University of Southern Queensland

Faculty of Engineering & Surveying

Speaker Identification - Prototype Development and Performance

A dissertation submitted by

David Michael Graeme Watts

in fulfilment of the requirements of

ENG4111/ENG4112 Research Project

towards the degree of

Bachelor of Engineering (Computer Systems)/Bachelor of Information Technology (Applied Computer Science)

Submitted: November, 2006

Abstract

Human speech is our most natural form of communication and conveys both meaning and identity. The identity of a speaker can be determined from the information contained in the speech signal through speaker identification.

Speaker identification is concerned with identifying unknown speakers from a database of speaker models previously enrolled in the system. The general process of speaker identification involves two stages. The first stage extracts features from speakers that are to be enrolled into the system. The second stage involves processing the identity of a speaker using features extracted from the speech and comparing these to the speaker models.

Several techniques available for feature extraction including Linear Predictive Coding (LPC), Mel-Frequency Cepstral Coefficients and LPC Cepstral coefficients. These features are used with a classification technique to create a speaker model. Vector Quantization is commonly used in speaker identification producing reliable results.

This project demonstrates a prototype speaker identification system tailored for utterances containing less than ten words and target sets of less than eight voice profiles.

VQ (codebook size = 128) with 20-dimension LPCC obtain accuracy results of 83% and 100% using 12 speakers with the NTIMIT and Alternative (own) corpus, respectively. Tests were conducted using 30s of training speech and 3s of testing speech.

Certification of Dissertation

I certify that the ideas, designs and experimental work, results, analyses and conclusions set out in this dissertation are entirely my own effort, except where otherwise indicated and acknowledged.
I further certify that the work is original and has not been previously submitted for assessment in any other course or institution, except where specifically stated.

David Michael Graeme Watts

Q12215217

Signature

Date

Acknowledgments

Firstly, I would like thank my supervisor, Associate Professor John Lies for his guidance through the duration of this project. Many thanks go to my family for supporting me in many ways throughout my time at university. I would also like to thank my friends Nick, Ben, Andrew and Troy for their help on this project. Finally a special thanks to Catherine and Nathan for their continued support, not only over the duration of this project, but for my entire university life.

David Michael Graeme Watts

University of Southern Queensland
November 2006

Dedicated to my mother, Alana Alison Watts

Contents

Abstract **i**

Acknowledgments **iv**

List of Figures **xii**

List of Tables **xv**

Chapter 1 Introduction **1**

1.1 Purpose 1

1.2 Overview of Speaker Recognition 3

1.3 Speech Analysis 4

1.4 General Process of Speaker Identification 7

1.5 Practical Implications 9

1.6 Applications of Speaker Identification 9

1.7 Speaker Corpus 10

1.8 Principle Objectives of Dissertation 11

1.9 Overview of Dissertation 11

Chapter 2 Literature Review 13

2.1 Introduction 13

2.2 Corpora for the Evaluation of Speaker Recognition Systems 13

2.3 Overview of Automatic Speaker Identification Results 15

2.4 Speaker Identification using Vector Quantization 15

2.5 Speaker Identification using NTIMIT 17

2.6 Speaker Identification For Large Set of Voices 18

2.7 Effects of Utterance Length on Speaker Identification 19

2.8 Chapter Summary 19

Chapter 3 Speaker Recognition 21

3.1 Introduction 21

3.2 Pre-Processing 21

3.2.1 DC Offset Removal 22

3.2.2 Silence Removal 22

3.2.3 Pre-emphasis 24

3.2.4 Frames and Windowing 25

3.2.5 Autocorrelation 26

3.3 Source-Filter Model of Speech 28

3.4 Speaker Modelling 30

3.4.1 Vector Quantization 30

3.4.2 Gaussian Mixture Models 31

3.4.3 Support Vector Machine 31

3.5 Distance Metrics 31

3.6 Chapter Summary 33

Chapter 4 Feature Extraction **35**

4.1 Introduction 35

4.2 Short-Term Fourier Transform 36

4.3 Cepstrum 37

4.3.1 Mel-Frequency Cepstrum Coefficients 39

4.4 Linear Prediction 40

4.5 Linear Predictive Cepstral Coefficients 44

4.6 Normalisation 44

4.7 Chapter Summary 45

Chapter 5 Speaker Modelling **46**

5.1 Introduction 46

5.2 Vector Quantization 46

5.2.1 Iterative Clustering Methods 51

5.2.2 Initial Codebook 51

5.2.3 Generalized Lloyd Algorithm (k-means) 51

5.2.4 Randomized Local Search 52

5.2.5 Hierarchical Clustering Methods 53

5.2.6 Pairwise Nearest Neighbor 53

5.2.7 Split 54

5.2.8 VQ Distortion 55

5.3 Chapter Summary 55

Chapter 6 Experimental Setup **57**

6.1 Introduction 57

6.2 Speech Corpora 57

6.2.1 NTIMIT 57

6.2.2 Alternative Corpus 58

6.3 Testing Procedures 59

6.3.1 Pre-processing 59

6.3.2 Feature Extraction 60

6.3.3 Speaker Model 60

6.3.4 Speech Corpus 60

6.4 Prototype 61

Chapter 7 Results 62

7.1 Results of Feature Extraction Methods 62

7.1.1 Linear Predictive Coefficients 63

7.1.2 Mel-Frequency Cepstral Coefficients 66

7.1.3 Linear Predictive Cepstral Coefficeients 70

7.1.4 Comparison of Feature Extraction Methods 73

7.1.5 Effects of Accuracy Results by Increasing Speakers 75

7.2 Chapter Summary 76

Chapter 8 Conclusion 77

8.1 Summary of Work 77

8.2 Conclusions 78

8.3 Further Work 78

References 80

Appendix A Project Specification 83

Appendix B Alternative Speaker Corpus 85

B.1 Speech 86

Appendix C Source Code 87

C.1 The AlternativeTest.m MATLAB Function 88

C.2 The timitPro.m MATLAB Function 89

C.3 The increaseSpeakers.m MATLAB Function 92

C.4 The kmgla.m MATLAB Function 98

C.5 The mfcc.m MATLAB Function 98

C.6 The lpcc.m MATLAB Function 99

C.7 The eucDist.m MATLAB Function 99

C.8 The mfcc.m MATLAB Function 100

C.9 The dist.m MATLAB Function 100

List of Figures

1.1 FOCAL meeting environment (Littlefield & Broughton 2005). 1

1.2 System design of a dual-type ASR (Littlefield & Broughton 2005). . . . 2

1.3 Categories of speaker recognition (Karpov 2003). 4

1.4 The human speech production systems (Johnson 2005). 5

1.5 Voiced (“a”) and Unvoiced Speech (“s”) 6

1.6 The spectral peaks, or formants with the original spectrum 6

1.7 The spectral peaks, or formants of the spectrum of a speech frame . . . 7

1.8 Speaker Identification Process (Willits 2003). 8

3.1 Silence Parts of the Speech Signal 23

3.2 Voice Degree Detector (Drygajlo 2005). 24

3.3 Frequency response of a pre-emphasis filter with a = 0.95 25

3.4 Frame without Pre-emphasis 26

3.5 Frame with Pre-emphasis 26

3.6 Hamming window applied to a framed speech signal 27

3.7 The process of extracting features from the speech signal (Willits 2003). 29

3.8 Euclidean and Manhattan metric in R^2 32

4.1 Source Filter Spectrum 36

4.2 Source Filter Spectrum (Karpov 2003) 38

4.3 Cepstrum 39

4.4 Relationship between the mel-frequency and the physical frequency . . 40

4.5 Mel Frequency Filter Bank 41

4.6 Source-filter relationship of LPC when used to synthesis speech. Source: *http://www.usq.edu.au/users/leis/courses/ELE4607/module8.pdf* . . . 42

4.7 LPC spectrum window, and STFT 44

5.1 Vector Quantization using two speakers (Do 2003). 47

5.2 Representation of a codebook (k=32) using 20 dimensional LPC reconstructed code vectors 48

5.3 Pairwise Nearest Neighbor VQ 54

5.4 Pairwise Nearest Neighbour VQ 55

7.1 Performance of the LPC on the NTIMIT and Alternative corpus for codebook size (k=32) 63

7.2 Performance of the LPC on the NTIMIT and Alternative corpus for codebook size (k=64) 64

7.3 Performance of the LPC on the NTIMIT and Alternative corpus for codebook size (k=128) 65

7.4 Performance of the LPC on the NTIMIT and Alternative corpus for codebook size (k=256) 65

7.5 Performance of the MFCC on the NTIMIT and Alternative corpus for codebook size (k=32) 67

7.6 Performance of the MFCC on the NTIMIT and Alternative corpus for codebook size (k=64) 67

7.7 Performance of the MFCC on the NTIMIT and Alternative corpus for codebook size (k=128) 68

7.8 Performance of the MFCC on the NTIMIT and Alternative corpus for codebook size (k=256) 69

7.9 Performance of the LPCC on the NTIMIT and alternative corpus for codebook size (k=32) 70

7.10 Performance of the LPCC on the NTIMIT and alternative corpus for codebook size (k=64) 71

7.11 Performance of the LPCC on the NTIMIT and alternative corpus for codebook size (k=128) 72

7.12 Performance of the LPCC on the NTIMIT and alternative corpus for codebook size (k=256) 72

7.13 Comparison of Accuracy and Coefficient Type using NTIMIT corpus . . 73

7.14 Comparison of Accuracy and Coefficient Type using Alternative corpus 74

7.15 The effect of increasing speakers on the performance of a VQ(codebook 128) with 20 dimensional LPCC 75

List of Tables

1.1 Factors affecting Speaker Identification 9

1.2 NTIMIT Corpus Description 10

1.3 Speakers used in alternative database 10

2.1 Performance of baseline system (TIMIT) 16

2.2 Performance of baseline system (NIST) 17

2.3 Impact of of the number of speakers on identification results 18

2.4 Accuracy results for different length speech input data 19

6.1 Summary of Corpora 59

Chapter 1

Introduction

1.1 Purpose

Speech recognition is currently an area of research and development by the Defence Science and Technology Organisation (DSTO). The DSTO is currently exploring the feasibility of a dual-type Automated Speech Recognition (ASR) system. This project combines both speaker-dependent and speaker-independent ASR systems, to overcome the shortcomings in each ASR type (Littlefield & Broughton 2005). The ultimate goal is to provide a system that is more robust and flexible than current ASR solutions.

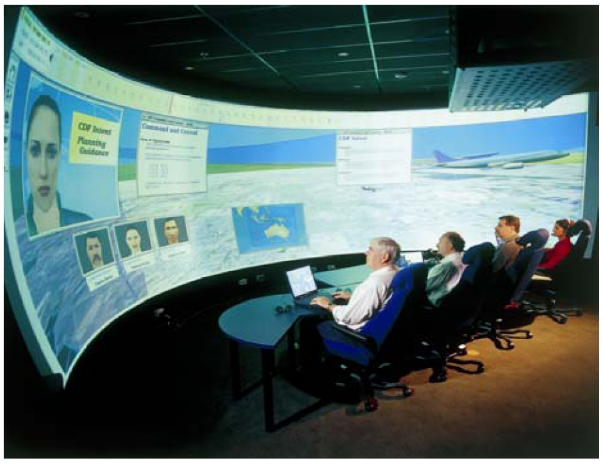

Figure 1.1: FOCAL meeting environment (Littlefield & Broughton 2005).

Within the proposed designs of the dual-type ASR system, the speaker-dependent component requires the use of a speaker identification tool. This tool will be used to load the corresponding user ASR profile of the identified speaker. Therefore this project aims to explore the technology of speaker recognition, specifically speaker identification, to research and ultimately develop a prototype speaker identification system. This prototype is to be tailored for utterances containing less than ten words and should be able to work from target sets of less than eight voice profiles.

Figure 1.1 illustrates a possible meeting environment the dual-type ASR system will be used in. Figure 1.2 shows one proposed design of the dual-type ASR system.

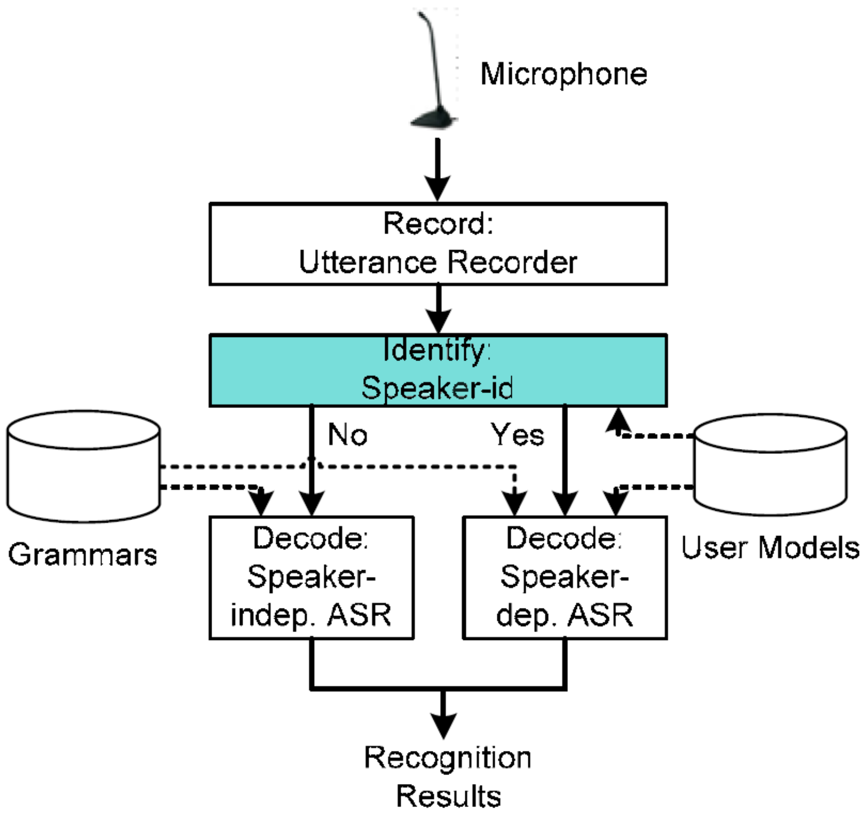

Figure 1.2: System design of a dual-type ASR (Littlefield & Broughton 2005).

Speech and speaker recognition have recently become an important research and development area. The driving factor behind much of the development is the desire to produce a natural form of communication between human and machine. Since speech is our most natural form of communication, it has the capability to impact on countless fields of research and development.

1.2 Overview of Speaker Recognition

Speaker recognition is the process of identifying a person, based on the physiological information contained in their speech. It differs from speech recognition in that the speaker is identified not the words (as in speech recognition). Speech recognition and speaker recognition are closely related topics and many of the same techniques are employed to extract information from the speech signal.

Speaker recognition uses the acoustic features of the speech signal to discriminate between individuals. These acoustic features can vary greatly from one speaker to another depending upon their anatomy and behavioural characteristics. Modelling these acoustic features is useful in speaker recognition, as they can be used to identify individuals.

Implementation of a speaker recognition system requires the human speech content to convey meaning to a machine. The human voice consists of sounds that are characterised by the behaviour and physiology of the individual. For instance, utterances produced by an individual are from the same vocal tract, and tend to have a typical pitch range, along with the characteristics associated with dialect or gender. This results in a highly correlated speech signal for any particular speaker.

The speech signal is a complex and variable signal that consists of many different harmonic components. The speech signal is the result of the vocal cords being excited by air flow from the lungs, which is then filtered through the vocal tract. The vocal tract then acts in varying ways to filter and ultimately create the collection of sounds perceived as speech (Johnson 2005).

Speaker recognition can be further broken into two categories: speaker identification and speaker verification. Identification takes the speech signal from an unknown speaker and compares this with a set of valid users. The best match is then used to identify the unknown speaker. Similarly, in verification the unknown speaker first claims identity, and the claimed model is then used for identification. If the match is above a predefined threshold, the identity is accepted. The fundamental difference between the two modes is the number of decision alternatives.

Speaker identification can be classified into two types, based on anonymity. These are open-set and closed-set speaker identification. Both sets use a database of registered speakers for identification, with the main difference being in the decision process. For open-set the decision is based upon the enrolled speakers together with the possibility that the speaker is unknown. Closed-set only considers the best match from the enrolled speakers.

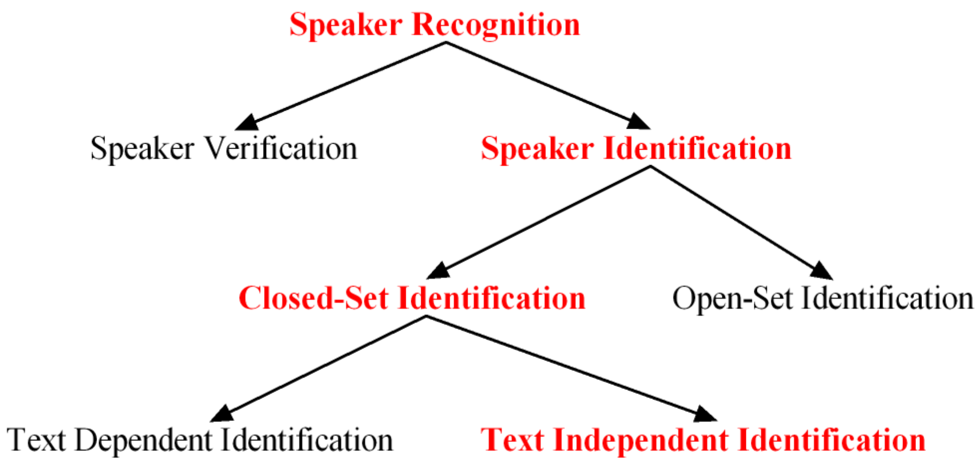
Speaker Recognition
Speaker Verification
Speaker Identification
Closed-Set Identification
Open-Set Identification
Text Dependent Identification
Text Independent Identification

Figure 1.3: Categories of speaker recognition (Karpov 2003).

Closed-set speaker identification can be further classified into text-dependent and text-independent. Text-dependent systems rely on prior knowledge of the text spoken by the individual for identification (Karpov 2003). This can be in the form of a phrase prompted by the system or a user specific phrase. Text-independent must be capable of recognising a speaker, without prior knowledge of the text (Karpov 2003). Text-independent systems are considerably more fixable than text-dependent, however they tend to produce lower accuracy levels. The varying categories of speaker recognition are shown in Figure 1.3.

1.3 Speech Analysis

In order to analyse and exploit characteristics of speech production, a method must be developed to describe and model the process. This involves determining how each sub-system of speech production produces its output. Consequently speech can be broken into two parts: (a) a sound source produced by the vibration of the vocal cords in response to air pressure from the lungs, and (b) a filter through the vocal tract, which creates acoustic disturbances. The vocal cords produce a periodic pulse, known as the pitch or the fundamental frequency of the speech signal (Johnson 2005). The vocal tract then serves to shape the spectrum produced by the vocal cords, acting as a filter. The vocal tract can be considered the Nasal Cavity, Lips, Tongue, Teeth and Oral Cavity. Figure 1.4 shows the human speech production system.

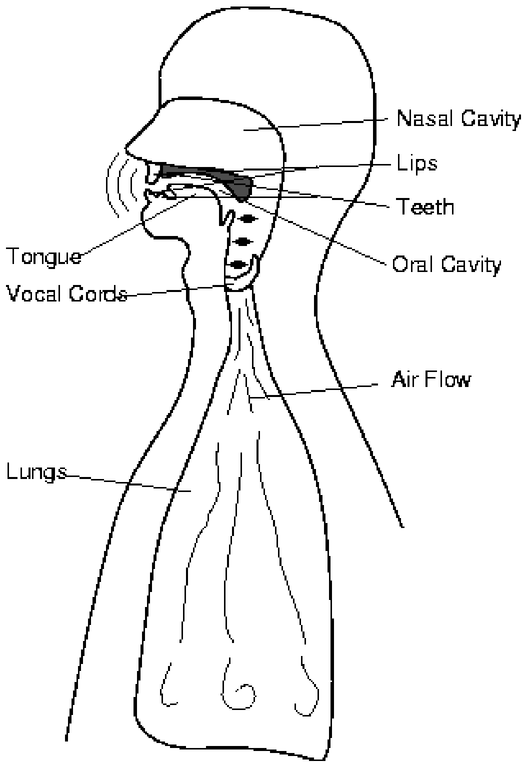

Figure 1.4: The human speech production systems (Johnson 2005).

The speech signal can be further broken into a number of categories based on the connection between the vocal cords and vocal tract. Voiced and unvoiced are the two main sounds that are produced, which also can be combined together. Voiced sounds are those predominately produced by the vocal cords, which are placed under tension to produce sounds like “a” and “u”. For unvoiced sounds the vocal cords are placed under much less tension, this results in sounds that are more noise like, and represent sounds like “s” and “f”.

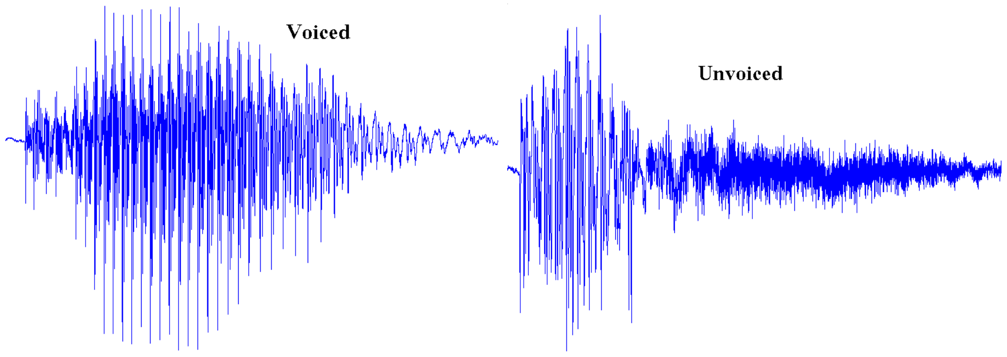

Figure 1.5: Voiced ("a") and Unvoiced Speech ("s")

Figure 1.6 illustrates the spectrum of speech sounds in three dimensions using a spectrogram. The top of Figure 1.6 shows the speech signal in the time domain, with the corresponding spectrogram below. In relation to the spectrogram, the horizontal axis is time, vertical axis is frequency and the amplitude is illustrated though shades of darkness.

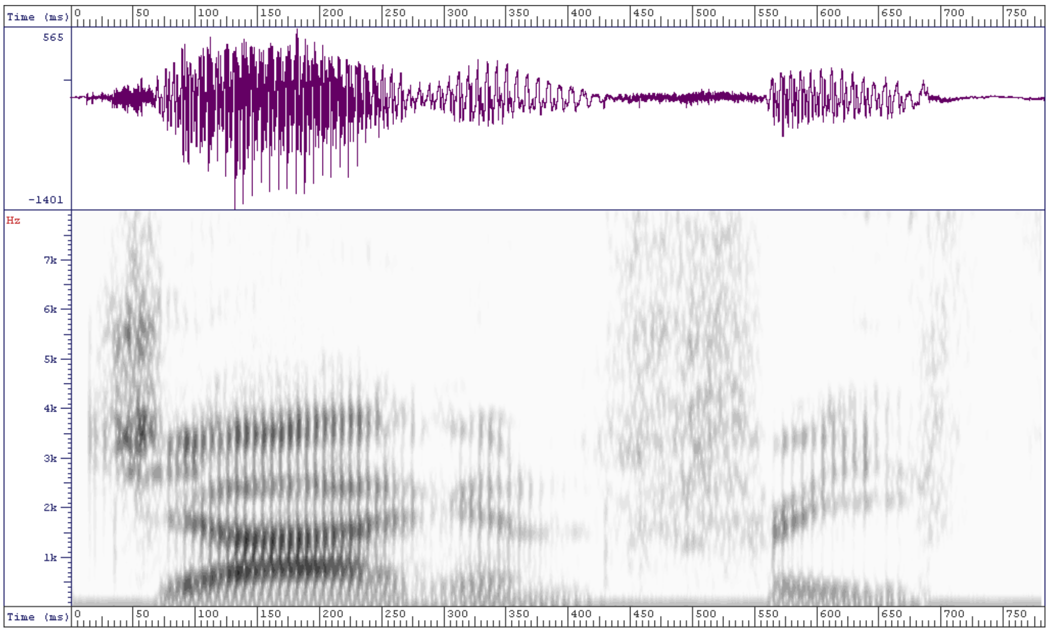

Figure 1.6: The spectral peaks, or formants with the original spectrum

The voiced and unvoiced sounds can also be seen in Figure 1.6 in both the time-domain and the spectrogram. The voiced part of the speech is evident in the time-domain and is also characterised by spectrum highs (darker areas) in the spectrogram. Unvoiced sounds are those where the frequency is slightly higher and noise-like in its appearance.

The vocal-cord-lung system can be described as the source which is used to produce a periodic pulse signal that can be then shaped by a filter. As the periodic acoustic signal passes through the vocal tract, its frequency content is altered by the resonances (Campbell 1999). The vocal tract serves to shape the spectrum of the signal from the vocal cords. Resonances produced by the vocal tract are called *formants* and can be seen in the spectrum of Figure 1.7. These formants can also be seen in the spectrogram (Figure 1.6) as the highs or dark spots and form *horizontal frequency bands* across the spectrum. These horizontal frequency bands illustrate the effect of the vocal tract over a period of time. Most speaker recognition systems use features extracted from these formants.

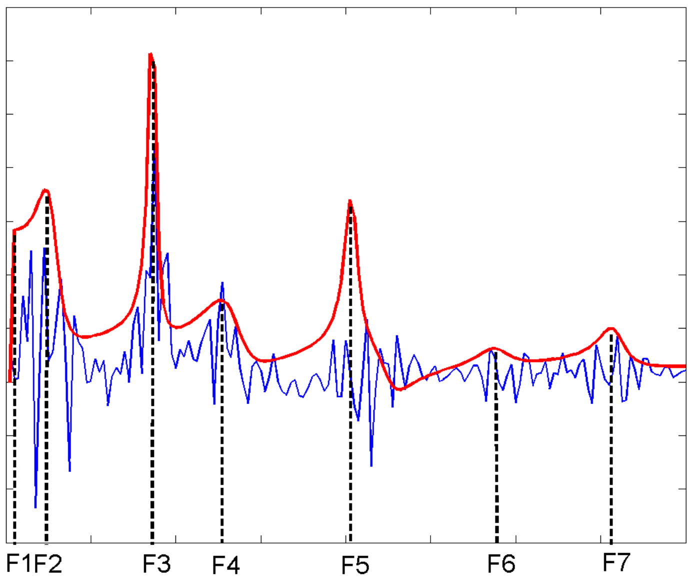

Figure 1.7: The spectral peaks, or formants of the spectrum of a speech frame

1.4 General Process of Speaker Identification

The general process of speaker identification involves two stages. The first stage is to enrol the speakers into the system. Enrolment involves determining distinct characteristics of the speaker's voice, to be used as a source in the modelling process. Speaker models are then created for each of the speakers and stored in a database. The second stage involves the identification of a speaker. Similar to the enrolment stage, this involves extracting distinct features from an unknown speaker to compare with the speaker database. The enrolment and identification processes are very similar, and both require distinct features to be extracted from the speech signal. The identification process depends on the modelling procedure used in the enrolment stage.

In order to construct a speaker identification system, there are two important aspects of the process that require further investigation; namely: feature extraction and classification. Both of these stages have a critical effect on the identification result. *Feature extraction* is the process of extracting distinct characteristics from the speech of an individual. *Classification* refers to the process of determining a speaker based upon previously stored models or information.

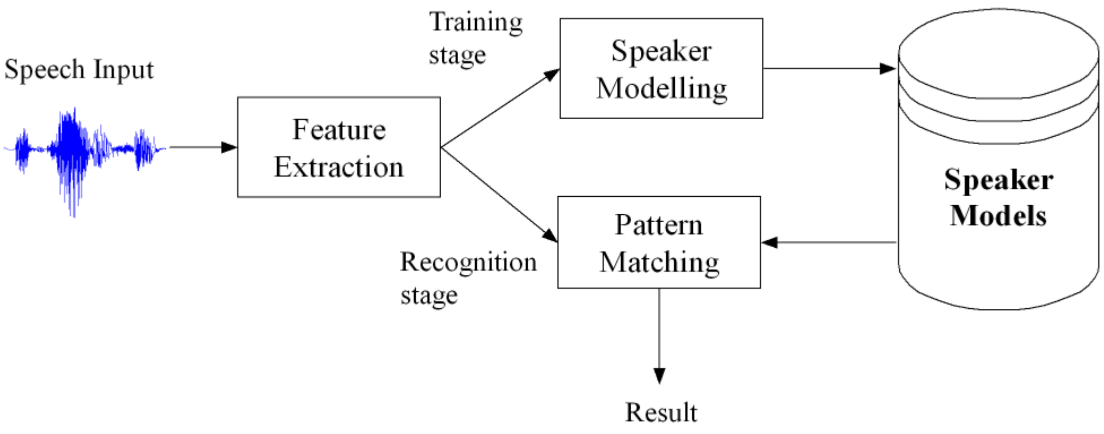

Figure 1.8: Speaker Identification Process (Willits 2003).

Pattern matching and speaker modelling are techniques used to classify and enrol speakers to an identification system. Speaker modelling constructs a model of an individual's voice based upon the features extracted from their speech signal. This is completed when speakers are enrolled in the speaker identification system to produce a database of registered speakers. This occurs through a training stage in which the system creates the speaker model. Pattern matching uses the models in the speaker database to calculate a matching score for each model. The final result is a measure of the similarity between the features extracted from the unknown speech signal and each of the models in the speaker database. Figure 1.8 illustrates how these components connect together.

1.5 Practical Implications

Practical implications of using a speaker identification system have to be assessed and if necessary, designed for. These include any factors that will affect the speaker identification results (Table 1.1).

Table 1.1: Factors affecting Speaker Identification

Situational Factors	*Environment (eg. room acoustics)*
	Quality of microphones
	Emotional state of speaker
	Computing power
Artifacts of the Speech Signal	*Silent parts of speech*
	Difference between male & female speakers
	Overall signal energy

Figure 1.1 illustrates a typical environment in which the prototype will could be employed. The dual-type ASR will also be used in a number of environments similar in design and acoustic properties. The type of speech recordings in these meeting environments will be of high quality, as the recording will also be need for speech recognition.

1.6 Applications of Speaker Identification

Speaker identification has many applications, from security to forensics and has the potential to impact upon many other areas. This technology can be used by financial institutions to protect accounts or as security defence for businesses. However an important aspect of speaker identification is its susceptibility to fraud and as such it should be used with other, existing measures of security (e.g. using speaker identification with PIN for an ATM withdrawal).

Many other areas exist where security is not of paramount importance, such as use in a meeting environment. Others include incorporating speaker identification into already existing systems that may use personalised greetings. The identification result can also be used to personalise and configure applications for particular individuals.

1.7 Speaker Corpus

Speaker recognition systems are normally evaluated using standard speech corpora. This project uses two corpora for evaluation, an Alternative corpus created specially for this project and the standard NTIMIT corpus.

NTIMIT Corpus

The TIMIT corpus was originally designed as speech data for acoustic studies in order to evaluate speech recognition systems. However due to the pristine recording environment it is poorly suited to speaker recognition systems, as these conditions are unrealistic.

The speaker corpus chosen for the evaluation of this project is the NTIMIT corpus. The NTIMIT corpus is a variation of the TIMIT corpus, where the TIMIT speech is played through an artificial mouth into a carbon-button telephone handset, transmitting the speech over local and long-distance telephone lines and recording the received signal (Joseph P. Campbell & Reynolds 1999). Table 1.2 shows the distribution of speakers.

Table 1.2: NTIMIT Corpus Description

No. of speakers	630(438M/192F)
Type of Speech	Sentences

Alternative Speaker Corpus

Recordings were also taken from a number of speakers, for use with evaluation in conditions matching those in the meeting environment. Table 1.3 shows the distribution of speakers.

Table 1.3: Speakers used in alternative database

Speaker	Male	Female
No. #	7	5

1.8 Principle Objectives of Dissertation

The principle objectives of this Dissertation are to:

1. Research and evaluate published results of speaker identification
2. Examine speaker identification methods and systems available, including
   - Pre-processing methods.
   - Feature extraction methods.
   - Classification techniques for creating speaker models.
   - Decision methods.
3. Choose a combination of suitable speaker identification techniques to design and develop a suitable prototype, tailored for utterances containing less than ten words and a speaker database of less than eight voice profiles.
4. Complete evaluation using both the NTIMIT speaker corpus and Alternative (own) speaker corpus.

1.9 Overview of Dissertation

The following chapters explore and examine methods used for development of a prototype speaker identification system.

Chapter 2: Literature Review. This chapter reports and evaluates published results of speaker identification.

Chapter 3: Speaker Recognition. This chapter describes the different techniques used in speaker identification. These involve techniques to improve the speech signal for analysis, feature extraction and classification techniques used to identify the speaker. This chapter also introduces the source-filter model of speech which is used as a basis for many speaker recognition techniques.

Chapter 4: Feature Extraction. Different techniques for feature extraction are explored and also their ability to represent the speech signal. These include the Short-Term Fourier Transform (STFT) Linear Predictive Coefficients (LPC) and cepstral analysis.

Chapter 5: Speaker Modelling. This involves Vector Quantization (VQ) techniques for creating speaker models from the feature vectors. These include descriptions of k-means, randomized local search and hierarchical clustering methods.

Chapter 6: Experimental Setup. This describes the prototypes and the experimental setup using two different speech corpora. The first being the NTIMIT corpus, which uses telephone grade speech quality and an alternative (own) speech corpora produced specifically for experimentation.

Chapter 7: Results. VQ techniques used with LPC, LPC cepstral and mel-frequency cepstral coefficients with various codebook sizes. Effects of increasing the number of speakers in the dataset, on the identification result.

Chapter 8: Conclusion. Conclusions of proposed prototype and further work.

Chapter 2

Literature Review

2.1 Introduction

The following section is an overview of published speaker identification systems and results. Results are also presented from studies demonstrating similar techniques discussed throughout this project.

2.2 Corpora for the Evaluation of Speaker Recognition Systems

Standard speech corpora are important factors behind speaker recognition as they allow experimentation and evaluation. Joseph P. Campbell & Reynolds (1999) have evaluated current publicly available speech corpora which are intended for use with evaluation of speaker recognition systems. These researchers's study outlines the corpora salient features and suitability for conducting speaker recognition experiments.

Four factors of the speech corpora where used to evaluate their suitability for speaker recognition, these include:

- Number and diversity of speakers.

- Time separation of sessions per speaker.
- Type of speech.
- Channel, microphone and recoding environment types and variability.

The degree to which a speech corpus exhibits these factors indicates its effectiveness for use with speaker recognition.

The following list gives a brief description of the five English speech corpora evaluated in Joseph P. Campbell & Reynolds (1999) study.

TIMT and Derivatives were designed for acoustic-phonetic studies and for developing and evaluating automatic speech recognition systems. TIMIT itself is poorly suited for evaluating speaker recognition systems due to the unrealistically pristine conditions. However, the NTIMIT corpus is more suitable, as noisy characteristics are introduced by transmitting speech over telephone lines. There are 630(438M/192F) speakers that make up this corpus.

KING-92 is a corpus collected under research contracted with the US government. It contains recorded speech from 51 male speakers who were recorded under 2 different settings. These settings differ in their channel characteristics: one from a telephone. handset and the other from a high-quality microphone.

YOHO corpus was designed to support text-dependent speaker verification evaluation for government security access applications. The speech was recorded in an environment with low-level office noise with high-quality microphones. There are 138(106M/32F) speakers making up this corpus.

Switchboard I-II Including NIST evaluation Subsets are corpora that represent one of the largest collections of conversational, telephone speech recordings. The numbers of speakers for Switchboard I-II are 543 and 657 (50% M/50% M), respectfully.

Speaker Recognition Corpus consists of 100(47M/53F) speakers calling from different telephone environments. Speakers called from quiet (e.g. closed room) and noisy (e.g. public area) locations using various types of phones (e.g. mobile, public). This corpus is useful for text-independent speaker identification and verification systems.

Throughout research undertaken in this area the most common speech corpora used for evaluation of speaker identification systems were the TIMIT, NTIMIT and NIST Corpora.

2.3 Overview of Automatic Speaker Identification Results

Research conducted by Reynolds (2002) provides an overview of studies performed on automatic speaker recognition. This work highlights some of the indicators of performance in different conditions. There have been two text-independent speaker identification results described, based on average performance of typical systems. The first systems using conversational speech, with two minutes of training data and 30 seconds of testing data. The accuracy results for these systems range from 85%-93%. The second system uses very noisy data from military radios and microphones with 30 seconds of training data and 15 seconds of testing data. The accuracy results for these systems typical range from 65%-80%.

2.4 Speaker Identification using Vector Quantization

Iyer et al. (2004) describes a speaker identification system, which focuses on speech detection and extraction. This system is based on the idea of the usable speech concept. The speech frames containing higher information content (i.e., usable) are separated form those containing lower information content (i.e., unusable).

The speaker identification system evaluated by Iyer et al. (2004) used Linear Predictive Cepstral Coefficents (LPCC) (refer to section 4) to extract features from the usable speech frames. The speaker model is created with Vector Quantization (VQ) (refer to section 5 on the features extracted, using the Euclidean distance metric. The system uses 14^{th} order LPCC with a codebook size of 128 (VQ codebook).

The TIMIT corpus was used for the experiments, with 48 speakers (24 male, 24 female) chosen from various dialects. Each speaker has ten utterances with four utterances used for training and the remaining six for testing. The results obtained with both usable and unusable speech frames gave an accuracy result of 94%, while for usable frames only, the accuracy an result of 100%.

It is evident from Iyer et al.'s (2004) research, that pre-processing techniques such as the usable speech concept can improve accuracy results. It also demonstrates accuracy results using a baseline system with LPCC and VQ on the TIMIT corpus.

Most of the computation time in speaker identification is spent on identifying matching scores between speakers and unknown utterances. Kinnuenen et al. (2004) demonstrates real-time speaker identification aimed at optimizing and reducing the matching process. This focuses on optimising a VQ based speaker identification system.

This study used the TIMIT corpus, with all 630 speakers being used for testing purposes. The baseline system for speaker identification used 12-dimensional Mel Frequency Cepstral Coefficients (MFCC) (refer to section 4) for feature extraction. VQ was then applied using the LBG (Linde-Buzo-Gray) algorithm on the feature vectors obtained using the MFCC, with a codebook sizes of 8, 64 and 512. The results obtained are shown in Table 2.1

Table 2.1: Performance of baseline system (TIMIT)

Codebook size	Accuracy Rate (%)
8	89.5
64	99.52
512	99.18
No model	98.41

These results illustrate the effectiveness of the VQ model using MFCC with the TIMIT corpus. Overfitting (refer to section 5) is also shown to occur when all data (No model) is used, as the accuracy decreases.

Results were also obtained using the NIST 1999 corpus, which includes recordings over telephone network similar to that of the NTIMIT. Results here are shown in Table 2.2

with the corresponding codebook size used.

Table 2.2: Performance of baseline system (NIST)

Codebook size	Accuracy Rate(%)
64	81.94
128	82.22
256	82.66

These results demonstrate the effect that degraded speech signals have on the identification results, reducing them significantly.

Kinnunen, Kilpelainen et al. (2004) have experimented with a number of different VQ techniques including GLA(Generalized Lloyd Algorithm), SOM (Self-Organising Maps), PNN(Pairwise Nearest Neighbour), Split and RLS(Randomized Local Search) (refer to section 5). This system used 12-dimensional MFCC with each of the VQ techniques. The results are shown with two different codebook sizes ($k = 64$ and $k = 256$).

The speaker corpus used for evaluation were collected from 25(14 males and 11 females) Finnish speakers. The quality of the speech was improved by; (a) removing the silent parts, (b) removing the DC offset, (c) applying pre-emphasis and (d) applauying 30ms Hamming window (refer to section 3).

These results indicate that the size of the codebook has a bearing on the accuracy of the system, along with the type of VQ technique used. Codebooks of 64 using the RLS method produced marginally better results than other VQ techniques. Increasing the size above 64 did not improve results significantly.

2.5 Speaker Identification using NTIMIT

Jr et al. (1995) reports speaker identification results using the NTIMIT corpus to examine performance degradation. The current speaker identification systems perform well with clean speech however, performance decreases significantly when speech is recorded under more realistic conditions, such as over telephone lines. Using the NTIMIT corpus, accuracy results of 81.7% for male speakers and 74.5% for females were found. This system used Gaussian Mixture Models (GMM) (refer to section 4) and 20-dimensional MFCC.

Reynolds et al. (1995) examined the effects of size of the population of speakers and degradations introduced by noisy communication channels. The study examined speaker identification performance on the complete 630 speaker TIMIT and NTIMIT corpora. Identification accuracies obtained were 99.5% and 60.7% for TIMIT and NTIMIT respectively. The speaker identification system used in this study consisted of using MFCC with GMM.

2.6 Speaker Identification For Large Set of Voices

Starnoiewicz & Majewski (1998) presented a text-dependent speaker identification system based on Support Vector Machines (SVM). These researches examined the usefulness of a SVM speaker identification system using a large set of speakers.

The Polish Speech Dat(E) corpus was used in the evaluation and consisted of 1300 speakers. Features were extracted using a Hamming window with 15-dimensional MFCC for each frame.

The speaker identification system was tested using three groups of speakers shown in Table 2.3

Table 2.3: Impact of of the number of speakers on identification results

Speakers	Accuracy Rate(%)
125	92.5
500	88.5
1300	88.5

These results indicate that the SVM implementation with MFCC is a relative effective method of identifying speakers. However due to the different corpus used in this study, comparisons between other speaker identification may be misleading.

2.7 Effects of Utterance Length on Speaker Identification

Kwon & Narayanan (2005) presented a speaker identification system which was tailored to use short utterances, by selectively using relatively robust feature vectors. The evaluation was performed using the NIST speech (1999) corpus with 400(200M/200F) speakers. This included 50 seconds of spontaneous speech for each speaker: 40 seconds of the speech was used for training and 10 seconds for identification.

The baseline speaker identification used 24-dimensional MFCC with a 30ms Hamming window shifted by 10ms. The speaker model was created using GMM with 16 mixtures. The results using different lengths of input speech are shown in Table 2.4

Table 2.4: Accuracy results for different length speech input data

Length of Input Data(seconds)	Accuracy Rate(%)
0.25	71
0.5	76
1	81
2	85

These results show the effect of identification using short utterances with GMM. This indicates that to gain reasonable identification results, at least two seconds of speech is needed. This can also be applied to other techniques such as VQ, were two seconds of speech can also produce reasonable accuracy results.

2.8 Chapter Summary

This chapter has described different studies and their relevance to speaker identification. These studies have shown the combination of methods used in a speaker identification system. These included the type of results expected with different parameters. These studies have examine the following areas

- Corpora for the Evaluation of Speaker Recognition Systems
- Automatic Speaker Identification Results

- Speaker Identification using Vector Quantization
- Speaker Identification using NTIMIT
- Speaker Identification For Large Set of Voices
- Effects of Utterance Length on Speaker Identification

As evident in the literature the results expected for a speaker identification system range from 60% to 99% depending on the corpora, utterances length and the number speakers used. Therefore it is these methods which will be described and focused on during the current study.

Chapter 3

Speaker Recognition

3.1 Introduction

Speaker Recognition can be divided into different areas as discussed in Chapter 1. Specifically this project is concerned with developing a Speaker Identification system using closed-set, text-independent speaker identification system. With this in mind, the following methods for speaker recognition will be directed specifically at closed-set, text-independent speaker identification.

Speaker identification involves two main stages, the enrolment stage and the identification stage described in section 1.4. These phases involve three main parts:

- Pre-Processing.
- Feature Extraction.
- Speaker Modelling.

3.2 Pre-Processing

Speech is recorded by sampling the input, which results in a discrete-time speech signal. Pre-processing is a technique used to make the discrete-time speech signal more amendable for the processes that follow. There are five pre-processing techniques that can be used to enhance feature extraction. These include DC offset removal, silence removal, pre-emphasis, windowing and autocorrelation.

3.2.1 DC Offset Removal

Often audio signals carry an inaudible, yet unwanted constant offset or DC offset. The DC offset can have an effect on the quality of the information extracted from the speech signal and may cause errors in the speaker model. There are a number of methods available to approximate the DC Offset, allowing for its removal.

The first method involves calculating the average value of the speech signal and subtracting this from itself. This effectively removes the DC offset, if it is constant over time. However the DC offset can vary over time, so a more effective method involves using a one-pole, one-zero high-pass filter. This filter considers the DC offset a zero-frequency component. A one-pole, one-zero high pass filter has the transfer function:

$$\begin{aligned} H(z) &= 1 - \frac{1-p}{1-pz^{-1}} \qquad (3.1) \\ &= \frac{p-pz^{-1}}{1-pz^{-1}} \end{aligned}$$

3.2.2 Silence Removal

Silence removal is used to remove silent parts of the speech signal, which contain little or no speaker specific characteristics. It is also useful in practical implementations such as in a meeting environment as it will eliminate inadvertent speech or prevent interference picked up by a microphone from being seen as the primary speaker source.

Silence removal is usually achieved using the energy of the signal and comparing this to the energy of each frame. This is achieved using a smaller overlapped range usually half the size of the frame size used for feature extraction.

The silent parts of the speech signal can be detected from the *short-time energy* defined

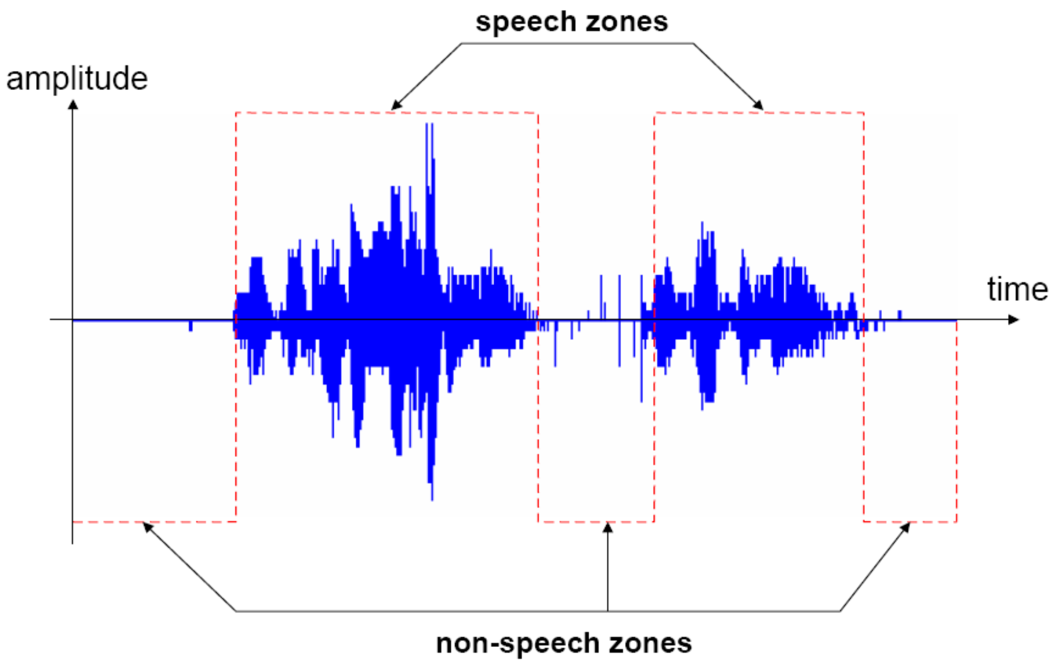

Figure 3.1: Silence Parts of the Speech Signal

as

$$E_n = \sum_{m=0}^{L-1} x_n(m)^2 \tag{3.2}$$

Voice degree detector can also be used to determine what parts of the speech signal are useful. A voice degree detector uses autocorrelation to determine the varying levels of speech present in the signal. Autocorrelation is defined as

$$R(k) = \sum_{n=0}^{N-1-k} s(n)s(n-k) \tag{3.3}$$

where $s(n)$ is the speech frame and $s(n-k)$ is the same speech frame delayed by k. A voice degree detector can then be created using

$$\alpha = \frac{R[1]}{R[0]} \tag{3.4}$$

where $R[0]$ and $R[1]$ is the autocorrelation sequence defined in Equation 3.3. α is a similarity measure of the speech frames. High values of α represent speech frames that are noise like or unvoiced sounds, while low values occur for periodic voiced sounds. Figure 3.2 shows a speech signal and the corresponding result of a voice detector using Equation 3.4.

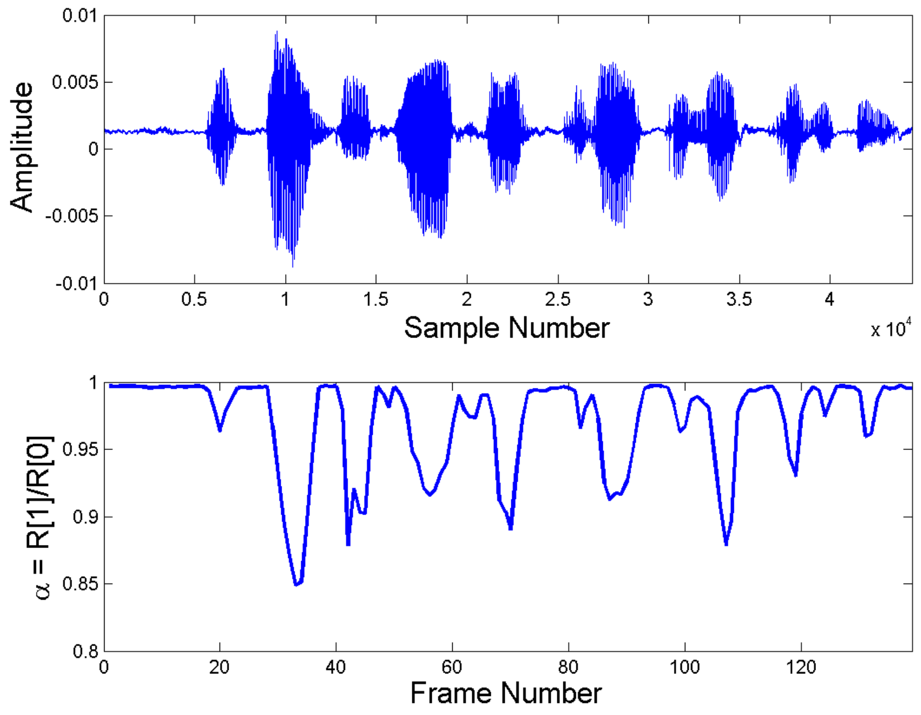

Figure 3.2: Voice Degree Detector (Drygajlo 2005).

3.2.3 Pre-emphasis

Pre-emphasis is a technique used in speech processing to enhance high frequencies of the signal. There are two important factors driving the need for pre-emphasis. Firstly, the speech signal generally contains more speaker specific information in the higher frequencies (Gravier 2004). Secondly, as the speech signal energy decreases the frequency increases. Therefore by applying pre-emphasis, the spectrum is flattened, consisting of formants of similar heights. This allows the feature extraction process to focus on all aspects of the speech signal.

Pre-emphasis is implemented as a first-order *Finite Impulse Response* (FIR) filter defined as:

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

Generally a is chosen to be between 0.9 and 0.95. Figure 3.3 shows the frequency response of a pre-emphasis filter.

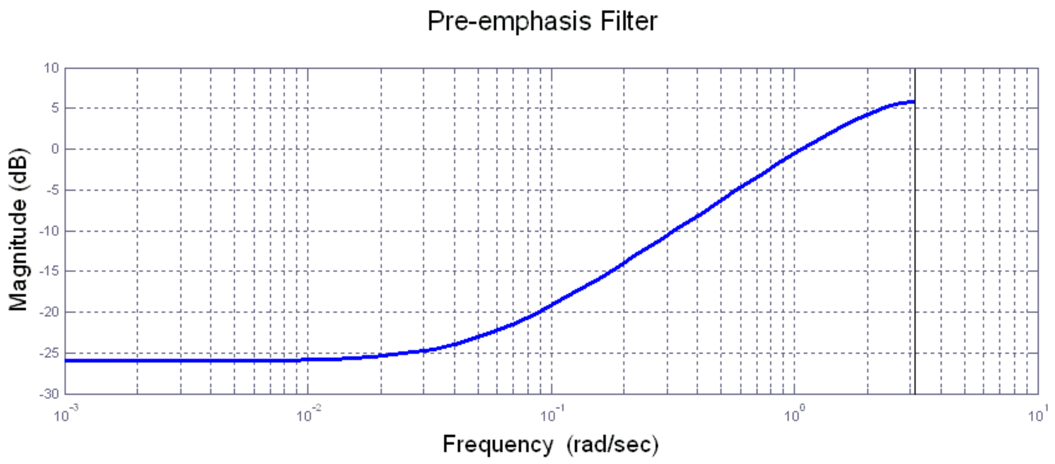

Figure 3.3: Frequency response of a pre-emphasis filter with a = 0.95

3.2.4 Frames and Windowing

The speech signal can be considered quasi-stationary over small intervals, making it amendable for analysis techniques at this level. Therefore we divide the speech signal into fixed-length frames, each considered as a stand-alone signal. This enables short-time analysis of the speech signal.

Speaker recognition takes advantage of the quasi-stationary speech signal through short-time analysis. A windowing function is used on each frame to smooth the signal and make it more amendable for spectral analysis. We can define this as:

$$x_n(m) = x(n - m)w(m) \tag{3.6}$$

Where $w(m)$ is a finite-length windowing function.

Windowing functions can be applied $w(m)$ to the frames. The simplest being a rectangle window shown at the top of Figure 3.6. This rectangle window however is not normally used, as it causes spectral distortion. This is due to abrupt frequencies caused by the end points of the frame. This distortion can be reduced by using a smoothing function to improve the short-time spectral analysis of the frames. Hamming, Hann and Blackman windows are three window functions used commonly in speech analysis to reduce the abrupt and undesirable frequencies occurring in the framed speech. Figure 3.6 shows a Hamming window applied to a speech frame.

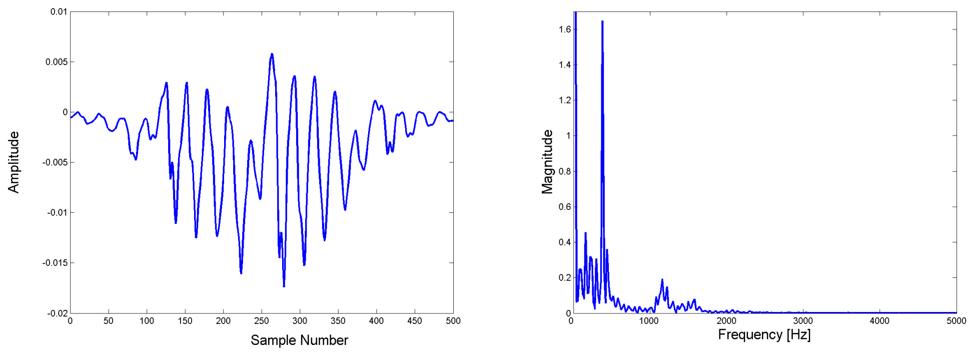

Figure 3.4: Frame without Pre-emphasis

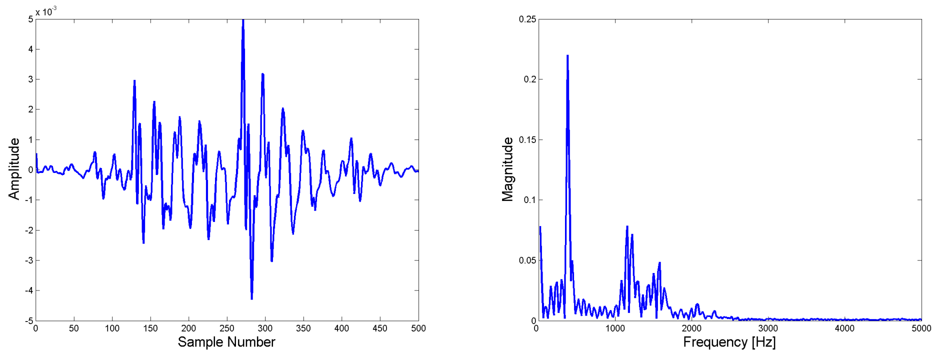

Figure 3.5: Frame with Pre-emphasis

The Hamming window is defined as:

$$w(n) = 0.54 - 0.46 \cos\left[\frac{2\pi n}{N-1}\right] \tag{3.7}$$

The Hann window is defined as:

$$w(n) = 0.5\left(1 - \cos\left(\frac{2\pi n}{N-1}\right)\right) \tag{3.8}$$

The Blackman window is defined as:

$$w(n) = 0.42 - 0.5\cos(\frac{2\pi n}{N}) + 0.08\cos(\frac{2\pi n}{N}) \tag{3.9}$$

where N represents the width of the frame and n is an integer, with values $0 \leq n \leq N-1$

3.2.5 Autocorrelation

Autocorrelation provides a mechanism for removing the noise-like parts of the speech signal, while preserving the spectrum of the speech signal and speaker information.

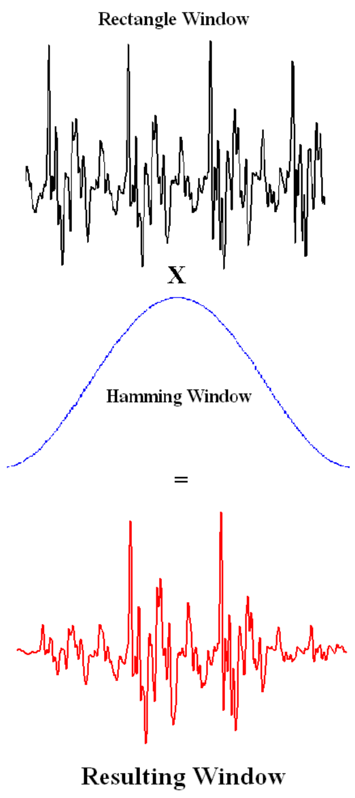

Figure 3.6: Hamming window applied to a framed speech signal

Autocorrelation is a measure of how well a signal matches a time-shifted version of itself. Two important pieces of information about autocorrelation signals are: (Leis 2002)

1. the autocorrelation of a periodic signal preserves the periodicity.
2. the autocorrelation of a random signal is zero at all lags except for a lag of zero.

Autocorrelation is defined as:

$$R_{xx}(k) = \frac{1}{N} \sum_{n=0}^{N-1} x(n)x(n-k) \tag{3.10}$$

where k is the lag or delay

Autocorrelation is taken over the frames to reduce the amount of noise and aims to improve the ability of feature extraction methods.

3.3 Source-Filter Model of Speech

There are numerous techniques available to extract features and model speakers. However, to do so requires an appreciation for the invariably complex signal that is our speech and the information obtained when captured.

Most characteristics of the speech signal are difficult to measure explicitly, however they are captured implicitly through short-time and long-time spectral energy. Characteristics are also captured implicitly through the fundamental frequency and overall energy. The implicit nature of the speech signal makes it difficult to differentiate one speaker from another without specific analysis techniques.

Most analysis techniques commonly employed in speaker identification are based on the short-time spectral analysis. This is due to the periodic and quasi-stationary characteristics of the speech signal when taken in small intervals or frames, usually around 10-30ms (Johnson 2005). In small frames like this, the statistical properties of the signal are slow moving and are amendable for speaker analysis. This leads us to the first component of speaker identification, feature extraction.

Designing a speaker identification tool first requires features to be extracted from the speech signal. Generally a 10-30ms windowing system, such as a Hamming window is used to sample the speech signal, as it can be considered quasi-stationary over a short period of time. From this windowed signal we can produce coefficients that represent the most dominant features of that part of the signal. This basic concept is known as feature extraction. There are many methods available that can be used to extract features from the speech signal, such as *cepstral* analysis, *Linear Prediction* (LP) and variations of these.

Feature extraction is used to parameterise the speech signal using short-time analysis. It is applied to framed speech to produce speaker-specific feature vectors. This process is shown in Figure 3.7.

There are a number of methods available to perform feature extraction and many incorporate the fundamental idea of the *source-filter* method defined as:

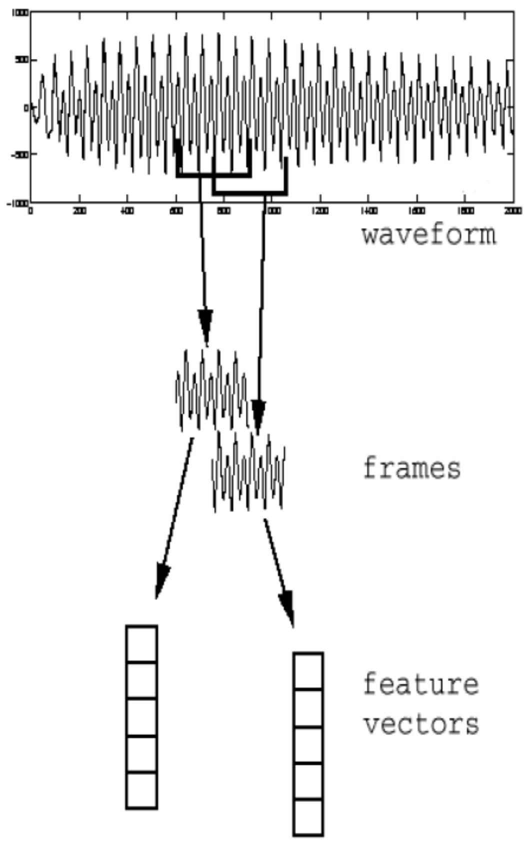

Figure 3.7: The process of extracting features from the speech signal (Willits 2003).

$$\widehat{s}(n) = e(n) \times f(n) \tag{3.11}$$

The speech signal can be modelled by the source-filter method of speech production. Cepstral and LP analysis is useful as a tool for feature extraction as they both try to separate the source and filter parts of the speech signal. The source-filter concept leads directly to engineering methods used separate the source from the filter (Gold & Morgan 2000). Here the source is the excitation signal, produced by the vocal cords and the filter is the vocal tract, which shapes the speech signal by emphasizing certain frequencies(Rabiner & Juang 1999). If the excitation signal is donated by $e(n)$ and the filter signal by $f(n)$ then the resulting speech waveform is the convolution of the two signals:

$$s(n) = e(n) \times f(n) \tag{3.12}$$

Since the source characteristics contain much less speaker specific information then filter or vocal cord characteristics, it would be useful to discard this information. Separation of the excitation and filter signals is, in a practical sense, impossible due to the mixed signals non-linearity. Chapter 4 details feature extraction methods.

3.4 Speaker Modelling

After extracting speaker-specific characteristics from the speech signal we need a method to classify the speaker in order to determine the author of a given speech signal. In order for identification a speaker must first be enrolled in the system using a modelling process. Once models for the speakers have been created, a matching or classification process is then used for identification.

Speaker modelling requires training, which is used to characterize speaker specific patterns of the speech signal for a given speaker. There exist a number of methods used for speaker modelling. Two commonly used methods are:

- *Vector Quantization*
- *Gaussian Mixture Models*

3.4.1 Vector Quantization

Vector Quantization (VQ) is a clustering method, which maps vectors from a vector space to a finite number of regions in that space. Each of these regions or clusters has a central vector or centroid. These clusters represent characteristics of similar vectors from the vector space (Kinnunen et al. 2004). Together these clusters form the vector space, known as a *codebook*. The codebook is created using features extracted from the speech signal, which are used to model specific components of a speakers voice (discussed in section 4.

3.4.2 Gaussian Mixture Models

Gaussian mixture models (GMM) are also a useful technique in speaker modelling. GMM is a statistical method used to model speaker-specific features, which aims to provide better estimates of data. A key advantage GMM have over VQ is that the clusters are able to overlap, which can lead to results that model the speakers more accurately

Mixture models are a type of density model, which comprise of a number component functions (Wu 2005). These component functions can be combined to provide a multi-modal Gaussian density representation that is unique for each speaker.

A GMM can be represented as a weighted sum of component densities, or mathematically as:

$$p(x) = \sum_{i=1}^{M} p_i b_i(x) \tag{3.13}$$

where M is the number of components, x is a feature vector, b_i are the components densities and p_i are the mixture weights.

3.4.3 Support Vector Machine

Support Vector Machines (SVM) are another method used recently in speaker identification and has achieved performance results that are greater or equal to other methods (Starnoiewicz & Majewski 1998). SVM are based on the principle of structural risk minimisation and are a binary classifier that makes decisions by constructing a linear hyperplane that optimally separates two classes.

3.5 Distance Metrics

An integral step in determining a speaker's identity is to determine the similarity between an unknown speaker's feature vectors and set of valid speaker models. This involves using a metric to calculate the distortion between unknown feature vectors and the speaker models. From these results, the lowest distortion can determine which speaker model is the closest match for the unknown speaker. Several different metrics can be used to calculate the distortion between feature vectors and a speaker model, or codebook in the case of VQ. Commonly the *Euclidean Distance Metric* is used to calculate the distance between two points.

$$\begin{aligned} d(\mathbf{x},\mathbf{y}) &= (x_1 - y_1)^2 + \ldots + (x_n - y_n)^2 \qquad (3.14) \\ &\textit{or} \\ d_i(\mathbf{x_i},\overline{\mathbf{x}}) &= (\mathbf{x_i} - \overline{\mathbf{x}})^T(\mathbf{x_i} - \overline{\mathbf{x}}) \end{aligned}$$

(3.15)

where x and y are feature vectors, $\overline{x}$ is the mean of the feature vectors, x_i is the mean of the i^{th} speaker's codebook and the d is Euclidean Distance between the two. A variation of this is the *Euclidean Squared Distance Metric.*

$$d(\mathbf{x},\mathbf{y}) = \sqrt{(x_1 - y_1)^2 + \ldots + (x_n - y_n)^2} \qquad (3.16)$$

The Euclidean measure is isotropic and is the best measure when all attributes are the same.

The *Manhattan Distance Metric* is also a useful metric and is more robust than the Euclidean Distance Metric. The Manhattan Distance Metric has the advantage of being less sensitive to large discrepancies.

$$d(\mathbf{x},\mathbf{y}) = |x_1 - y_1| + \ldots + |x_n - y_n| \qquad (3.17)$$

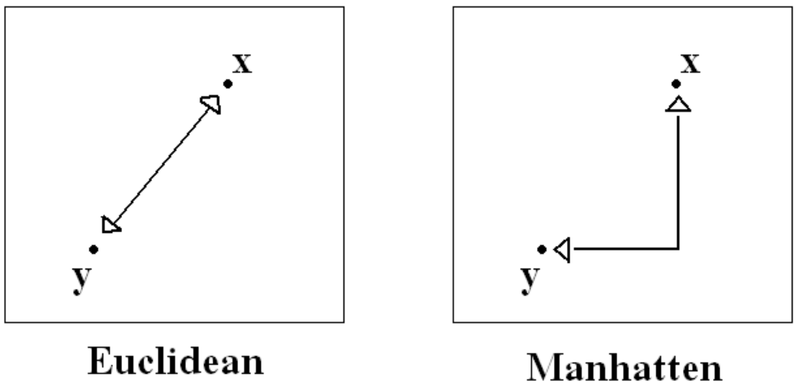

Figure 3.8: Euclidean and Manhattan metric in R^2

Another commonly used metric is the *Mahalanobis Distance.* This distance metric is based on the Euclidean distance metric where a weighting matrix is added to give less weights to components having more variance. This is defined as

$$d_i(\mathbf{x_i}, \overline{\mathbf{x}}) = (\mathbf{x_i} - \overline{\mathbf{x}})^{\mathbf{T}} \mathbf{W} (\mathbf{x_i} - \overline{\mathbf{x}}) \tag{3.18}$$

where W is the weighting matrix. The weighting matrix W is the covariance matrix corresponding to the mean $\overline{x}$. If weighting matrix W, were an identity matrix of $\overline{x}$ the distance would be Euclidean.

The covariance of two points:

$$cov(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=1}^{n} (\mathbf{x_i} - \overline{\mathbf{x}})(\mathbf{y_i} - \overline{\mathbf{y}})}{(n-1)} \tag{3.19}$$

Therefore the W weight matrix is the covariance matrix defined as:

$$W = \begin{pmatrix} cov(x_1, x_1) & cov(x_1, x_2) & \ldots & cov(x_1, x_n) \\ cov(x_2, x_1) & cov(x_2, x_2) & \ldots & cov(x_2, x_n) \\ \vdots & \vdots & \vdots & \vdots \\ cov(x_n, x_1) & cov(x_n, x_2) & \ldots & cov(x_n, x_n) \end{pmatrix} \tag{3.20}$$

where n is the number of dimensions in $\mathbf{x}$

3.6 Chapter Summary

This chapter has presented an overview of speaker identification and the processes involved. This includes pre-processing, feature extraction, speaker determination and pattern matching. Several techniques were demonstrated for each of the processes involved.

Firstly, pre-processing techniques are used to improve the quality of the speech signal to make it amendable for feature extraction. These pre-processing techniques include

- DC offset removal.
- Silence Removal.

- Pre-emphasis.
- Windowing.
- Autocorrelation.

Secondly the process of extracting information from the speech signal was presented which demonstrated a number of methods briefly. These techniques included

- Cepstral Analysis.
- Linear Predictive Coding.

The above methods were based on the source-filter concept of speech modelling, which can give a good approximation of the information obtained in the speech signal.

Lastly, techniques used for speaker modelling, which included methods of how to construct representations of the speakers speech from the features extracted from their speech signal. These included

- Vector Quantization.
- Gaussian Mixture Models.
- Support Vector Machines.

In conjunction with these speaker modelling techniques, a distance metric is needed, to determine the distortion between points. Three commonly used metrics examined were

- Eucildean.
- Manhatten.
- Mahalanobis.

Chapter 4

Feature Extraction

4.1 Introduction

In order to create a speaker profile, the speech signal must be analysed to produce some representation that can be used as a basis for such a model. In speech analysis this is known as feature extraction. Feature extraction allows for speaker specific characteristics to be derived from the speech signal, which are used to create a speak model. The speaker model uses a distortion measure to determine features which are similar. This places importance on the features extracted, to accurately represent the speech signal.

A number of speech coding methods used for compression, are prime candidates for extracting features from a signal. Notably *Linear Prediction (LP)* and its associated variants are examples of speech coding techniques applied to speaker identification. Another important method used involves a spectral technique called *cepstral analysis*. Both methods provide a good approximation of the deconvolution of the source and filter, described in Section 3.3

4.2 Short-Term Fourier Transform

The *Short-Term Fourier Transform* (STFT) can be used to extracted spectral information from the speech signal. The Fourier transform is commonly used in speaker recognition techniques to extract information. It is normally used in combination with other methods to emphasis certain spectral components.

The *Discrete Fourier Transform* (DCT) of a sampled signal is:

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-jn\omega_k} \tag{4.1}$$

where N is the number of samples and ω_k is the frequency of the k^{th} sinusoid

$$\omega_k = \frac{2\pi k}{N} \tag{4.2}$$

The spectral energy in the speech signal is characterized by the speech content and physiology of the speaker.

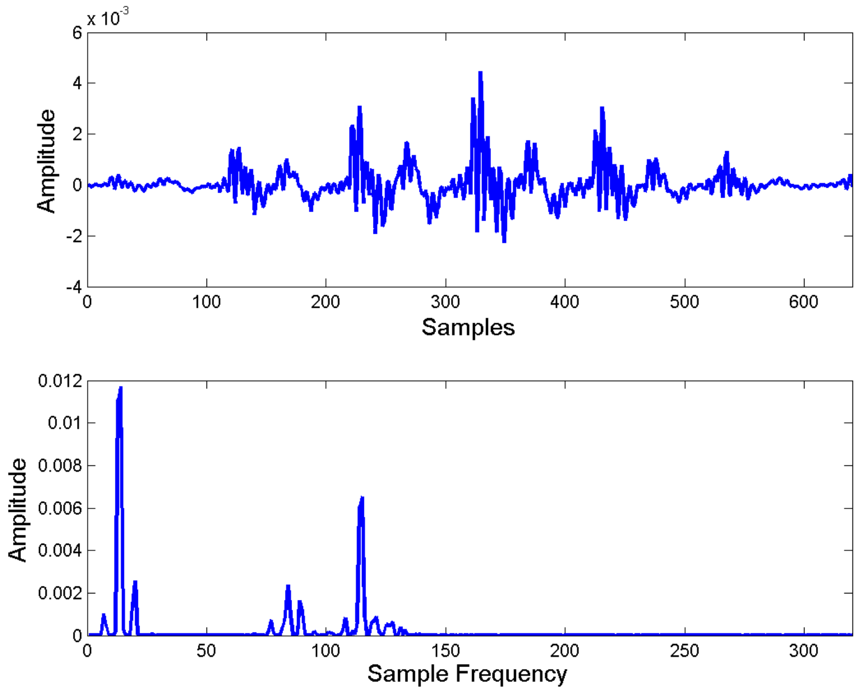

Figure 4.1: Source Filter Spectrum

Figure 4.1 shows the a voiced segment of speech and the corresponding STFT of the same frame. The spectrum of a speech frame illustrates that the signal has a high amplitude component with a low frequency and a number of higher frequency components.

Features can be extracted using the STFT, such as pitch information and formants, however the STFT is rarely used by itself. Instead the STFT is used with other methods which rely on this transformation into the frequency domain. A method used commonly is the cepstrum, which takes advantage of the characteristics of STFT.

4.3 Cepstrum

The fundamental idea of the cepstrum is to take the inverse Fourier transform of the logarithm of the spectrum. Mathematically it can be defined as:

$$cepstrum = IFT(\log(FT(signal))) \tag{4.3}$$

There are two forms of the cepstrum, the real and the complex. The real ceptrsum uses the logarithm defined for real values while the complex cepstrum users the logarithm defined for complex values (Osdol 2004). The outcome is that the real cepstrum retains information only on the magnitude of the spectrum while the complex cepstrum retains both magnitude and phase information.

For applications in signal processing the complex cepstrum is used most often, due to the information retained in the phase and magnitude. The complex cepstrum of a signal $s(n)$ can be defined in terms of its Z transform.

The cepstrum can be used to approximate the separation of the source and filter signals. The convolution of the excitation signal $s(n)$ and the vocal tract filter $f(n)$, when represented in the frequency-domain become the multiplication of the respective Fourier transform:

$$s(\omega) = e(\omega) \times f(\omega) \tag{4.4}$$

Then by taking the logarithm of the magnitude of both sides we can separate the

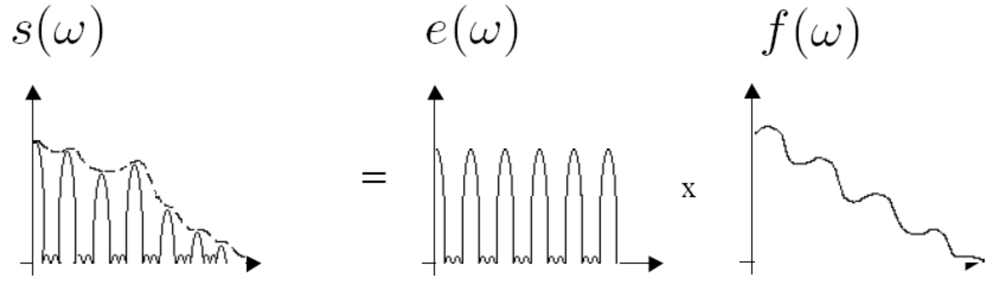

Figure 4.2: Source Filter Spectrum (Karpov 2003)

multiplied variables. This transforms the equation from multiplication to addition.

$$\log|s(\omega)| = \log|e(\omega)| \times \log|f(\omega)| \tag{4.5}$$

Finally by taking the inverse Fourier transform of the logarithm of the magnitude spectrum we get the frequency distribution of the fluctuations in the curve of the spectrum of the original signal.

$$IFT(\log|s(\omega)|) = IFT(\log|e(\omega)|) \times IFT(\log|f(\omega)|) \tag{4.6}$$

The inverse Fourier transform separates the quickly varying and slowly varying parts from the log of the spectrum. The cepstrum basically decomposes the signal into the source and filter characteristics and can be considered a deconvolution operator. The final result of the cepstrum is an approximate separation of the source and filtered signals. The source information contains little speaker-specific information, while the filter signal does contain speaker specific information that can be used for identification. This allows us to produce speaker-specific vectors that can be used to model and identify speakers. Figure 4.3 shows the cepstrum and correspond speech frame.

There a number of feature extraction methods that rely on modelling the human speech as a convolution of two signals, the source and the filtered signals. In order to separate the speech signal, methods such as the cepstral analysis can be used to approximate the deconvolution.

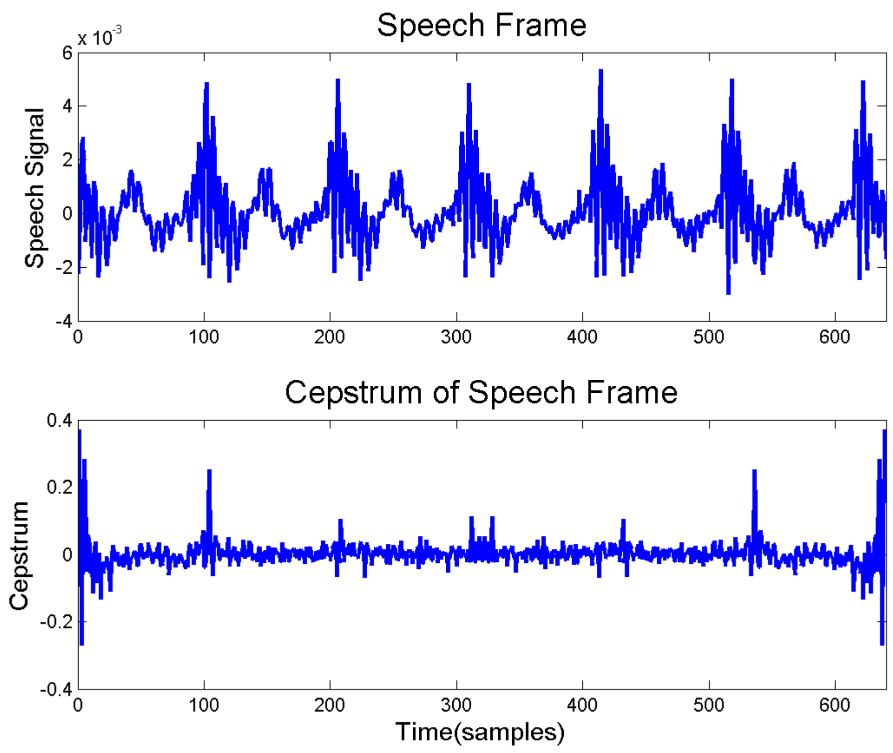

Figure 4.3: Cepstrum

4.3.1 Mel-Frequency Cepstrum Coefficients

A method used for feature extraction is Mel-frequency cepstrum coefficients (MFCC). MFCC are based upon short-time spectral analysis in which MFCC vectors are computed. MFCC analysis is similar to that of cepstral analysis, however the frequency axis is warped according to a *mel-scale*. The mel-scale is a perceptual scale of pitches judged by listeners to be equal in distance from one another (Raja, Y. (1999)). MFCC is based on the information carried by low-frequency components of the speech signal, in which less emphasis is placed on the high frequency components. The aim MFCC is to better approximate the human auditory system's response.

The mel-scale shown in Figure 4.4 illustrates that the spacing below 1kHz is linearly spaced while the that above 1kHz is spaced logarithmically (Shah, Iyer et al. 2004). This non-linearity of the mel-scale approximates, that of the human ear, which roughly hears frequencies lower than 1kHz linearly and frequencies higher than 1kHz logarithmically.

The relationship between the mel-frequency and normal frequency is given by

$$mel(f) = 2595 \times \log_{10}(1 + f/700) \tag{4.7}$$

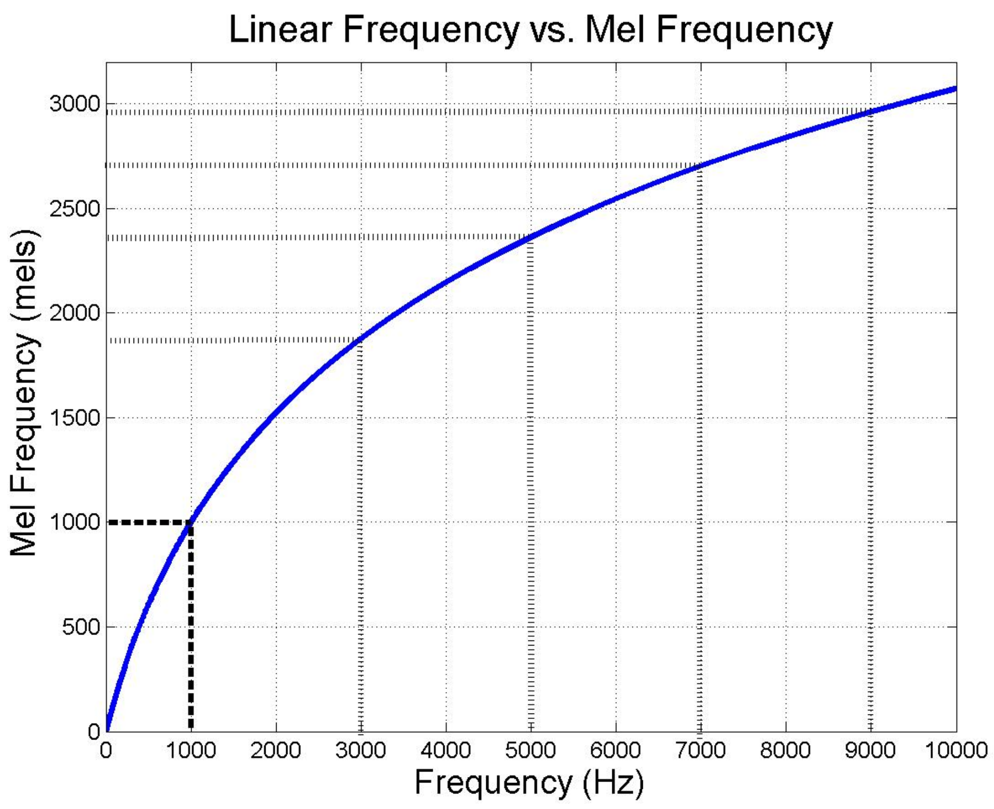

Figure 4.4: Relationship between the mel-frequency and the physical frequency

A method used commonly to create the mel-spectrm is to use a filter bank. Each filter in this bank has a triangular bandpass frequency response and corresponding to a desired mel-frequency component. Each filter computes the average spectrum around each centre frequency with increasing bandwidths as shown in Figure 4.5

4.4 Linear Prediction

LP is a speech coding technique used to reduce the amount of information needed to represent the signal. It is based on the source-filter method of speech production. LP analysis is similar to that of cepstral analysis in that the source is considered to be pulses from the vocal cords, which are then passed through an all-pole transfer

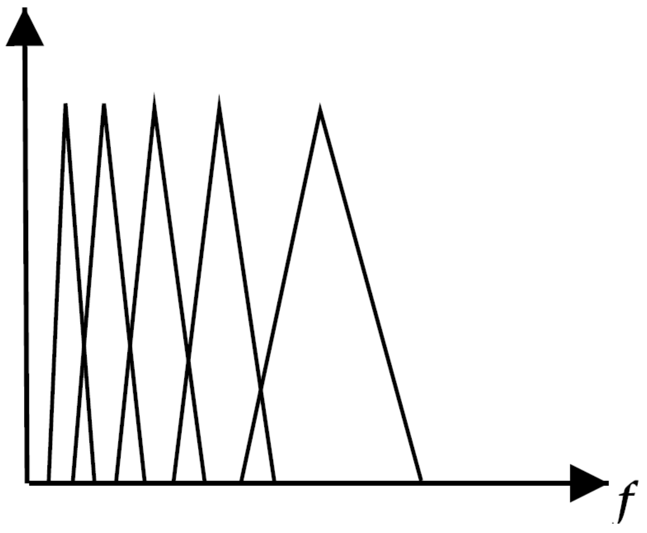

Figure 4.5: Mel Frequency Filter Bank

function representing the effect of the vocal tract (Markel & Gray 1976). Specific poles of the transfer function (4.8) can then be used as a representation of the signal. The *LP Coefficients* (LPC) are the result of attempting to predict the speech samples as a linear combination of past samples. Certain weights that give the best prediction, and form the coefficients.

$$H(z) = \frac{G}{A(z)} \tag{4.8}$$

Generally, LP analyses the past speech samples in order to predicted a given speech sample. The prediction signal $\widehat{s}(n)$ can be represented as a summation of the previous samples $s(n)$ multiplied weights or coefficients.

$$\begin{aligned} \widehat{s}(n) &= a_1 s(n-1) + a_2(n-2) + a_P s(n-P) \qquad (4.9) \\ &= \sum_{k=1}^{P} a_k s(n-k) \end{aligned}$$

Here P is the number of coefficients and n is the sample. The speech signal $s(n)$ can then be modelled based on this prediction

$$e(n) = s(n) - \widehat{s}(n) \tag{4.10}$$

$$e(n) = s(n) - \sum_{k=1}^{P} a_k s(n-k) \tag{4.11}$$

In the z-domain we can represent $e(n)$ as

$$A(z) = 1 - \sum_{k=1}^{P} a_k z^{-k} \tag{4.12}$$

If we combined equation 4.8 and 4.12 we get the all-pole transfer function

$$H(z) = \frac{G}{1 - \sum_{k=1}^{P} a_k z^{-k}} \tag{4.13}$$

An all-pole transfer function may not retain some the phase characteristics of the speech signal, however this information is less important as the human ear is fundamentally phase deaf (Karpov 2003). The all-pole model also preserves the spectral information almost exactly and is why LPC and variations of this method are used in speech coding and speaker recognition systems. Figure 4.6 shows the process of LP, importantly the Pulse Train (vocal cords).

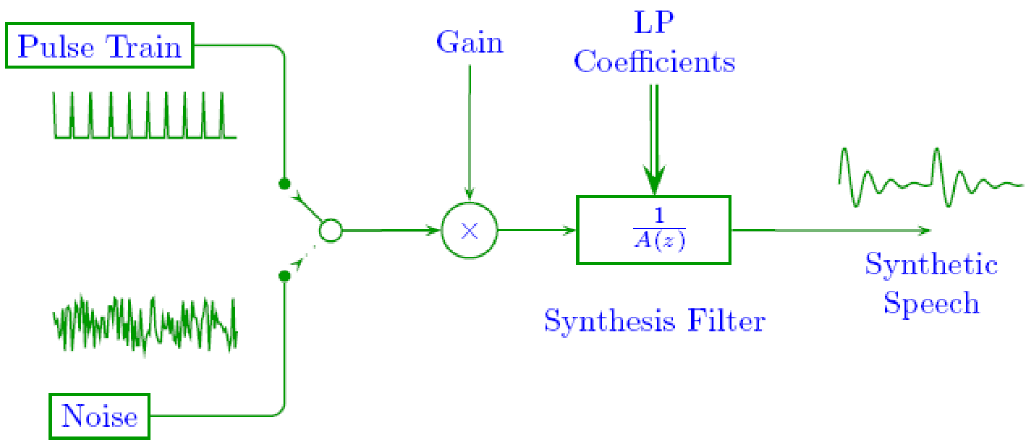

Figure 4.6: Source-filter relationship of LPC when used to synthesis speech. Source: *http://www.usq.edu.au/users/leis/courses/ELE4607/module8.pdf*

The next problem is how to calculate the prediction coefficients. There exists two methods, autocorrelation and covariance method. Both of these methods minimize the mean-square value of error, equation 4.11.

$$\begin{aligned} E &= e^2(n) \qquad (4.14) \\ &= \left[s(n) - \sum_{k=1}^{P} a_k s(n-k) \right]^2 \end{aligned}$$

To find the optimal value, the partial derivative of E with respect to a_k is set to zero.

$$\frac{\partial E}{\partial a_k} = 0, k =, \ldots P. \tag{4.15}$$

This can be solved using the autocorrelation method, which has the form.

$$Ra = r \tag{4.16}$$

where R is a special type of matrix called the *Toeplitz matrix*(reference), a is the vector of the LPC and r is the autocorrelation matrix. The autocorrelation of $s(n)$ is defined as $r_s(k)$ in equation

$$R(k) = \sum_{n=0}^{N-1-k} s(n)s(n-k) \tag{4.17}$$

The predictor coefficients can then be found by solving the matrix equation 4.16, where

$$R = \begin{bmatrix} R_n(0) & R_n(1) & R_n(2) & \dots & R_n(P-1) \\ R_n(1) & R_n(0) & R_n(1) & \dots & R_n(P-2) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ R_n(P-1) & R_n(P-2) & R_n(P-3) & \dots & R_n(0) \end{bmatrix} \tag{4.18}$$

R is the Toeplitz matrix or autocorrelation matrix

$$a = \begin{bmatrix} a_1 \\ a_2 \\ \dots \\ a_P \end{bmatrix} \tag{4.19}$$

a is the predictor coefficients

$$r = \begin{bmatrix} R_n(1) \\ R_n(2) \\ \dots \\ R_n(P) \end{bmatrix} \tag{4.20}$$

r is the autocorrelation vector

LPC are calculated over each speech frame to produce feature vectors. The number of coefficients used to represent each frame generally ranges from 10 to 20 depending on the application, sampling rate, and number of poles in the model.

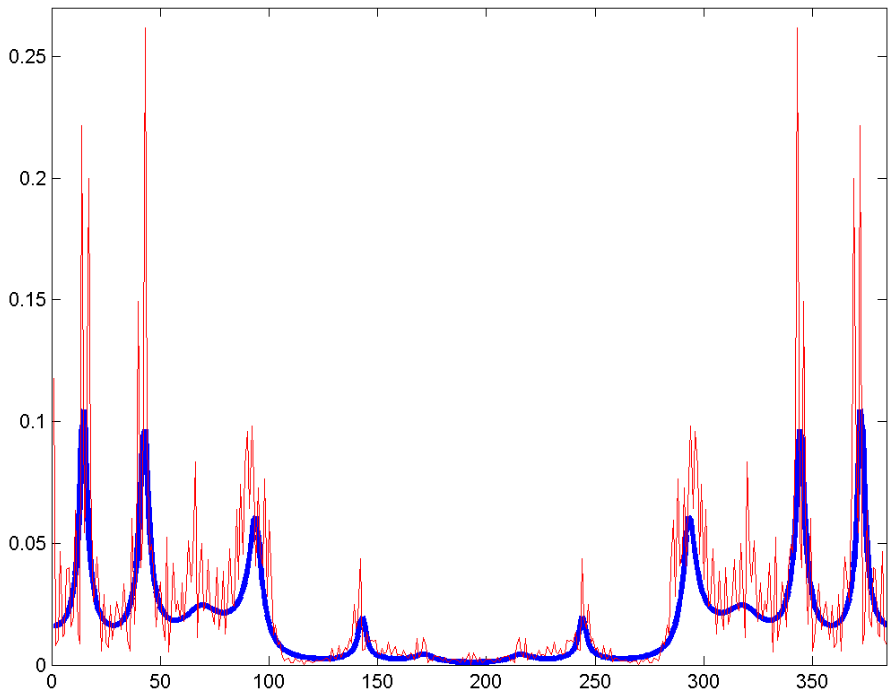

Figure 4.7: LPC spectrum window, and STFT

4.5 Linear Predictive Cepstral Coefficients

Combining both LPC and cepstral analysis of a signal gives benefits of both techniques and improves the accuracy of the features extracted. The basic idea of *Linear Predictive Cepstral Coefficients* (LPCC) is instead of taking the inverse Fourier transform of the logarithm of the spectrum of a signal, its taken from the LPC. By taking the LPC $[a_k]_{k=1}^{P}$ the cepstral coefficients $c(n)$ can be computed using a recursive formula, without computing the Fourier Transform (Campbell 1999).

$$c(n) = a(n) + \sum_{k=1}^{n-1} \left[\frac{k}{n}\right] c(k)a(n-k) \tag{4.21}$$

4.6 Normalisation

Normalisation is used to reduce the mismatch between signals which have been recorded in different environments (Gravier 2004). Normalising the data can be achieved by calculating the mean $\overline{x}$ and variance s^2 for the features and then applying a normalisation function. That is

$$\overline{x}_j = \frac{1}{m}\sum_{i=1}^{m} x_{ij} \tag{4.22}$$

$$s^2 = \frac{1}{m-1}\sum_{i=1}^{m}(x_{ij} - \overline{x}_j)^2$$

Then to normalise

$$z_{ij} = \frac{x_{ij} - \overline{x}_j}{s_j} \tag{4.23}$$

4.7 Chapter Summary

This chapter has presented techniques for feature extraction. These include STFT, Ceptral analysis and LP analysis. These methods are based on the source-filter method of speech production the speech signal can be represented by the convolution of a source and filter. These methods described seek to separate the two and model the filter to determine speaker specific characteristics.

The cesptrum approximates the deconvolution of the source from the vocal cords $s(n)$ and filter $f(n)$ caused by the vocal tract. This can be improved by warping the frequencies according to a mel-scale, which approximates the human auditory system.

LP is a speech coding method which can be used to extract speaker specific features. LP can be combined with the cepstrum method to produce features which include the benefits of both methods.

Finally normalisation is applied to the feature vectors to account for any mis-match between speech signals.

Chapter 5

Speaker Modelling

5.1 Introduction

A number of different techniques exist that can be used to model speakers based on the features extracted from there speech. Vector Quantization (VQ) is commonly used as a technique for lossy data compression which can be also be used to create a speaker model. Much research has been invested into VQ as it applies to Speaker Recognition and results have shown that it can produce practical results and compares well to other techniques such as Gaussian Mixture Models (GMM).

5.2 Vector Quantization

VQ is a method used predominantly for image and speech coding as a compression tool. However, VQ is also applied in biometrics to classify data and can be used to model speaker specific characteristics. VQ is similar to scalar quantization and involves the process of approximating a continuous or large set of values by a small set of discrete values. The important difference is that scalar quantization considers only R^1 while VQ can be used in R^N, where N is an positive integer.

VQ works be taking a large set of vectors that form a vector space, and mappings these into a smaller number of finite regions in that space. This process finds clusters of vectors with similar values and uses the central vectors or *centroids* to create a codebook. This codebook represents the vector space from which the vector set was created. When applied to speaker identification this codebook can be used to represent a speaker, created from the features extracted from the speech signal. The process of creating a codebook is illustrated in Figure 5.1.

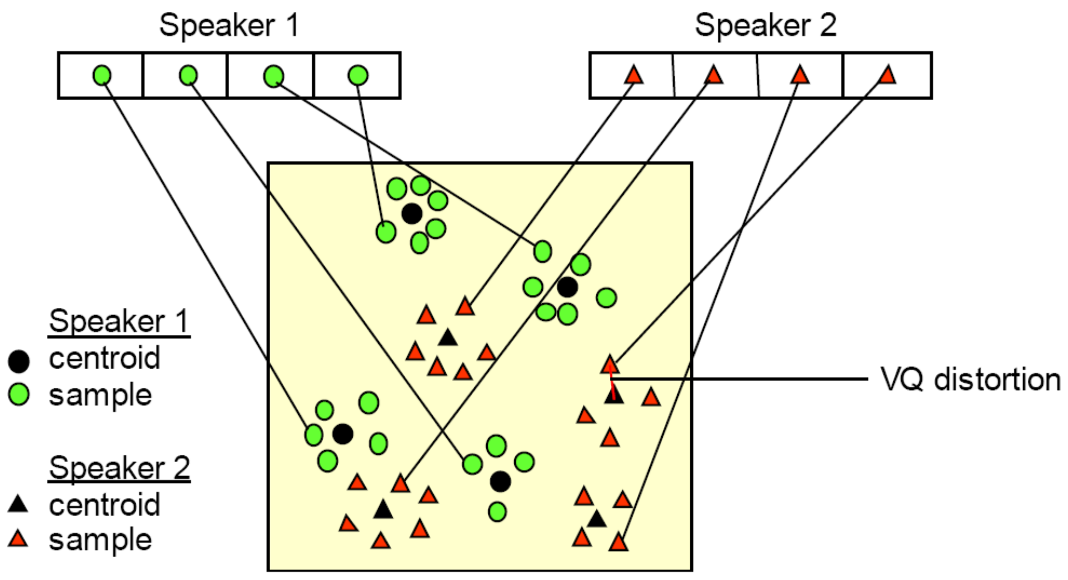

Figure 5.1: Vector Quantization using two speakers (Do 2003).

VQ is used to form k non-overlapping clusters of the feature vectors. Each cluster is represented by a *code vector* C_k or centroid. The set of code vectors for a speaker is then known as the code book and serves to model the speaker.

Mathematically, VQ can be defined as:

$$\begin{aligned} X &= \overline{x}_1, \overline{x}_2, \ldots, \overline{x}_T \\ C &= \overline{c}_1, \overline{c}_2, \ldots, \overline{c}_k \end{aligned}$$

where a centroid c_n is

$$\overline{c}_n = \frac{\sum_{\overline{x}_m \epsilon S_n} \overline{x}_m}{\sum_{\overline{x}_m \exists S_n} 1} \tag{5.1}$$

Where X is the set of feature vectors for a specific speaker, C is the corresponding feature codebook and S_n are the vectors that represent a particular cluster.

The aim of the codebook C is to model a specific speaker by reducing the amount of data in speech signal, while preserving the distribution and essential information contained in the speaker's voice. Figure 5.2 illustrates a speakers codebook (k = 32) showing the centroids as reconstructed LPC signals. Each of these signals represents the information used to identify a speaker.

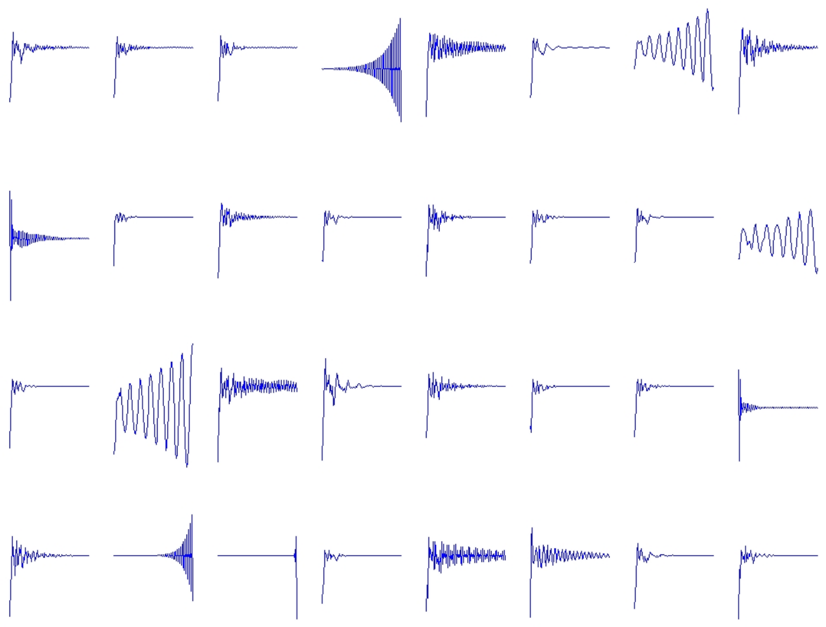

Figure 5.2: Representation of a codebook (k=32) using 20 dimensional LPC reconstructed code vectors

VQ can be decomposed into two operations, the *vector encoder* $E(x)$ and *vector decoder* $D(x)$. These operations can occur in R^n and together can be used to create specific speaker codebooks. This process can be regarded as

$$\begin{aligned} E(x) &: R^n \longrightarrow C \\ D(x) &: C \longrightarrow R^n \end{aligned} \tag{5.2}$$

where C is the codebook.

In the context of speaker identification vector quantization will match a code vector c with an approximate and similar representation of the input vector x. The vector x

may represent a number of different possible feature extraction parameters, including LPC and MFCC discussed in Chapter 3.

Generally given vector quantize, the encoder $E(x)$ must satisfy the *nearest-neighbour rule*, where each input vector x is quantized to an output vector c which is at least as close to x as any other code vector. The measure between the, vector is assumed to be the squared error distortion

To assess the quality of codebooks, a performance measure is used, normally a distortion function $d(x, y)$, which measures the distortion between two vectors, in this case the code vector c and input vector x. A common distortion measure used is the *mean squared error* or squared Euclidean distance metric(Eq. 3.17)

$$d(x, y) = \| x - y \|^2 \tag{5.3}$$

Another useful distortion function is the weighted squared error function

$$d(x, y) = (x - y)^t W (x - y) \tag{5.4}$$

where W is a symmetric non-negative definite matrix.

The main disadvantage of VQ is that clusters cannot overlap, and such interconnections between clusters cannot be modelled. Simply features vectors can only be associated with one cluster, and partially with many. This ability to allowed clusters to overlap is an advantage of using probability methods such as GMM, as this allows for interconnection.

VQ is used to model speaker-specific characteristics, how well this is achieved is based on the generation method and the size of the codebook. Two issues which affect the accuracy of the codebook to model a speaker, include:

- Size of the code book (e.g. Typical sizes include 64, 128, 256 code vectors).
- Algorithm used to generate the codebook.

Research shows that the size of the codebook used and directly effect the accuracy results (refer to section 2. Generally increasing the codebook size reduces the error rates, however the codebook can be set to high as to *overfit* the data. This overfitting is artefact from using a large codebook with training data. The codebook would model the specific training data, more than the general speaker specific characteristics.

The second important aspect of VQ is the algorithm used to create the codebook. These can be classed into supervised and unsupervised generation of the codebook. Supervised codebook generation involves creation of the codebooks independent of other speakers codebook. Unsupervised codebook generation uses the intercorrelations between the codebooks to minimize any overlap between them.(Reference) The most popular method used in speaker identification systems is that of the unsupervised methods, as supervised can become complex and involve tweaking of certain parameters. For the purpose of this project, supervised algorithms will be considered only.

The following list shows a number of clustering methods used for codebook generation:

- GLA: Generalized Lloyd(Voronoi) Aalgorithm, starts with an initial codebook, which is iteratively improved until a local minimum is reached (Kinnunen et al. 2004).
- SOM: Self-Organising Maps, uses neural network concepts, to develop code vectors.
- PNN: Pairwise Nearest Neighbour, creates the codebook hierarchically.
- Split: Iterative splitting technique, top down-approach starting with a single cluster including all feature vectors.
- RLS: Randomized Local Search, starts with a random codebook, which is improve iteratively.

Two main approaches are used in VQ, for creating the codebooks, they are iterative methods and hierarchical methods.

5.2.1 Iterative Clustering Methods

Iterative methods are those where an initial solution is iteratively improved to produce an optimal solution. These improvements can be random (RLS) or descendent (GLA) where certain rules are used to improve the codebook. However it is important for some methods to start with an initial codebook that will allow for the best globally optimal solution.

5.2.2 Initial Codebook

Vector quantization requires an initial codebook in order to begin optimising based on a speaker features. There are a number of different methods for selecting these initial codebooks, with varying effects on the solution, such as whether the codebook is globally or locally optimal. Four methods of selecting the initial codebooks are:

- Random generation of the codebook,
- Splitting (discussed in section 5.2.7),
- Create codebook with different subset using the appropriate method,
- Pairwise Nearest Neighbour (discussed in section 5.2.6),

5.2.3 Generalized Lloyd Algorithm (k-means)

Many different supervised algorithms exist which are variations on the basic k-means algorithm, also known as the *Generalized Lloyd algorithm(GLA)*(reference) or *Linde-Buzo-Gray algorithm*(reference).

The k-means algorithm is used to find k clusters of N data points R^n, with each cluster represented by k centroids. The clusters are found by starting with k tentative centroids and allocating each point to the closest centroid, which is based on a particular distance metric (Euclidean in k-means). The centroids are then recalculated to by taking the mean of the associated data points for each centroid and allocating this, as the new centroid. The closest data points are allocated to these new centroids and the process is continued iteratively until each of the centroids converge. The aim of the k-means algorithm is to minimize an objective function, in this case J

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \|x_i^{(j)} - c_j\|^2 \tag{5.5}$$

Where the x_i is the data point and c_j is the cluster centre or centroid and the distance measure $\|x_i^{(j)} - c_j\|^2$.

k-means is guaranteed to find the local minimum, however this may not be the global minimum and therefore the algorithm is not always optimal. Problems also exist in the initial centroids or codebook to begin with and the criteria used to stop the iterations. k-means is the simplest of methods of implementing VQ, but forms the basis for related algorithms are based upon.

5.2.4 Randomized Local Search

Randomized Local Search (RLS) using a random codebook, which is iteratively improved by a predefined number of iterations. The codebook is created using a *random swap* technique, in which a randomly chosen code vector c is replaced by another randomly chose input vector x. Two iterations of GLA are then applied to fine-tune the trail solution. The codebook is accepted if the solution improves the previous solution and is repeated for a fixed number of iterations (Kinnunen et al. 2004).

The random swap occurs as follows:

$$c_j \longleftarrow x_i \mid j = random(1, M),\ i = random(1, N) \tag{5.6}$$

where M and N are the size of the codebook and feature space (input vectors) respectively. The new codebook is accepted if:

$$d(x_i, c_k') \ < \ d(x_i, c_k) \tag{5.7}$$

That is, the new codebook gives a smaller distortion than the original. This codebook is then accepted and the process can continue again, until the require number of iterations has elapsed. An important note is that when the random swap occurs, the codebook vector that is replaced, ceases to exist. The final solution using RLS can be improved by using the k-means algorithm described previously.

5.2.5 Hierarchical Clustering Methods

Hierarchical methods of forming codebooks, involve building the cluster structure step-wise. Two hierarchical methods are the splitting (top-down) and the merge-based (bottom-up). The splitting approach increases the codebook size by new code vectors until the size required is reached. These are normally started from an initial codebook of one. Merge-based approach is the opposite of this, it starts with a large codebook, which decrease by removing existing code vectors and merging clusters. Pairwise nearest neighbour and Split are two common approaches to VQ by hierarchical means.

5.2.6 Pairwise Nearest Neighbor

Pairwise nearest neighbour (PNN) generates the codebook hierarchically, by initializing every training vector as a code vector. Two code vectors which increase the distortion the least are merged at each step until the desired size codebook is obtained.

Figure 5.3 shows the basic idea of PNN. Here each of the feature vectors are allocated to a particular centroid. Two centroids or code vectors are then chosen which produce the lowest increase in distortion. These two code vectors are then merged and the result is a new cluster with the total codebook decreased by one. This process would be repeated until the required size codebook is obtained.

Using the PNN approach to VQ, can become computationally expensive for large data sets compared to k-means, and other hierarchically methods. The PNN method of VQ is not guaranteed to generate the most optimal codebook for a feature space, however in practice it generates good codebooks for data sets (Teo & Garfinkle 1998).

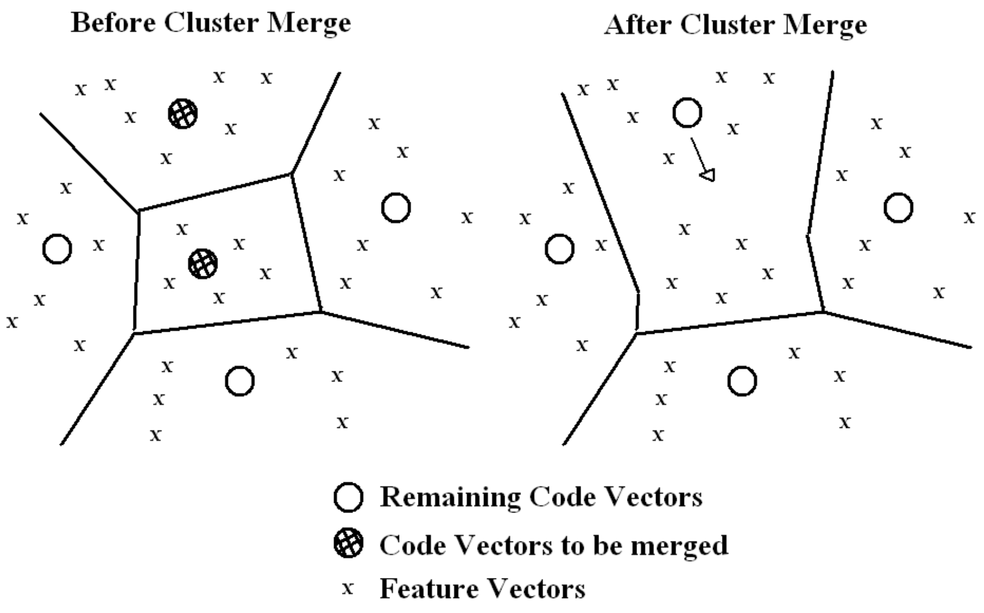

Figure 5.3: Pairwise Nearest Neighbor VQ

5.2.7 Split

This is essentially the opposite of PNN, where the codebook is started as a single code vector. Each code vector is then split to give the best distortion and optimized using k-means. This process completed until the required codebook is reached.

Figure 5.4 shows the process of splitting single initial code vector. Generally the mean of the feature vectors is used as the initial codebook. The code vector is split by a perturb $\pm \epsilon$ distance which is fixed over the entire duration of the algorithm.

The splitting approach is more efficient than PNN due to the nature of the exponential rate at which the code book increases (Teo & Garfinkle 1998). The number of iterations obtained a desired codebook is the logarithm to base 2 of the number of elements require.

$$\log_2 N = i \tag{5.8}$$

where i is the number of iterations to produce a codebook of size N

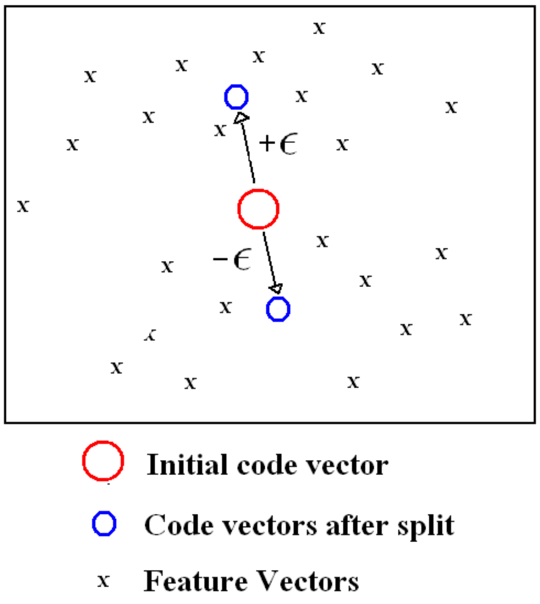

Figure 5.4: Pairwise Nearest Neighbour VQ

5.2.8 VQ Distortion

VQ will produce a codebook which can be measure to the degree at which it represents the training feature vectors. This is known as the distortion measure, and uses techniques discussed in section 3.5. The dissimilarity between the codebook C_i and feature vectors X used to create the codebook is:

$$d(X, C_i) = \frac{1}{L} \sum_{j=1}^{L} min_{k=1}^{K} \; d(x_j, c_{ik}) \tag{5.9}$$

where $d(x_j, c_{ik})$ is the distance metric used to create the codebook. This distortion measure is also known as the Mean Square Error (MSE).

5.3 Chapter Summary

This chapter has introduced VQ as a classification technique that is able to take feature vectors and convert them into a feature space that can be used for identification of speakers.

There are many algorithms that can be used in VQ, this section has presented four of the most commonly used methods including k-means, hierarchical and random methods. Generally these methods can be used together to produce better results, however results do not increase significantly from one method to another as shown by Kinnunen et al. (2004)

Chapter 6

Experimental Setup

6.1 Introduction

Experimentation is an important procedure in which techniques and algorithms are explored and assessed. This will determine the viability of such techniques for speaker identification.

6.2 Speech Corpora

For the evaluation of the speaker identification methods, two speech corpora were used, namely the NTIMIT corpus and an Alternative corpus created specifically for this project.

6.2.1 NTIMIT

NTIMIT is an American English corpus based on the original TIMIT corpus. The NTIMIT corpus is a variation on the TIMIT corpus, where the speech is played through telephone lines, which reduces the quality and information in the speech. The corpus consists of 8 dialect regions, with 10 speech files for each speaker. Two of these files (the SA sentences) contain the same linguistic content for all speakers and are meant to expose the dialectal variants of the speakers (UPenn 1990). Five sentences (the SX sentences) are phonetically-compact and three sentences (the SI sentences) are phonetically diverse (UPenn 1990).

In order to make comparisons between the corpora, 12 speakers were arbitrary from dialect region DR3. Three of these were female and seven were male. For training purposes 8 files were used, the SX and SI sentences, 2 files were used for testing, SA sentences. The training sentences together were on average 18 seconds, while the test sentences were 6 seconds.

6.2.2 Alternative Corpus

An alternative corpus was used to compare and contrast between the results of the different corpora. This corpus was created using Australian English. The recordings were taking in a low-level noise room with an inexpensive microphone. Three recordings were taken:

- One training speech file, which last between 22 to 29 seconds depending on the speaker (see Appendix B for details)
- Two test speech files, these are sentences which range from 3 to 4 seconds (see appendix for details).

The Alternative corpus provides a reasonably accurate representation of the speech and variability of the quality which would be expected in a possible meeting environment. The advantages of such a corpus over the NTIMIT, is that the degraded quality of the NTIMIT may effect results excessively. The effect of telephone speech and the associated information loss could greatly impact results in some of the prototypes. The attributes of both corpora are shown in Table 6.1

Speaker identification prototype is to be used with 8 voice profiles. In these experiments we are assessing the methods using 12 speakers to add some flexibility in the indented

Table 6.1: Summary of Corpora

	NTIMIT	Alternative
Language	American English	Australian English
Speakers	12(7M,5F)	10(7M,5F)
Speech Type	Read Speech	Read Speech
Recording Conditions	Nosiy	Clean
Sampling Frequency	16 kHz	16 kHz
Training Speech	18s	22s
Evaluation Speech	6s	3s

use. The testing speech is also completed using evaluation speech of between 3 and 6 seconds, to comply with aim of using utterances of less than 10 words.

6.3 Testing Procedures

This section examines the procedures that will be used in the experiments. This includes:

- pre-processing;
- feature extraction; and
- speaker modelling.

6.3.1 Pre-processing

Pre-processing will be applied for each of the experiments, with the following combination.

1. DC offset removal (via the removal of the mean).
2. Silence removal.
3. Pre-emphasis using $a = 0.95$.

4. 20ms Hamming window, overlapped by 10ms.
5. Autocorrelation .

These pre-processing techniques are applied to both speech corpora before feature extraction.

6.3.2 Feature Extraction

The experimentation will be used to assess the most effective feature extraction method. Three methods will be assess from Chapter 5.

- LPC
- MFCC
- LPCC

Each of these methods will be assessed by varying the order of coefficients from 1 to 20. This will be used to asses the quality of each method.

6.3.3 Speaker Model

Speaker models will be created using VQ (GLA method). VQ will be used with codebook sizes of 32, 64, 128 and 256. The Euclidean distance metric will be used to find the distortion between data points.

Results for other VQ implementations have shown that accuracy does not improve significantly (Kinnunen et al. 2004) and therefore will not be assessed in this experiment.

6.3.4 Speech Corpus

The NTIMIT and Alternative speech corpora will be used to evaluate the methods discussed above, using 12 speakers from each, as seen in Table 6.1.

The NTIMIT corpus will also be used to assess the effect of increasing the speakers in that dataset. This will be achieved by increase the number of speakers in the dataset from 1 through to 40 and determining the accuracy result at each level.

6.4 Prototype

A prototype will be recommended based on the accuracy results obtained in these experiments.

Chapter 7

Results

7.1 Results of Feature Extraction Methods

This section presents the results obtained using different feature extraction methods, for codebook sizes of 32, 64, 128 and 256. The feature extraction methods evaluated include

- Linear Predictive Coefficients (LPC)
- Mel Frequency Cepstral Coefficients (MFCC)
- Linear Predictive Cepstral Coefficients (LPCC)

The results for the different codebook sizes and features, are shown for both the NTIMIT and Alternative corpora.

Euclidean distance metric is used to determine the distortion between two feature vectors, for both the codebook generation (training) and the identification phase. The pre-processing stages are the same as those described in chapter 6 (silence removal, DC offset removed, 20ms Hamming window, auto-correlation, 10ms overlap, pre-emphasis).

7.1.1 Linear Predictive Coefficients

The first method evaluated uses LPC to derived features. LPC is rarely used directly by themselves and they serve as a basis for comparison between other methods. The results are shown in Figures 7.1 7.2 7.3 and 7.4

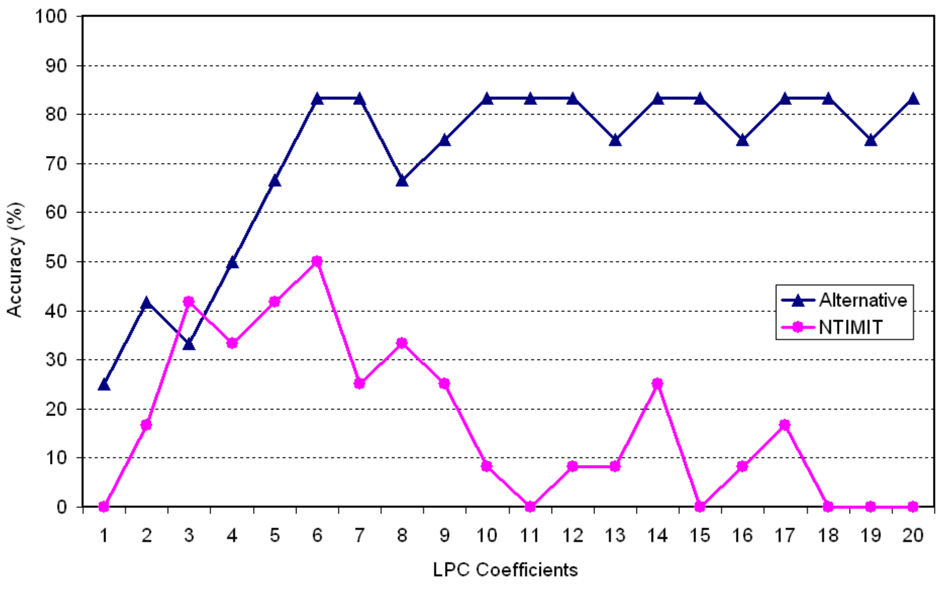

Figure 7.1: Performance of the LPC on the NTIMIT and Alternative corpus for codebook size (k=32)

As shown in Figure 7.1 there is an effect of varying the order of LPC with a codebook size of 32. A prominent observation of these results it that the difference in the accuracy obtained between the NTIMIT and Alternative corpora. The accuracy results increase for both corpora for the first 6 LPC coefficients. The results for the NTIMIT corpus then dramatically decrease until they are completely inaccurate, while the Alternative corpus continues to increase, oscillating at 80% accuracy.

These results are not totally unexpected and can be explained by two significant factors: the quality the speech signal combined with the quality of the codebook and LPC features. The size of the codebook and LPC parameters have also effected the results. Increasing the number of LPC parameters may also increase the effect of noise, as the coefficients have less signal information. This allows the noise to become a mitigating factor. It would be expected that results for an increased sized codebook would produce better results for both the NTIMIT and Alternative corpora. The training data for the both corpora is unlikely to cause a difference between these recognition results, as speech times were similar (see Table 6.1).

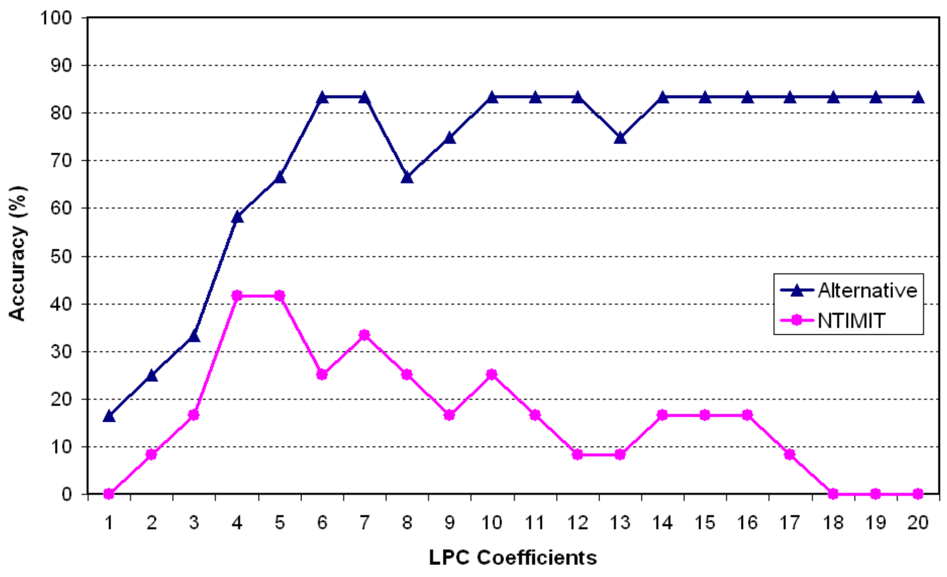

Figure 7.2: Performance of the LPC on the NTIMIT and Alternative corpus for codebook size (k=64)

The results shown in Figure 7.2 using a codebook of 64, are very similar to those in Figure 7.1, using a codebook of 32. Again the NTIMIT performs poorly, while the Alternative corpus accuracy results stabilise with more than 10-dimensional LPC.

Figure 7.3 shows similar results using a codebook of 128, as those using a codebook of 32. The main difference here is the slight increase in the accuracy found using the NTIMIT corpus. However, these results are still poor, as they continually decrease after the first three LPC. Again noise is a mitigating factor, but the increase in the codebook size has enhanced accuracy results. The alternative corpus remains around the 80% accuracy mark.

Finally using a codebook size of 256, Figure 7.4 shows results that almost mimic those of Figure 7.4, with only slight increases overall for both corpora.

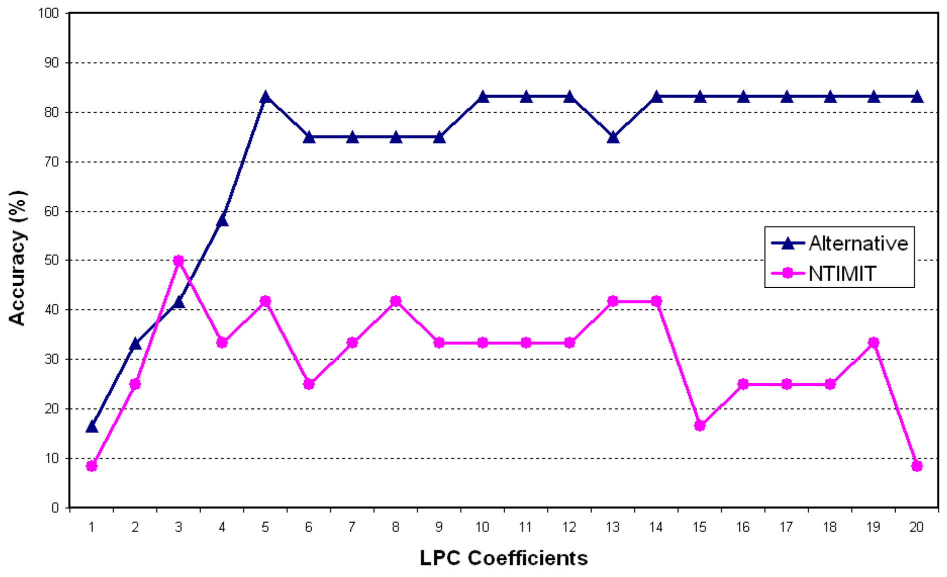

Figure 7.3: Performance of the LPC on the NTIMIT and Alternative corpus for codebook size (k=128)

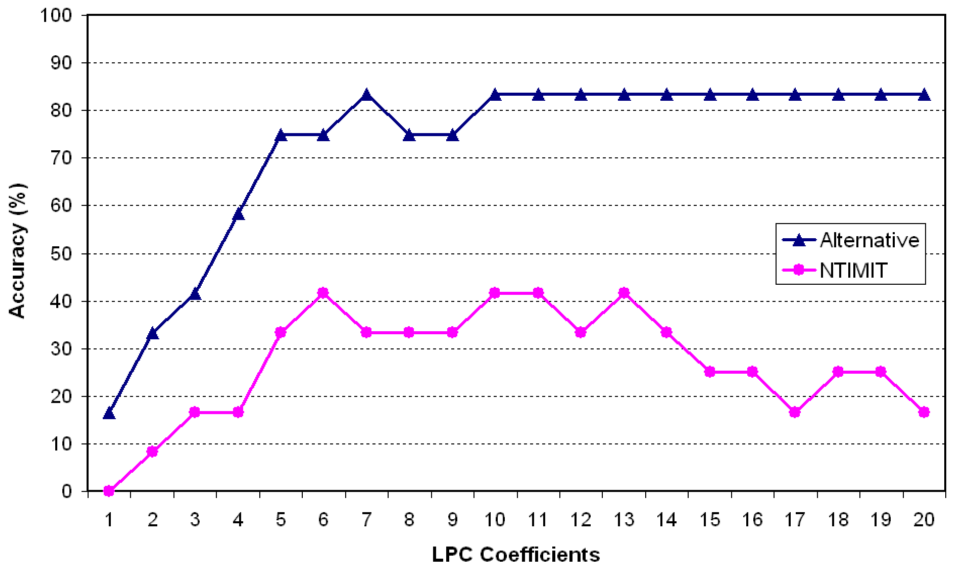

Figure 7.4: Performance of the LPC on the NTIMIT and Alternative corpus for codebook size (k=256)

Conclusion

Results using LPC on the Alternative corpus are similar to those discussed in chapter Chapter 2. These results stabilised using 14^{th} order LPC and a codebook of size 64. Conversely, the results for the NTIMIT were very poor due in part, to the quality of the speech signal. This reduction in quality caused the LPC to inaccurately capture speaker specific characteristics.

Results were also found using LPC coefficients as the feature extraction method for both clean and noisy signals. The results are far below the desirable, even with only 12 speakers. The findings show that LPC is especially poor for noisy signals however, moderate results are produced with relatively clean speech.

Variations in the number of coefficients used, codebook size, training and testing speech could also be altered to possible improve the recognition results. However, this would not be significant enough to warrant such investigation (Kinnunen et al. 2004). Instead another approach to feature extraction is implemented, namely MFCC.

7.1.2 Mel-Frequency Cepstral Coefficients

The second method evaluates how cepstral analysis derives features from the speech frames. In the case of the cepstrum, the frequency axis is warped by a mel-scale which produce the MFCC. This method is commonly used with probability methods of analysis such as GMM. However here, VQ is used to determine MFCC suitable for speaker identification. The results of this process are shown in Figures 7.5 7.6 7.7 and 7.8

Figure 7.5 shows the effect of using MFCC with a codebook size of 32. These results are similar to those obtained in Figure 7.1, with reasonable results using the Alternative corpus and poorer results for the NTIMIT. However the MFCC do seem to give moderately better results for the NTMIT corpus with an average of approximately 50% accuracy. This shows that the MFCC are more resistant to the amount of noise contained in the signal than to LPC.

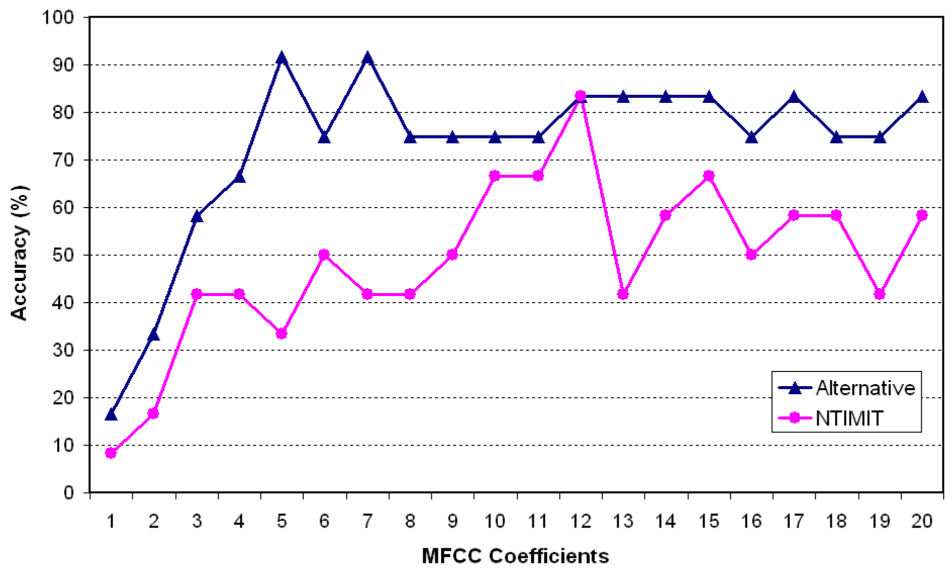

Figure 7.5: Performance of the MFCC on the NTIMIT and Alternative corpus for codebook size (k=32)

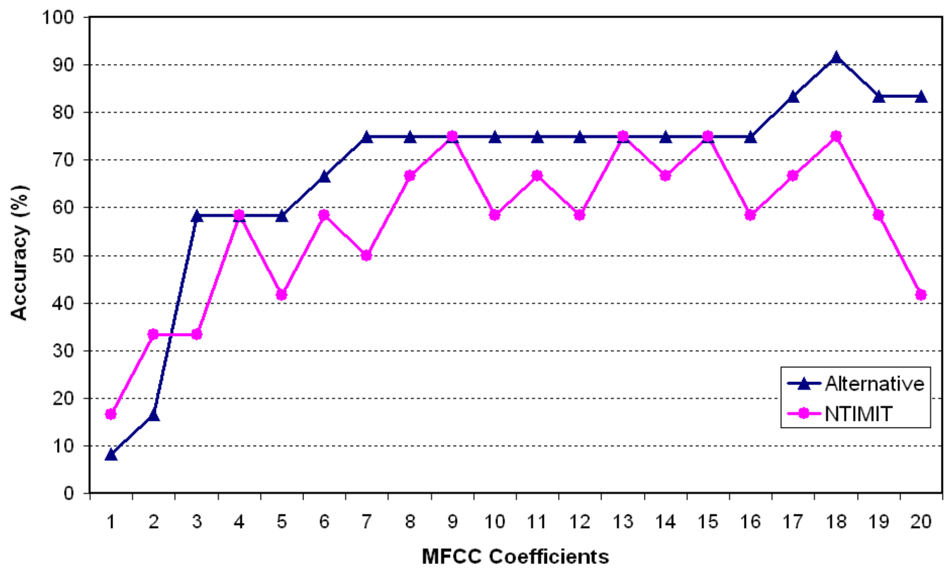

Figure 7.6: Performance of the MFCC on the NTIMIT and Alternative corpus for codebook size (k=64)

Using a codebook of 64 with MFCCs produces the result shown in Figure 7.6. Here the main change has occured in the accuracy results from the NTIMIT corpus which have increasing slightly, with a sharp decline using 20 MFCC. The NTIMIT accuracy results also oscillate greatly around 70%. The Alternative corpus results are slightly worse in this case, increasing around 17 MFCC.

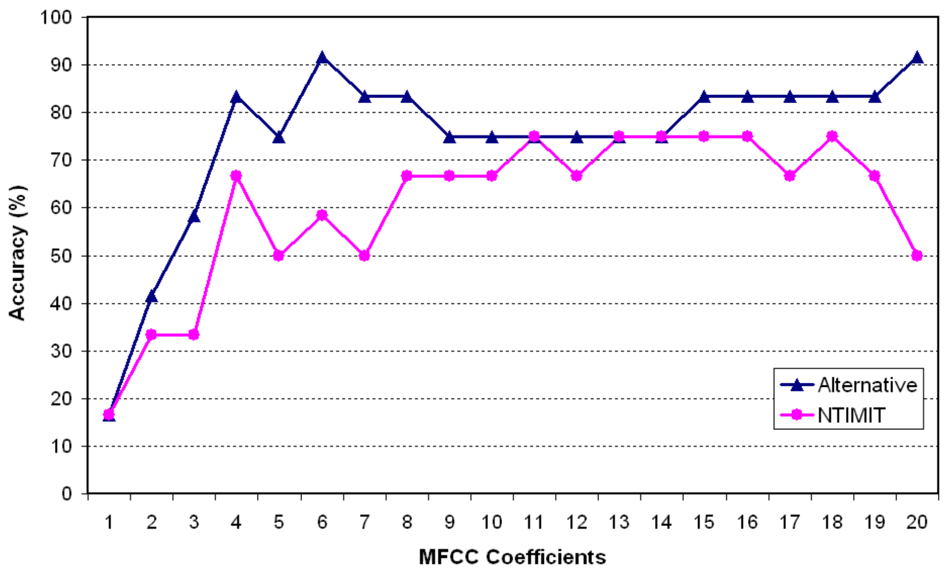

Figure 7.7: Performance of the MFCC on the NTIMIT and Alternative corpus for codebook size (k=128)

Figure 7.7 shows the accuracy results using a codebook of 128 with MFCC. The main changes include a greater accuracy result in the Alternative corpus early in the order (4th order) of MFCC and a stabilising effect on the NTIMIT corpus at about 70%.

Lastly Figure 7.8 shows results using MFCC with a codebook size of 256. These results are almost identical to those obtained for a codebook size of 128, with just a few accuracy points changing. This illustrates that increasing the size of the codebook from 128 to 256 does not significantly improve the result. In fact increasing the codebook size any more than 256 is likely to result in overfitting, causing a decrease in performance.

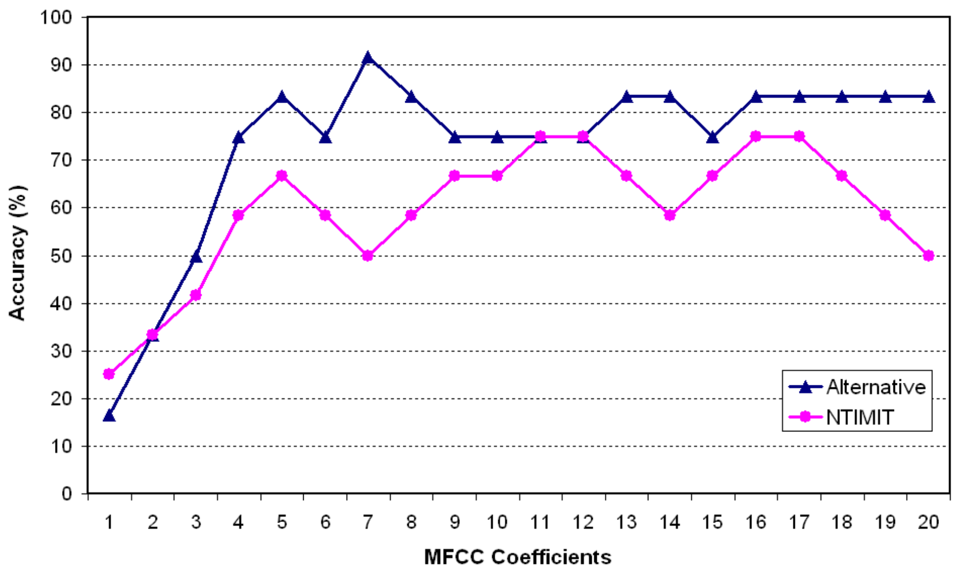

Figure 7.8: Performance of the MFCC on the NTIMIT and Alternative corpus for codebook size (k=256)

Conclusion

The results for MFCC indicated a significant increase in accuracy over LPC for the NTIMIT corpus. These results improved from around 40% to 70% with a codebook of 128 and using 14 or more coefficients. This illustrates that the MFCC are better equipped for modelling noisy and degraded signals than LPC. The results for the Alternative corpus are very similar for MFCC and LPC. These are slightly less with MFCC approximately 75% compared with 80% for LPC.

Again these results have demonstrated that the number of coefficients and codebook size does effect the accuracy results. The results from both the experiments using LPC and MFCC, with different codebooks show that results are not significantly improved using more then 14 coefficients and a codebook of greater than 128. A combination of these two methods is demonstrated next, which hopes to improve accuracy results.

7.1.3 Linear Predictive Cepstral Coefficeients

The final method evaluates LPCC features, in which the MFCC are taken from the reconstructed signal created by the LPC. This method aims to combine the benefits of both methods to produce speaker specific features that improve identification results. The results shown in Figures 7.9 7.10 7.11 and 7.12

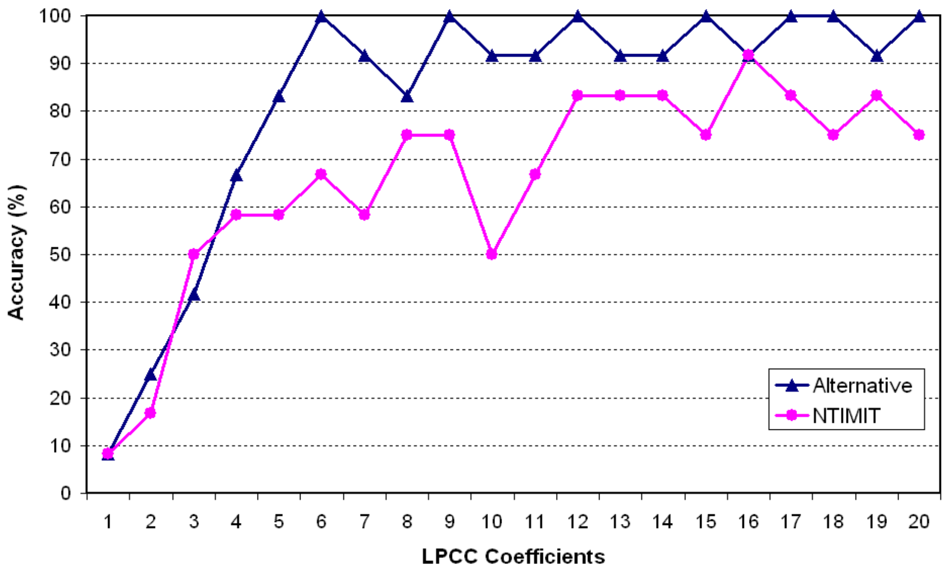

Figure 7.9: Performance of the LPCC on the NTIMIT and alternative corpus for codebook size (k=32)

Figure 7.9 shows the results for a codebook size of 32 using LPCC. These accuracy results are already significantly Improved over LPC and MFCC features. The Alternative corpus after only 6 LPCC starts to oscillate around 95%. This improves greatly over LPC and MFCC features for a codebook size of 32. The NTIMIT corpus is also significantly better with accuracy of around 80% after 12 LPCC. This method indicates that the LPCC method is far better equipped for use with VQ using both clean and noisy speech.

Figure 7.10 shows improved results using a codebook of 64 for both the NTIMIT and Alternative corpora. Importantly here the Alternative corpus is stable with 100% accuracy using 13 LPCC. This demonstrates the improvement in both the LPCC technique and the codebook size on accuracy results. The NTIMIT corpus has also improved

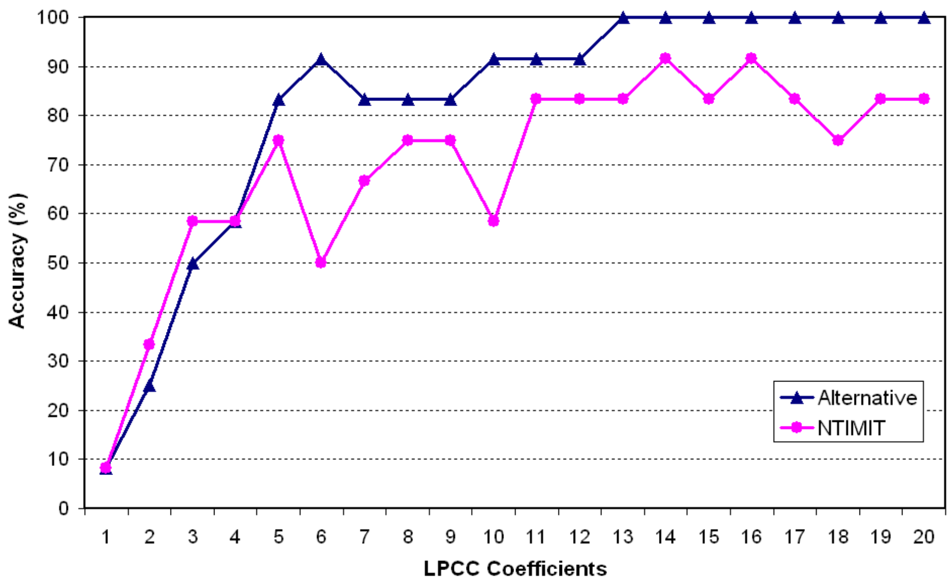

Figure 7.10: Performance of the LPCC on the NTIMIT and alternative corpus for codebook size (k=64)

with stable results after 10 LPCC.

The results shown in Figure 7.11 using a codebook of 128 shows improvements in the stability of the accuracy results for both corpora. This illustrates that increasing the codebook size from 64 to 128 may not significantly increase the overall results after 14 LPCC. However, it does make the overall accuracy results improve incrementally preventing oscillation with both lower and higher order coefficients.

LPCC are demonstrating in Figure 7.12 using a codebook of 256. These results are almost identical to those used with 128 for both corpora. Slight improvements occur for lower coefficients. The NTIMIT corpus reaches an accuracy results of 92%, with the Alternative corpus still at 100%. Results indicate using 14^{th} order LPCC for both speech corpora yield the best results.

Conclusion

LPCC produced the best performance by far, for both the NTIMIT and Alternative corpora. The accuracy results for the NTIMIT corpus peaked at 92%, using 14 LPCC

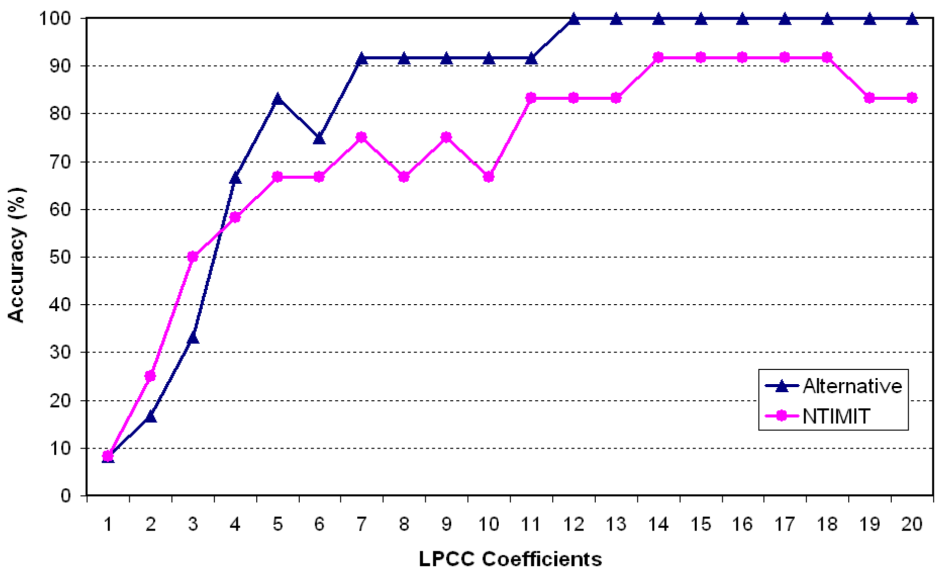

Figure 7.11: Performance of the LPCC on the NTIMIT and alternative corpus for codebook size (k=128)

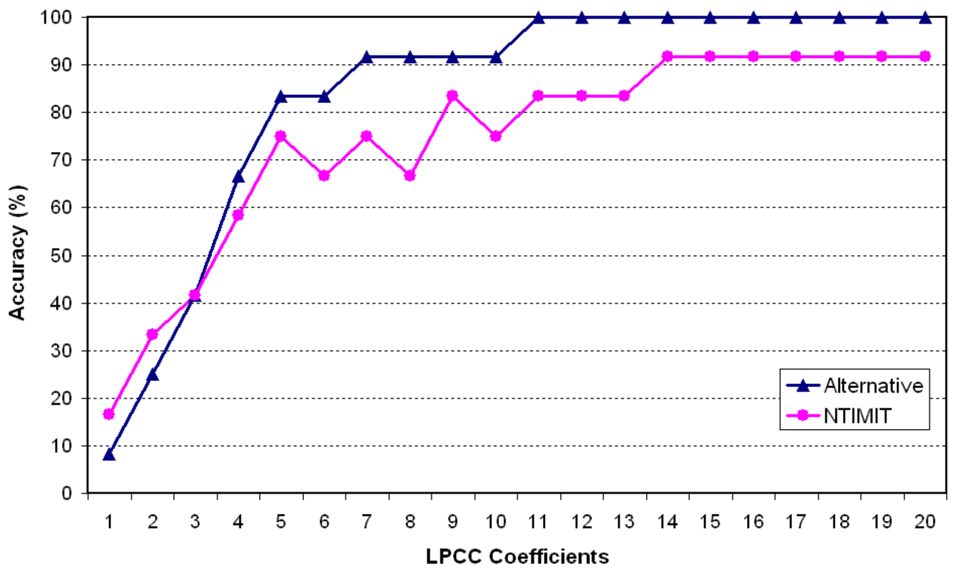

Figure 7.12: Performance of the LPCC on the NTIMIT and alternative corpus for codebook size (k=256)

with a codebook of 256. For the Alternative corpus the accuracy results peaked using 13 LPCC with a codebook of just 64. These results indicate that when determining what parameters a speaker identification system should use, the quality of the speech signal is critical to its performance.

7.1.4 Comparison of Feature Extraction Methods

This section compares the different feature extraction methods, using 14 and 20 coefficients. Most speaker recognition systems used between 10 to 20 dimension feature vectors to model a speakers vocal characteristics. Figure 7.13 and 7.14 show a comparison of the different types of coefficients.

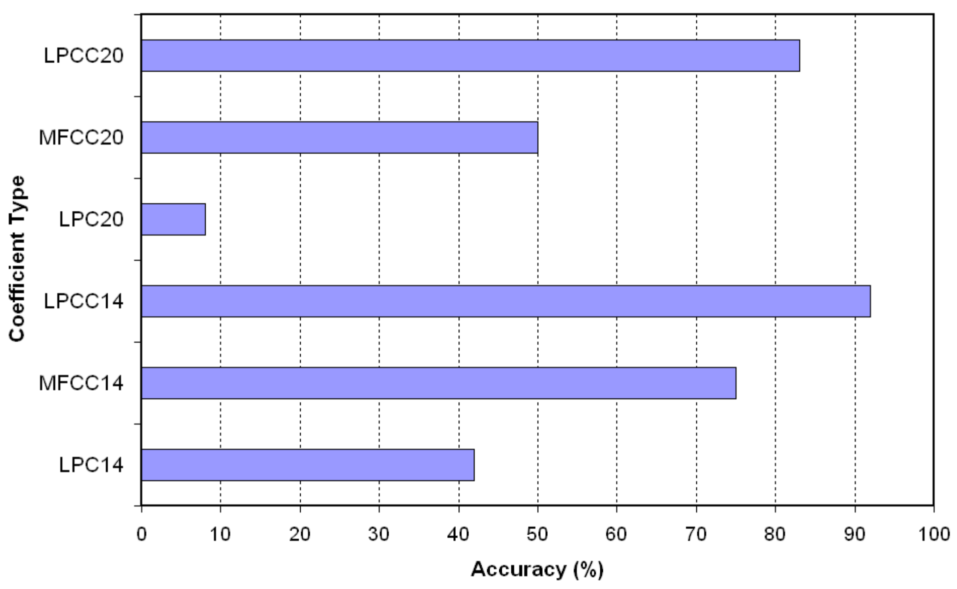

Figure 7.13: Comparison of Accuracy and Coefficient Type using NTIMIT corpus

Figure 7.13 displays accuracy results using the NTIMIT with 14 and 20 dimensional LPC, MFCC and LPCC. The performance using LPCC easily out performs the other extraction methods, while LPC performs poorly. Surprisingly the results for LPC get worse for high dimensional coefficients. The apparent loss of information can be explained by the noise in the speech signal. Since the lower coefficients (higher dimensional coefficients) will have progressively less information, the noise will have an increasing detrimental effect on the lower the coefficients.

The use of MFCC produces results that are a significant improvement over LPC. MFCC are the result of the approximate deconvolution of the source and filter. These results are less sensitive to noise, as the noise and filter characteristics are separated from the source.

Finally using LPCC produce results that best model the speech. Reconstructing the signal from the LPC coefficients results in a speech signal which models the fundamental formants of the speaker. By applying cepstral analysis, the signal is separate into the source and filter parts, which is less sensitive to noisy. This is due to both the reconstructed signal and cepstral analysis.

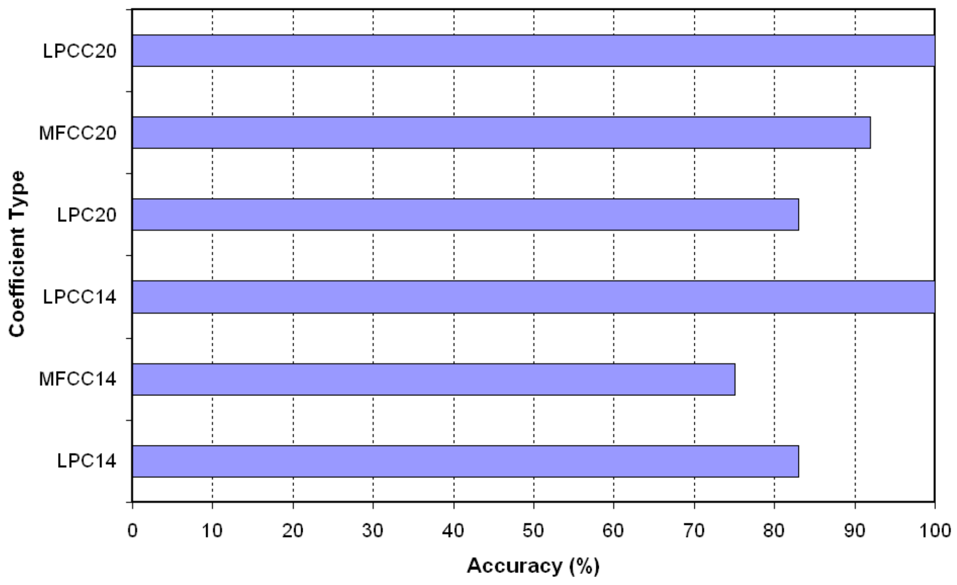

Figure 7.14: Comparison of Accuracy and Coefficient Type using Alternative corpus

Figure 7.14 shows accuracy results for the Alternative corpus similar to that for the NTIMIT. The obvious change here is the increased accuracy obtained for the 14 and 20 dimensional LPC, MFCC and LPCC coefficients. This is a direct result of the quality of the speech signal used in the Alternative corpus compared with that of the NTIMIT.

LPCC produces easily the best results for both the NTIMIT and Alternative corpus. VQ using LPCC features produces 100% accuracy with the 12 speakers in the Alternative corpus.

7.1.5 Effects of Accuracy Results by Increasing Speakers

It is also of interest to see the effect increasing the number of speakers on the accuracy results of identification. Here we use the NTIMIT corpus only for these results increasing the number of speakers to 40. The alternative corpus has a limited number of speakers and could not be used for this experiment. The system used to assess the effect of increasing speakers, is the 20 dimensional LPCC technique of feature extraction with a codebook size of 128.

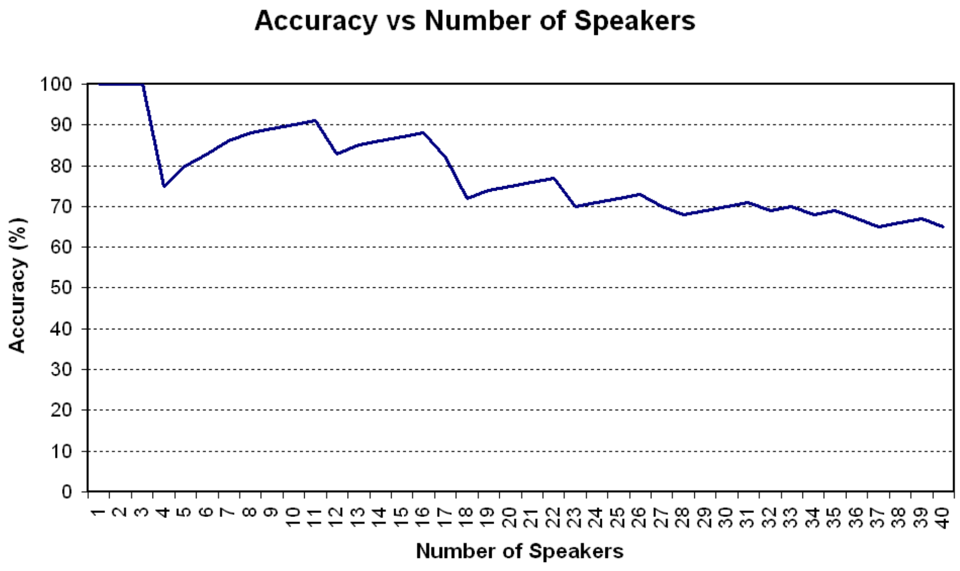

Figure 7.15: The effect of increasing speakers on the performance of a VQ(codebook 128) with 20 dimensional LPCC

Figure 7.15 shows the effect of increasing the speakers on performance of the speaker identification system. Accuracy starts off highly as would be expected, and slowly declines to approximately 65%. It would be expect that the results would plateau around 60%, which is congruent with accuracy results found in Reynolds et al. (1995). These performance results illustrate that the number of speakers in the database significantly influences the results. The performance of only 8 speakers in the database using the NTIMIT corpus would at worst produce an accuracy result of 88%. These results serve to show a worse case scenario using VQ with LPCC. Ultimately performance in the purposed meeting environment would produce results that closely match those obtained using the Alternative corpus, rather than the NIMIT corpus.

7.2 Chapter Summary

The experiment results have produced the following findings:

- LPC performs poorly with noisy speech signals.
- Codebook sizes greater than 128 do not significantly improve results.
- Coefficients greater than 14, do not significantly improve results.
- Overall LPC performance is the least effective method to extract features.
- The most effective method for extracting features are LPCC.

From the experiment results, the preposed prototype will used a codebook of 128, with 14-dimensional LPCC. These parameters produce the most practically effective results.

Chapter 8

Conclusion

8.1 Summary of Work

Before a speech signal can be used for speaker identification, pre-processing is required to improve the speech for feature extraction. These techniques involve DC offset removal, silence detection, pre-emphasis, windowing and autocorrelation. This leaves the signal ready for use with speaker identification.

The fundamental process in speaker identification is feature extraction. This involves extraction of speaker specific features from the speech signal. These features can be extracted using LPC, MFCC and LPCC. The coefficients form the feature space of a speaker, which can then be used to create a speaker model.

VQ is used to create speaker models from the feature space of a speaker. The speaker model or codebook is then used in the identification stage. The codebook is compare via a distance metric to the feature space of an unknown speaker. The best match in the speaker database is found, indicating the identify of the speaker

8.2 Conclusions

This dissertation has presented the design and development of a speaker identification system, tailored for use with utterances of less than ten words and a speaker database of eight voice profiles. This system is intended for use with a dual-type ASR, currently being researched by the DSTO.

The experiment results identify which methods, used commonly in speaker recognition, produce the best performance under the conditions describe for this project. The proposed design of the speaker identification system uses 14 dimensional LPCC with a VQ codebook of 128. The parameters of the proposed design are the result of evaluation of different speaker identifications designs, evaluated using the NTIMIT and Alternative corpora. The Alternative corpus achieved 100% accuracy using 12 speakers while the NTIMT achieved 83% accuracy. This demonstrates the performance with a slightly higher number of speakers, than needed, illustrating some flexibility for use with the dual-type ASR.

The results indicate that the speaker identification system would work well in the environment with of dual-type ASR system. The quality of the speech used with the purposed system would match that of the Alternative corpus, rather than speech from the NTIMIT. The NTIMIT speech demonstrates a worse case scenario and performance would be far greater when uses with dual-type ASR system. Results using the NTIMIT corpus, illustrate that increasing the number of speakers can significantly affect the accuracy of the system.

8.3 Further Work

1. Comparison of the prototype speaker identification system to a commercial produce such as Nuance Verifier.
2. Further work on VQ could be undertake to determine the most accurate method of creating the codebook. Kinnunen et al. (2004) has shown results using a number of VQ methods which could form a basis of investigation.

3. Collection and experimentation on a speech corpus created in the intended environment would be derisible.

4. Investigation of the effectiveness of SVM for more robust speaker identification.

References

Burges, C. J. (1988), A Tutorial on Support Vector Machines for Pattern Recognition, Technical report, Microsoft Research, Lucent Technologies.

Campbell, J. P. (1999), Speaker Recognition, Technical report, Department of Defence, Fort Meade.

Do, M. N. (2003), An Automatic Speaker Recognitio System, Technical report, Swiss Federal Institute of Technology, Lausanne Switzerland.

Drygajlo, A. (2005), Biometrics, Technical report, University of Lausanne.

Ethicity Group (1998), 'LPC for Speech Recognition'. `http://www.owlnet.rice.edu/~elec532/PROJECTS98/speech/cepstrum` viewed 01/06/2006.

Gold, B. & Morgan, N. (2000), *Speech and Audio Signal Processing*, John Wiley & Sons, Inc.

Gravier, G. (2004), 'Speech Analysis Techniques'. `http://www.irisa.fr/metiss/guig/spro/spro-4.0.1/spro_2.html#SEC8` viewed 12/09/2006.

Husband, M. (2004), 'Linear Predictive Coding in Voice Conversion'. `http://cnx.org/content/m12473/latest` viewed 19/09/2006.

Iyer, A. N. et al. (2004), Speaker Identification Improvement using the Usable Speech Concept, Technical report, Temple University.

Johnson, D. (2005), 'Modeling the Speech Signal'. `http://cnx.org/content/m0049/latest` viewed 29/05/2006.

Joseph P. Campbell, J. & Reynolds, D. A. (1999), Corpora for the Evaluation of Speaker Recognition Systems, Technical report, Department of Defence, USA.

Jr, C. R. J. et al. (1995), Measuring Fine Structure in Speech: Application to Speaker Identification, *in* 'ICASSP 1995', MIT Lincoln Laboratory.

Karpov, E. (2003), Real-Time Speaker Identification, Masterś thesis, University of Joensuu.

Kinnuenen, T. et al. (2004), Real-Time Speaker Identification, Technical report, University of Joensuu.

Kinnunen, T., Kilpelainen, T. et al. (2004), Comparison of Clustering Algorithms in Speaker Identification, Technical report, Department of Computer Science, University of Joensuu, Joensuu, Finland.

Kwon, S. & Narayanan, S. (2005), Robust Short-Segment Speaker Identification base on Selective Use of Feature Vectors, Technical report, University of Southern California.

Leis, J. (2002), *Digital Signal Processing - MATLAB-Base Tutorial Approach*, Research Studies Press LTD.

Littlefield, J. & Broughton, M. (2005), Dual-Type Automatic Speech Recogniser Designs for Spoken Dialogue Systems, *in* 'Australasian Language Technology Workshop 2005', Defence Science and Technology Organisation.

Lovekin, J. M. et al. (2001), Developing Usable Speech Criteria for Speaker Identification Technology, Technical report, Temple University.

Markel, J. & Gray, A. (1976), *Linear Prediction of Speech*, Springer-Verlag, New York.

Osdol, B. V. (2004), 'Cepstrum'.
`http://cnx.org/content/m12469/latest` viewed 01/06/2006.

Parsons, P. (2006), 'Focus', *Focus Magazine* .

Phan, T. T. & Soong, T. (1999), Text-Indepentdent Speaker Identification, Technical report.

Rabiner & Juang (1999), 'Cepstrum Analysis'.
`http://shay.ecn.purdue.edu/~ee649/notes/cepstrum.html` viewed 01/06/2006.

Reynolds, D. A. (2002), An overview of Automatic Speaker Recognition Technology, *in* 'ICASSP 2002', MIT Lincoln Laboratory.

Reynolds, D. A. et al. (1995), The Effects of Telephone Transmission Degradations on Speaker Recognition Performance, *in* 'ICASSP 1995', MIT Lincoln Laboratory.

Shah, J. K., Iyer, A. N. et al. (2004), Robust Voice/Unvoiced Classification using Novel Featuresand Guassian Mixture Model, Technical report, Temple University, Philadelphia, USA.

Starnoiewicz, P. & Majewski, W. (1998), SVM Based Text-Dependent Speaker Identification for Large Set of Voices, Technical report, Department of Analysis and Processing of Acoustic Signal, Institute of Telecommunications and Acoustics, Worclaw Univesity of Technology.

Teo, P. C. & Garfinkle, C. D. (1998), Fast Resampling Using Vector Quantization, Technical report, Department of Computer Science, Standford University.

UPenn (1990), 'The DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus'.
`http://www.ldc.upenn.edu/Catalog/readme_files/timit.readme.html` viewed 03/08/2006.

Villazon, L. (2006), 'How do seabirds warm their feet', *Focus* p. 39.

Wan, V. & Renals, S. (2004), SVMSVM: Support Vector Machine Speaker Verification Methodology, Technical report, Department of Computer Science, University of Sheffield.

Willits, A. E. (2003), Low Resource Speaker Identification, Masterś thesis, University of Edinburgh.

Wu, Y.-C. (2005), 'Gaussian Mixture Model'.
`http://cnx.org/content/m13205/latest` viewed 05/06/2006.

Appendix A

Project Specification

University of Southern Queensland

FACULTY OF ENGINEERING AND SURVEYING

ENG 4111/4112 Research Project
PROJECT SPECIFICATION

FOR:	David Michael Graeme **WATTS**
TOPIC:	SPEAKER IDENTIFICATION – PROTOTYPE DEVELOPMENT AND PERFORMANCE
SUPERVISORS:	John Leis Michael Broughton & Jason Littlefield, Defence Science and Technology (DSTO)
ENROLMENT:	ENG4111 – S1, D, 2006 ENG4112 – S2, D, 2006
SPONSORSHIP:	Defence Science and Technology Organisation (DSTO)
CONDIFDENTIALITY:	UNCLASSIFIED, COMMERCIAL IN CONFIDENCE
PORJECT AIM:	The objective of this project is to develop a demonstrable set of software tools and techniques for Speaker Identification.

PROGRAMME: **Issue B, 24th July 2006**

1. Research background information relating to speaker identification systems and conduct a literature review of the published results.

2. Examine and evaluate existing speaker identification algorithms and systems available.

3. Experiment with a number of methods, using MATLAB and other suitable software, comparing the difference between clean and noisy signals.

4. Examine the effects of various techniques and parameters to maximise the speaker identification system effectiveness for the specified application.

5. Develop a prototype speaker identification system that is tailored for utterances containing less than ten words, and is able to work from target sets of less than eight voice profiles.

If time permits

6. Compare the prototype speaker identification system to a commercial product such as Nuance Verifier

AGREED : [illegible] (student) [illegible] (supervisor)

Date: 30 / 10 / 06

Appendix B

Alternative Speaker Corpus

B.1 Speech

Speakers were given a one paragraph and two sentences for the recordings. The paragraph was the same for all the speakers. This speech from this paragraph was used for training. These were taken from the June issue of Focus:

"How do seabirds warm their feet? They don't. If seabirds were to pump bloody at body temperature round their legs, they would dump their body heat into the water much faster than it was generated and die of hypothermia. So they extract the heat from the blood that's on its way down to the feet, using a contraflow system. As this cold blood returns to the body, it is warmed by the heat from the arterial blood that is heading out to the feet. The muscles (which require warm blood) are all at the top of the leg and work the toes remotely by pulling tendons."(Villazon 2006)

A number of sentences were used for testing these included (Parsons 2006):

- *"You'll find detailed test about almost everything from digital cameras to cars."*
- *"You'll get expert but jargon-free features, from complementary therapies to the lastest technology"*
- *"When I went to school it was drummer into me that the number of planets in out Solar System was nine"*
- *"It wouldn't be the first time a planet has been demoted"*

Appendix C

Source Code

C.1 The AlternativeTest.m MATLAB Function

This is used to test the Alternative corpus using 12 speakers

```
% Tests the alternative corpus
% David Watts, 2006

% Alternative speaker corpus
speechFiles1 = {'darcy1','david1','delan1','catherine1','john1','ben1','matt1','
    nick1','andrew1','anne1','rach1','wens1'};
speechFiles2 = {'darcy2','david2','delan2','catherine2','john2','ben2','matt2','
    nick2','andrew2','anne2','rach2','wens2'};

% stors all the results
allRes = [];
kiter = [32 64 64 256];
for ite = 1:4
    k = kiter(ite);
    % clears old results for next iteration
    res = [];
    for P = 20:20
        codeBook = [];
            for codeTest = 1:2
                codeTest
                % stores the feature vectors
                dataD = [];
            for files = 1:12
                % determines if this is a training or testin run
                if codeTest == 1
                    speechFiles = speechFiles1;
                    [y sam] = wavread(char(strcat(speechFiles(files),'T.wav')))
                        ;
                else
                    speechFiles = speechFiles2;
                    [y sam] = wavread(char(strcat(speechFiles(files),'R.wav')))
                        ;
                end
                % the length of the speech signal
                ny = length(y);
                % samples
                % frameSize is 200;
                frameTime = 0.020;
                frameSize = frameTime * sam;
                % for use with indexes
                fs = frameSize-1;
                % Stores the codebook for each iteration
                lpck = [];
                % for the new signal
                newsound = [];
                % for lpc
                lpcoeff = [];
                overLap = frameSize/2;
                % loops through the signal at the "frameSize"
                for f = 1:frameSize-overLap:ny-fs+1;
                    frame = y(f:f+fs);
                    % length of the frame
                    n = length(frame);
                    % Hamming window
                    h = hamming(frameSize);
                    frame = frame .* h;
                    % uncomment to use feature extraction method
                    % MFCC
                    % pc = mfcc(frame,P);
                    % LPCC
                     pc = lpcc(frame,P)
                    % LPC
                    lpck = [lpck pc];
                    % pc = real(lpc(frame,P));
                    %lpck = [lpck pc(2:P+1)'];
                end
                if (codeTest == 1)
                    % calculates codebook, not necessary for training data
                    c = kmgla(lpck',k)
                    codeBook = [codeBook c];
                end
                % stors data for use with comparison of codebooks and
                % unknown speakers
                data1 = lpck';
                switch(files)
                case(1)
                    darcy = data1;
                case(2)
                    david = data1;
                case(3)
                    delan = data1;
                case(4)
                    cat = data1;
                case(5)
                    john = data1;
                case(6)
                    ben = data1;
                case(7)
                    matt = data1;
                case(8)
                    nick = data1;
                case(9)
                    and = data1;
                case(10)
                    ann = data1;
                case(11)
                    rach = data1;
                case(12)
                    wens = data1;
                otherwise
                    -1
                end
            end
        end
        % identifies the speakers
        res=[res; dist(darcy,codeBook,P) dist(david,codeBook,P) dist(delan,
            codeBook,P) dist(cat,codeBook,P)  dist(john,codeBook,P)  ...
        dist(ben,codeBook,P) dist(matt,codeBook,P) dist(nick,codeBook,P) dist(
            and,codeBook,P) dist(ann,codeBook,P) ...
        dist(rach,codeBook,P) dist(wens,codeBook,P)];
    end
    % stores all the results
    allRes = [allRes; res];
end
```

C.2 The timitPro.m MATLAB Function

This is used to test the NTIMIT corpus using 12 speakers

```
% Tests the NTIMIT Corpus using 12 speaker
% David Watts

codeBook = [];
dataD = [];

addpath('C:\dev\project\timitmfiles');
addpath('C:\dev\project\new');


% Test NTIMIT Files
speechFiles1 = {'FALK0','SA1','SA2','SI456','SI658','SI1086','SX6','SX96','SX186
    ','SX276','SX366'};
speechFiles2 = {'FCKE0','SA1','SA2','SI481','SI1111','SI1741','SX31','SX121','
    SX211','SX301','SX391'};
speechFiles3 = {'FCMG0','SA1','SA2','SI1142','SI1242','SI1872','SX72','SX162','
    SX252','SX342','SX432'};
speechFiles4 = {'FDFB0','SA1','SA2','SI1318','SI1948','SI2010','SX58','SX148','
    SX238','SX328','SX418'};
```

```matlab
speechFiles5 = {'FDJH0','SA1','SA2','SI935','SI1565','SI2195','SX35','SX125','SX215','SX305','SX395'};
speechFiles6 = {'MADC0','SA1','SA2','SI737','SI1367','SI1997','SX17','SX107','SX197','SX287','SX377'};
speechFiles7 = {'MAKB0','SA1','SA2','SI1016','SI1646','SI2276','SX26','SX116','SX206','SX296','SX386'};
speechFiles8 = {'MAKR0','SA1','SA2','SI722','SI1352','SI1982','SX92','SX182','SX272','SX272','SX452'};
speechFiles9 = {'MAPV0','SA1','SA2','SI663','SI1293','SI1923','SX33','SX123','SX213','SX303','SX393'};
speechFiles10 = {'MBEF0','SA1','SA2','SI651','SI1281','SI1911','SX21','SX111','SX201','SX291','SX381'};
speechFiles11 = {'MCAL0','SA1','SA2','SI508','SI1138','SI1768','SX58','SX148','SX238','SX328','SX418'};
speechFiles12 = {'MCDC0','SA1','SA2','SI662','SI1292','SI1922','SX32','SX122','SX212','SX302','SX392'};

for fil = 1:12
switch(fil)
    case(1)
        speechFiles = speechFiles1;
    case(2)
        speechFiles = speechFiles2;
    case(3)
        speechFiles = speechFiles3;
    case(4)
        speechFiles = speechFiles4;
    case(5)
        speechFiles = speechFiles5;
    case(6)
        speechFiles = speechFiles6;
    case(7)
        speechFiles = speechFiles7;
    case(8)
        speechFiles = speechFiles8;
    case(9)
        speechFiles = speechFiles9;
    case(10)
        speechFiles = speechFiles10;
    case(11)
        speechFiles = speechFiles11;
    case(12)
        speechFiles = speechFiles12;
    otherwise
        -1
end

y = [];
totLen = 1;
for files = 2:11
    [insY, BufLen, sam] = nistread(char(strcat('testFiles\',speechFiles(1),'\',speechFiles(files),'.WAV')));
    if (files == 2 | files == 3 )
 y = [y (insY(10:length(insY)))'];
    else
%   y = [y (insY(10:length(insY)))'];
    end
end
k = 128;

% Original Signal
ny = length(y);
t = 1/sam:1/sam:ny/sam;
% Pre-Processing
signal = y;
len = length(signal);
t = 1/sam:1/sam:len/sam;
avg = mean(signal);
stddev = std(signal);
% Remove dc offset
signal = signal - avg;
frameTime = 0.020;
frameSize = frameTime * sam;
newSignal = [];
engSignal = sum(signal.^2)/(len/frameSize);

for f = 1:frameSize:len-frameSize
    frame = signal(f:f+frameSize-1);
    engFrame = (sum(frame.^2));
    stdFrame = std(frame);
    if (stdFrame > stddev/9)
        newSignal = [newSignal frame];
    end
end

y = filter([-0.95 1],[1],newSignal(:));

ny = length(y);
t = 1/sam:1/sam:ny/sam;

% samples
%frameSize = 200;
frameTime = 0.020;
frameSize = frameTime * sam;
% for use with indexes
fs = frameSize-1;
% Stores the lpc coefficients
lpck = [];
% for the new signal
newsound = [];
lpcoeff = [];
R = [];
overLap = frameSize*0.375;
% loops through the signal at the "frameSize"
for f = 1:frameSize-overLap:ny-fs+1;
         frame = y(f:f+fs);
      % length of the frame
      n = length(frame);
      % Hamming window
      h = hamming(frameSize);
      frame = frame .* h;
      % uncomment to use feature extraction method
      % MFCC
      % pc = mfcc(frame,P);
      % LPCC
      pc = lpcc(frame,P)
      % LPC
      lpck = [lpck pc];
      % pc = real(lpc(frame,P));
      %lpck = [lpck pc(2:P+1)'];
end

c = kmgla(lpck',k);
data1 = lpck';
codeBook = [codeBook c;];
switch(fil)
    case(1)
        dat1 = data1;
    case(2)
        dat2 = data1;
    case(3)
        dat3 = data1;
    case(4)
        dat4 = data1;
    case(5)
        dat5 = data1;
    case(6)
        dat6 = data1;
    case(7)
        dat7 = data1;
    case(8)
        dat8 = data1;
    case(9)
        dat9 = data1;
    case(10)
        dat10 = data1;
    case(11)
        dat11 = data1;
    case(12)
        dat12 = data1;
    otherwise
        gh = 1
end
end
```

C.3 The increaseSpeakers.m MATLAB Function

This test the effect of increasing speakers on the NTIMIT corpus

```
% Tests the NTIMIT Corpus increasing the speaker from 1 through to 40
% David Watts, 2006
codeBook = [];
dataD = [];

addpath('C:\dev\project\timitmfiles');
addpath('C:\dev\project\new');

% Test NTIMIT Files
speechFiles1 = {'FALK0','SA1','SA2','SI456','SI658','SI1086','SX6','SX96','SX186
    ','SX276','SX366'};
speechFiles2 = {'FCKE0','SA1','SA2','SI481','SI1111','SI1741','SX31','SX121','
    SX211','SX301','SX391'};
speechFiles3 = {'FCMG0','SA1','SA2','SI1142','SI1242','SI1872','SX72','SX162','
    SX252','SX342','SX432'};
speechFiles4 = {'FDFB0','SA1','SA2','SI1318','SI1948','SI2010','SX58','SX148','
    SX238','SX328','SX418'};
speechFiles5 = {'FDJH0','SA1','SA2','SI935','SI1565','SI2195','SX35','SX125','
    SX215','SX305','SX395'};
speechFiles6 = {'MADC0','SA1','SA2','SI737','SI1367','SI1997','SX17','SX107','
    SX197','SX287','SX377'};
speechFiles7 = {'MAKB0','SA1','SA2','SI1016','SI1646','SI2276','SX26','SX116','
    SX206','SX296','SX386'};
speechFiles8 = {'MAKR0','SA1','SA2','SI722','SI1352','SI1982','SX92','SX182','
    SX272','SX272','SX452'};
speechFiles9 = {'MAPV0','SA1','SA2','SI663','SI1293','SI1923','SX33','SX123','
    SX213','SX303','SX393'};
speechFiles10 = {'MBEF0','SA1','SA2','SI651','SI1281','SI1911','SX21','SX111','
    SX201','SX291','SX381'};
speechFiles11 = {'MCAL0','SA1','SA2','SI508','SI1138','SI1768','SX58','SX148','
    SX238','SX328','SX418'};
speechFiles12 = {'MCDC0','SA1','SA2','SI662','SI1292','SI1922','SX32','SX122','
    SX212','SX302','SX392'};
speechFiles13 = {'FGCS0','SA1','SA2','SI856','SI1486','SI2116','SX46','SX136','
    SX226','SX316','SX406'};
speechFiles14 = {'FEME0','SA1','SA2','SI875','SI1505','SI2135','SX65','SX155','
    SX245','SX335','SX425'};
speechFiles15 = {'FGRW0','SA1','SA2','SI1152','SI1782','SI1990','SX72','SX162','
    SX252','SX342','SX432'};
speechFiles16 = {'FJLG0','SA1','SA2','SI1506','SI1889','SI2306','SX89','SX179','
    SX269','SX359','SX449'};
speechFiles17 = {'FJLR0','SA1','SA2','SI601','SI1231','SI1861','SX61','SX151','
    SX241','SX331','SX421'};
speechFiles18 = {'FLAC0','SA1','SA2','SI901','SI1339','SI2161','SX91','SX181','
    SX271','SX361','SX451'};
speechFiles19 = {'FLJD0','SA1','SA2','SI886','SI1516','SI2146','SX76','SX166','
    SX256','SX346','SX436'};
speechFiles20 = {'FLTM0','SA1','SA2','SI1070','SI1700','SI2330','SX80','SX170','
    SX260','SX350','SX440'};
speechFiles21 = {'MCDD0','SA1','SA2','SI883','SI1513','SI2143','SX73','SX163','
    SX253','SX343','SX433'};
speechFiles22 = {'MCEF0','SA1','SA2','SI842','SI1135','SI1765','SX55','SX145','
    SX235','SX325','SX415'};
speechFiles23 = {'MDBB1','SA1','SA2','SI1006','SI1636','SI2056','SX16','SX106','
    SX196','SX286','SX376'};
speechFiles24 = {'MDDC0','SA1','SA2','SI789','SI1419','SI2049','SX69','SX159','
    SX249','SX339','SX429'};
speechFiles25 = {'MDEF0','SA1','SA2','SI1123','SI1563','SI2193','SX33','SX123','
    SX213','SX303','SX393'};
speechFiles26 = {'MDHS0','SA1','SA2','SI900','SI1530','SI2160','SX90','SX180','
    SX270','SX360','SX450'};
speechFiles27 = {'MDJM0','SA1','SA2','SI825','SI1455','SI2085','SX15','SX105','
    SX195','SX285','SX375'};
speechFiles28 = {'MDLC0','SA1','SA2','SI765','SI1395','SI2025','SX45','SX135','
    SX225','SX315','SX405'};
speechFiles29 = {'MDLH0','SA1','SA2','SI574','SI700','SI1960','SX70','SX160','
    SX250','SX340','SX430'};
```

```
speechFiles30 = {'MDNS0','SA1','SA2','SI873','SI1011','SI2271','SX21','SX111','
    SX201','SX291','SX381'};
speechFiles31 = {'MDSS1','SA1','SA2','SI697','SI1327','SI1713','SX67','SX157','
    SX247','SX337','SX427'};
speechFiles32 = {'MDTB0','SA1','SA2','SI570','SI1200','SI1830','SX120','SX210','
    SX300','SX321','SX390'};
speechFiles33 = {'MDWM0','SA1','SA2','SI916','SI1546','SI2176','SX16','SX106','
    SX286','SX376','SX433'};
speechFiles34 = {'MFMC0','SA1','SA2','SI502','SI1132','SI1762','SX52','SX142','
    SX232','SX322','SX412'};
speechFiles35 = {'MGAF0','SA1','SA2','SI652','SI1282','SI1912','SX22','SX112','
    SX202','SX292','SX382'};
speechFiles36 = {'MHJB0','SA1','SA2','SI1017','SI1647','SI2277','SX27','SX117','
    SX207','SX297','SX387'};
speechFiles37 = {'MHMR0','SA1','SA2','SI489','SI1119','SI1692','SX39','SX129','
    SX219','SX309','SX399'};
speechFiles38 = {'MILB0','SA1','SA2','SI807','SI903','SI2163','SX3','SX93','
    SX183','SX273','SX363'};
speechFiles39 = {'MJDA0','SA1','SA2','SI1031','SI1661','SI2291','SX41','SX131','
    SX221','SX311','SX401'};
speechFiles40 = {'MJJB0','SA1','SA2','SI1139','SI1277','SI1769','SX59','SX149','
    SX239','SX329','SX419'};


for incr = 1:2
for fil = 1:40
switch(fil)
    case(1)
        speechFiles = speechFiles1;
    case(2)
        speechFiles = speechFiles2;
    case(3)
        speechFiles = speechFiles3;
    case(4)
        speechFiles = speechFiles4;
    case(5)
        speechFiles = speechFiles5;
    case(6)
        speechFiles = speechFiles6;
    case(7)
        speechFiles = speechFiles7;
    case(8)
        speechFiles = speechFiles8;
    case(9)
        speechFiles = speechFiles9;
    case(10)
        speechFiles = speechFiles10;
    case(11)
        speechFiles = speechFiles11;
    case(12)
        speechFiles = speechFiles12;
    case(13)
        speechFiles = speechFiles13;
    case(14)
        speechFiles = speechFiles14;
    case(15)
        speechFiles = speechFiles15;
    case(16)
        speechFiles = speechFiles16;
    case(17)
        speechFiles = speechFiles17;
    case(18)
        speechFiles = speechFiles18;
    case(19)
        speechFiles = speechFiles19;
    case(20)
        speechFiles = speechFiles20;
    case(21)
        speechFiles = speechFiles21;
    case(22)
        speechFiles = speechFiles22;
    case(23)
        speechFiles = speechFiles23;
    case(24)
        speechFiles = speechFiles24;
    case(25)
        speechFiles = speechFiles25;
```

```matlab
    case(26)
        speechFiles = speechFiles26;
    case(27)
        speechFiles = speechFiles27;
    case(28)
        speechFiles = speechFiles28;
    case(29)
        speechFiles = speechFiles29;
    case(30)
        speechFiles = speechFiles30;
    case(31)
        speechFiles = speechFiles31;
    case(32)
        speechFiles = speechFiles32;
    case(33)
        speechFiles = speechFiles33;
    case(34)
        speechFiles = speechFiles34;
    case(35)
        speechFiles = speechFiles35;
    case(36)
        speechFiles = speechFiles36;
    case(37)
        speechFiles = speechFiles37;
    case(38)
        speechFiles = speechFiles38;
    case(39)
        speechFiles = speechFiles39;
    case(40)
        speechFiles = speechFiles40;
    otherwise
        -1
end
y = [];
totLen = 1;
for files = 2:11

    [insY, BufLen, sam] = nistread(char(strcat('testFiles\',speechFiles(1),'\',
        speechFiles(files),'.WAV')));
    if ((files == 2 | files == 3 ) )
        if(incr == 2)
            y = [y (insY(10:length(insY)))'];
        end
    else
        if (incr == 1)
            y = [y (insY(10:length(insY)))'];
        end
    end
end

k = 128;

ny = length(y);
t = 1/sam:1/sam:ny/sam;

% Pre-Processing
signal = y;
len = length(signal);
t = 1/sam:1/sam:len/sam;
avg = mean(signal);
stddev = std(signal);
% Remove dc offset
signal = signal - avg;

frameTime = 0.020;
frameSize = frameTime * sam;
newSignal = [];
engSignal = sum(signal.^2)/(len/frameSize);

for f = 1:frameSize:len-frameSize
    frame = signal(f:f+frameSize-1);
    engFrame = (sum(frame.^2));
    stdFrame = std(frame);
    % Removal of Silence Component
    if (stdFrame > stddev/9)
        newSignal = [newSignal frame];
    end
end
y = filter([-0.95 1],[1],newSignal(:));
```

```

ny = length(y);
t = 1/sam:1/sam:ny/sam;

frameTime = 0.020;
frameSize = frameTime * sam;
% for use with indexes
fs = frameSize-1;
% Stores the lpc coefficients
lpck = [];
% for the new signal
newsound = [];
lpcoeff = [];
R = [];
overLap = frameSize*0.020;
% loops through the signal at the "frameSize"
for f = 1:frameSize-overLap:ny-fs+1;
        frame = y(f:f+fs);
        % length of the frame
        n = length(frame);
        % Hamming window
        h = hamming(frameSize);
        frame = frame .* h;
        % uncomment to use feature extraction method
        % MFCC
        % pc = mfcc(frame,P);
        % LPCC
        pc = lpcc(frame,P)
        % LPC
        lpck = [lpck pc];
        % pc = real(lpc(frame,P));
        %lpck = [lpck pc(2:P+1)'];
end
if (incr == 1)
     [idx c] = kmeans(lpck',k);
     codeBook = [codeBook c;];
end
data1 = lpck';

switch(fil)
     case(1)
          dat1 = data1;
     case(2)
          dat2 = data1;
     case(3)
          dat3 = data1;
     case(4)
          dat4 = data1;
     case(5)
          dat5 = data1;
     case(6)
          dat6 = data1;
     case(7)
          dat7 = data1;
     case(8)
          dat8 = data1;
     case(9)
          dat9 = data1;
     case(10)
          dat10 = data1;
     case(11)
          dat11 = data1;
     case(12)
          dat12 = data1;
     case(13)
           dat13 = data1;
     case(14)
           dat14 = data1;
     case(15)
           dat15 = data1;
     case(16)
           dat16 = data1;
     case(17)
           dat17 = data1;
     case(18)
           dat18 = data1;
     case(19)
           dat19 = data1;
```

```
    case(20)
         dat20 = data1;
    case(21)
         dat21 = data1;
    case(22)
         dat22 = data1;
    case(23)
         dat23 = data1;
    case(24)
         dat24 = data1;
    case(25)
         dat25 = data1;
    case(26)
         dat26 = data1;
    case(27)
         dat27 = data1;
    case(28)
         dat28 = data1;
    case(29)
         dat29 = data1;;
    case(30)
         dat30 = data1;
    case(31)
        dat31 = data1;
    case(32)
        dat32 = data1;
    case(33)
        dat33 = data1;
    case(34)
        dat34 = data1;
    case(35)
        dat35 = data1;
    case(36)
        dat36 = data1;
    case(37)
        dat37 = data1;
    case(38)
        dat38 = data1;
    case(39)
        dat39 = data1;
    case(40)
        dat40 = data1;;
    otherwise
         -1
end
end
end

% for testing by increasing the number of speakers
for speakers = 2:40
    res =[];
    speakers
    for spk = 1 : speakers
    switch(spk)
    case(1)
        res = [res dist(dat1,codeBook(:,1:speakers*P),P) ];
    case(2)
        res = [res dist(dat2,codeBook(:,1:speakers*P),P) ];
    case(3)
        res = [res dist(dat3,codeBook(:,1:speakers*P),P) ];
    case(4)
         res = [res dist(dat4,codeBook(:,1:speakers*P),P) ];
    case(5)
       res = [res dist(dat5,codeBook(:,1:speakers*P),P) ];
    case(6)
         res = [res dist(dat6,codeBook(:,1:speakers*P),P) ];
    case(7)
        res = [res dist(dat7,codeBook(:,1:speakers*P),P) ];
    case(8)
        res = [res dist(dat8,codeBook(:,1:speakers*P),P) ];
    case(9)
        res = [res dist(dat9,codeBook(:,1:speakers*P),P) ];
    case(10)
       res = [res dist(dat10,codeBook(:,1:speakers*P),P) ];
    case(11)
       res = [res dist(dat11,codeBook(:,1:speakers*P),P) ];
    case(12)
```

```
            res = [res dist(dat12,codeBook(:,1:speakers*P),P) ];
        case(13)
            res = [res dist(dat13,codeBook(:,1:speakers*P),P) ];
        case(14)
             res = [res dist(dat14,codeBook(:,1:speakers*P),P) ];
        case(15)
          res = [res dist(dat15,codeBook(:,1:speakers*P),P) ];
        case(16)
            res = [res dist(dat16,codeBook(:,1:speakers*P),P) ];
        case(17)
            res = [res dist(dat17,codeBook(:,1:speakers*P),P) ];
        case(18)
           res = [res dist(dat18,codeBook(:,1:speakers*P),P) ];
        case(19)
            res = [res dist(dat19,codeBook(:,1:speakers*P),P) ];
        case(20)
             res = [res dist(dat20,codeBook(:,1:speakers*P),P) ];
        case(21)
           res = [res dist(dat21,codeBook(:,1:speakers*P),P) ];
        case(22)
            res = [res dist(dat22,codeBook(:,1:speakers*P),P) ];
        case(23)
             res = [res dist(dat23,codeBook(:,1:speakers*P),P) ];
        case(24)
             res = [res dist(dat24,codeBook(:,1:speakers*P),P) ];
        case(25)
              res = [res dist(dat25,codeBook(:,1:speakers*P),P) ];
        case(26)
             res = [res dist(dat26,codeBook(:,1:speakers*P),P) ];
        case(27)
          res = [res dist(dat27,codeBook(:,1:speakers*P),P) ];
        case(28)
              res = [res dist(dat28,codeBook(:,1:speakers*P),P) ];
        case(29)
             res = [res dist(dat29,codeBook(:,1:speakers*P),P) ];
        case(30)
            res = [res dist(dat30,codeBook(:,1:speakers*P),P) ];
        case(31)
             res = [res dist(dat31,codeBook(:,1:speakers*P),P) ];
        case(32)
           res = [res dist(dat32,codeBook(:,1:speakers*P),P) ];
        case(33)
             res = [res dist(dat33,codeBook(:,1:speakers*P),P) ];
        case(34)
             res = [res dist(dat34,codeBook(:,1:speakers*P),P) ];
        case(35)
            res = [res dist(dat35,codeBook(:,1:speakers*P),P) ];
        case(36)
         res = [res dist(dat36,codeBook(:,1:speakers*P),P) ];
        case(37)
            res = [res dist(dat37,codeBook(:,1:speakers*P),P) ];
        case(38)
             res = [res dist(dat38,codeBook(:,1:speakers*P),P) ];
        case(39)
             res = [res dist(dat39,codeBook(:,1:speakers*P),P) ];
        case(40)
            res = [res dist(dat40,codeBook(:,1:speakers*P),P) ];
        otherwise
            -1
        end
end
end
```

C.4 The kmgla.m MATLAB Function

Performs the VQ

```
% Implementation of GLA
% Inputs: feature vectors and size of required codebook
% Output: Codebook
% David Watts

function[codeBook] = kmgla(feat,k)
% size of the features
[rfeat cfeat] = size(feat);
% random vectors for initial code vector
rdn = ceil(rand(k,1)*rfeat);
initialCodebook = feat(rdn,:);
% for check of while loop
oldCodebook = ones(k,cfeat);
distances = [];
allDist = [];
it = 0;
while( (sum((sum((oldCodebook-initialCodebook).^2)))) > 0.01 )
        % calculates the distances for each feature vector and associates the
            features
        % to the centroids
         for (i = 1:k)
             for (j = 1:rfeat)
                distances = [distances sqrt(sum((initialCodebook(i,:) - feat(j
                    ,:)).^2))];
            end
            allDist = [allDist; distances];
            distances = [];
         end
         % new codebook is update
         [val vector] = min(allDist) ;
         allDist = [];
         oldCodebook = initialCodebook;
         % calculates the new distortion measure
         for (i = 1:k)
             if sum(find(vector == i)) > 1
                 initialCodebook(i,:) = mean(feat(find(vector == i),:));
             end
         end
end
codeBook = initialCodebook;
```

C.5 The mfcc.m MATLAB Function

```
% calculates the lpcc
% David Watts, 2006

function [d] = mfcc(frame, features);

fier = fft(frame);
fourier = log10(real(fier(1:length(frame/2))));
% warps the frequency using mel-scale
mel = 2595*log10(fourier/700);
% working mel
allCoef = real(ifft(mel));
% removes the DC component
d = allCoef(2:features+1);
```

C.6 The lpcc.m MATLAB Function

```
% calculates the lpcc
% David Watts, 2006

function [d] = lpcc(frame, features);

% calculates lpc and reconstructs the signal
e = zeros(length(frame),1);
e(1) = 0.01;
pc = real(lpc(frame,features));
yr = filter(1,pc,e);
fier = fft(yr);
fourier = log10(real(fier(1:length(frame/2))));
% warps the frequency using mel-scale
mel = 2595*log10(fourier/700);
% working mel
allCoef = real(ifft(mel));
% removes the DC component
d = allCoef(2:features+1);
```

C.7 The eucDist.m MATLAB Function

Calculates the Euclidean distance

```
% Calculats the Ecludian distance Measure for input features and a codebook
% David Watts
% Inputs, featurs = lcpk, and codebook = clust

function [d] = eucDist(lpck, clust);

% used for normalization
lpckMean = mean(lpck);
lpckStd = std(lpck);
[k1 dontcar] = size(clust);
% length of features
s = length(lpck);
% Noramlizes input featurs
z1 = [];
for k = 1:s
    z1 = [z1; (lpck(k,:)-lpckMean)./lpckStd];
end
% Normalizes codebook
clustMean = mean(clust);
clustStd = std(clust);
z2 = [];
for m = 1:k1
    z2 = [z2; (clust(m,:)-clustMean)./clustStd];
end
% sets new values
lpck = z1;
clust = z2;
distAll = [];
totalDist = 0;
for j = 1:s
    for i = 1:k1
        % Euculidean Distance
        dist = sqrt(sum((lpck(j,:) - clust(i,:)).^2));
        %dist = (sum((lpck(j,:) - clust(i,:)).^2));
        %dist = sum(abs(lpck(j,:) - clust(i,:)));
        distAll = [distAll dist];
    end
    [val pos] = min(distAll);
    totalDist = totalDist + val;
    distAll = [];
end
% returns total distortion
d = totalDist/s   ;
```

C.8 The mfcc.m MATLAB Function

```
% calculates the lpcc
% David Watts, 2006

function [d] = mfcc(frame, features);

fier = fft(frame);
fourier = log10(real(fier(1:length(frame/2))));
% warps the frequency using mel-scale
mel = 2595*log10(fourier/700);
% working mel
allCoef = real(ifft(mel));
% removes the DC component
d = allCoef(2:features+1);
```

C.9 The dist.m MATLAB Function

Finds the distance between features and codebooks

```
% Calculates the dist between all codebooks and the features
% returns the distortion between codebooks and input features
% Inputs feature vectors, codebooks, order of coefficients
% David Watts, 2006

function [d] = dist(dat, cdbk, P);

[r c] = size(cdbk);
siz = c/P;
ee = [];
for i=1:P:(P*siz)-1
    % calculates the euclidean distance
    ee = [ee eucDist(dat,cdbk(:,i:i+P-1))];
end
[xx yy] = min(ee);
d = yy;
```