predictor coefficients a_k, directly from the speech signal so that the spectral properties of the digital filter in the previous figure match those of the actual speech waveform. We can achieve this, usually by trying to minimise the mean square difference between the true samples x[n] and our estimates $\hat{x}[n]$, over a range of N samples. Hence, we are trying to minimise the following expression:

$$E = \sum_{n=1}^{N} (x[n] - \hat{x}[n])^2 \tag{4.5}$$

We can even try to predict future samples of x[n], but then in order to calculate the error we will have to minimise the following expression:

$$E = \sum_{n=1}^{N} (x[n+r] - \hat{x}[n])^2 = \sum_{n=1}^{N} (x[n+r] - \sum_{k=1}^{p} a_k x[n-k])^2 \tag{4.6}$$

The coefficients a_k that can minimise the above equation are those that can set the partial derivative of E to zero. More specifically, the equation we have to solve is the following:

$$\frac{\partial E}{\partial a_k} = 0 \text{, for k=1,…..,p} \tag{4.7}$$

If we assume that x[n] is speech signal of length L then we can produce the following set of equations, exploiting the partial derivative:

x[1+r]	=	$a_1x[0]+ a_2x[-1]+…+ a_px[1-p]$	(n=1)
……	=	……	……
x[p+r]	=	$a_1x[p-1]+ a_2x[p-2]+…+ a_px[0]$	(n=p)
……	=	……	……
x[L-1]	=	$a_1x[L-2-r]+ ……+ a_px[L-1-r-p]$	(n=L-1-r)
……	=	……	……
x[L-1+p+r]	=	$0+ …+ a_px[L-1]$	(n=L-1+p)

,or equivalently:

$$\begin{bmatrix} x[1+r] \\ \dots \\ x[p+r] \\ \dots \\ x[L-1+p+r] \end{bmatrix} \approx \begin{bmatrix} x[0] & x[-1] & \dots & \dots & x[1-p] \\ .... & \dots & \dots & \dots & \dots \\ x[p-1] & x[p-2] & \dots & \dots & x[0] \\ \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & \dots & x[L-1] \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ .. \\ .. \\ a_p \end{bmatrix} \Rightarrow$$

$$x = Xa \tag{4.8}$$

When r=0 there are two methods of linear prediction, which are distinguished solely by which equations are included in the error sum:

1. ***Autocorrelation method:***

All possible equations from n=1 to n=L-1+p are included. Thus, if the extent of the input date x[n] is finite $0 \le n \le L$, the prediction distance is r, and the length of the predictor is p, there will be L-1+p equations. In some cases the predictor will be trying to match 0,because x[L]=0, x[L+1]=0, ……, x[L-1+p+r]=0.

In order to solve the equations presented before, we are going to use the *orthogonality principle*, which is stated below:

"The linear predictor $\hat{x}[n]$ of x[n], that minimises the mean square error, should always fulfill the following equation:

$$E\{(x[n]-\hat{x}[n])x[n-k]\} = 0 \quad \text{,for k=1,….p "} \tag{4.9}$$

Manipulating the above equation, we can get that:

$$E\{x[n]x[n-k]\} - E\{\hat{x}[n]x[n-k]\} = 0 \Rightarrow E\{x[n]x[n-k]\} = E\{\hat{x}[n]x[n-k]\} \Rightarrow$$

$$R_{xx}(k) = \sum_{i=1}^{p} a_k E\{x[n-i]x[n-k]\} \Rightarrow$$

$$R_{xx}(k) = \sum_{i=1}^{p} a_k R_{xx}(|i-k|) \text{ , for } k,i=1,\dots\dots,p \tag{4.10}$$

If we apply all possible values for k,i on this equation , we can form the following matrices:

$$\begin{bmatrix} R_{xx}(1) \\ R_{xx}(2) \\ \dots \\ R_{xx}(p) \end{bmatrix} = \begin{bmatrix} R_{xx}(0) & R_{xx}(1) & \dots & R_{xx}(p-1) \\ R_{xx}(1) & R_{xx}(0) & \dots & R_{xx}(p-2) \\ \dots & \dots & \dots & \dots \\ R_{xx}(p-1) & R_{xx}(p-2) & \dots & R_{xx}(0) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \dots \\ a_p \end{bmatrix} \Rightarrow$$

$$\Rightarrow r = Ra \Rightarrow a = R^{-1}r \tag{4.11}$$

,where **R** is a pxp Toeplitz matrix made up of values of the autocorrelation sequence for x[n], **a** is a px1 vector of prediction coefficients, and **r** is a px1 vector of autocorrelation values.

It is obvious that it would be necessary to invert the pxp **R** matrix, in order to calculate the coefficients a_k. Inverting a pxp matrix is a very difficult procedure (especially for large values of p), and the traditional method of inverting a matrix can not be applied efficiently in our case. However, **R** is a *Toeplitz* matrix and there are several recursive methods that can facilitate the inversion, such as the *Levinson-Durbin* method.

In brief, the *Levinson-Durbin* method can be described by the following steps:

1. $E^{(0)} = R(0)$ (4.12)
2. $k_i = \dfrac{\left(R(i) - \sum_{j=1}^{i-1} a_j^{i-1} R(i-j) \right)}{E^{(i-1)}}$, for $1 \le i \le p$ (4.13)
3. $a_i^{(i)} = k_i$ (4.14)
4. $a_j^{(i)} = a_j^{(i-1)} - k_i a_{i-j}^{(i-1)}$, for $1 \le j \le i\text{-}1$ (4.15)
5. $E^{(i)} = (1 - k_i^2) E^{(i-1)}$ (4.16)

(4.17)

The steps 2 to 5 are repeated recursively for i = 1,2,…..,p and the final solution is given as:

$$a_j = a_j^{(p)} \quad , \text{for } 1 \le j \le p$$

Levinson-Durbin method is a very efficient way of solving simultaneous equations. The resulting H(z) will always be stable, i.e. all of its pole will be inside the unit circle.

2. *Covariance method:*

Only those equations for which all values of x[n] needed on both sides are present in the data. This method uses fewer equations, only L-p-r, but does not predict past the end of data.

Using the notation $R_n(|i-k|) = \phi_n(i,k)$, we can turn the equation [] into the following:

$$\begin{bmatrix} \phi_n(1,0) \\ \phi_n(2,0) \\ \dots \\ \phi_n(p,0) \end{bmatrix} = \begin{bmatrix} \phi_n(1,1) & \phi_n(1,2) & \dots & \phi_n(1,p) \\ \phi_n(2,1) & \phi_n(2,2) & \dots & \phi_n(2,p) \\ \dots & \dots & \dots & \dots \\ \phi_n(p,1) & \phi_n(p,2) & \dots & \phi_n(p,p) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \dots \\ a_p \end{bmatrix} \tag{4.18}$$

The resulting matrix is symmetric, as $\phi_n(i,j) = \phi_n(j,i)$, but it is not Toeplitz and can be only solved by Cholesky decomposition method. However, the transfer function found from these coefficients may not be stable. Therefore, the covariance method is not used in common speech processing applications.

The LPC coefficients generally feature great performance. However, there are several points somebody should be aware when using the LPC coefficients [20].

- The frequency response of the all-pass filter is very sensitive to small changes in a_k (such as quantising errors in coding)
- There is no easy way to verify that the filter is stable
- Interpolating between the parameters that correspond to two different filters will not vary the frequency response smoothly from one to the other: stability is not even guaranteed.

Although the linear prediction coefficients a_k are often thought to be the fundamental set used for linear predictive analysis, sometimes we need to transform this set of coefficients to a number of parameter sets, so as to get alternative representations of speech [3]. These alternative representations often are more convenient for certain applications of linear predictive analysis. We will examine two other basic parameter sets that can be directly derived from LP coefficients.

4.6 Reflection Coefficients

The Reflection (PARCOR) coefficients can act as a replacement to the well-known LP parameters [3,20].

We can calculate the reflection coefficients r_i directly from the set of LP coefficients a_i using the following algorithm:

1. Set $a_j^{(p)} = a_j$, for $1 \le j \le p$ (4.19)
2. $k_i = a_i^{(i)}$ (4.20)
3. $a_j^{(i-1)} = \dfrac{a_j^{(i)} + a_i^{(i)} a_{i-j}^{(i)}}{1 - k_i^2}$ (4.21)
4. Repeat step 3 for $1 \le j \le i\text{-}1$ (4.22)
5. Repeat steps 2,3,4 for i=p to 1 with negative step (4.23)
6. $r_i = -k_i$ (4.24)

The reflection coefficients are bounded.

$$0 \le |r_i| \le 1 \tag{4.25}$$

This is an extremely useful result for storage and compression applications involving LP models [29].

4.7 Cepstrum Coefficients

A transform that is commonly used in speech processing is called the *cepstrum* [3,20]. In fact, the *cepstrum* is the inverse fourier transform of the log spectrum and is calculated by the following equation:

$$c[n] = \frac{1}{2\pi}\int_{-\pi}^{+\pi} \log(X(e^{j\omega}))e^{j\omega n}d\omega \qquad (4.26)$$

Another alternative set are the *cepstrum* coefficients, which in fact are the cepstrum of the impulse response h[n] of the overall LP system. We can calculate the cepstrum coefficients c_j directly from the set of LP coefficients a_i using the following formula:

$$c_j = a_j + \frac{1}{j}\sum_{k=1}^{j-1}(j-k)c_{j-k}a_k \quad \text{,for } 1 \le j \le p \qquad (4.27)$$

The use of a non-linear transform that follows the superposition property, such as the logarithm, was introduced as it offered a methodology for separating the excitation signal from the vocal tract shape [29].

4.8 Mel-Frequency Cepstral Coefficients (MFCC)

Many psychophysical studies have shown that human perception of the frequency content of sounds, either for pure tones or for speech signals, does not follow a linear scale [2]. This conclusion has led to the idea of introducing subjective pitch of pure tones, therefore introducing a new frequency scale, known as the *mel scale*. In other words, for each tone with an actual frequency f (in Hz), a subjective pitch is measured on the mel scale (in mels). As a reference point, the pitch of a 1 kHz tone, 40 dB above the perceptual threshold is defined as 1000 mels.

Figure 4.3 shows the relation between subjective pitch and frequency. The upper curve shows the relationship of the subjective pitch to frequency on a linear scale; it shows that subjective pitch in mels increases less and less rapidly as the stimulus frequency is increased linearly. The lower curve, on the other hand, shows that the subjective pitch is essentially linear with the logarithmic frequency. It shows that the subjective pitch is essentially linear with the logarithmic frequency beyond about 1000 Hz.

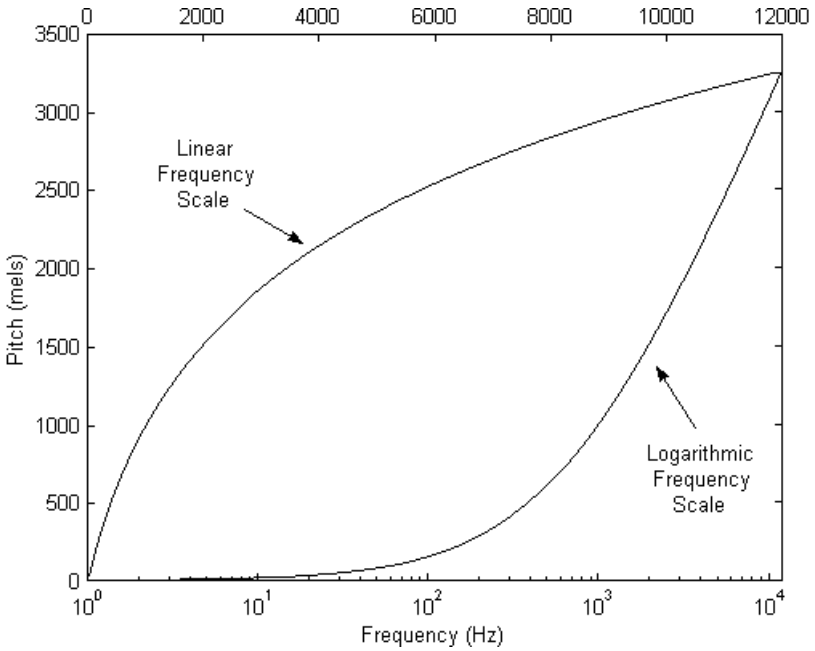


Figure 4.3: The Mel-scale

There have been many approximations of the mapping function between the perceived frequency scale F_{mel} and the real frequency scale f_{Hz}. The most commonly used approximation is the one described by the following equation:

$$F_{mel} = 2595 \log\left(1 + \frac{f_{Hz}}{700}\right) \tag{4.28}$$

Moreover, it has been found that the perception of a particular frequency by the auditory system is influenced by the energy in a critical band around that frequency. In this idea the mel-scale spectrum is simulated using a filter bank spaced uniformly on a mel scale, where the output energy from each filter band approximates the modified spectrum.

The Mel-Frequency Cepstral Coefficients are produced by extracting information from the mel-scale spectrum of the speaker's speech samples. In order to calculate MFCC, first of all, we have to design a *filter bank* that will extract information from the mel-scale spectrum and then perform a Discrete Cosine Transform to the log-outputs of the filter bank. We will look into the whole procedure in detail.

A *filter bank* [1,2,3] consists of a set of band-pass filters, used to extract information from certain areas of the input signal's spectrum. In our case, the filter bank consists of a set of triangular band-pass filters with their centre frequency uniformly distributed in the mel-scale. Triangular filters are used in this design instead of rectangular filters so as to avoid large changes in spectrum with small changes in frequency [21]. The following figure shows a *mel-scale filter bank* consisting of m_p filters.

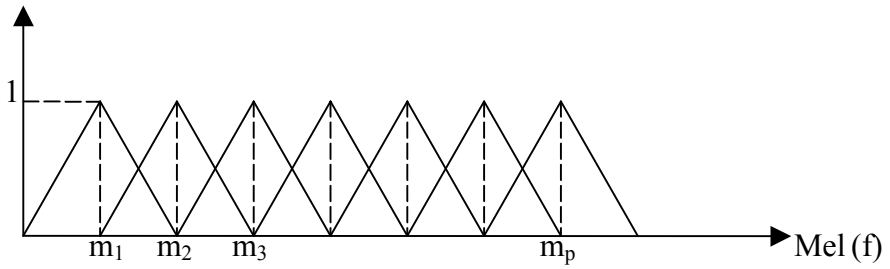


Figure 4.4: Equally distributed Triangular Filter Bank on mel space

The next figure shows the triangular mel-scale filter bank represented in the real frequency domain.

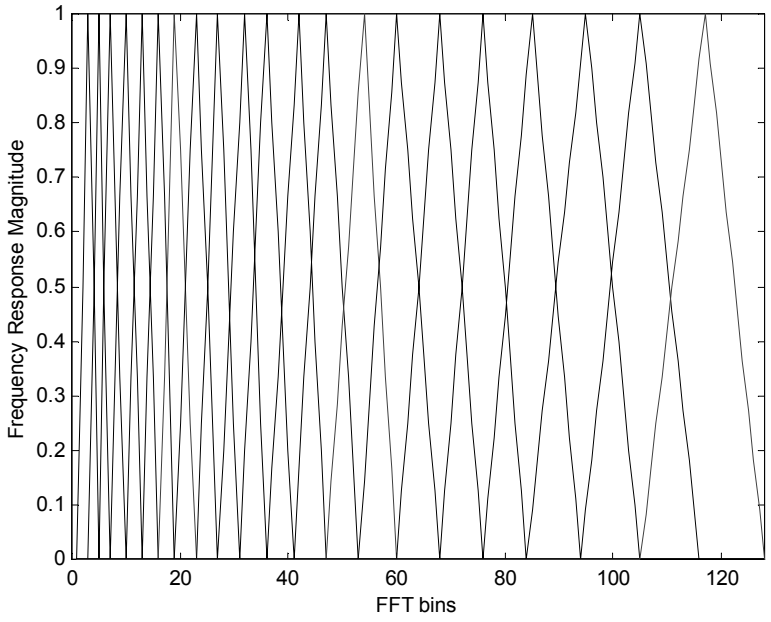


Figure 4.5: The previous filter bank represented on real axis.

In practice, we design the mel-scale filter bank and then using the mapping function between the mel and the real frequency, we transform it into a real frequency filter bank.

In order to calculate the MFCCs of a speech segment, we take the FFT of the segment and then find the power spectrum by squaring the absolute value of the FFT, so as to perform the filtering in the power domain. Then, we pass the power spectrum through each filter of the filter bank, calculating the output power of the filters in the filter bank. Hence, we get a set of m_p parameters S_k (m_p is the number of filter-bank filters). Normally, we take the log of S_k.

The final step is to apply a sort of discrete cosine transform to $\log(S_k)$, using the following formula that will give us the MFCC.

$$c_n = \sqrt{\frac{2}{m_p}} \left\{ \sum_{k=1}^{m_p} (\log(S_k) \cos\left[n\left(k - \frac{1}{2} \right) \frac{\pi}{m_p} \right] \right\}, \text{ for } n = 1,\ldots..,\text{Ncep} \qquad (4.29)$$

, where m_p is the number of filter-bank filters and Ncep is the number of desired MFCC.

This transform is identical to the common DCT, but we don't calculate the DC value of the DCT, so as to make MFCC independent of signal level [20]. The introduction of the DCT reduces correlation between coefficients and compressed information into low-order coefficients (well-known property of the DCT).

The application of MFC coefficients in modern speaker verification application is more than widespread. The MFCC are maybe one of the most important speech feature sets.

4.9 Delta and Delta-Delta MFCC Coefficients

A very important set of speech feature parameters are the *delta-MFC coefficients (ΔMFCC)* and the *delta-delta-MFC coefficients (Δ^2MFCC)* and can act as replacement to the normal MFCC set.

In order to calculate the ΔMFCC and the Δ^2MFCC, we have to calculate the MFCC of the corresponding speech segment. The ΔMFCC set is the product of the derivative of the MFCC set and the Δ^2MFCC set is the product of the derivative of the ΔMFCC set.

A very important issue is the calculation of the derivative of a discrete set of points x[n]. One could use the definition of the derivative as an approximation. In other words, we can say:

$$\frac{d}{dn}(x[n]) \approx x[n] - x[n-1] \tag{4.30}$$

However, this approach is not very accurate and introduces much noise to the derivative. An alternative approach is by using *polynomial fitting* [20]. According to this technique, we are trying to fit a polynomial to a given set of points, or in other words to calculate the coefficients of a polynomial that can pass from the given set of points. In our case, we will try to fit a line around the T closest points from either side to the point we are interested in calculating the derivative (2T+1 points in total)

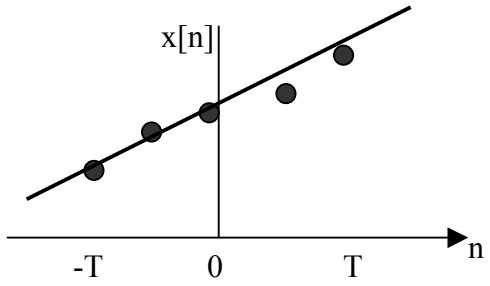


Figure 4.6: Calculating the derivative of a discrete set of points

This line can be a very good approximation of the tangent on that point, therefore the slope of that line represent the derivative at that point. According to a mathematical analysis [20], we can say that the derivative is given by the following formula:

$$x'[n] = \frac{\sum_{k=-T}^{T} kx[n-k]}{\sum_{k=-T}^{T} k^2} \tag{4.31}$$

The above formula can be calculated much more easily (in MATLAB) by calculating the convolution between the input x[n] and the discrete signal with values –T, -(T-1),…..,0,….,T-1,T . Of course, we have to discard the first and the last T points of the result and then scale the result with the denominator of the equation above.

4.10 Perceptual Linear Predictive Coefficients

The *perceptual linear predictive* analysis is a relatively new technique for analysis of speech [15]. In comparison with conventional linear predictive analysis (LP), PLP analysis is more consistent with human hearing. This technique uses three concepts from the psychophysics of hearing to derive the estimate of the auditory spectrum: (i) the critical-band spectral selectivity, (ii) the equal-loudness curve and (iii) the intensity-loudness power law. It also resembles the mel-frequency cepstrum analysis, as we will see further on.

The PLP analysis calculates the coefficients of an all-pole model, as in the LP analysis, according to the following steps:

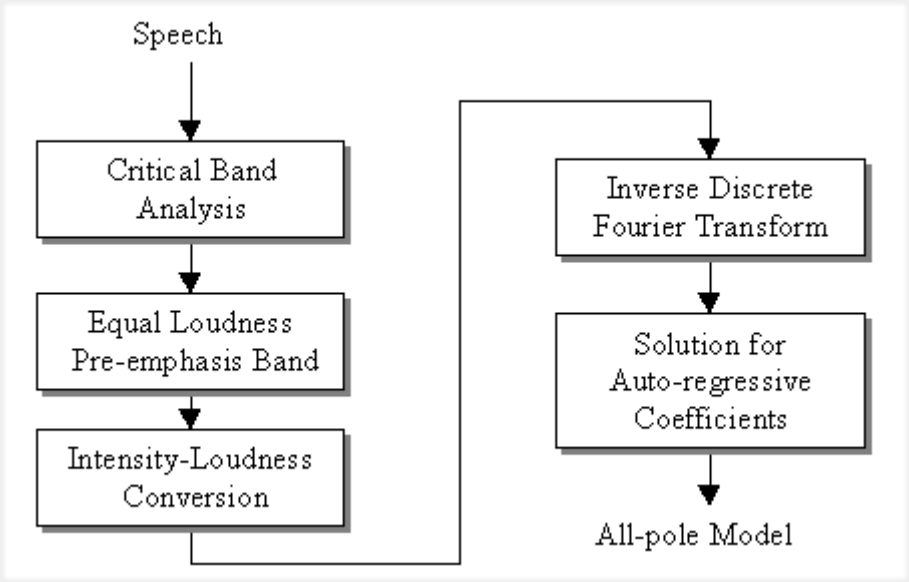


Figure 4.7: Perceptual Linear Predictive analysis

The first two steps of the above block diagram result in the design of a filter bank of band pass filters, quite in the same nature as in the MFCC case. The bandpass filters designed in this case are not triangular, but a short of rectangular windows with smooth edges and they are warped around another scale that models human auditory perception, named the *Bark* scale. We can define a mapping of acoustic frequency, to the "perceptual" frequency scale (*Bark scale*), as follows [29]:

$$F_{Bark} = 13\tan^{-1}\left(\frac{0.76f}{1000}\right) + 3.5\tan^{-1}\left(\frac{f^2}{7500^2}\right) \tag{4.32}$$

The equation that describes the shape of each bandpass filter is given by the following equation:

$$\Psi(\Omega) = \begin{cases} 0 & , for \ \Omega < -1.3 \\ 10^{2.5(\Omega+0.5)} & , for \ -1.3 \le \Omega \le -0.5 \\ 1 & , for \ -0.5 \le \Omega \le 0.5 \\ 10^{-1(\Omega-0.5)} & , for \ 0.5 \le \Omega \le 2.5 \\ 0 & , for \ \Omega > 2.5 \end{cases} \tag{4.33}$$

A filter bank consisting of 16 band pass filters is shown in the next figure:

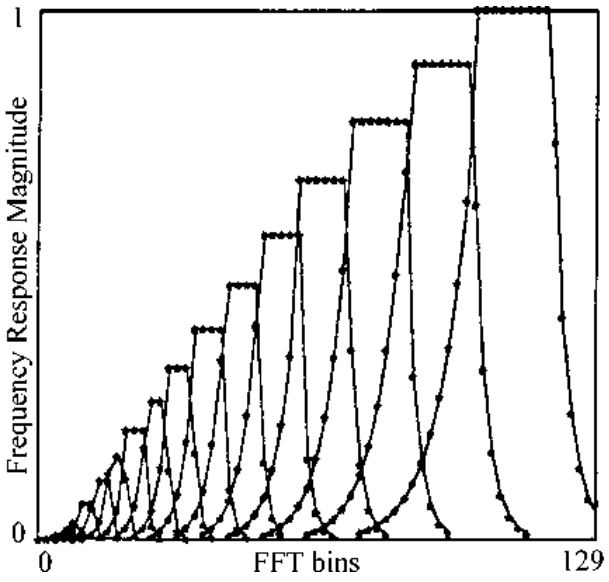


Figure 4.8: Filter Bank used for PLP Calculation.

In order to calculate the PLP coefficients of a speech segment, we take FFT of the segment and then find the power spectrum by squaring the absolute value of the FFT, so as to perform the filtering in the power domain. Then, we pass the power spectrum through each filter of the filter bank (like the one shown above), calculating the output power of the filters in the filter bank. Hence, we get a set of m_p parameters S_k (m_p is the number of filter-bank filters).

The last operation prior to the all-pole modeling is the cubic-root amplitude compression, i.e. the *intensity-loudness power-law*:

$$F_k = S_k^{0.33} \tag{4.34}$$

This operation is an approximation to the power law of hearing and simulates the non-linear relation between the intensity of sound and its perceived loudness.

The next step is to take real part of the inverse DFT of the F_k parameters and then use the *Levinson-Durbin* algorithm in order to estimate the parameters of the all-pole model. These parameters constitute the Perceptual Linear Prediction Coefficients.

The Mel scale cepstral analysis is very similar to perceptual linear predictive analysis of speech, because they are both based on psychologically based spectral transformation. However, in the MFCC the spectrum is warped according to the Mel scale, whereas in PLP the spectrum is warped according to the Bark scale. The main difference between Mel scale cepstral analysis and perceptual linear prediction is related to the output cepstral coefficients. The PLP model uses an all-pole model to smooth the modified power spectrum. The output cepstral coefficients are then computed, based on this model. In contrast, Mel scale cepstral analysis uses cepstral smoothing to smooth the modified power spectrum. This is done by using a DCT on the log power spectrum.

4.11 Rasta – PLP Coefficients

The PLP speech analysis technique is based on the short-term spectrum of speech, subsequently modified by several psychophysically based spectral transformations. However, the PLP technique (just like most other short-term spectrum based techniques) is vulnerable when these short-term spectral values are modified by the frequency response of the communication channel. The *RelAtive SpecTrAl (RASTA)* methodology makes PLP (and possibly also some other short-term spectrum based techniques) less sensitive to linear spectral distortions [16].

CHAPTER 5

GAUSSIAN MIXTURE MODELS (GMM)

5.1 Introduction

The next step of a speaker verification system is to arrange the features data obtained from the training speech samples into a reference template, so that any time the user requires authorisation, we can refer to it and verify his identity.

In this chapter, we are going to examine the use of a mixture of gaussian distributions for modelling the speaker's phonetic classes. We will look into the mathematical definition, the theoretical and practical interpretations of GMM, as well as some extensions to the common GMM. Moreover, we will make an overview of other common reference template building techniques used in speaker verification.

5.2 Definition

In this section, we are going to present and interpret the Gaussian Mixture Modelling technique for speakers. More specifically, we are going to look into the mathematical definition of GMM, as well as some considerations concerning their practical applications.

A Gaussian Mixture Model consists of M components [17]. The conditional probability density function of the vector $\underline{X} = \underline{x}$, given the parameters of the model is given by the following equation:

$$P(\underline{x} \mid \lambda) = \sum_{i=1}^{M} p_i b_i(\underline{x}) \tag{5.1}$$

,where p_i are the weights for each component and $b_i(\underline{x})$ is the gaussian probability function of a random vector $\underline{x}$ of size D, with mean $\underline{\mu}_i$ and covariance matrix C_i.

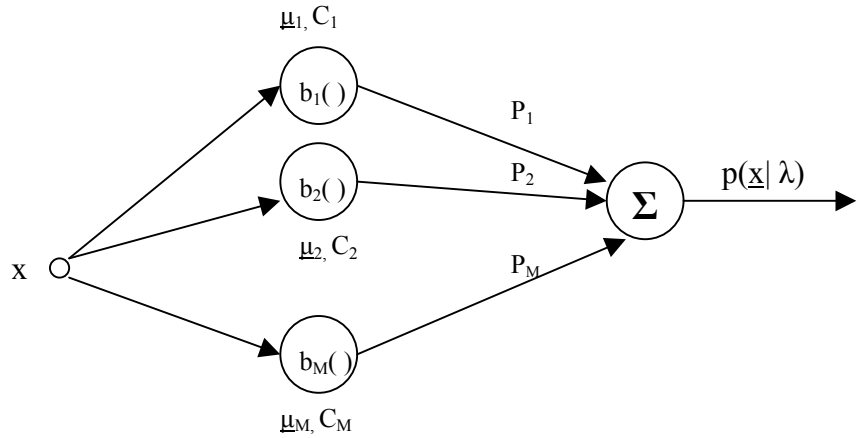


Figure 5.1: Graphical Representation of a Gaussian Mixture Model

Of course,

$$\sum_{i=1}^{M} p_i = 1 \text{ and} \tag{5.3}$$

$$b_i(\underline{x}) = \frac{1}{(2\pi)^{D/2} \mid C_i \mid^{1/2}} \exp\left\{-0.5(\underline{x} - \underline{\mu}_i)^T C_i^{-1} (\underline{x} - \underline{\mu}_i)\right\} \tag{5.4}$$

In other words, a GMM can be represented only by the mean vectors, covariance matrices and mixture weights from all component densities. Therefore, we can describe a GMM using the following notation:

$$\lambda = \{p_i, \underline{\mu}_i, C_i\} \qquad \text{for i = 1,….,M} \tag{5.5}$$

In speaker verification, each speaker is represented by a separate GMM, described by his/her model λ. Figure 5.1 graphically depicts the principle of a GMM, consisting of M components.

The calculation of a covariance matrix requires a considerable great processing time. As we can easily see the calculation of a great number of covariance matrices can considerably increase the processing time of a practical speaker verification application. There-

fore, many approaches to the covariance matrix calculation have been proposed. In order to reduce the number of calculations, a GMM can have several different forms depending on the choice of covariance matrices:

1. *Nodal covariance*: each component of a GMM has its own covariance matrix.
2. *Grand covariance*: the components of each GMM have the same covariance matrix
3. *Global covariance*: a single covariance matrix is used for all models.

Moreover, covariance matrices can be full or be regarded as diagonal. In other words, we can assume that the elements of the random vector x (features) are uncorrelated, which makes the covariance matrix diagonal. Unfortunately, this estimate can undermine the performance of our system for certain feature sets, but it can reduce the total processing time of our system. For example, the cepstral vectors can be modelled by GMMs with diagonal matrices without much deviation from reality [23].

However, we can decorrelate the input feature vectors, using a linear transform, and therefore diagonalise non-diagonal covariance matrices. This procedure will be presented further on.

Practically, you can use any combination of the above modes. An approximation, which is very close to reality, is the use of nodal, diagonal covariance matrices. In other words, each component of a GMM model has its own diagonal covariance matrix,

5.3 Gaussian Mixture Model Interpretations

Basically, there are two motivations behind the use of the Gaussian Mixture Density modelling for speaker verification [17]. First of all, the individual-component Gaussians in a speaker-dependent GMM are interpreted to represent some broad acoustic classes. It is reasonable to assume the acoustic space corresponding to a speaker's voice can be characterised by a set of acoustic classes representing some broad phonetic events, such as vowels, nasals, or fricatives. These acoustic classes reflect some general speaker-dependent vocal tract configurations that are very useful for modelling speaker identity.

One can also look at gaussian mixtures as describing alternative pronunciations of a particular speech sound [20]. Moreover, the spectral shape of the ith acoustic class [17], for example, can in turn be represented by the mean vector μ_i, and variation of the average spectral shape can be represented by the covariance matrix C_i.

The second motivation for using GMM for speaker verification is the empirical observation that a linear combination of Gaussian basis function is capable of representing a large class of sample distributions. One of the powerful attributes of the GMM is its ability to form smooth approximations to arbitrarily - shaped densities. Using a discrete set of Gaussian functions, each with their own mean and covariance matrix, GMM allows a more versatile modeling capability.

Moreover, because the component Gaussians are acting together to model the overall pdf, full covariance matrices are not necessary even if the features are not statistically independent. The linear combination of diagonal covariance Gaussians is capable of modeling the correlation between feature vector elements. The use of a set of M full covariance Gaussians can be compensated by using a larger set of diagonal covariance Gaussians.

5.4 Gaussian Mixture Model Training

Using training speech from a speaker, the goal of speaker model training procedure is to estimate the parameters λ of the GMM that matches the distribution of the training feature vectors.

There are several techniques for estimating the parameters of a GMM [17]. The most popular, well-established and robust method is the *Maximum Likelihood (ML)* estimation. This training procedure is described directly

5.4.1 Expectation-Maximisation algorithm (EM)

Suppose we have a set of T training vectors $X=\{x_1,x_2,x_3\ldots,x_T)$. We would like to estimate the λ of our model, so that

$$P(X \mid \lambda) = \prod_{t=1}^{T} P(X_t \mid \lambda) \quad \text{is maximum} \tag{5.6}$$

Unfortunately, this is a non-linear problem, therefore λ can not be directly calculated. Therefore, we will use an iterative algorithm, called *Expectation-Maximisation (EM)* algorithm, so as to estimate ML parameters. The EM algorithm consists of the following steps:

EM algorithm

1. Choose an initial model λ
2. Find a new model λ' so that P(X|λ') > P(X|λ).
3. Repeat 2 until the difference P(X|λ')-P(X|λ) has reached a convergence threshold, or you have reached the maximum number of iterations.

The EM algorithm resembles the *Baum-Welch reestimation* algorithm, used for estimating HMM parameters [17,21].

On each EM iteration, there are certain reestimation formulas that can guarantee a monotonic increase in the model's likelihood value. Using the GMM notation, we have

Mixture Weights:

$$p_i' = \frac{1}{T} \sum_{t=1}^{T} p(i \mid \vec{x}_t, \lambda) \tag{5.7}$$

Means:

$$\vec{\mu}_i' = \frac{\sum_{t=1}^{T} p(i \mid \vec{x}_t, \lambda) \vec{x}_t}{\sum_{t=1}^{T} p(i \mid \vec{x}_t, \lambda)} \quad \text{,for i=1,…..,M} \tag{5.8}$$

Variances:

$$\sigma_i'^2 = \frac{\sum_{t=1}^{T} p(i \mid \vec{x}_t, \lambda) x_t^2}{\sum_{t=1}^{T} p(i \mid \vec{x}_t, \lambda)} - \mu_i'^2 \tag{5.9}$$

, where μ_i, x_t, σ_t are arbitrary elements of the corresponding vectors.

The a posteriori probability for acoustic class i is given by the following formula:

$$p(i \mid \vec{x}_t, \lambda) = \frac{p_i b_i(\vec{x}_t)}{\sum_{k=1}^{M} p_k b_k(\vec{x}_t)} \tag{5.10}$$

As it's well obvious, each Gaussian component has its own covariance matrix (nodal) and we assume that it's a diagonal matrix.

A very useful hint in the EM algorithm is the use of log-probabilities instead of the real values, while calculating the P(X|λ). The logarithm is a monotonous rising function and therefore we can write :

$$P(X \mid \lambda) = \prod_{t=1}^{T} P(x_t \mid \lambda) \xrightarrow{\log()} \log\big(P(X \mid \lambda)\big) = \log\left(\prod_{t=1}^{T} P(x_t \mid \lambda)\right) = \sum_{t=1}^{T} \log\big(P(x_t \mid \lambda)\big) \tag{5.11}$$

The transformation above helps us to calculate the convergence threshold much easier and gives us a more illustrative form for the P(X|λ). If we look at the definition of P(X|λ), we will see that it is a number with at least T decimal points, as it is the product of T probabilities, where T is the number of the training vectors. Using a logarithm, we can process P(X|λ) much easier.

5.4.2 Practical Issues of the Training Algorithm

- *Choice of M*

Unfortunately, there is no good theoretical guide for selecting the proper number of gaussian components for our speaker model. For speaker modelling the objective is to choose the minimum number of components necessary to adequately model a speaker for good speaker identification. Choosing too few mixture components can produce a

speaker model, which does not accurately model the distinguishing characteristics of a speaker's distribution. Choosing too many components can reduce performance, when there are a large number of model parameters relative to the available training data and can also result in excessive computational complexity and processing time both in training and classification.

- *Algorithm Initialisation*

A very important issue is the initialisation of the EM algorithm with some starting λ values. The EM algorithm is guaranteed to find a local maximum likelihood model regardless of the starting point. However, the likelihood equation for a GMM has several local maxima and different starting models can lead to different local maxima [17]. There have been several algorithms for initialisation of the EM algorithm. We are going to describe two basic methods.

The *first method* doesn't perform any special initialisation. It randomly selects M feature vectors from the speaker's training data (after silence removal) as starting values for the initial model means and uses identity matrices for the initial covariance matrices. This initialisation method is very simple and even though it is not very much sophisticated, produces satisfactory results in terms of speaker verification performance.

The *second method* employs a *K-means* algorithm in order to perform a clustering of the training data (speech feature vectors) into M acoustic classes [20]. After that, we calculate the M initial estimates for the mixture means and covariance matrices. The K-means algorithm applied in this case follows:

K-Means algorithm

1. Set the mean vectors μ_i to M randomly chosen training frames
2. Allocate each training frame to whichever μ_i it is nearest to.
3. Update each μ_i to the mean of all the frames that were allocated to it.
4. If no frames were allocated to μ_i, set it to a randomly chosen point from one of the other distributions.

5. Repeat steps 2,3,4, until convergence occurs
6. Set C_i to the covariance of the frames allocated to μ_i.

Both initialisation methods finally converge to a local maximum, which indeed is different. Both methods feature very good performance.

- *Variance Limiting*

While training a *nodal variance* GMM, it has been observed that variance elements can become very small in magnitude and therefore can degrade the performance of our system. This is particularly true for a mixture model with a large number of component densities. These small variances produce a singularity in the model's likelihood function and can degrade identification performance by distorting speaker model scores used in the maximum likelihood classifier. These singularities are likely to appear when there is not enough data to train a component's variance vector sufficiently or when using noise-corrupted data [17].

In order to avoid these effects, we can apply a variance limiting constraint. In other words, we set a minimum variance value on elements of all variance vectors in a speaker's model. More specifically,

$$\sigma_i^2 = \begin{cases} \sigma_i^2 & ,\text{if} \quad \sigma_i^2 \geq \sigma_{min}^2 \\ \sigma_{min}^2 & ,\text{if} \quad \sigma_i^2 < \sigma_{min}^2 \end{cases} \tag{5.12}$$

This constraint is applied to the variance estimates after each EM iteration, making the EM algorithm more robust. Again, attention must be paid in the selection of the variance threshold value. Setting the threshold very low may not eliminate the singularities in the covariance matrices. On the other hand, setting the threshold very high, the component variances are continuously masked to the same value and therefore totally distort the model, thus deteriorating the performance of the speaker verification system. The threshold should be empirically set according to the performance of the system.

5.5 Speaker Verification using Gaussian Mixture Models

Suppose we have a group of S speakers {1,2,…,S}, represented by $\lambda_1,\lambda_2,\ldots\ldots,\lambda_S$ GMMs and we have a set of speech data $X=\{x_1,x_2,x_3\ldots,x_N\}$ from the claimed user. The identity of this user should maximise the following probability.

$$\max_{1\le k\le S}\{P(\lambda_k \mid X)\} = \max_{1\le k\le S}\frac{P(X \mid \lambda_k)P(\lambda_k)}{P(X)} \tag{5.13}$$

Assuming equally likely speakers, implies that $P(\lambda_\kappa)=1/S$. Moreover, P(X) is the same for all speaker models, therefore P(X) = 1/S. The above equation can be modified as following:

$$P(\lambda_k \mid X) = \frac{P(X \mid \lambda_k)}{1/S}1/S = P(X \mid \lambda_k) \tag{5.14}$$

We can see that the probability $P(\lambda_\kappa \mid X)$ can now be easily calculated using the equations derived so far. Now, for the reasons mentioned earlier, we are going to use log probabilities. So, the expression that should be maximised by the claimed user's speech data is the following.

$$\max_{1\le k\le S}\{\log(P(\lambda_k \mid X))\} = \max_{1\le k\le S}\{\log(P(X \mid \lambda_k))\} = \max_{1\le k\le S}\left\{\sum_{t=1}^{N}\log(P(\vec{x}_t \mid \lambda_k))\right\} \tag{5.15}$$

This procedure can be employed in two different versions of speaker recognition applications. The first application is called *speaker identification*, where we are going to find the claimed user from its test data. In this case, the λ_κ that maximises the above expression denotes the identity of the user providing the test data. The other application is *speaker verification*, where we are going to verify that the identity of the claimed user. In other words, we have to verify that the λ_κ that maximises the above expression belongs to

the claimed user. In that case, we accept the identity of the user. Otherwise we reject authorisation. to the user.

5.6 Cohort normalisation on Speaker verification with GMM

A common technique used in speaker verification systems is *cohort normalisation* [19,21], aiming to enhance the performance of our system. According to this technique, we use a normalised score in order to decide whether to accept or reject the claimed identity. The log-probability log($P(X \mid \lambda_\kappa)$) of the claimed user is normalised to the log-probabilities of the cohort speakers. The term 'cohort speakers' refers to imposters impersonating the true speaker [21]. This creates a normalised score P_{norm} described by the following equation:

$$P_{norm} = \log(P(X \mid \lambda_i)) - \frac{1}{K}\sum_{k=1}^{K} \log(P(X \mid \lambda_k)) \qquad (5.16)$$

The verification is performed by comparing P_{norm} to a set threshold. If P_{norm} exceeds the threshold we accept or otherwise reject the user.

A very important issue here is determining the number K and the identity of the impostors. Logically, the impostors will identify themselves by producing great log-probabilities log($P(X \mid \lambda_\kappa)$). There are several proposals for the formation of the impostor group. We can choose the K models with the greatest log-probability (optimum selection) or even K randomly selected models as impostors. It's more than obvious that the performance of the optimum selected impostor group is much greater [19]. We can choose the K/2 models with the greatest log-probability and the K/2 models with the smallest log-probability, creating the *near and far background* set [24].

5.7 Using Orthogonal GMM (OGMM)

In the previous sections, we have described the operation and the theory behind Gaussian Mixture Models. The basis of the GMM approach is to represent the distribution of training vectors from each speaker with a weighted sum of several multivariate Gaussian

functions. Each Gaussian function may have a full covariance matrix, however, in practice, the diagonal covariance matrix has been continuously used, employing more Gaussian components for compensation. A consequence of using a large GMM is that its training procedure is time-consuming and its response speed is very slow.

Generally, the elements of feature vectors extracted from a speech signal are correlated. An alternative solution can be to use a linear transform so as to decorrelate the elements of speech feature vectors [18].

Suppose we have a random vector $\underline{x}$, with D elements. The mean and the covariance of the random vector is given by the following equations:

$$\underline{\mu}_x = E\{\underline{x}\} \tag{5.17}$$

$$C_x = E\{\underline{x}\,\underline{x}^T\} \tag{5.18}$$

The covariance matrix C_x is a DxD real symmetric matrix. If we want to diagonalise this matrix, or in other words decorrelate the elements of the random vector, we have to do the following linear transform to $\underline{x}$:

$$\underline{y} = \Omega\underline{x} \tag{5.19}$$

, where Ω is a DxD matrix whose rows are the eigenvectors $\underline{e}_i$ of C_x. As C_x is a real symmetric matrix, it is certain that C_x has D eigenvectors. Hence,

$$\Omega = \begin{bmatrix} \underline{e}_1^T \\ \underline{e}_2^T \\ \cdots \\ \underline{e}_D^T \end{bmatrix} \tag{5.20}$$

Every eigenvector $\underline{e}_i$ of the C_x matrix is connected to the corresponding eigenvalue with the following equation:

$$C_x\,\underline{e}_i = \lambda_i\,\underline{e}_i \tag{5.21}$$

Now, we are going to calculate the mean and the covariance of $\underline{y}$.

$$\underline{m}_y = E\{\underline{y}\} = E\{\Omega\underline{x}\} = \Omega E\{\underline{x}\} \Rightarrow \underline{m}_y = \Omega\underline{m}_x \tag{5.22}$$

$$C_y = E\{\underline{y}\,\underline{y}^T\} = E\{\Omega\underline{x}\,\underline{x}^T\Omega^T\} = \Omega E\{\underline{x}\,\underline{x}^T\}\Omega^T \Rightarrow C_y = \Omega C_x \Omega^T \tag{5.23}$$

We can easily prove that the new covariance matrix C_y is a diagonal matrix. Note that the eigenvectors are orthonormal vectors, implying that

$$\underline{e}_i^T \underline{e}_j = \begin{cases} 0 & , i \neq j \\ 1 & , i = j \end{cases} \tag{5.24}$$

We have that

$$C_y = \Omega C_x \Omega^T = \begin{bmatrix} \underline{e}_1^T \\ \underline{e}_2^T \\ \dots \\ \underline{e}_D^T \end{bmatrix} C_x \begin{bmatrix} \underline{e}_1^T \\ \underline{e}_2^T \\ \dots \\ \underline{e}_D^T \end{bmatrix}^T = \begin{bmatrix} \underline{e}_1^T \\ \underline{e}_2^T \\ \dots \\ \underline{e}_D^T \end{bmatrix} C_x \begin{bmatrix} \underline{e}_1 & \underline{e}_2 & .... & \underline{e}_D \end{bmatrix} =$$

$$= \begin{bmatrix} \underline{e}_1^T \\ \underline{e}_2^T \\ \dots \\ \underline{e}_D^T \end{bmatrix} \begin{bmatrix} \lambda_1 \underline{e}_1 & \lambda_2 \underline{e}_2 & .... & \lambda_D \underline{e}_D \end{bmatrix} \Rightarrow C_y = \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \dots & \dots & .. & 0 \\ 0 & 0 & \dots & \lambda_D \end{bmatrix} \tag{5.25}$$

We have shown that by using this linear transform the final covariance matrix in the y-plane is diagonal.

As it is quite evident here, the greatest problem concerning the practical application of this technique is the calculation of the covariance matrix. Suppose we have a population of M vectors, then the mean vector and the covariance matrix can be well approximated by the following summations:

$$\underline{\mu}_x = \frac{1}{M} \sum_{k=1}^{M} \underline{x}_k \tag{5.26}$$

$$\underline{C}_x = \frac{1}{M} \sum_{k=1}^{M} \underline{x}_k \underline{x}_k^T - \underline{m}_x \underline{m}_x^T \tag{5.27}$$

The procedure described above can be also found in the well-known *Karhunen-Loeve* or *Hotelling* transform, used in signal or image compression.

5.8 Other reference template building techniques

Several methods have been developed, which deal with the building of the user's model (reference template) from a group of input feature vectors. Most of these methods

were borrowed from *speech recognition*, as speaker verification is a very special speech recognition case.

One of the first approaches was *Vector Quantisation (VQ)*. In vector quantisation approach to speaker verification, we can use a parameter template, which is called the speaker's codebook [13]. The codebook is a series of vectors in Euclidean N-space (N is the order of the feature vectors) that represent the phonetic clusters in the user's speech. Template building in this case refers to training the codebook vectors. Feature vectors used to train the codebook and they are known as training vectors. An iterative technique is used to train the codebook, using about ten times as many training vectors as codebook vector. In the verification procedure the feature vectors extracted from the test utterance, are quantised to the closest codebook vector. The quantisation error can be used to verify the speaker's identity.

Artificial Neural Networks can also be used in speaker verification. Rather than train individual models to represent particular speakers, discriminative Neural Networks are trained to model decision function which best discriminates speakers within a known set. Several neural network topologies have recently been applied to speaker recognition techniques, such as multi-layer perceptrons. Generally, neural networks require a smaller number of parameters than independent speaker models [17]. The major drawback to many of the NN techniques is that the complete network must be retained when a new speaker is added to the system.

A very common approach is the use of *Hidden Markov Models (HMM)* [17,19,20,21]. According to this approach. There is an explicit or implicit segmentation of the speech into phonetic sound classes prior to speaker model training. This segmentation is mainly performed using the Viterbi alignment algorithm. Each cluster produced by this segmentation is represented by a state in the Markov chain. The Markov model employed here is a left-to-right model. As we can see in this model, the states proceed from left to right as time increases, and therefore is not an ergodic model. This Markov model is called *Hidden*, because the actual state sequence is not directly observable, but can be observed only through another set of stochastic processes that produce the sequence of observations.

The calculation of the transition probabilities is done via an iterative algorithm, called the *Baum-Welch Reestimation* technique, using the segmentation performed earlier by the Viterbi alignment algorithm.

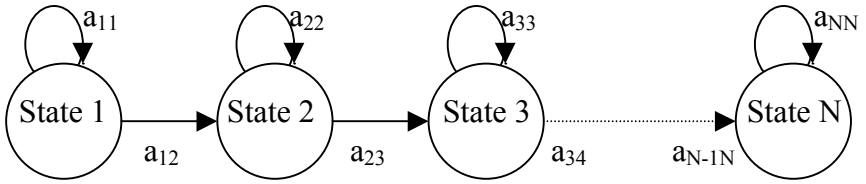


Figure 5.2: A left to right Hidden Markov Model

There has been considerable research effort on HMM so far, and the greatest part of the commercial speaker verification products are based on HMM recognisers. Therefore, there have several versions of the classic HMM recogniser, such as the integration of a HMM and a Multi-layer Perceptron (MLP) [22], in order to boost its performance.

Hidden Markov Models are based on the same concept as Gaussian Mixture Models: the classification of the input feature vectors into phonetic/acoustic classes. The difference between the two models is that in HMM, the state transitions are Markovian, whereas the state transition of GMM are not [23]. The close relationship between GMMs and HMMs will allow speaker recognition systems developed using GMMs to take advantage of the considerable research efforts currently being undertaken the scientific community.

CHAPTER 6

GRAPHICAL USER INTERFACE (GUI) DESIGN FOR A SPEAKER VERIFICATION SYSTEM

6.1 Basic Properties of a Graphical User Interface (GUI)

Formally, a *Graphical User Interface (GUI)* is a computer program that enables a person to communicate with a computer through the use of symbols, visual metaphors, and pointing devices. Some people consider the design of a GUI quite elementary in programming and therefore of less importance. However, it is very important to build a GUI that can facilitate the human-computer interaction in the application that runs beneath it.

There are three basic qualities of GUI design [4]:

- *Simplicity*

Simplicity in GUI design is the first goal. A GUI should feature a clean, elegant look and a sense of unity. A GUI that puzzle the users with many dialog boxes and buttons arranged in a clumsy way can not be of great value. The programmer should arrange the GUI layout so that the users can advance gradually and in simple steps towards the completion of the task.

- *Consistency*

The further users are from their base of experience, the more likely they are to feel disoriented. Anything you can do to keep the user from feeling confused is extraordinarily important.

- *Familiarity*

If the GUI that we create is in some sense familiar to its users, then they can generally learn how to use it more quickly. This is value of basing the GUI on a good metaphor.

They might not know how to do a given task, but the metaphor helps them make a good guess [4].

We can not suggest a certain methodology in GUI design, as it is not a step-by-step recipe. However, it's helpful to think about the GUI creation process as breaking into a design phase and an implementation phase. This concept is illustrated in the diagram below.

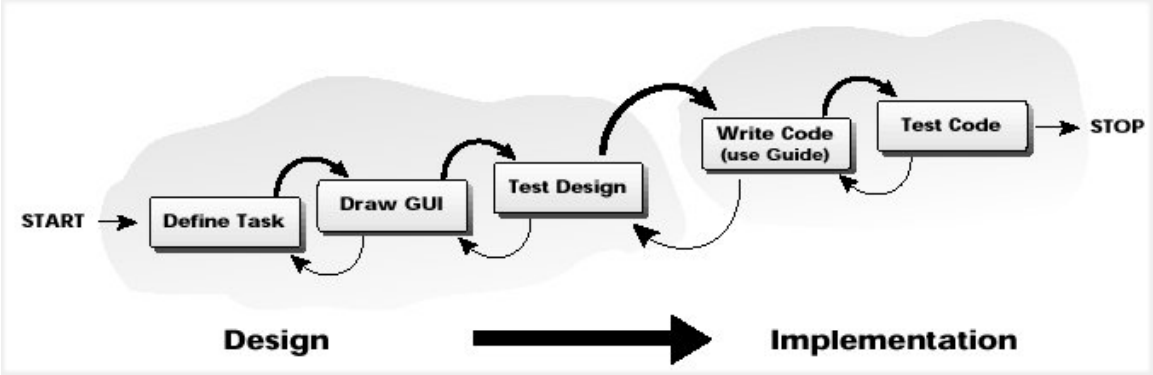


Figure 6.1: A model GUI design technique [4].

Of course the design may change once we start coding, and there may well be design decisions that we can't properly make until we have written some code. The above diagram simply proposes a logical order of steps in GUI design.

6.2 Building GUIs in MATLAB

GUI design in MATLAB follows the general principles, found in all object-oriented programming languages with visual interface (Visual C++, Visual J++, etc). In MATLAB, the GUI is implemented using the user interface (UI) controls [4,27]. One can also use this approach in design oriented problems that require comparison of several different techniques.

Building GUI applications has been considerably simplified in version 5. The addition of Guide, a set of MATLAB tools designed to make building GUIs easier and faster, has made GUI design almost as easy as program development using a Rapid Application Development RAD) package.

The five tools that together make up the Guide are

(i) the Property Editor,

(ii) the Guide Control Panel,

(iii) the Callback Editor,

(iv) the Alignment Editor, and

(v) the Menu Editor.

Each tool performs a distinct task and also is aware of and interacts with other tools.

The *Property Editor* will basically allow one to change different properties of a figure. Each item (button, slider etc) is represented as an object like in many other visual object-oriented languages. Each object has its own variables that represent certain properties of the object (such as the button's colour, its position etc). Using the property editor, we can have a look and even change the properties of every object in every figure we create. The objects are presented in hierarchical order according to the figure they belong to.

Axes and uicontrols (e.g., list box, checkbox, slider, etc) can be added by selecting the appropriate plate located on the button of the *Guide Control Pane*l. There is a graphical drag-and-drop interface where you can easily design the structure of your application window (called 'figure' in MATLAB). We can easily position our buttons, sliders, pop-up menus and plots on our figure window.

The *Alignment Tool* will allow one to align selected objects using a collection of vertical and horizontal alignment or distribution push buttons.

The *Menu Editor* will let the user customise the menus in the designed figures. We can add, delete, create new options to the figure we design, according to the demands of the figure.

On the other hand, the *Callback Editor* is a tool, where you can define the *callbacks* of each object (element) of the figure. With the term *callback*, MATLAB refers to the actions (commands) connected with each object, i.e. the commands that are executed when you press a button

In order to produce an interactive simulation program, we also need to write the necessary MATLAB programmes that will produce the actual simulation results and then de-

sign the interfaces. The interface portion usually consists of one (or more) plot area, which displays the output, and several control objects, such as buttons, sliders, etc. that are designed to change particular parameters.

6.3 Speaker Verification GUI

In the design of a speaker verification GUI, we have to take into account the structure of a speaker verification system, as discussed analytically in previous chapters. First of all, we have to decide the number of windows that are needed for our application and assign a specific task to it.

As we already know, a speaker verification system performs two basic tasks: *enrol a new user* to the system and *verify the identity of a user*. Moreover, the aim of this project is to design a modular speaker verification system, implying that some parameters of the system can be adjusted by the user, so as to encourage and help experimentation on the use of different feature sets combinations into various speaker modelling algorithms. Therefore, we can see that we obviously need three windows for these mentioned tasks: enrolment, verification and system configuration. Indeed, we can design a main control window that can co-ordinate the whole system and help users make their way through the other three functions of the system.

To sum up, our speaker verification GUI design consists of the following windows:

- the *Main* Speaker Verification window,
- the *Configuration* window,
- the *Enrollment* window, and
- the *Verification* window

Each of these windows performs a different specific task. In the next paragraphs, we will analyse the role of each window in detail.

6.3.1 The main Speaker Verification window

As we mentioned above, the main Speaker Verification window acts as a central menu to the other three functions of the GUI. Therefore, as we can see, the whole window is divided into three areas: enrolment, verification and configuration.

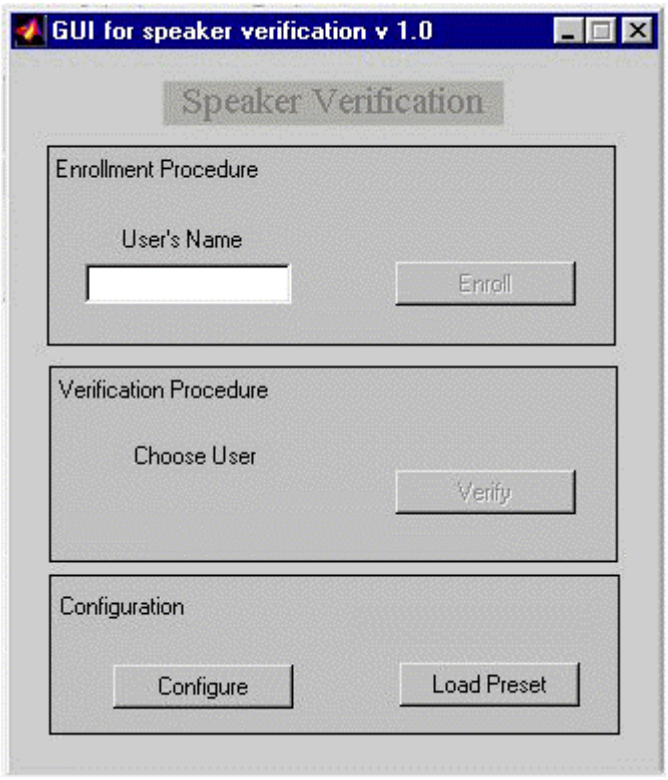


Figure 6.2: The main Speaker Verification window

We have said that GUI encourages the use of different configurations (feature sets, speaker modelling) for the speaker verification system. However, it is evident that the programme can not keep the same speaker database for different configurations. For each different configuration, the GUI builds a new database. The configuration details are stored in a text file with the extension .ftr, which is created by the configuration window. This file (preset) keeps the parameters of the feature sets that will be used to create the feature vector, as well as the method of speaker modelling.

We can see in the above figure that the enrolment and verification areas are inactive for the time being. This is because we haven't specified which configuration we are going

to use. In order to create a new configuration, we can press the 'Configure' button and jump to the configuration window and construct ours. The GUI obliges the user to save the configuration he is going to use as a preset (.ftr). It is also very important to save every preset (configuration) in a different directory, as the programme builds the database as well as other batch files in the directory of the preset file. This will be also beneficial for the user, so as to keep distinct records of several experiments.

If we have already saved our configuration, we can load it using the 'load preset' button. The name of the preset is now shown above the 'load preset' button and the other areas of the window are now active. If we had previously added some users to the system using this configuration, they would appear in the pop-down menu in the verification area. Otherwise this pop-down menu will be empty.

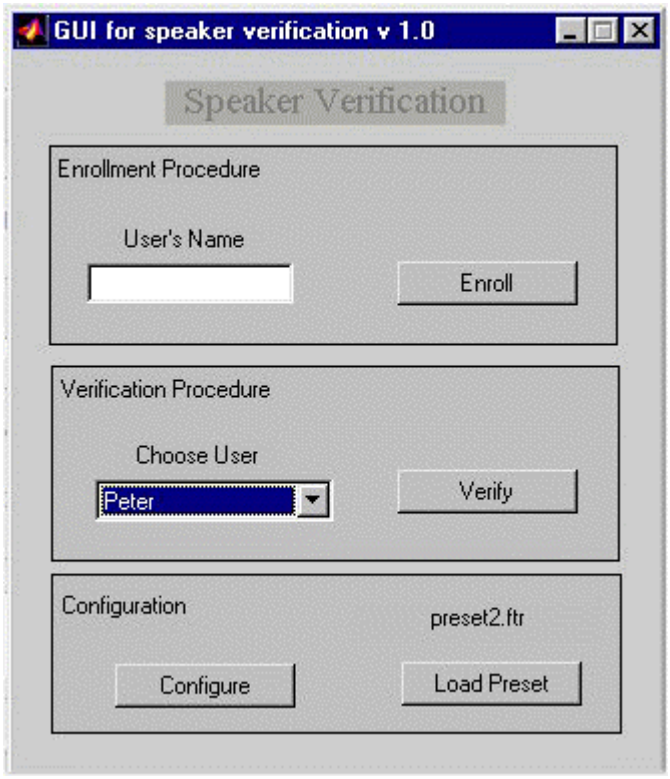


Figure 6.3: The main Speaker Verification window with loaded configuration

We can use the enrolment area to enrol new users to the system. We can type the name in the corresponding window and press the 'Enroll' button. The GUI also ensures that you provide a username before you proceed, otherwise you are prompted to do so. Then, it

checks whether that name exists in the database and in that case you are prompted to change the user name. If all the username details are checked, the main window calls the *Enrolment* window. After completing enrolment, the new username will appear in the pop-down menu in the verification area of the main window.

We can use the verification area to verify the identity of a user. We can simply select a username from the pop-down menu and press the 'Verify' button. The main window calls the verification window, passing the username of the user, whose identity is to be verified.

6.3.2 The Configuration Window

The Configuration Window creates the configuration file that specifies the parameters of the speaker verification system. Here, we can select the feature sets that will form the feature vectors that will be extracted from the speaker's input speech samples, as well as the algorithm that will be used to model each speaker.

In order to build the feature vector, we have to follow a set of steps. First of all, select from the corresponding pop-down menu, the type of feature set we want to add to our feature vector. We can select between LPC, reflection, complex cepstrum, mel-cepstrum, delta mel-cepstrum, delta-delta mel-cepstrum, PLP coefficients. Underneath the pop-down menu, we can enter the parameters for each feature set. First of all, we can specify the window length and the overlap ratio that will used. We have mentioned in a previous chapter, that for each feature set we can define separate segmentation of the input speech data (see also 4.2). We can also define the order of the feature set, or in other words the number of coefficients we want. In the case of mel-cepstrum coefficients, we can specify the number of filters used to create the filterbank (see also 4.8). After filling in all the parameters, we can press the 'Add' button and add the feature set to the feature vector. We can see all the feature sets we have added to the Feature Vector, in the form of MATLAB commands, in the 'Feature Vector' pull-down menu. In case we are not happy with a feature set we can remove it from the feature vector by selecting it and pressing the 'Remove' button.

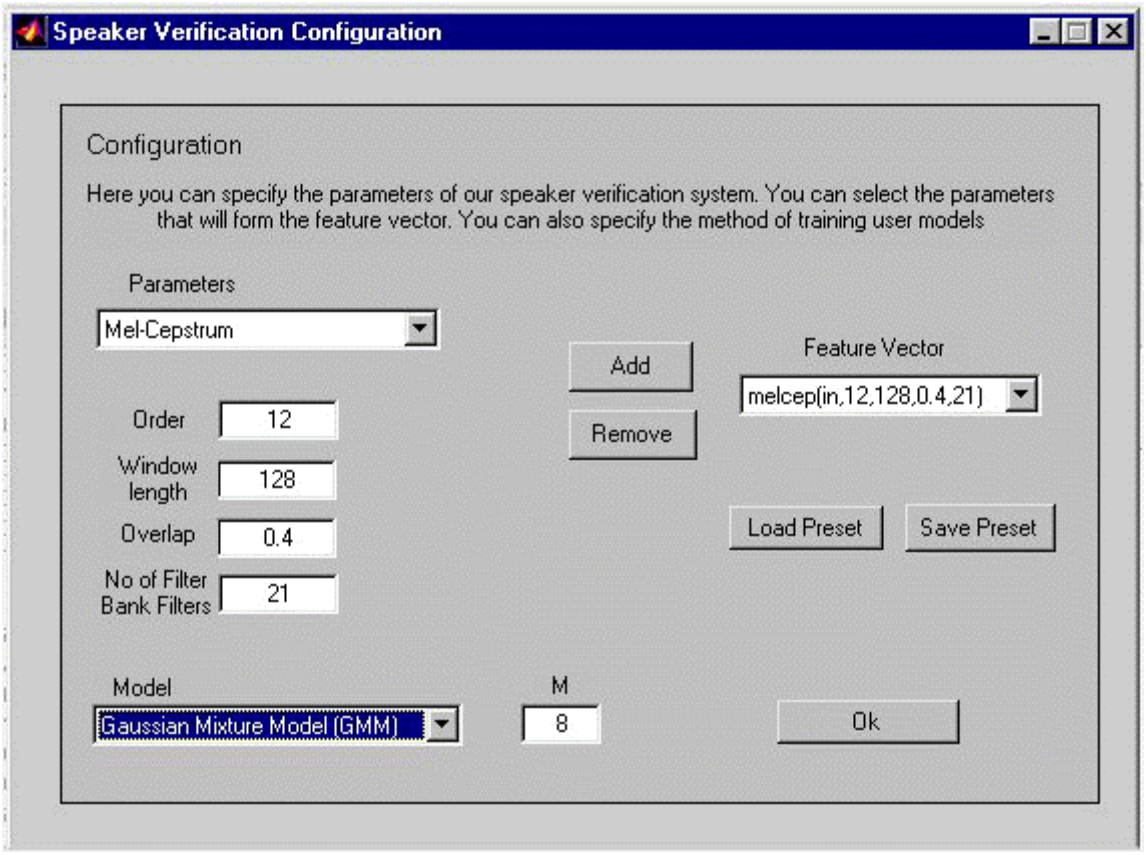


Figure 6.4: The Configuration window

We can also select the method of modeling speaker's identity. We can choose between Neural Networks, Vector Quantisation, HMM, GMM and Dynamic Time Warping. However, in this version of the programme, we can only work with Gaussian Mixture Models. We can also specify the number of Gaussian Mixtures that will be used to form each model.

After building the Feature Vector and defining the speaker modelling technique, we can save our configuration by pressing the 'Save Preset' button. The GUI saves our configuration into the *.ftr file and simultaneously builds two MATLAB files in the directory of the configuration file. One named 'Feat_Batch.m', which will be used in the Enrollment window and the Verification window to perform all the essential feature extraction to the input speech waveforms and one named 'Model_Batch.m' that would be used to train the GMM. We have to say that in this callback, we have not taken into account the case of a model other than GMM. If you select any other modelling technique, the GUI

gives you an error message. Moreover, at this point GUI initiates an empty user list saved in the text file 'Users'.

We can also load an already saved configuration file and make some changes in its configuration. Then, we can easily save it using the procedure described above. However, in this case all the users trained and all other data kept from previous sessions with that preset will be lost.

On completion of our configuring session, we can press the 'Ok' button to get back to the main Speaker Verification window

6.3.3 The Enrollment Window

In the *Enrollment Window,* the new user enrols himself to the system. In order to do that the new user has to record five phrases written in the left part of the window. He can record each of these phrases by pressing the corresponding 'Record' button on the right. Once pressed, each of the 'Record' buttons calls the 'soundrec.m' function that records the user's utterance from the microphone connected to the PC's soundcard. After that the callback calls a MATLAB batch file, named 'preprocess.m', that performs the essential preprocessing to the recorded waveform. This file can be edited by the user and add or remove preprocessing functions.

In some cases, we can not have many speakers available to perform our experiments. Therefore, we might as well use some of the standard speaker databases for speech processing applications that are available. These databases contain sets of speech samples of many speakers that can be used as training and testing data to speaker verification systems. In that case, these speech samples will be in the form of a computer sound file. Therefore, we have provided a set of 'From File' buttons that enables us to load the corresponding phrase from a computer file. However, the GUI does not support every computer sound format. Only Microsoft .wav mono files with 8 KHz sampling rate are supported. If we try to load a different type of file, the GUI will inform us that it can not support that kind of sound format. Moreover, when we record or load a phrase, the corresponding button changes colour. This to help us check the number of phrases we have recorded, and the method (record or load).

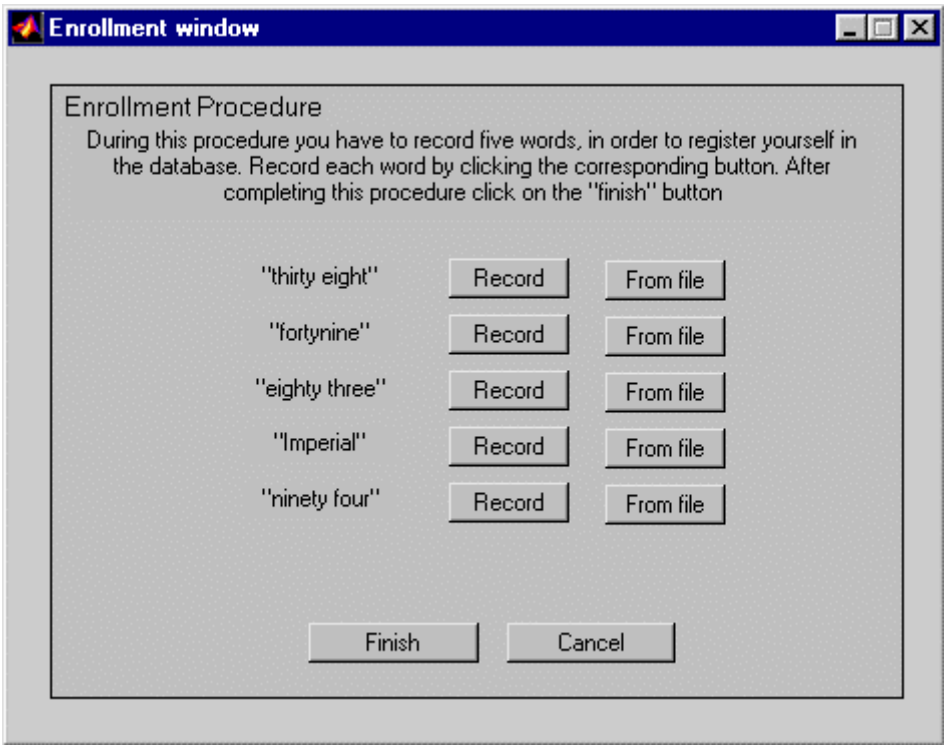


Figure 6.5: The Enrolment Window

When we have recorded all the samples we only have to press the 'Finish' button. If we haven't recorded all the phrases, the GUI will show an error message. At this time the GUI connects all recorded and preprocessed waveforms to form a single waveform. After that, it calls the 'Feat_Batch.m' and 'Model_Batch.m' functions, so as to extract the feature vectors and finally, to train the GMM for the user. When the systems completes all this processing, it saves the parameters that represent a GMM (see chapter 5) in a binary file that is named after the username of the user in the system. The structure of this file is shown in the following matrix. Moreover, it adds the user's name into the username database (Users file) and refresh the User pop-down menu in the main Speaker Verification window, before returning to it.

$$\begin{bmatrix} P_1 & P_2 & \dots & \dots & P_M \\ \underline{m}_1 & \underline{m}_2 & \dots & \dots & \underline{m}_M \\ \underline{C}_1 & \underline{C}_2 & \dots & \dots & \underline{C}_M \end{bmatrix}$$

If at any time we want to cancel the enrolment procedure and return to the main Speaker Verification window, we simply have to press the 'Cancel' button.

6.3.4 The Verification Window

In the *Verification Window,* a user can authenticate himself to the system. As in the Enrollment window, the user is prompted to record a phrase so as to prove his identity.

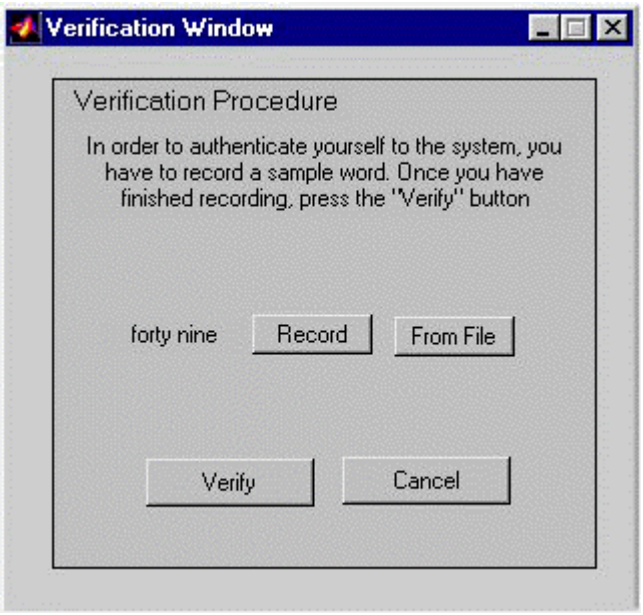


Figure 6.6: The Verification window

For the same reasons mentioned before, we have provided the option of loading the phrase from a speech file, so as to take advantage of the standard speaker databases. The same sound format restrictions apply here. Only 8 KHz – mono – Microsoft wav files are accepted again in this case. Moreover, we get the same type of colour indication, as a confirmation of recording or loading the phrase. After that automatically, the callback performs preprocessing, using 'preprocess.m'.

When we have recorded the sample phrase, we only have to press the 'Verify' button. If we haven't recorded the phrase, the GUI will show an error message. Then it calls 'Feat_Batch.m', so as to extract the feature vectors from the recorded and preprocessed waveform. Then, the programme calculates the logarithmic propabilities, described by

the equation (5.15), for each user. In order to perform that task the programme loads the model parameters stored for each user and calls the function 'GMM_verify.m', which calculates the log-probability for each user, given the input feature vectors. After that, we calculate the cohort-normalised probability of the claimed user, using the principles described in chapter 5.5. If this probability (score) exceeds the set threshold, then the user is given authorisation to the system, otherwise he is not. The cohort speakers are defined in this application as the K users with the greatest probability (score). The number of cohort speakers, the threshold as well as the whole normalisation procedure are set in the function 'cohort.m' and can be changed for experimentation by editing the corresponding file.

If we want to get the cohort-normalised score of a user after verification, we can type in the MATLAB command window the command 'cohort_sc(result,index)'.

If at any time we want to cancel the enrolment procedure and return to the main Speaker Verification window, we simply have to press the 'Cancel' button.

CHAPTER 7

IMPLEMENTATION AND TEST RESULTS

7.1 Introduction

The Graphical User Interface for Speaker Verification is a very powerful tool for evaluating Speaker Verification algorithms. One can choose between many different combinations of parameters and therefore examine the way each parameter can actually affect the performance of a speaker verification system.

In this chapter, we are going to present the implementation of a GMM speaker verification system with many different configurations, using the GUI developed. Several performance results for various GMM configurations are presented and analysed.

7.2 Speaker Database

In order to evaluate a speaker verification system, we need to create a database of speakers. A set of sample phrases is recorded for each user, so as to be used for enrolment and verification in the system.

In our case, we have used the TIMIT speaker database so as to acquire the essential speech samples. This database offers a number of sample phrases of several American men and women. Unfortunately, they are stored in the SPHERE/TIMIT format that is actually used by the National Institute of Speech Technology (NIST). In order to facilitate programming, we have used the *'readsph.m'* routine, provided by the *VoiceBox,* to read these files in MATLAB and transform them to '.wav' files that can be easily manipulated by almost any Windows programme. We have saved them as Microsoft wave mono files, using 8 KHz sampling rate and 8 bits/sample. This is a common choice, used in many speech processing applications.

For each speaker, we have used a set of five phrases with average duration of 4 secs each. This set is used in the user's enrolment to the system, providing the essential input data for the system to train. One of the five phrases is then used for the verification of the user's identity. Moreover, there is no need for endpoint detection, as these speech samples are pre-recorded and therefore the removal of the silent parts is already done. However, in the case that we want to make real time enrolment and verification of a user using the 'record' buttons provided in the GUI, we definitely need to add one of the two endpoint implementations described earlier in the function 'preprocess.m'.

7.3 Experiment Configuration

In these series of experiments, we are going to evaluate the performance of the developed GMM verification system. We are going to use a set of eight (8) speakers, four female (F0, F1, F2, F3) and four male (M0, M1, M2, M3) as a database for our experiments. We are going to use four cohort speakers in the cohort normalisation procedure. *Cohort speakers are considered the speakers that give the greatest probability (score) to the presented input feature vectors.*

We are going to segment our input signal using 50% overlapping frames of 256 samples. We are also going to build a GMM using 10 Gaussian Mixtures, so as to model each user. In this series of experiments, we will look at the effect of using different feature vectors on the system's performance. We will even try several combinations of feature vectors on our system. We will also look at the effect of varying the number of gaussian mixtures used. We will look at the performance of the system, where the speakers are only of the same sex. Finally, we will look at the performance of OGMM.

In order to measure the *false rejection rate*, we use the five recorded sample phrases from each user, whereas in order to measure the *false alarm rate*, we use seven recorded sample phrases one from each user of the system. This is done in order to capture the score of each user to the user who claims identification.

7.4 Test 1 (Using MFCC)

In the first series of experiments, we are going to construct our feature vectors using 12 mel-frequency cepstrum coefficients (MFCC) extracting from each input frame. The next tables exhibit the system's performance for speakers F0, F1 and M0, M1. Some plots of the False Alarm and False Rejection rate for these cases follow:

Average Cohort Normalised Score for F0	Cohort Normalised Scores for F0	Cohort Normalised Scores for Impostors	Threshold	FA (7 max)	FR (5 Max)
			0	7	0
			32.0407	7	0
		F1 155.0088	64.0814	7	0
	342.7975	F2 126.3993	96.1221	7	0
	242.0395	F3 126.3993	128.1628	4	0
320.4076	458.4545	M0 132.6076	160.2035	1	0
	244.1264	M1 112.4896	192.2442	0	0
	314.6201	M2 168.2342	224.2849	0	0
		M3 152.7599	256.3256	0	2
			288.3663	0	2
			320.4070	0	3

Average Cohort Normalised Score for F1	Cohort Normalised Scores for F1	Cohort Normalised Scores for Impostors	Threshold	FA	FR
			0	5	0
			14.3740	3	0
		F0 27.7236	28.7480	2	0
	259.4252	F2 5.0238	43.1220	0	0
	79.2164	F3 34.8345	57.4960	0	1
143.7403	158.1164	M0 -10.4946	71.8700	0	2
	167.0854	M1 -5.2652	86.2440	0	2
	54.8582	M2 40.7397	100.6180	0	2
		M3 10.5784	114.9920	0	2
			129.3660	0	2
			143.7400	0	2

Average Cohort normalised score for M0	Cohort Normalised Scores for M0	Cohort Normalised Scores for Impostors	Threshold	FA	FR
18.3770	84.0158 -2.5079 8.5141 30.8641 -29.0010	F0 -237.2457 F1 -382.6884 F2 -243.1720 F3 -203.0030 M1 -148.0065 M2 -123.7555 M3 -60.2785	0 1.8377 3.6754 5.5131 7.3508 9.1885 11.0262 12.8639 14.7016 16.5393 18.3770	0 0 0 0 0 0 0 0 0 0 0	2 2 2 2 2 3 3 3 3 3 3

Average Cohort normalised score for M1	Cohort Normalised Scores for M1	Cohort Normalised Scores for Impostors	Threshold	FA	FR
26.5078	72.6883 -14.2364 3.7305 11.1960 59.1608	F0 -220.0958 F1 -193.4146 F2 -227.7684 F3 -152.0800 M0 -133.6616 M2 -110.3620 M3 -160.8010	0 2.6508 5.3016 7.9523 10.6031 13.2539 15.9047 18.5555 21.2062 23.8570 26.5078	0 0 0 0 0 0 0 0 0 0 0	1 1 2 2 2 3 3 3 3 3 3

If we have a look at these tables and the figures in the next page, we can clearly see that by using only 12 MFCC, we can have a speaker verification system that works efficiently. We can see that each user is well discriminated from his imposters, giving a very low EER (almost equal to zero). In other words, the difference between the score of the claimed user is much higher than his imposter's score. We include even speakers of the opposite sex as impostors, making the system more sophisticated.

However, we can clearly see the difference between the score levels of the users. We can see that women tend to score much higher than men, making it difficult to set a global threshold for the system.

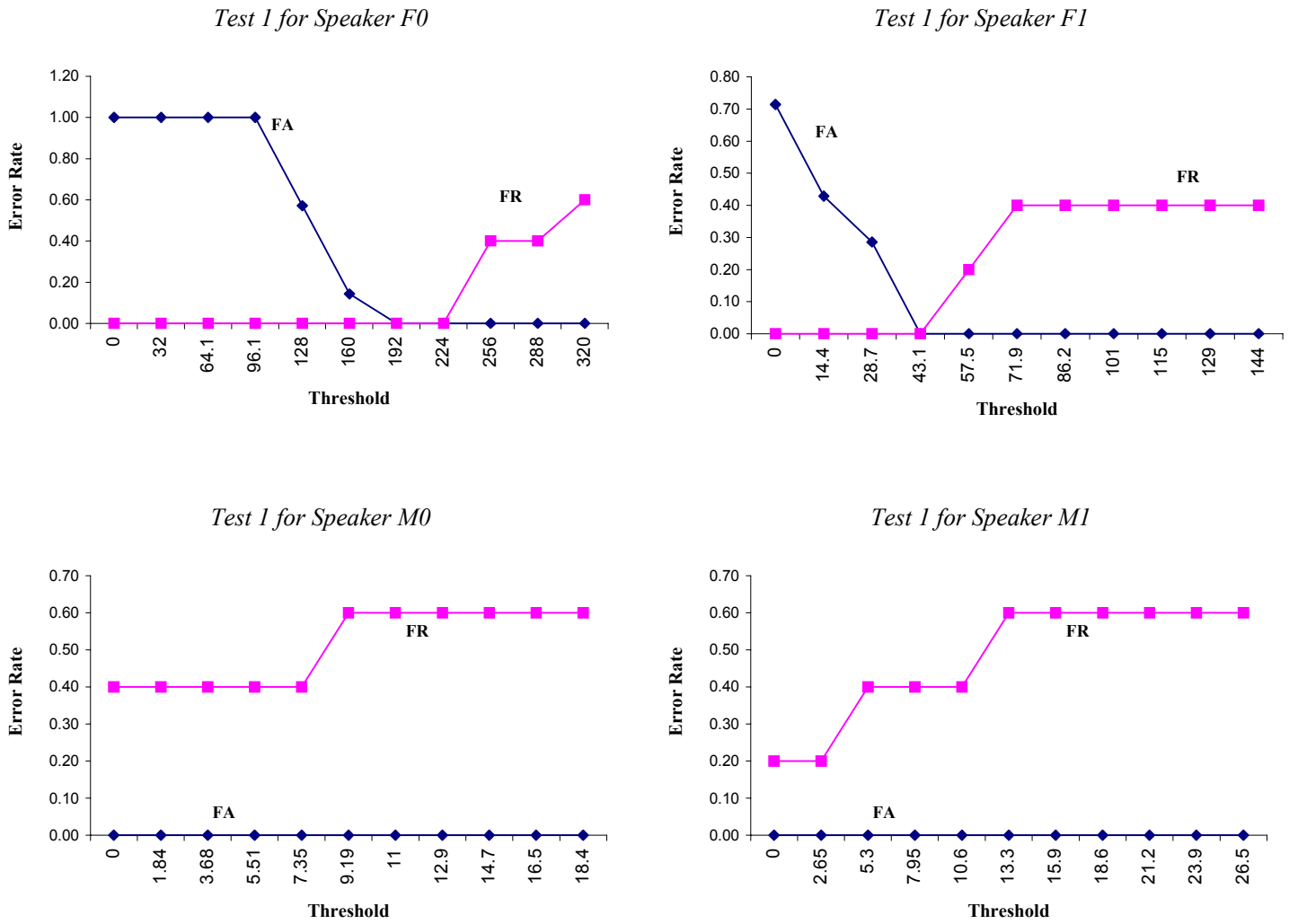


Figure 7.1: Using 12 MFCC coefficients

7.5 Test 2 (Using ΔMFCC)

In the second series of experiments, we are going to construct our feature vectors using 12 delta mel-frequency cepstrum coefficients (ΔMFCC) extracting from each input frame. The next tables exhibit the system's performance for speakers F0,F1 and M0,M1. Some plots of the False Alarm and False Rejection rate for these cases follow:

Average Cohort Normalised Score for F0	Cohort Normalised Scores for F0	Cohort Normalised Scores for Impostors	Threshold	FA (7 max)	FR (5 Max)
204.688	234.4640 136.5894 349.8669 124.6306 177.8891	F1 -33.4396 F2 -94.4966 F3 12.1073 M0 -37.9261 M1 -25.0339 M2 90.0656 M3 2.8629	0 20.4688 40.9376 61.4064 81.8752 102.3440 122.8128 143.2816 163.7504 184.2192 204.6880	3 1 1 1 1 0 0 0 0 0 0	0 0 0 0 0 0 0 2 2 3 3

Average Cohort Normalised Score for F1	Cohort Normalised Scores for F1	Cohort Normalised Scores for Impostors	Threshold	FA	FR
64.5655	102.6068 42.2289 92.1263 83.5857 2.2796	F0 -30.4692 F2 -20.4188 F3 0.4219 M0 -37.2810 M1 -46.3205 M2 -35.5674 M3 -6.2051	0 6.4566 12.9131 19.3697 25.8262 32.2828 38.7393 45.1959 51.6524 58.1090 64.5655	1 0 0 0 0 0 0 0 0 0 0	0 1 1 1 1 1 1 2 2 2 2

Average Cohort Normalised Score for M0	Cohort Normalised Scores for M0	Cohort Normalised Scores for Impostors	Threshold	FA	FR
			0	0	1
			2.7920	0	1
		F0 -128.4336	5.5841	0	1
	67.5347	F1 -120.0323	8.3761	0	1
	15.3149	F2 -109.7734	11.1682	0	1
27.9204	33.6769	F3 -61.8854	13.9602	0	1
	46.9246	M1 –20.8861	16.7522	0	2
	-23.8490	M2 –54.5156	19.5443	0	2
		M3 –69.0369	22.3363	0	2
			25.1284	0	2
			27.9204	0	2

Average Cohort Normalised Score for M1	Cohort Normalised Scores for M1	Cohort Normalised Scores for Impostors	Threshold	FA	FR
			0	0	0
			7.8374	0	0
		F0 -40.8414	15.6749	0	0
	129.8293	F1 -8.6079	23.5123	0	0
	66.3454	F2 -78.9852	31.3498	0	0
78.3744	48.9340	F3 –17.2950	39.1872	0	0
	64.1930	M0 –40.3675	47.0246	0	0
	82.5703	M2 –15.6735	54.8621	0	1
		M3 –44.6298	62.6995	0	1
			70.5370	0	3
			78.3744	0	3

If we have a look at these tables and the figures in the next page, we can clearly see that by using only 12 ΔMFCC, we can have a speaker verification system that works efficiently. We can see that each user is well discriminated from his imposters, giving a very low EER. In other words, the difference between the score of the claimed user is much higher than his imposter's score.

However, we can see that there is an improvement in the score levels of the users. We can see that the distance between female and male scores is much less and therefore we can easily set a very good threshold for the score. A global set threshold can cause only very small errors.

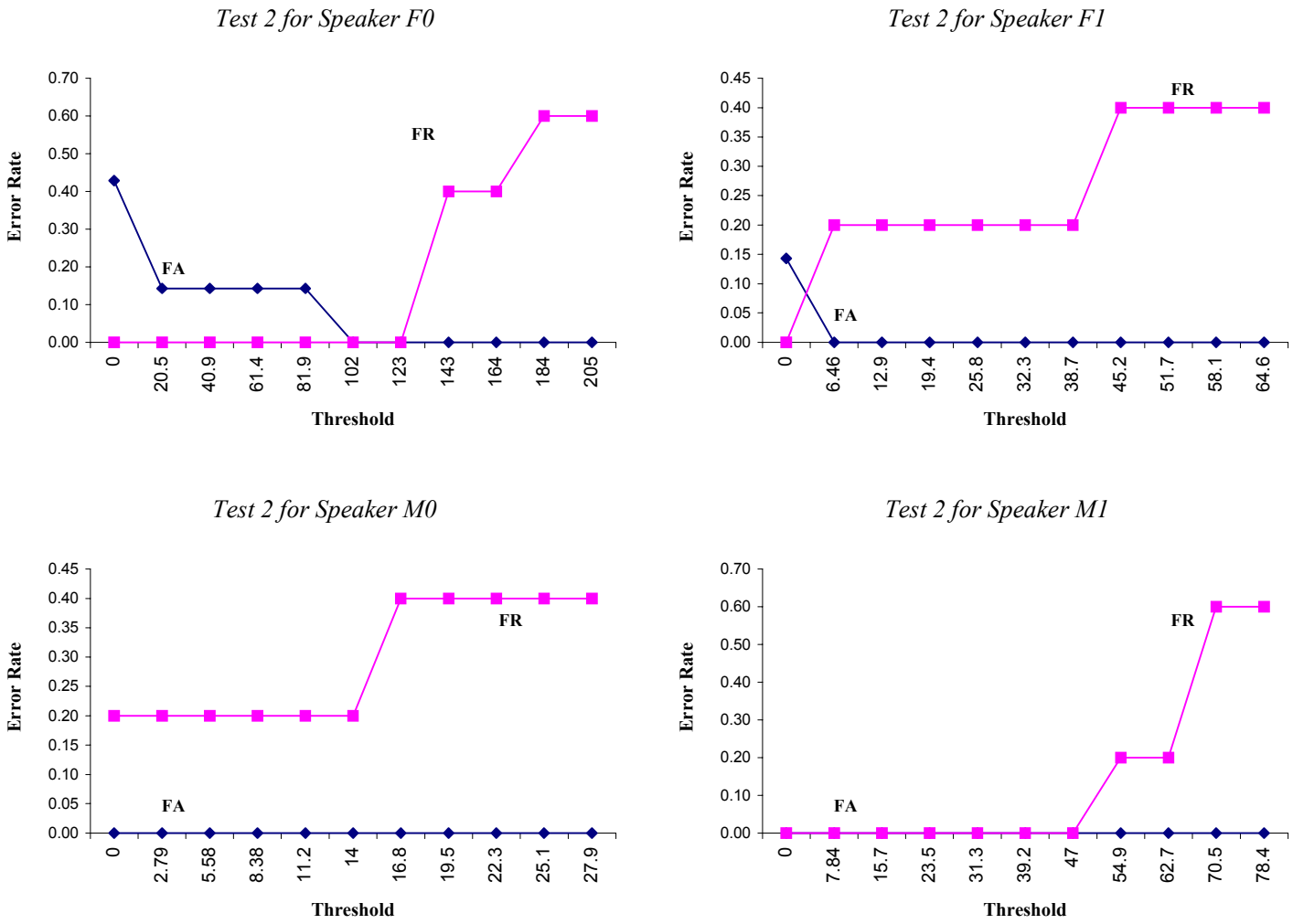


Figure 7.2: Using 12 ΔMFCC coefficients

7.6 Test 3 (Using PLP)

In the second series of experiments, we are going to construct our feature vectors using 12 Perceptual Linear Predictive coefficients (PLP) extracting from each input frame. The next tables exhibit the system's performance for speakers F0,F1 and M0,M1. Some plots of the False Alarm and False Rejection rate for these cases follow:

Average Cohort Normalised Score for F0	Cohort Normalised Scores for F0	Cohort Normalised Scores for Impostors	Threshold	FA (7 max)	FR (5 Max)
			0	2	0
			6.8553	0	0
		F1 -6.8252	13.7105	0	0
	87.5915	F2 –44.5732	20.5658	0	0
	43.7822	F3 2.2219	27.4210	0	0
68.5526	99.3773	M0 0.1234	34.2763	0	0
	45.8457	M1 -4.3967	41.1316	0	0
	66.1664	M2 -22.7332	47.9868	0	2
		M3 -36.0392	54.8421	0	2
			61.6973	0	2
			68.5526	0	3

Average Cohort Normalised Score for F1	Cohort Normalised Scores for F1	Cohort Normalised Scores for Impostors	Threshold	FA (7 max)	FR (5 max)
			0	1	0
			5.6960	0	0
		F0 –13.5140	11.3920	0	0
	79.6295	F2 –29.1973	17.0880	0	0
	35.8952	F3 1.3042	22.7840	0	1
56.9599	73.8769	M0 –46.8732	28.4799	0	1
	74.3494	M1 –15.1412	34.1759	0	1
	21.0484	M2 –35.3177	39.8719	0	2
		M3 -6.1054	45.5679	0	2
			51.2639	0	2
			56.9599	0	2

Average Cohort Normalised Score for M0	Cohort Normalised Scores for M0	Cohort Normalised Scores for Impostors	Threshold	FA (7 max)	FR (5 max)
			0	0	0
			4.8260	0	0
		F0 –69.7298	9.6521	0	0
	89.3925	F1 –80.8268	14.4781	0	0
	18.6698	F2 –28.1286	19.3041	0	1
48.2603	35.3915	F3 -5.7748	24.1302	0	1
	62.7609	M1 –35.9042	28.9562	0	1
	35.0866	M2 -49.1091	33.7822	0	1
		M3 -41.0306	38.6082	0	3
			43.4343	0	3
			48.2603	0	3

Average Cohort Normalised Score for M1	Cohort Normalised Scores for M1	Cohort Normalised Scores for Impostors	Threshold	FA (7 max)	FR (5 Max)
			0	2	0
			3.4339	1	0
		F0 –16.2601	6.8677	0	0
	44.6773	F1 –27.0337	10.3016	0	0
	27.8135	F2 –66.1427	13.7355	0	0
34.3387	40.2587	F3 –26.9344	17.1694	0	0
	22.1097	M0 -10.6571	20.6032	0	0
	36.8343	M2 1.3436	24.0371	0	1
		M3 7.1980	27.4710	0	1
			30.9048	0	2
			34.3387	0	2

If we have a look at these tables and the figures in the next page, we can clearly see that by using only 12 PLP, we can have a robust speaker verification system. We can see that each user is well discriminated from his imposters, giving a very low EER (close to zero). In other words, the difference between the score of the claimed use is much higher than his imposter's score.

However, we can see that there is an improvement in the score levels of the users, compared to other coefficients. We can see that female and male scores are indeed on the same level and therefore we can easily set a very good threshold for the score. We can see that we can easily choose many threshold values that can give zero EER.

Another important feature of PLP coefficients is that they feature almost zero correlation, a very important property for GMM modeling.

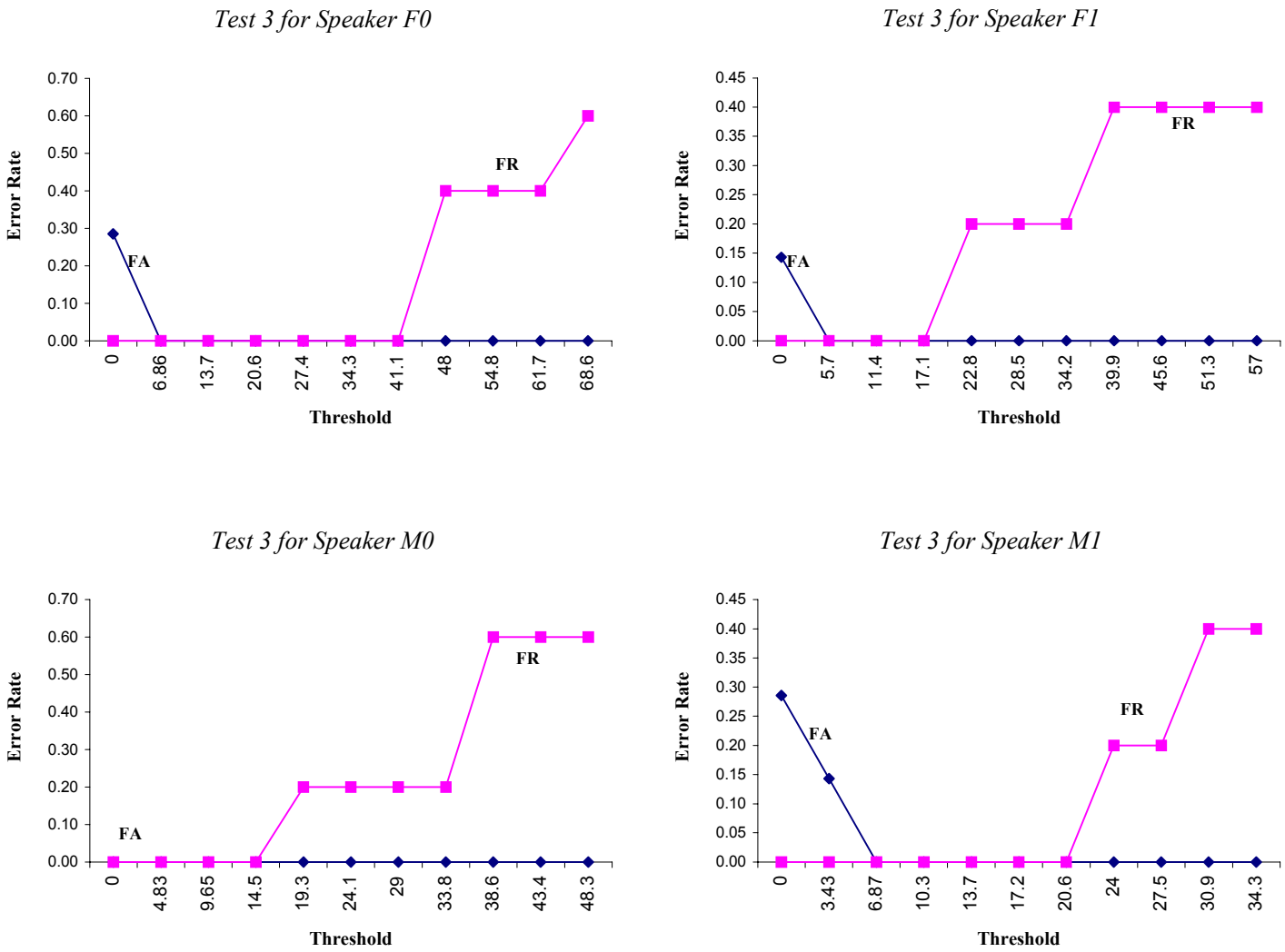


Figure 7.3: Using 12 PLP coefficients

7.7 Test 4 (Using PLP and ΔMFCC)

In the fourth series of experiments, we are going to construct our feature vectors using 12 Perceptual Linear Predictive coefficients (PLP) and 12 delta mel-cepstrum coefficients extracting from each input frame. The next tables exhibit the system's performance for speakers F0,F1 and M0,M1. Some plots of the False Alarm and False Rejection rate for these cases follow:

Average Cohort Normalised Score for F0	Cohort Normalised Scores for F0	Cohort Normalised Scores for Impostors	Threshold	FA (7 max)	FR (5 Max)
350.3836	384.5219 282.0019 564.0734 238.1528 283.1679	F1 86.3916 F2 40.2127 F3 -193.7869 M0 -5.7216 M1 –27.9095 M2 83.1976 M3 –32.8901	0 35.0384 70.0767 105.1151 140.1534 175.1918 210.2302 245.2685 280.3069 315.3452 350.3836	3 3 2 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 1 1 3 3

Average Cohort Normalised Score for F1	Cohort Normalised Scores for F1	Cohort Normalised Scores for Impostors	Threshold	FA	FR
141.9896	236.5447 76.2235 210.3921 135.5856 51.2020	F0 -6.6128 F2 -0.4183 F3 -117.4875 M0 –107.7117 M1 -21.4172 M2 -21.1655 M3 -41.8083	0 14.1990 28.3979 42.5969 56.7958 70.9948 85.1938 99.3927 113.5917 127.7906 141.9896	0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 1 1 2 2 2 3 4

Average Cohort Normalised Score for M0	Cohort Normalised Scores for M0	Cohort Normalised Scores for Impostors	Threshold	FA	FR
100.8200	195.2111 57.2766 62.5647 168.3659 20.6818	F0 -233.4315 F1-299.0766 F2 -56.3673 F3 -45.0496 M1 -52.3311 M2 -80.2822 M3 -61.5213	0 10.0820 20.1640 30.2460 40.3280 50.4100 60.4920 70.5740 80.6560 90.7380 100.8200	0 0 0 0 0 0 0 0 0 0 0	0 0 0 1 1 1 2 3 3 3 3

Average Cohort Normalised Score for M1	Cohort Normalised Scores for M1	Cohort Normalised Scores for Impostors	Threshold	FA	FR
127.5224	205.7477 65.7902 130.3971 95.0139 140.6633	F0 -153.7269 F1 -61.9602 F2 -81.8730 F3 -157.9676 M0 -92.6618 M2 -79.7210 M3 -68.1201	0 12.7522 25.5045 38.2567 51.0090 63.7612 76.5134 89.2657 102.0179 114.7702 127.5224	0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 1 1 2 2 2

If we have a look at these tables and the figures in the next page, we can clearly see that by using only 12 PLP and 12 ΔMFCC, we can have a robust speaker verification system. We can see that each user is well discriminated from his imposters, giving a very low EER (close to zero). In other words, the difference between the score of the claimed user is much higher than his imposter's score.

However, we can see that again we have the introduction of difference in threshold between male and female users and this is maybe due to the introduction of ΔMFCC. However, we can still set a threshold that can guarantee low EER.

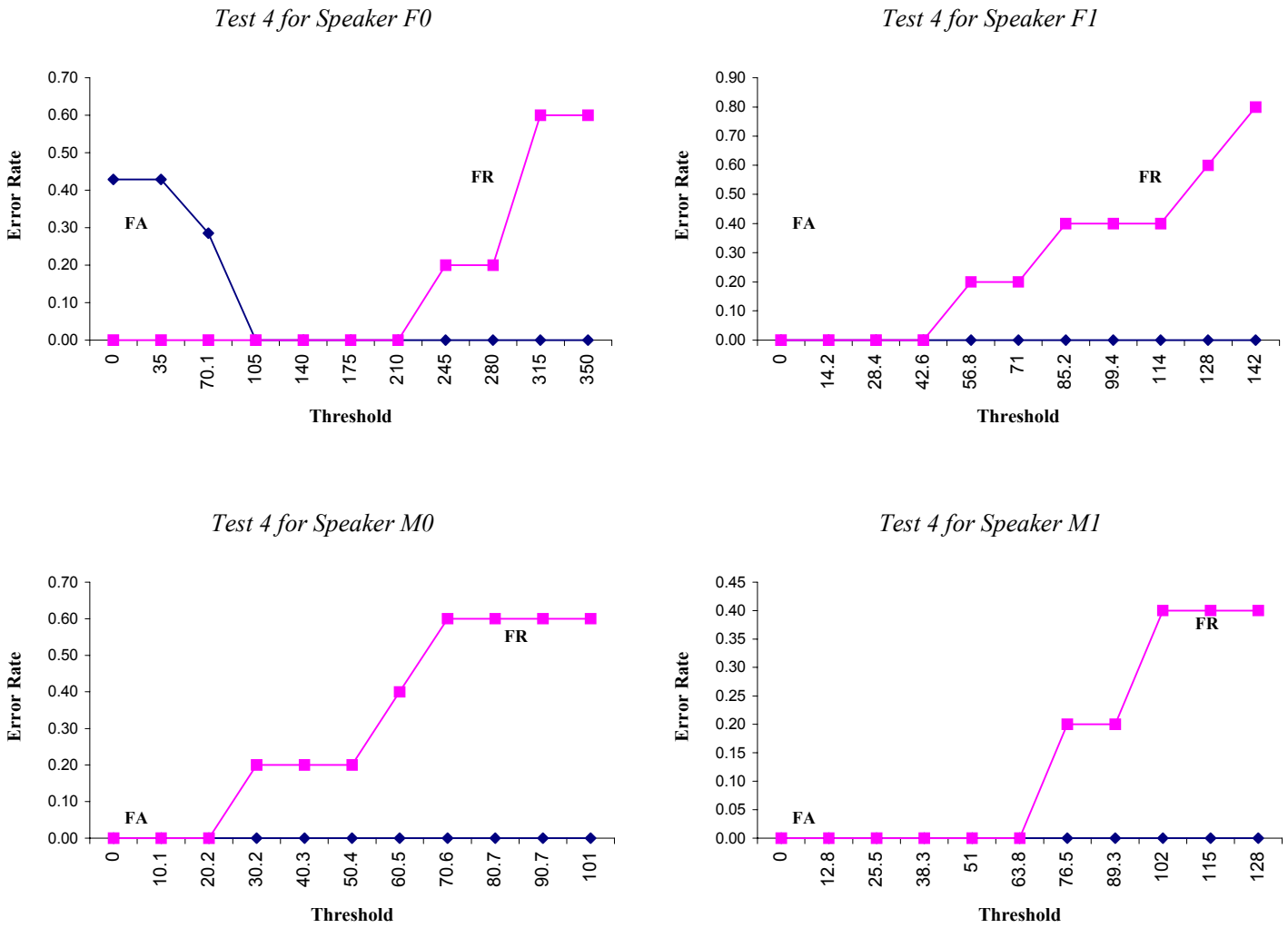


Figure 7.4: Using 12 PLP and ΔMFCC coefficients

7.8 Test 5 (Using MFCC and more Gaussian Mixtures)

In the fifth series of experiments, we are going to construct our feature vectors using 12 Mel-Frequency Cepstrum coefficients (MFCC) coefficients extracting from each input frame. In this case, we are going to increase the number of Gaussian Mixtures (M=20) and investigate its influence on the system. The next tables exhibit the system's performance for speakers F0, F1 and M0, M1. Some plots of the False Alarm and False Rejection rate for these cases follow:

Average Cohort Normalised Score for F0	Cohort Normalised Scores for F0	Cohort Normalised Scores for Impostors	Threshold	FA (7 max)	FR (5 Max)
342.5401	387.6811 228.0665 472.7063 257.6353 366.6111	F1 146.9709 F2 81.7497 F3 99.7286 M0 98.1175 M1 102.7716 M2 167.8868 M3 131.1468	0 34.2540 68.5080 102.7620 137.0160 171.2701 205.5241 239.7781 274.0321 308.2861 342.5401	7 7 7 4 2 0 0 0 0 0 0	0 0 0 0 0 0 0 1 2 2 2

Average Cohort Normalised Score for F1	Cohort Normalised Scores for F1	Cohort Normalised Scores for Impostors	Threshold	FA	FR
175.8333	307.0346 115.3491 150.4034 215.0552 91.3242	F0 -8.4269 F2 -22.0588 F3 25.0010 M0 2.4398 M1 -11.9561 M2 -39.1441 M3 4.8098	0 17.5833 35.1667 52.7500 70.3333 87.9167 105.5000 123.0833 140.6666 158.2500 175.8333	3 1 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 1 2 2 3 3

Average Cohort Normalised Score for M0	Cohort Normalised Scores for M0	Cohort Normalised Scores for Impostors	Threshold	FA	FR
76.94944	146.5605 41.9161 53.3392 96.3669 46.5645	F0 -273.3069 F1 -355.8456 F2 -178.5954 F3 -191.5491 M1 -156.7696 M2 -73.7540 M3 -120.4764	0 7.6949 15.3899 23.0848 30.7798 38.4747 46.1697 53.8646 61.5596 69.2545 76.9494	0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 1 3 3 3 3

Average Cohort Normalised Score for M1	Cohort Normalised Scores for M1	Cohort Normalised Scores for Impostors	Threshold	FA	FR
40.77892	99.3159 -5.3707 9.5633 35.5354 64.8507	F0 -250.9470 F1 -228.3324 F2 -270.3649 F3 -171.3418 M0 -191.8554 M2 -112.8495 M3 -241.3352	0 4.0779 8.1558 12.2337 16.3116 20.3895 24.4674 28.5452 32.6231 36.7010 40.7789	0 0 0 0 0 0 0 0 0 0 0	1 1 1 2 2 2 2 2 2 3 3

If we compare the results presented here and the tables referring to Test 1, we can see that there is a slight improvement to the system by using more Gaussian Mixtures. There is also a chance that the system with more Gaussian Mixtures demands more training input data and therefore we don't see a radical improvement.

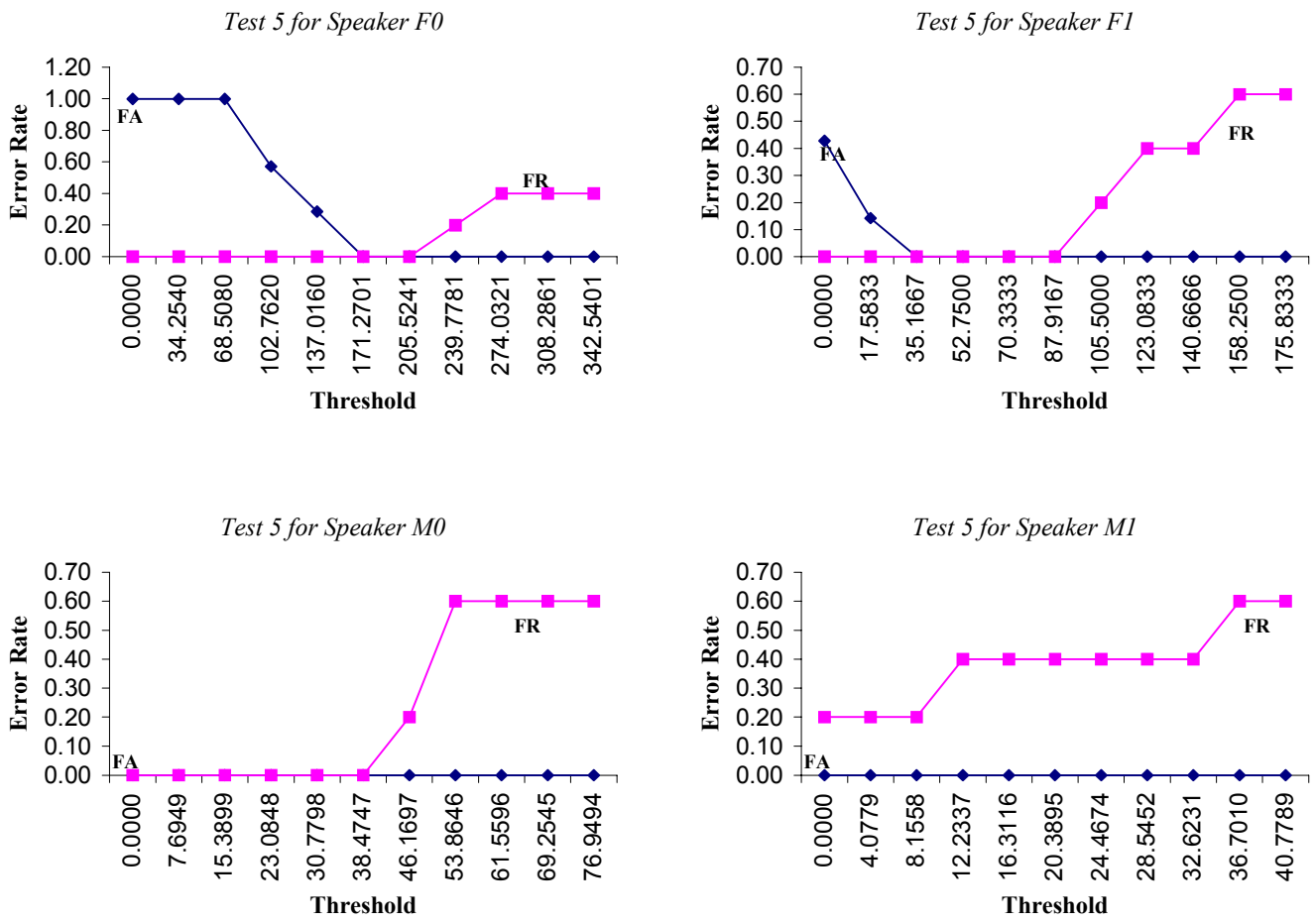


Figure 7.5: Using 12 MFCC coefficients and 20 Gaussian Mixtures

7.9 Test 6 (Using more MFCC coefficients)

In the sixth series of experiments, we are going to construct our feature vectors using 20 Mel-Frequency Cepstrum coefficients (MFCC) coefficients extracting from each input frame. In this case, we have increased the number of MFCC coefficients and investigate its influence on the system. The next tables exhibit the system's performance for speakers F0, F1 and M0, M1. Some plots of the False Alarm and False Rejection rate for these cases follow:

Average Cohort Normalised Score for F0	Cohort Normalised Scores for F0	Cohort Normalised Scores for Impostors	Threshold	FA (7 max)	FR (5 Max)
677.6584	657.1187 554.0639 995.6068 505.6067 675.8961	F1 101.5020 F2 126.1821 F3 127.4300 M0 -0.4797 M1 54.1754 M2 91.9601 M3 44.9211	0 67.7658 135.5317 203.2975 271.0634 338.8292 406.5950 474.3609 542.1267 609.8926 677.6584	6 4 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 1 2 3

Average Cohort Normalised Score for F1	Cohort Normalised Scores for F1	Cohort Normalised Scores for Impostors	Threshold	FA	FR
301.7609	467.8506 206.3815 334.7282 367.8810 131.9632	F0 -210.6930 F2 -88.0537 F3 11.1710 M0 79.8697 M1 96.0981 M2 115.4771 M3 65.3944	0 30.1761 60.3522 90.5283 120.7044 150.8805 181.0565 211.2326 241.4087 271.5848 301.7609	5 4 4 2 0 0 0 0 0 0 0	0 0 0 0 0 1 1 2 2 2 2

Average Cohort Normalised Score for M0	Cohort Normalised Scores for M0	Cohort Normalised Scores for Impostors	Threshold	FA	FR
59.5317	119.9461 52.0795 43.1246 74.1355 8.3728	F0 -657.7771 F1 -410.3113 F2 -523.7886 F3 -433.5923 M1 -208.0058 M2 -149.6053 M3 -138.0398	0 5.9532 11.9063 17.8595 23.8127 29.7659 35.7190 41.6722 47.6254 53.5785 59.5317	0 0 0 0 0 0 0 0 0 0 0	0 0 1 1 1 1 1 1 2 3 3

Average Cohort Normalised Score for M1	Cohort Normalised Scores for M1	Cohort Normalised Scores for Impostors	Threshold	FA	FR
27.27598	86.3189 66.2110 -3.1400 -45.6736 32.6636	F0 -555.9564 F1 -299.7352 F2 -619.4888 F3 -418.8712 M0 -88.1881 M2 -69.1476 M3 -188.9657	0 2.7276 5.4552 8.1828 10.9104 13.6380 16.3656 19.0932 21.8208 24.5484 27.2760	0 0 0 0 0 0 0 0 0 0 0	2 2 2 2 2 2 2 2 2 2 2

In this series of experiments, we can easily see that by employing more MFCC coefficients, there is a slight improvement in the system's performance.

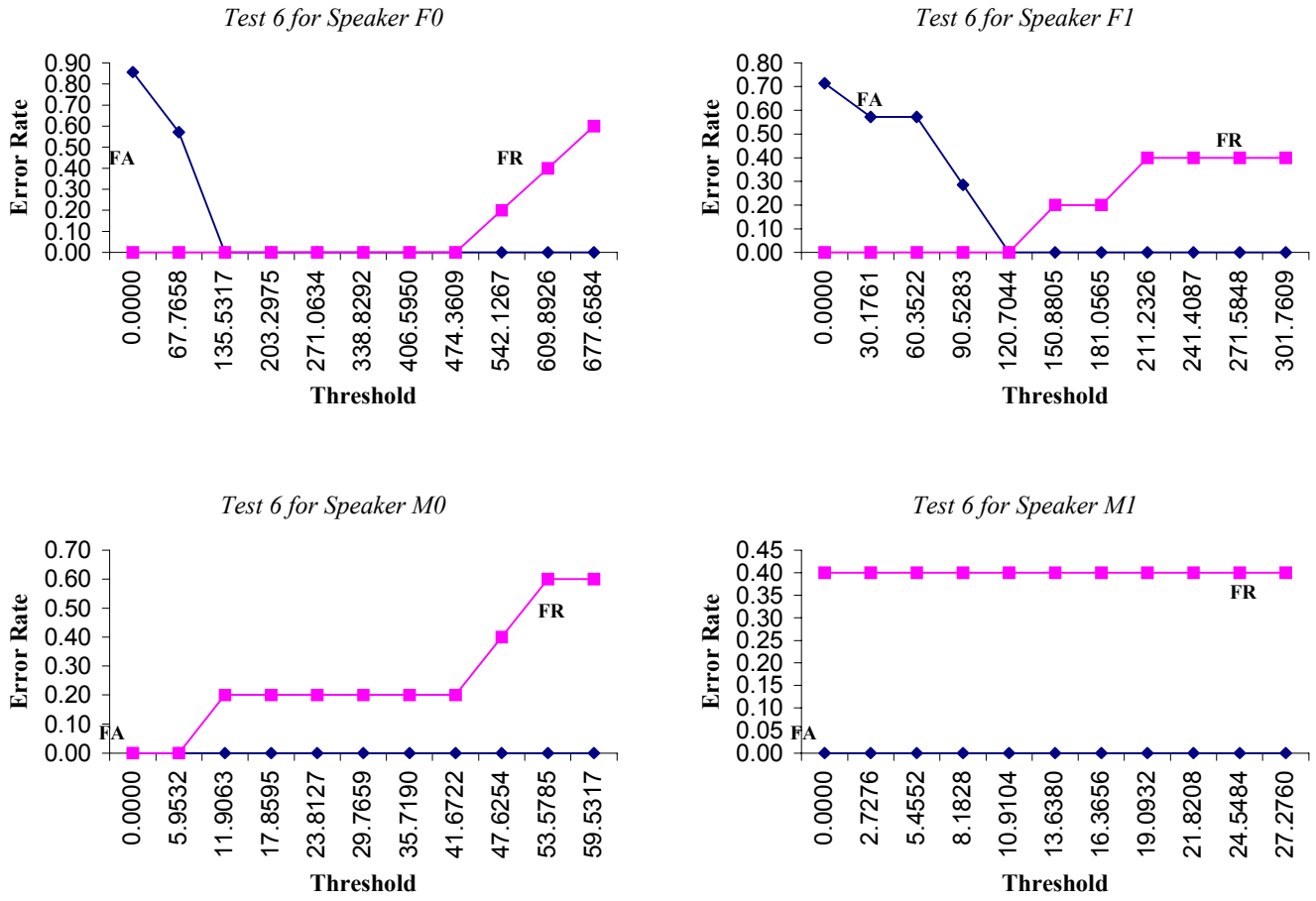


Figure 7.6: Using 20 MFCC coefficients

7.10 Test 7 (Using only Male speakers)

In the seventh series of experiments, we are going to construct our feature vectors using 12 Mel-Frequency Cepstrum coefficients (MFCC) coefficients extracting from each input frame. In this case, we are going to explore the effect of using a database of speakers having the same sex. Our database consists of seven male speakers M0 to M6. The next tables exhibit the system's performance for speakers M0, M1. Some plots of the False Alarm and False Rejection rate for these cases follow:

Average Cohort Normalised Score for M0	Cohort Normalised Scores for M0	Cohort Normalised Scores for Impostors	Threshold	FA (6 max)	FR (5 Max)
145.7617	204.3041 109.3478 86.5073 215.3207 113.3284	M1 100.0428 M2 75.5022 M3 61.7807 M4 64.6873 M5 81.3638 M6 59.6305	0 14.5762 29.1523 43.7285 58.3047 72.8808 87.4570 102.0332 116.6094 131.1855 145.7617	6 6 6 6 6 4 1 0 0 0 0	0 0 0 0 0 0 0 1 3 3 3

Average Cohort Normalised Score for M1	Cohort Normalised Scores for M1	Cohort Normalised Scores for Impostors	Threshold	FA	FR
191.2873	254.5638 140.4670 268.3636 121.1841 171.8582	M0 42.4796 M2 67.4456 M3 -34.2930 M4 56.5593 M5 169.0151 M6 -84.1360	0 19.1287 38.2575 57.3862 76.5149 95.6436 114.7724 133.9011 153.0298 172.1586 191.2873	4 4 4 2 1 1 1 1 1 0 0	0 0 0 0 0 0 0 1 2 3 3

Average Cohort Normalised Score for M3	Cohort Normalised Scores for M3	Cohort Normalised Scores for Impostors	Threshold	FA	FR
92.54184	156.1221 53.7950 109.4784 76.0151 67.2986	M0 -164.1685 M1 -140.6598 M2 -169.4607 M4 -164.9297 M5 -237.2176 M6 50.2905	0 9.2542 18.5084 27.7626 37.0167 46.2709 55.5251 64.7793 74.0335 83.2877 92.5418	1 1 1 1 1 1 0 0 0 0 0	0 0 0 0 0 0 1 1 2 3 3

This series of experiments demonstrates the performance of the system in a more efficiently. In the case of using male and female speakers, the term impostor is a bit exaggerated, as it is less possible to have a naturally speaking female as impostor to a man. However, in the opposite case, the considered impostors can really be impostors to a male or a female.

We can see that the system now has a measurable EER of about 15%, using 12 MFCCs. Indeed, this is a very good performance for the system. Moreover, we can witness the difference between the score levels of each male user, generating a slight difficulty in setting a global threshold.

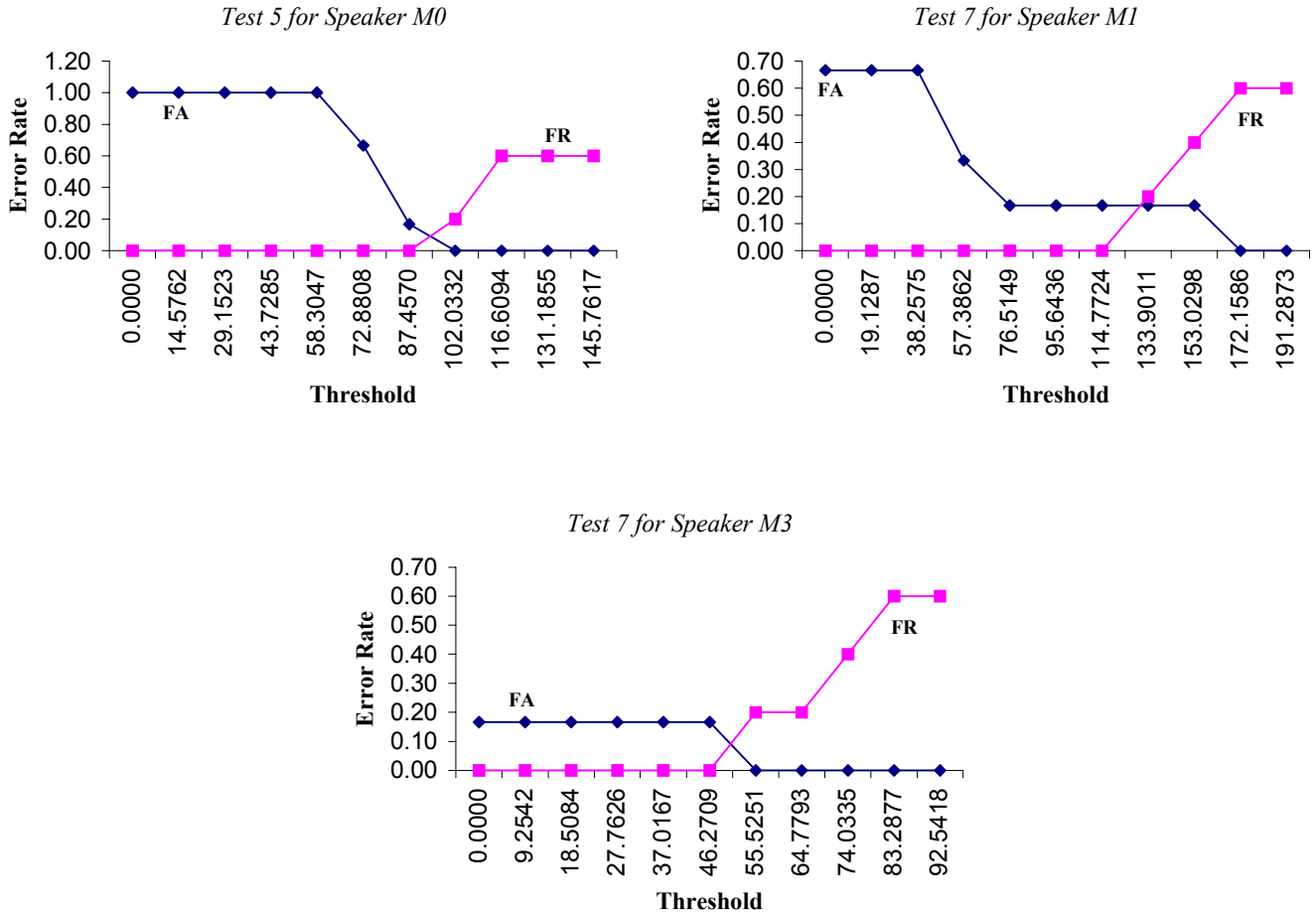


Figure 7.7: Using 12 MFCC coefficients and male only speakers

7.11 Test 8 (Using MFCC and OGMM)

In the first series of experiments, we are going to construct our feature vectors using 12 mel-frequency cepstrum coefficients (MFCC) extracting from each input frame. However, in this case we are going to investigate the effect of using OGMM to the system. The next tables exhibit the system's performance for speakers F0, F1 and M0, M1. Some plots of the False Alarm and False Rejection rate for these cases follow:

Average Cohort Normalised Score for F0	Cohort Normalised Scores for F0	Cohort Normalised Scores for Impostors	Threshold	FA (7 max)	FR (5 Max)
276.6359	323.4577 184.3985 366.2036 222.4624 286.6575	F1 95.4314 F2 46.7526 F3 43.3030 M0 56.6342 M1 81.2328 M2 39.7502 M3 50.6812	0 27.6636 55.3272 82.9908 110.6544 138.3180 165.9815 193.6451 221.3087 248.9723 276.6359	7 7 3 1 0 0 0 0 0 0 0	0 0 0 0 0 0 0 1 1 2 2

Average Cohort Normalised Score for F1	Cohort Normalised Scores for F1	Cohort Normalised Scores for Impostors	Threshold	FA (7 max)	FR (5 Max)
215.7034	349.3045 121.4806 249.4051 232.9532 125.3735	F0 41.0159 F2 64.6283 F3 75.8681 M0 39.5745 M1 59.3024 M2 53.9489 M3 64.0044	0 21.5703 43.1407 64.7110 86.2814 107.8517 129.4220 150.9924 172.5627 194.1331 215.7034	7 7 5 1 0 0 0 0 0 0 0	0 0 0 0 0 0 2 2 2 2 2

Average Cohort normalised score for M0	Cohort Normalised Scores for M0	Cohort Normalised Scores for Impostors	Threshold	FA (7 Max)	FR (5 Max)
			0	0	2
			0.2858	0	2
		F0 -296.8173	0.5716	0	2
	48.8897	F1 -499.8620	0.8575	0	2
	-11.1191	F2 -272.1557	1.1433	0	2
2.85822	13.9653	F3 -276.0006	1.4291	0	2
	7.0340	M1 -217.5887	1.7149	0	2
	-44.4788	M2 -136.9311	2.0008	0	2
		M3 -211.9389	2.2866	0	2
			2.5724	0	2
			2.8582	0	2

Average Cohort normalised score for M1	Cohort Normalised Scores for M1	Cohort Normalised Scores for Impostors	Threshold	FA (7 Max)	FR (5 Max)
			0	0	0
			5.6148	0	0
		F0 -194.1452	11.2296	0	0
	99.0288	F1 -203.9644	16.8444	0	0
	24.0260	F2 -143.7516	22.4592	0	0
56.1481	48.1572	F3 -138.9906	28.0741	0	1
	48.4501	M0 -83.7530	33.6889	0	1
	61.0784	M2 -81.2212	39.3037	0	1
		M3 -181.1984	44.9185	0	1
			50.5333	0	3
			56.1481	0	3

During the verification procedure, in order to calculate the log probability scores of each user, it's very important to multiply the feature vectors extracted from the test speech samples with the corresponding transform matrix of each user.

In the next figure, we can see that a OGMM system performs better than common GMM. Even in the case of MFCC, we can see the difference between the score levels for each user is much less, making it more facile to set a global threshold.

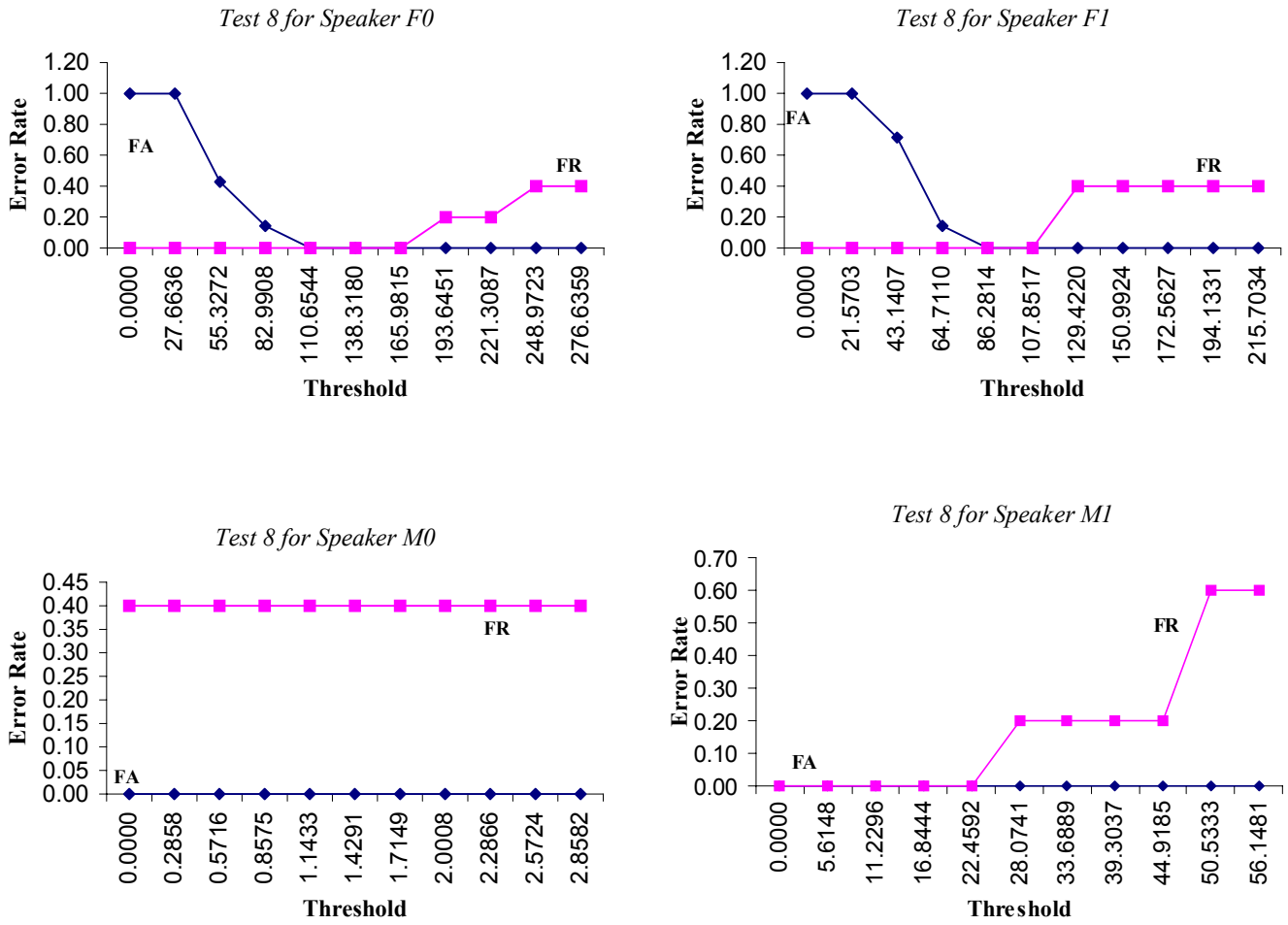


Figure 7.8: Using 12 MFCC coefficients and OGMM operation

7.12 Performance Evaluation

In order to study the performance of our system during the previous series of experiments, we are going to employ a statistics tool, called *Receiver Operator Characteristics (ROC) curves*. ROC curves are normally used to examine the performance of a diagnostic test over all possible decision thresholds, or to compare many alternative diagnostic tests so the best can be chosen. One basic field where ROC curves are used is clinical medicine. However, as we can see ROC curves match exactly our case, where we need to examine the performance of the previously described speaker verification configurations over all possible decision thresholds.

For our ROC curve analysis, we have processed the results gathered through the previous series of experiments in *Microsoft Excel 97*, using it to plot the corresponding ROC curves.

First of all, we are going to compare the performance of our system using MFCC, ΔMFCC and PLP coefficients. The figure 7.9 shows the ROC curve

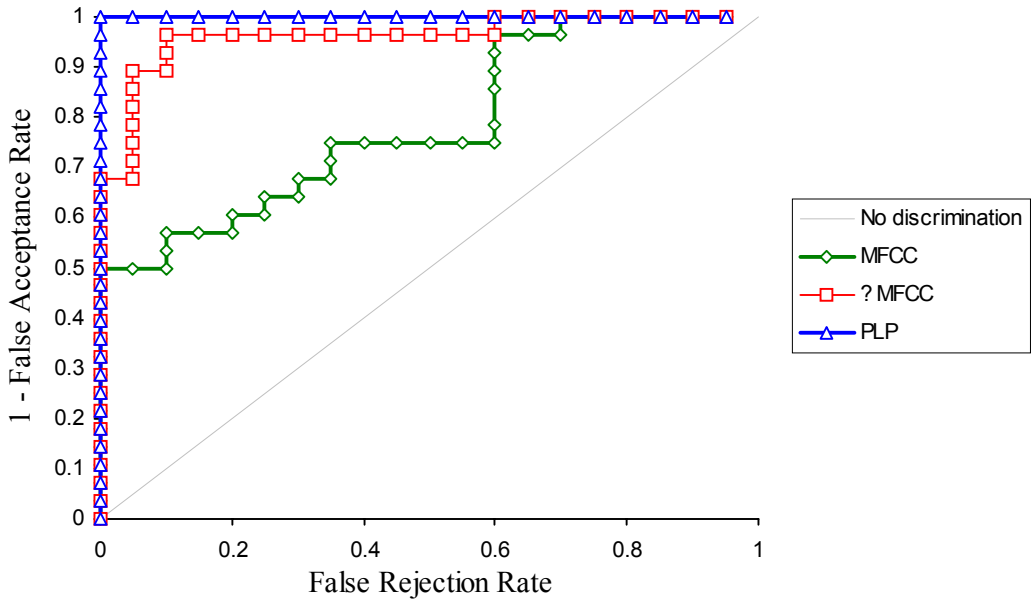


Figure 7.9: Comparing the performance of MFCC, ΔMFCC and PLP coefficients.

As we can see in the previous figure our system performs better with PLP coefficients than with every other configuration. Moreover, ΔMFCC coefficients are a better choice than MFCCs as it is much evident in the previous figure. Of course, the results presented can be more accurate, by adding more users to the system and taking more measures. The following tables contain a comparison of the three methods.

Curve	Area
MFCC	0.788
ΔMFCC	0.961
PLP	1.000

Contrast	Difference
MFCC v ΔMFCC	-0.173
MFCC v PLP	-0.213
ΔMFCC v PLP	-0.039

Figure 7.10 depicts the performance of a system that combines 2 kinds of feature sets, i.e. 12 PLP coefficients and 12 ΔMFCC coefficients. As we can see, the system performs betters with that configuration than with only 12 ΔMFCC. Nonetheless, a system with 12 PLP performs better. Therefore, we can see that the system combines some of the characteristics of the two feature sets, finally improving the performance.

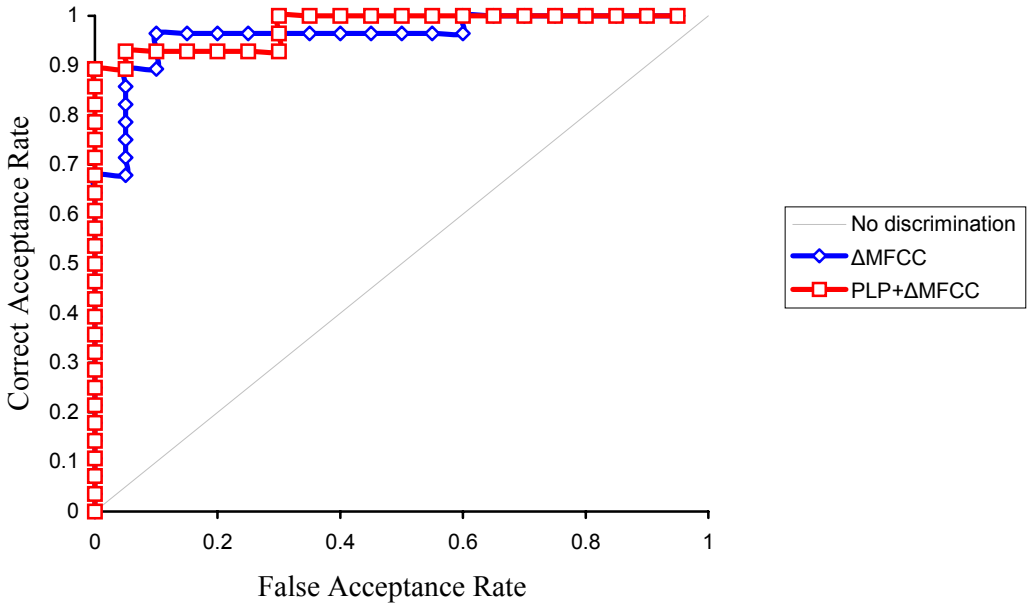
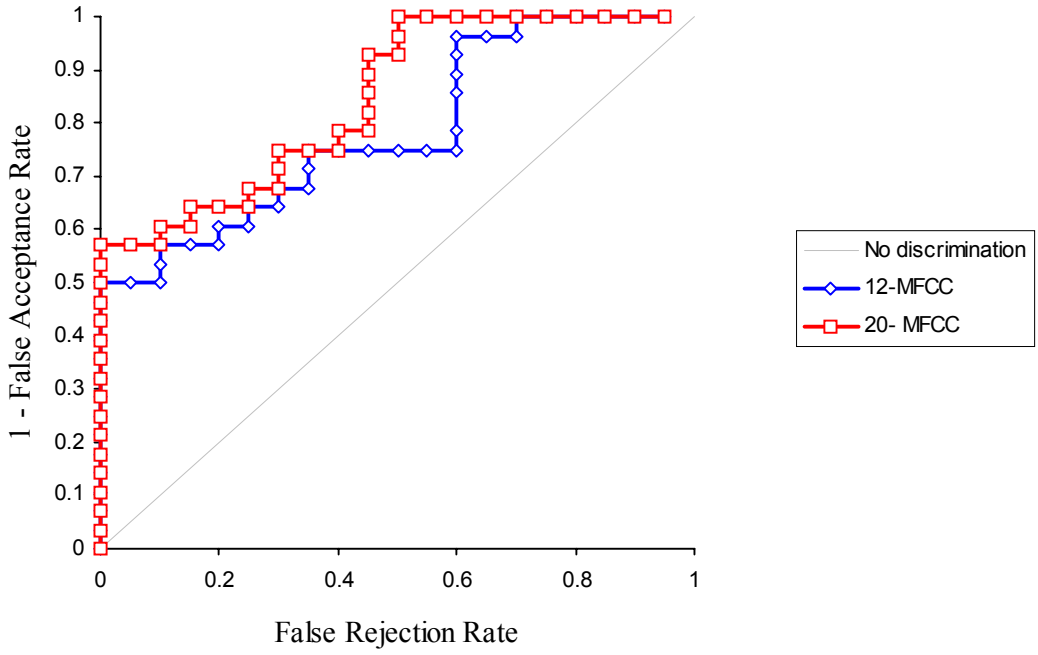


Figure 7.10: Comparing the performance of 12 ΔMFCC and a combination of 12 PLP and 12 ΔMFCC coefficients

The next figure depicts the performance of the system with 12 MFCC and with 20 MFCC. If we have a look at Figure 7.11, we can clearly see that using more coefficients of a feature set can enhance the performance of our system. This is mainly due to the fact that by increasing the order of the coefficients, the vocal tract is modelled more efficiently. The relationship between the system's performance and the order of the feature set can not be linear, implying that the performance of our system may not improve after a certain number of coefficients.



Curve	Area
12 MFCC	0.788
20 MFCC	0.846

Contrast	Difference
12 MFCC v 20 MFCC	-0.059

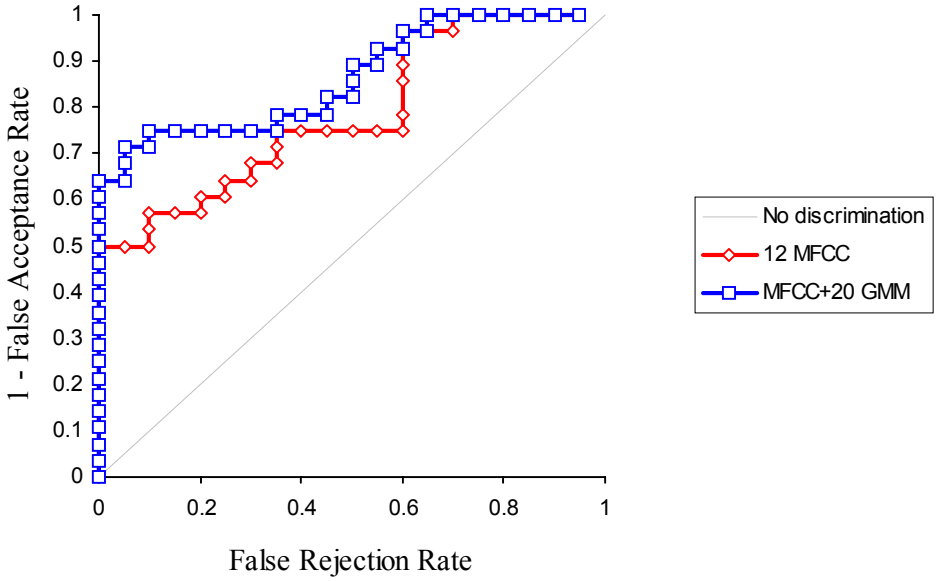


Figure 7.12: Comparing the performance of 12 MFCC and a system with 12 MFCC and 20 Gaussian Mixtures.

In Figure 7.12, we can see the effect of using more Gaussian mixtures on a system using MFCC coefficients. More specifically, we have used 20 Gaussian mixtures instead of the 10 Gaussian mixtures used in the other experiments. We can see that there is great improvement compared to the MFCC system. However, there is really no great improvement in the system's performance, after we exceed a certain number of Gaussian Mixtures [17].

Figure 7.13 just exhibits the performance of the system using a database of 6 male users and 12 MFCC coefficients. This kind of experiments reflects the real performance as the inter-gender tests sometimes mislead the verification system. It is more difficult for the system to distinguish between users of the same sex, as they possess similar voice characteristics. Moreover, the term impostor when referring to a naturally speaking member of the opposite sex from the claimed user is a little bit ambiguous in real life.

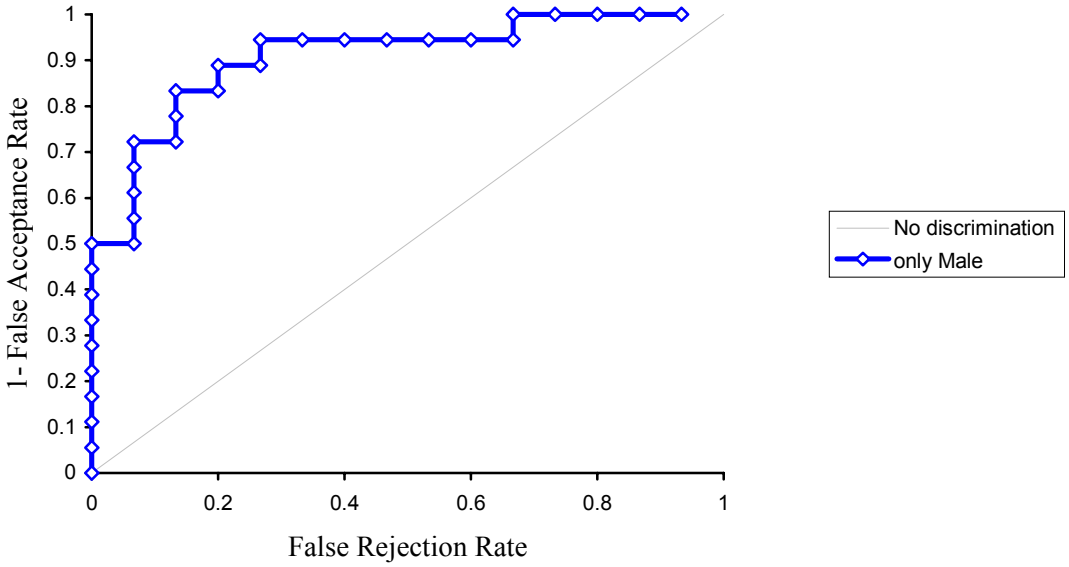


Figure 7.13: The performance of 12 MFCC coefficients using 6 male speakers

In Figure 7.14, we can see the performance of an OGMM system. We can clearly see that the decorrelation of the input feature vectors, using a linear transform, boosts the performance of the system. Practically, the introduction of decorrelation in the feature extraction module adds a very small delay in the total processing time of the programme.

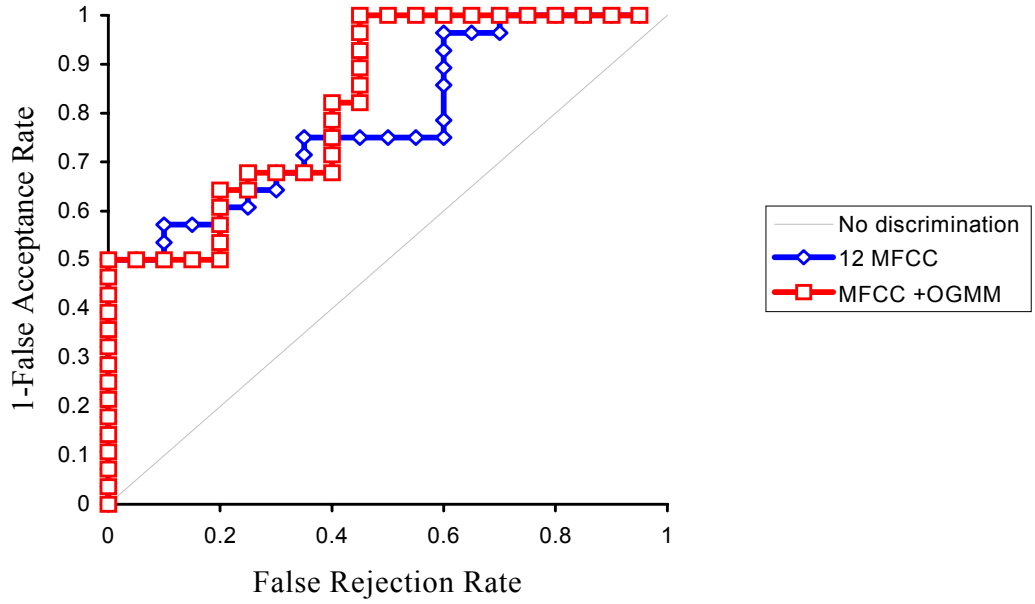


Figure 7.14: The performance of OGMM-MFCC compared to MFCC

However, this method requires storing the transform matrix for each user. This increases the storage needed for every user, but the extra storage space needed is insignificant.

Curve	Area
12 MFCC	0.788
MFCC +OGMM	0.825

In the table presented above, we can see the difference between the two methods. The area covered by the MFCC-OGMM is greater than that of the MFCC case. Generally, by using OGMM, we can achieve the same performance as a common GMM system that employs more mixtures.

CHAPTER 8

CONCLUSION AND PROPOSALS

8.1 Conclusion

In this chapter, we will try to summarise the basic accomplishments in this project and point out some basic observations that we think were crucial for speaker verification systems development.

First of all, we have discussed several practical and social aspects of biometrics identification and pointed out that a speaker verification system can have a wide range of real life application as well as very good performance, despite the great variability of human voice. Moreover, we have discussed the structure of a speaker verification system both in the enrolment and the verification operation:

$$\text{Preprocessing} \rightarrow \text{Feature Extraction} \rightarrow \text{Speaker Modelling}$$

We have studied and implemented a series of algorithms that are used for preprocessing and Feature Extraction and finally, we investigated and implemented the Gaussian Mixture Modelling technique giving a small reference to other common speaker modelling techniques. To facilitate the whole experimentation on speaker verification, we have built a Graphical User Interface in MATLAB, where we can configure our speaker verification system and then perform a series of experiments enrolling and verifying the identity of several users. Once setting up a configuration, the GUI resemble a commercial Speaker verification system, granting or rejecting authorisation to users.

In order to evaluate the system, we have set up a series of eight experiments, using a database of 8 people (4 male and 4 female) and several configurations. The speech samples were taken from the TIMIT database and each user uses 5 phrases to train the system. On the whole, we can say that the system works very efficiently, with small EER rates, regardless of the configuration used. This is mainly due to the fact that we are using

almost perfect recordings, belonging to a standards database. If we had used real life recordings, i.e. in an office environment, with constant background noise, then the performance of our system would have deteriorated.

However, we have seen that PLP coefficients have the best performance for a speaker verification system. The score presented may be too ideal, but even in real life situation they are bound to have a very good performance. We have seen that scores of the four users are in the same level and the system can easily apply a proper global threshold according to which the system can detect the true user from his impostors. The good performance of PLP coefficients is justified because they feature small correlation between them. This is very important for Gaussian Mixture Modelling, where we suppose that the covariance matrix of each Gaussian Muxture is diagonal.

The performance of MFCC and ΔMFCC coefficients is equally very good, with a drawback of MFCC giving different cohort score levels for each user, making it difficult to set a global error-free threshold.

In test 4, we have seen that by using combinations of different feature sets, we can combine their properties and improve the system's performance.

In test 6, we have seen that by increasing the order of a feature set, we can enhance the system's performance. Using 20 MFCCs instead of 10 MFCCs, the system operates more efficiently.

Increasing the number of Gaussian Mixtures can also improve the performance of the system. However, after a certain number of Gaussian mixtures, the system tends to have the same performance.

Moreover, we have seen the effective modelling performed by Gaussian Mixture Models. By using very few mixtures and a relatively simple to understand and implement training algorithm (compared to HMM), we can produce speaker models very quickly and easily. We may need to increase the number of mixtures, when using with a bigger speaker database.

We have also seen that by decorrelating the input feature vectors, we can improve the performance of a common GMM system. OGMM systems introduce a little more processing to the modelling algorithm, in order to calculate the transform matrix, but the performance of the system is much enhanced.

One very important issue is the choice of the test words and phrases that will be used to train and test the system. This is equally important in an almost text-independent system like the one described in this thesis. Over these tests, we have seen that there are certain speech files that tend to produce a low score, regardless of the type of coefficients used. This implies that the choice for the phrase may not be very good. Generally, phrases and words containing many voiced phonemes are much more preferable.

Finally, the development of a library of MATLAB routines as well as a Graphical User Interface for testing and evaluating speaker verification algorithms was the outcome of this project. Hoping that this infrastructure can be used as a starting point for further research on speaker verification, we suggest some possible extensions of this project.

8.2 Proposals for Future Work

The design of a graphically configurable speaker verification system lights the way for a wide range of possible tests and experiments that could not be conducted during the short period of this project.

Using the already developed infrastructure one can conduct several series of experiments in order to explore the effect of the following on the system's performance:

- The length of each frame.
- The overlapping ratio in segmentation.
- Combinations of several feature types, for example MFCC and Reflection Coefficients etc.
- Cohort size

One can also look into the extraction of RASTA-PLP coefficients from input speech samples, as an extra feature set, that can be integrated in the speaker verification GUI [16].

Moreover, more speaker modelling techniques, such as *Vector Quantisation*, *Hidden Markov Models*, *Neural Networks* can be added to the GUI, so as to enable further experimentation.

Furthermore, if we are to use a speaker verification system in a real life environment, implying that we are not going to use sample speech recordings from a speech database, then maybe we will have to deal with the possible background noise. Several speech enhancement techniques have been proposed for use in speaker verification systems, such as using spectral subtraction techniques or even RASTA filtering [30] and one can experiment with some of them.

REFERENCES

[1] J.R. Deller Jr, J.G. Proakis, John H.L. Hansen, *Discrete-Time Processing of Speech Signals,* Macmillan, 1993

[2] L.R. Rabiner, B.H. Juang, *Fundamentals of Speech Recognition*, Prentice Hall, 1993

[3] L.R. Rabiner, R.W. Schafer, *Digital Processing of Speech Signals*, Prentice Hall, 1978

[4] MATLAB, *Building GUIs with MATLAB version 5*, The MathsWorks Inc., 1997

[5] MATLAB, *Application Program Interface Guide version 5*, The MathsWorks Inc., 1997

[6] Jayant M. Naik, *Speaker Verification: A Tutorial*, IEEE communications magazine, Jan 1990, pp. 42-48

[7] Daniel Jurafsky, James H. Martin, *Speech and Language Processing*, Prentice Hall, 2000

[8] P. J. Philips, A. Martin, C.L. Wilson, M. Przybocki, *An introduction to evaluating Biometric systems*, IEEE Computer, Feb. 2000, pp. 56-63

[9] Sh. Pankanti, R.M. Bolle, An. Jain, *Biometrics: The future of Identification*, IEEE Computer, Feb. 2000, pp. 46-49

[10] L.R. Rabiner, M.R. Sambur, *An algorithm for determining the endpoints of isolated utterances*, Bell System Technical Journal, Vol. 54, pp. 297-315, February 1975

[11] J. Taboada, S. Feijoo,R. Balsa,C. Hernandez, *Explicit estimation of speech boundaries*, IEE Proc.-Sci. Meas. Technol., vol. 141, No.3, May 1994

[12] H. Qiang, Z. Youwei, *On Prefiltering and endpoint detection of speech signal*, Proceedings of ICSP '98, IEEE 1998, pp. 749, 752

[13] D. Etter, B. Bradley, M. Dickeson, *Development and implementation of speaker verification algorithms*, Proceedings of the ICASSP 1997, IEEE 1997, pp. 753-757

[14] J. Makhoul, *Linear Prediction: A tutorial review*, Proceedings of the IEEE, vol. 63, no.4, April 1975

[15] H. Hermansky, *Perceptual Linear Predictive (PLP) analysis of speech*, J. Acoust. Soc. Amer., pp.1738-1752, 1990

[16] H. Hermansky, N. Morgan, *RASTA Processing of Speech*, IEEE transactions on speech and audio processing, vol. 2, no.4, October 1994

[17] D.A. Reynolds, R.C. Rose, *Robust Text-Independent Speaker Identification using Gaussian Mixture Speaker Models*, IEEE transactions on Speech and Audio Processing, Vol.3, No. 1, January 1995.

[18] L. Liu, J. He, *On the use of orthogonal GMM in Speaker Recognition*, Proceedings of the ICASSP 1999, IEEE,1999, pp. 845, 848

[19] C.W. Che, Q. Lin, D.S. Yuk, *An HMM approach to text-prompted speaker verification*, Proceedings of the ICASSP 1996, IEEE 1996, pp. 673-676

[20] D.M. Brookes, *Lecture notes on Speech Processing,* Imperial College, Spring 2000

[21] W.C. Ho, *Text-dependent Speaker Verification using HMM*, MEng Thesis, Imperial College, 1996

[22] Jayant M. Naik, David M. Lubensky, *A Hubrid HMM-MLP Speaker Verification Algorithm for telephone Speech,* Proceedings of the ICASSP 1994, IEEE 1994, pp. I-153, I-156

[23] W.J.J. Roberts, J.P. Willmore, *Automatic Speaker Recognition using Gaussian Mixture Models*, Defence Science and Technology Organisation Salisbury, South Australia, 1999

[24] B.L Pellom, J.H.L. Hansen, *An experimental study of speaker verification sensitivity to computer voice-altered imposters*, Proceedings of the ICASSP 1999, IEEE 1999, pp. 837, 840

[25] K.K. Ang, A.C. Kot, *Speaker Verification for home security system*, Proceedings of the ICASSP 1997, IEEE 1997, pp. 27, 30

[26] R.W. Frischolz, U. Dieckmann, BioID: A Multimodal Biometric Identification System, IEEE Computer, Feb. 2000, pp. 64-68

[27] A. Azemi, E. E. Yaz, *Using Graphical User Interface Capabilities of MATLAB in Advanced Electrical Engineering Courses*, Proceedings of the 32th Conference on Decision & Control, Phoenix, 1999

[28] J. Gonzalez- Rodriguez, S. Cruz-Llanas, J. Ortega-Garcia, *Biometric identification through speaker verification over telephone lines*, Proceedings of the ICASSP 1999, IEEE 1999, pp. 238, 242

[29] J. Picone, *Signal Modeling Techniques In Speech Recognition*, Proceedings of the IEEE, IEEE 1993,

[30] D. Hardt, K. Fellbaum, *Spectral Subtraction and RASTA-filtering in text-dependent HMM-based speaker verification*, Proceedings of the ICASSP 1997, IEEE 1997, pp. 867-870