Speaker Verification in JAVA

A thesis submitted in partial fulfillment of
the requirements for the degree of

Master of Computer and Information Engineering

School of Microelectronic Engineering
Griffith University

By
Magnus Nilsson
October 2001

Statement of Originality

This work has not been previously submitted for a degree or diploma in any university. To the best of my knowledge, the thesis contains no material previously published or written by any other person except where a reference is made.

Magnus Nilsson

Abstract

The Signal Processing Laboratory at Griffith University conducts research in the digital signal processing area, specialized on image and speaker verification, under the supervision of Professor Kuldip K. Paliwal.

In this report the design and implementation of a speaker verification system in Java is described, and the use of Java as a research language, due to its platform independency. The system is using Vector Quantization, VQ, and Mel Frequency Cepstral Coefficients, MFCC's, since the concept of using VQ and MFCC's is a popular technique with high accuracy.

Simulation in Matlab and implementation in Java 2 version 1.3 show a total error rate of four percent.

You are welcome to visit the Signal Processing Laboratory on the web: http://spl.me.gu.edu.au

Preface

This thesis is a part of my education towards a Master degree in Computer and Information Engineering at Griffith University, Brisbane, Australia. Project 4, MEE8099.

The work is conducted at the Signal Processing Laboratory, Griffith University, Brisbane, Australia.

I would like to thank the following people who has been of great help to me during my work.

My supervisor Professor Kuldip K. Paliwal, Griffith University.

My friends, the research staff at the Signal Processing Laboratory, Griffith University, Conrad Sanderson, Mohammad Dehghani, Brett Wildermoth, Stephen So, Nanda Koestoer, Xuechuan Wang and Ganesh Nanik.

I would also like to thank my family back home in Sweden and my friends here in Brisbane, without whom this degree never would have been possible, Anne Ragnefors, Anna-Stina Nilsson, Ove Nilsson, Erik Nilsson, Maj-Britt Nilsson, Mattias Berglund, Scott Jones, Mikael Bergman, Joakim Blomkvist, David Thunmarker, Krister Nyman, Mattias Almlund, Bryan Wythe, Tarik Hammadou, Alok Sharma, Satwant Sandhu, and the rest of the people I have met here which have made my time in Brisbane enjoyable.

Thanks all!

Table of Contents

1.0 Introduction

1.1 Background

It would be interesting to develop a Speaker Verification system/software in JAVA, since the JAVA language is said to be platform independent and would be interesting as a research language.

1.2 Task

To study, implement and evaluate a VQ (Vector Quantization) Speaker Verification system in JAVA, using MFCC's (Mel Frequency Cepstral Coefficients).

1.3 Technical Function

A graphical software implementation which shall record speech from a person throw a microphone, verify the person as true speaker or false speaker.

2.0 JAVA

2.1 History of JAVA

The history of Java design began in 1991 at Sun Microsystems by James Gosling & Co. The idea behind the Java platform was to develop a programming language that address the problem of building software for network consumer devices. To be able to meet all these requirements, the compiled code had to survive transport across networks, operate on any client, assure the client that it was safe to run, and possess the capability to work on a wide range of platforms and CPU's [1]. These requirements turned out to form the Java programming language as a general-purpose object-oriented concurrent language. Its syntax is similar to C and C++, but it leaves out many of the features that make C and C++ complex, confusing, and unsafe.

The first name for Java was Oak, but this name was already taken by another language, so the creators decided upon the name Java. Their first product was a remote control that did not get much support from the customers.

Everything changed in the early 1990's when the Internet quickly gained popularity. The first browser for surfing the net was called Gopher and since it allowed only text based pages to be displayed, people found it pretty boring. After a while, the web browser Mosaic appeared, (later became Netscape), allowing graphical images to be displayed on web pages and, as a result, more people became interested in the net surfing and web developers started to build pages with graphics contents. Internet enthusiasts quickly understood that the content supported by the web's HTML document format was too limited. HTML extensions, such as forms, only highlighted those limitations, while making it clear that no browser could include all the features users wanted, something that extended the browser was the answer.

The Java development team saw the Internet as a very interesting and promising way of implementing the language and making web pages look and function better. They also realized that it is interesting to build a new browser that would allow the Java enabled web pages to be displayed effectively. The first of the web browsers that made it possible to embed program inside a HTML page, was Sun's HotJava browser, which highlighted the interesting properties of the Java programming language and platform portability. The Java programs are transparently downloaded into the browser together with the HTML pages in which they appear. Be course of the safety of Java, before being accepted by the browser, the programs are carefully checked to make sure they are safe. Due to Java’s portability, the

programs behave the same way regardless of where they come from or what kind of machine they are being loaded into and run on.

The first big step for Java came on May 23 1995, when Java Technology was for the first time shown to the world. However, one of the biggest step forward for Java, came when Netscape also decided to enable its browser to display web pages that contained Java. This was allowed in Netscape version 2 for the first time, and also Microsoft saw the benefits in using Java and enabled its browser to display Java applets as well. The development of Java has being continued with growing speed, and the JavaTM 2 platform is now available for the public (http://www.java.sun.com). The Java developers say that the Java 2 platform is the definitive environment for projects intending to build and install web-centric software applications that will be capable to run on different computers, servers and other computing devices [1].

All web browser incorporating the Java or Java 2 platform is no longer limited to a fixed set of capabilities. Internet surfers can now visit web pages that holds a dynamic content, and can be assured that their machines cannot be damaged by that content. Programmers can write a program once, and it will run on any machine supplying a Java or Java 2 runtime environment.

2.2 Virtual Machine

The Java virtual machine is the cornerstone of Java. This component is technology responsible for Java's hardware- and operating system- independence, the small size of its compiled code, and its ability to protect users from malicious programs [1]. The Java virtual machine can be defined as an abstract computing machine, and when being compared with real computing machines, it has a similar instruction set and manipulates various memory areas at run time. Java is not the first programming language using a virtual machine. Possibly the best-known virtual machine before Java was the P-Code machine of UCSD Pascal.

The first Java virtual machine prototype implementation, done at Sun Microsystems, emulated the Java virtual machine instruction set in software hosted by a handheld device that resembled a contemporary Personal Digital Assistant (PDA). Sun's current Java virtual machine implementations (version 1.3), emulate the Java virtual machine on Win32, Linux RedHat and Unix Solaris hosts in much more sophisticated ways. However, the Java virtual machine does not assume any particular implementation technology, host hardware, or host operating system. Due to this fact, it can just as well be implemented by compiling its instruction set to that of a silicon CPU. It may also be implemented in microcode or directly in silicon.

2.3 Applet and Application

There are two types of Java programs, the stand-alone program, also known as a Java application, or the web browser based, that runs within a Java-compatible browser, the Java applet.

A Java applets have access to the libraries of the Java Application Programming Interface (API) that is supported by all Java capable browsers, such as Microsoft Internet Explorer and Netscape Communicator. The API currently includes libraries for applets, input and output, language, network access, GUI and general utilities. An applet is defined by class files that can be downloaded from one or more other computers on the internet. Security is a critical issue that has dominated the design, and as mentioned earlier, an applet is always subject to a verification process to check for language compliance. After the security check the memory layout is determined and execution begins. Applets can be written with with a basic function that is customized by parameters that are specified in the web page. The Java language allows dynamic linking to software libraries written in other languages. However, for security reasons this feature is usually disabled for software that is downloaded from the net.

When dealing with a Java stand-alone application, one has to have the Java virtual machine, i.e. the Java runtime environment installed. This software pack is free of charge and can be downloaded from the SUN web page, or will be distributed together with the application. A Java stand-alone application can be compared with the well known executable files one get when compiling a C or C++ code. An application does not have the same security constrains as an applet, but there is still a certain amount of problems to access the actual hardware of the host computer, i.e parallel/serial ports and soundcard.

2.4 Platform Portability

In the research world most computational research is done in C or C++, and on a wide variety of workstations and personal computers, using different operating systems. When publishing a paper, there would be interesting to be able to distribute the research publication together with an executable file that displays the results. It might appear that any author could easily make the C/C++ programs and data available for others to reproduce the results of the publication, but in fact, moving C/C++ programs from one compiler to another or from one computer to another is very difficult [2]. For starters, there is no standard definition of what an integer or floating point is. The definition of data types is implementation dependent for example. As a result, without significant modifications, a program may not execute correctly, or maybe not at all when moved from one machine to another.

The burden of portability has always fallen on the developer of the code when dealing with C/C++ or any other platform dependent language. When dealing with platform dependent languages the provider can choose to address this problem by using only those parts of the C/C++ language that are “highly portable” and thus offer the best chance of executing correctly on a variety of platforms, computers and compilers. This is not a simple task. Another approach is to maintain different versions of the program for different platforms and computers. A provider can use either approach or some combination. Both approaches are difficult and, this task is never completed, to the fact that changes are made in the language and different kind of platforms, compilers and computers are introduced, This means that the careful provider has to constantly monitor the developments on the computers that he has chosen to support.

One way of dealing with this fact is to distribute the code for the program. Even this is not bullet proof. If the user does not port the code correctly, the user may claim that the results reported in the publication can not be reproduced, meaning that there could be a question of validity of the original results. Other users may, frustrated by the difficulty of porting the code, fully discard the results.

All is said to be addressed in the Java language, and the big idea behind Java is that the responsibility for porting computer programs has been removed from the hands of the providers and users and made it the responsibility of the network, that means the Java virtual machine. Java programs are compiled and executed on a specific computer, but the linking and loading of programs is a network capability and the execution produces standard results regardless of the computer and platform. By shifting the responsibility for porting programs from the provider and users, a major advance has been made.

One significant drawback of Java’s portability benefits is that it will have to pay in expense of performance characteristics. The characteristics can be summarized as:

> Any application built to run on a specific computer with specific hardware and software resources executes better than the same application written in Java that runs on all computers [2].

These facts are also to be taken into consideration and whether the tradeoffs are justified. It is possible to dynamically link an applet to C libraries. This, of course, destroys the portability of the program.

3.0 Speaker Verification and Vector Quantization

The following section will describe the consept of speaker verification and speaker recognition

3.1 Speaker Recognition and Speaker Verification

A speaker recognition system involves the process of automatically recognizing who is speaking on the basis of individual information included in the speakers speech waves. Speaker verification, on the other hand, is the process of accepting or rejecting the identity claim of a speaker. The difference of these concepts are described in figure 1 and figure 2 (derived from [3].

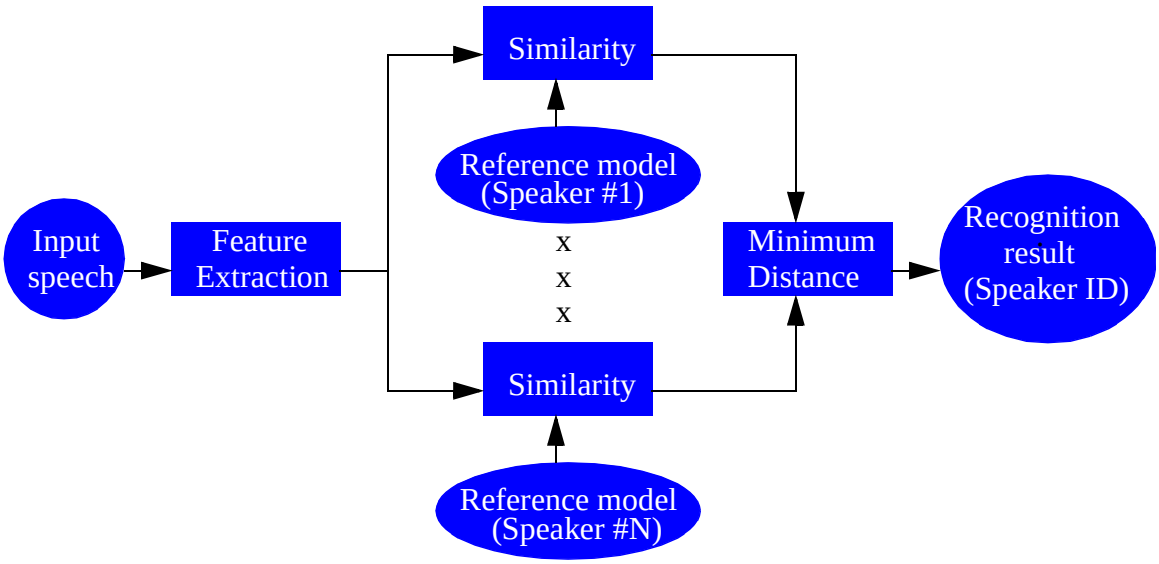

FIGURE 1. Speaker Recognition system

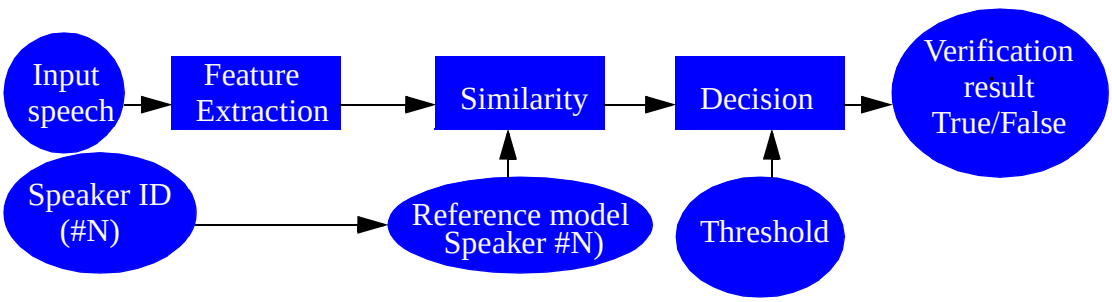

FIGURE 2. Speaker Verification system

The concept of recognizing a person through their voice can be useful as security control when accessing confidential information areas, access to remote computers, voice dialing, banking by telephone, telephone shopping, database access services, information services, voice mail or as PIN code for your ATM.

Speaker recognition and verification methods can be divided into text-independent and text-dependent methods. In a text-independent system, speaker models capture characteristics of a persons speech which show up irrespective of what one is saying. In a text-dependent system, on the other hand, the recognition of the speakers identity is based on his or her speaking one or more specific phrases, like passwords, card numbers, PIN codes, etc. All technologies of speaker recognition, identification and verification, text-independent and text-dependent, each has its own advantages and disadvantages and may require different treatments and techniques.

The goal of this project is to develop a speaker verification system in the platform independent software language Java, and since the system will be using Vector quantization and Mel frequency cepstral coefficients it is classified as a text-independent speaker verification system since its task is to verify the person who speaks regardless of what this person is saying.

All speaker recognition/verification system contains of two basic building blocks (see figure 1 and 2), feature extraction and feature matching.

The first part, the feature extraction is a process that will extract a small amount of speech data from the voice signal recorded. This data can later be used to represent each speaker. The second part, the feature matching involves the actual procedure to identify the unknown speaker by comparing extracted features from this persons voice input with the ones from a set of known speakers. All speaker verification systems also have to serve two distinguish phases. The first one is referred to as the training phase and the second as the verification or testing phase. For the training phase, the speaker has to provide samples of their speech so that the verification system can build a model for the speaker. For our verification system, a threshold is also computed from the training samples and cohort speakers. Later, during the verification phase, the input speech is matched with the earlier stored models and a decision calculation is made, deciding if the speaker is the claimed or not.

Speaker verification and speaker recognition are complex areas and still hot research subjects, with many different interesting ways of doing the feature extraction. An automatic speaker verification system works based on the premise that a persons speech exhibits characteristics that are unique to the speaker [4]. However, this task is a bit complicated due to variance of the speaker, such as peoples voice change with time, health conditions (i.e. the speaker has a cold), speaking rates, etc. and under which conditions the

speech is recorded. Examples of these are acoustical noise and variations in recording environments.

3.2 Speech Feature Extraction

This phase includes converting the speech waveform into a parametric representation with a considerably low information rate for further analysis and processing. This phase is often referred to as the signal processing front end.

The speech signal can be described as a slowly timed varying signal, or quasi-stationary. A sample of speech from the well known speech database TIMIT [5], in this case from a version of TIMIT with noise added and a sample rate of 8000 Hz, can be seen below.

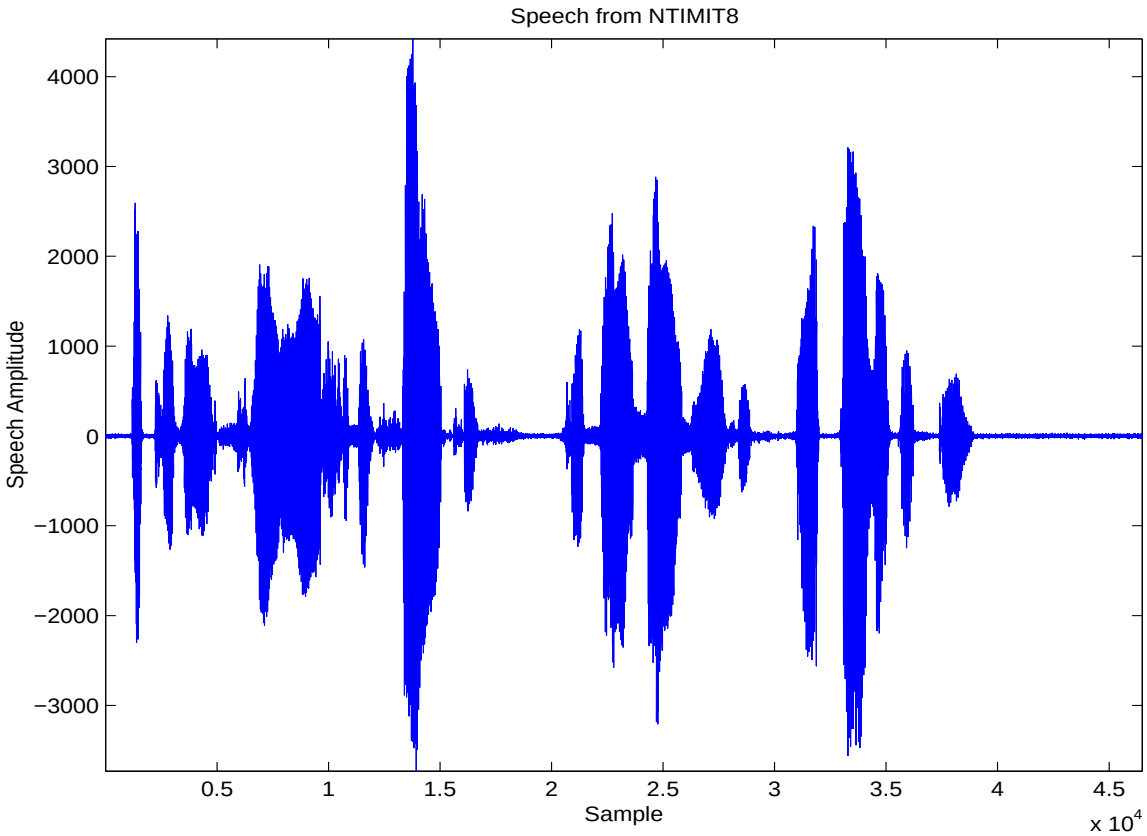

FIGURE 3. Speech data from TIMIT, Fs = 8000Hz, 16-bits, telephone noise added

When examined over a short period of time, somewhere between 20 and 30 msec, the characteristics are assumed to be fairly stationary.

3.3 Mel-frequency cepstral coefficients processor

There exists a wide range of possibilities for parameter extraction from a speech frames. As mentioned earlier, we will use one of the most popular methods, Mel-Frequency Cepstral Coefficients (MFCC). MFCC's are based on the known variation of the human ears critical bandwidths with frequency, filters spaced linearly at low frequencies and logarithmically at high frequencies have been used to capture the phonetically important characteristics of speech [6]. The characteristics is expressed on the mel-frequency scale, which is a linear frequency spacing below 1000 Hz and a logarithmic spacing above 1000 Hz. In addition, rather than the speech waveforms themselves, MFCC's are shown to be less susceptible to the above mentioned variation of the speakers voice and surrounding environment. The basic concept of a mel-frequency cepstral coefficient processor is described below.

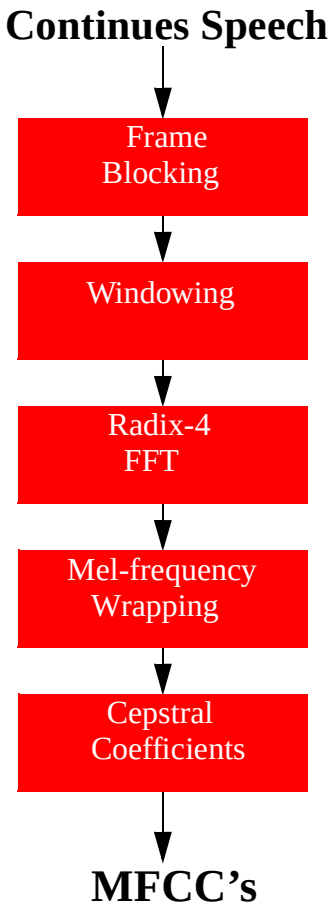

FIGURE 4. Block diagram of MFCC processor

3.3.1 Frame Blocking

The first step of the feature extraction is to frame the speech into frames of approximately 30 msec (30 msec at Fs = 8000Hz gives 240 samples).

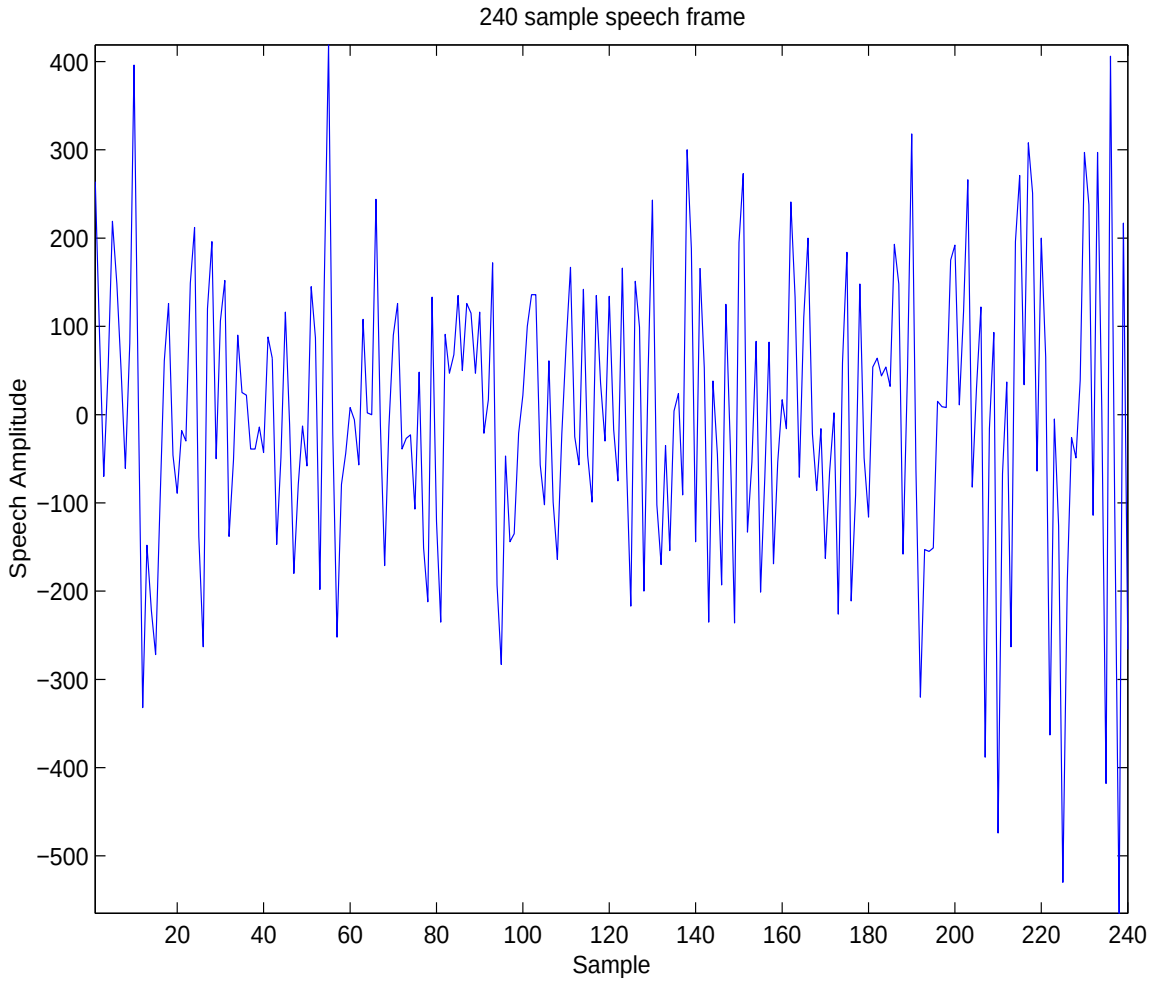

FIGURE 5. Speech frame with 240 samples i.e. 30 msec at Fs = 8000Hz

To be able to extract as much features as possible from a speech sample, the technique of overlapping frames is used [7]. The speech is blocked into frames of N samples (N = 240 in our case). With a overlapping of 50% one will get M number of frames out of a speech sample consisting of S samples:

$$NoOfFrames = \frac{2 \times SampleLength}{FrameLength} - 1$$

Number of frames from a speech sample with 50% overlapping (EQ 1)

3.3.2 Windowing

The next step in the processing is to window each individual frame so as to minimize the signal discontinuity at the beginning and end of each frame. The concept here is to minimize the spectral distortion by using the window to taper the signal to zero at the beginning and end of each frame. A typical window utilized for speaker verification is the Hamming window [8]. The equation for a Hamming window is as follows:

$$w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right) \qquad 0 \le n \le N-1$$

Hamming window equation (EQ 2)

This will be applied by:

$$y(n) = x(n) \times w(n) \qquad 0 \le n \le N-1$$

Frame smoothing or windowing (EQ 3)

3.3.3 Fast Fourier Transform (FFT)

The next step is to apply a Fourier Transform on the windowed speech frame. A Radix-4 Fast Fourier Transform is utilized, converting each frame from the time domain into the frequency domain. The FFT is a fast algorithm to implement the Discrete Fourier Transform (DFT). The Fourier Transform is defined as:

$$Y(\omega) = \int_{-\infty}^{\infty} y(t)e^{-j\omega t}$$

Definition of the Fourier Transform (EQ 4)

The Fast Fourier Transform and the Radix-4 algorithm is further described in [9]. To get a better display of the Fourier Transform, the process of zero padding is applied. It is important to note that zero padding does not provide any additional information about the spectrum Y(w) of the sequence {x(n)}[7].

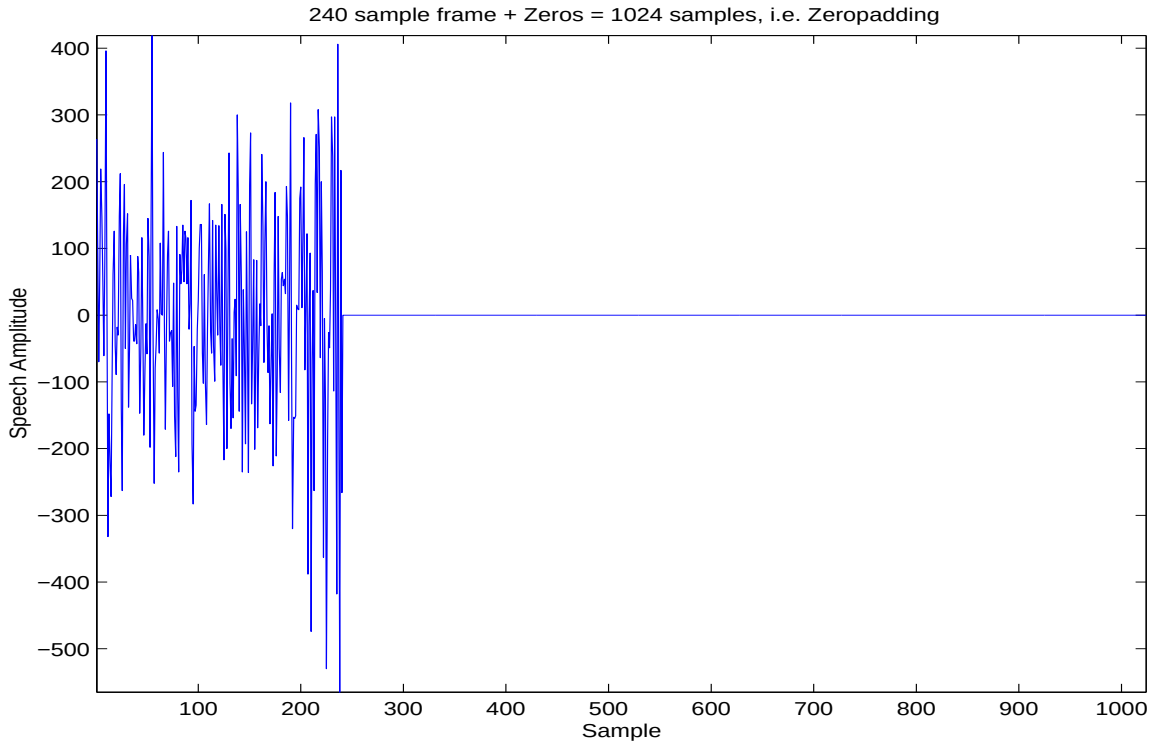

FIGURE 6. Zeropadding of speech frame

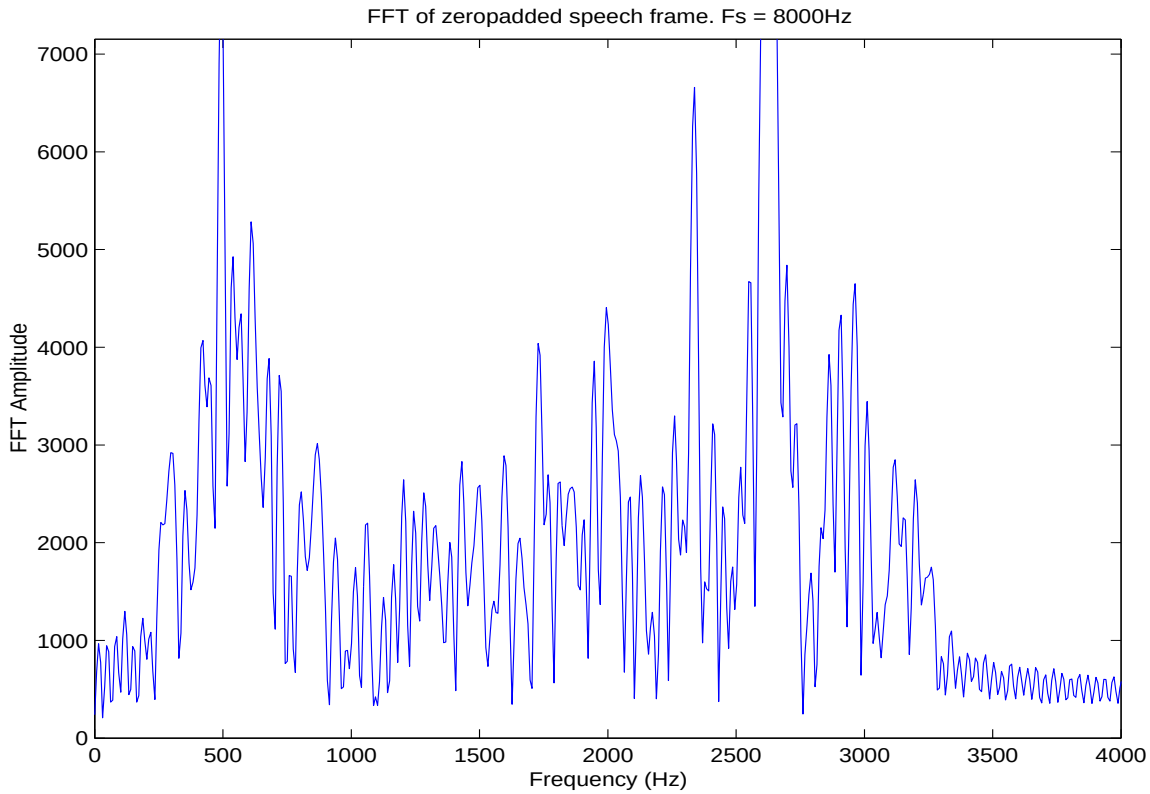

FIGURE 7. Radix-4 FFT of a zero padded speech frame

To be able to extract the power from the Fourier Transformed speech frame is calculated.

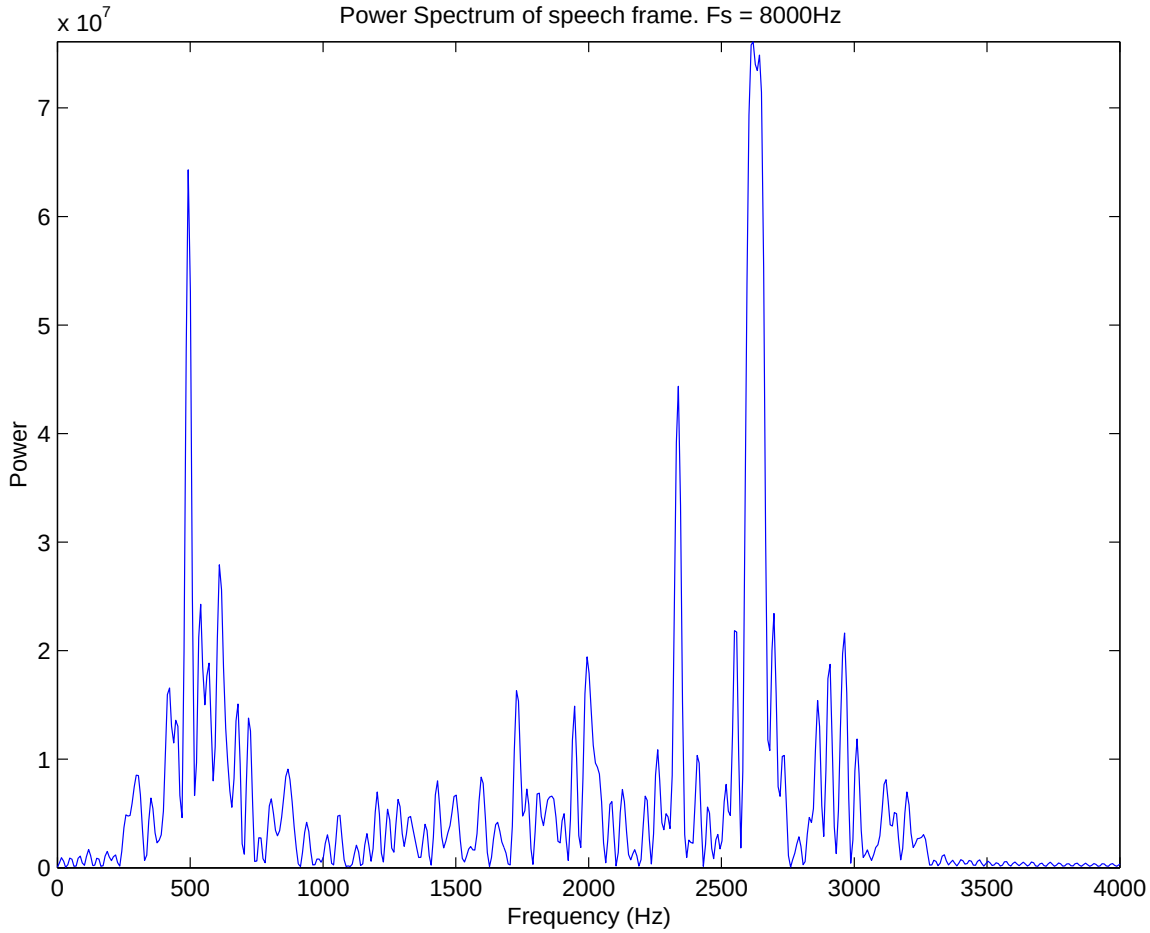

FIGURE 8. Power Spectrum/Periodogram

The result after this step is often referred to as Power Spectrum or Periodogram.

3.3.4 Mel-frequency Wrapping

As mentioned above, studies have been conducted that show that the human perception of the frequency contents of sounds for speech signals does not follow a linear scale. Thus for each tone with an actual frequency, f, measured in Hz, a subjective pitch is measured on a scale called the mel scale. The mel-frequency scale is a linear frequency spacing below 1000 Hz and a logarithmic spacing above 1000 Hz. As a reference point, the pitch of a 1 kHz tone, 40 dB above the perceptual hearing threshold, is defined as 1000 mels. Therefore we can use the following approximate formula to compute the mels for a given frequency f in Hz [10]:

$$mel(f) = 2595 \times \log_{10}\left(1 + \frac{f}{700}\right)$$

Melfrequency frequency calculation (EQ 5)

Our approach to simulate the ears way of extracting the power from the speech is to apply a filterbank to the Power Spectrum. This filterbank is uniformly spaced on the mel scale, has a triangular bandpass frequency response, and the spacing as well as the bandwidth is determined by a constant mel frequency interval [10]. The number of mel spectrum coefficients, K, is typically chosen as 20, but will vary a little depending on the sampling frequency. To be observed is that we are applying these filters in the frequency domain, therefore we simply multiply those triangle-shape windows in figure 9 on the Power Spectrum.

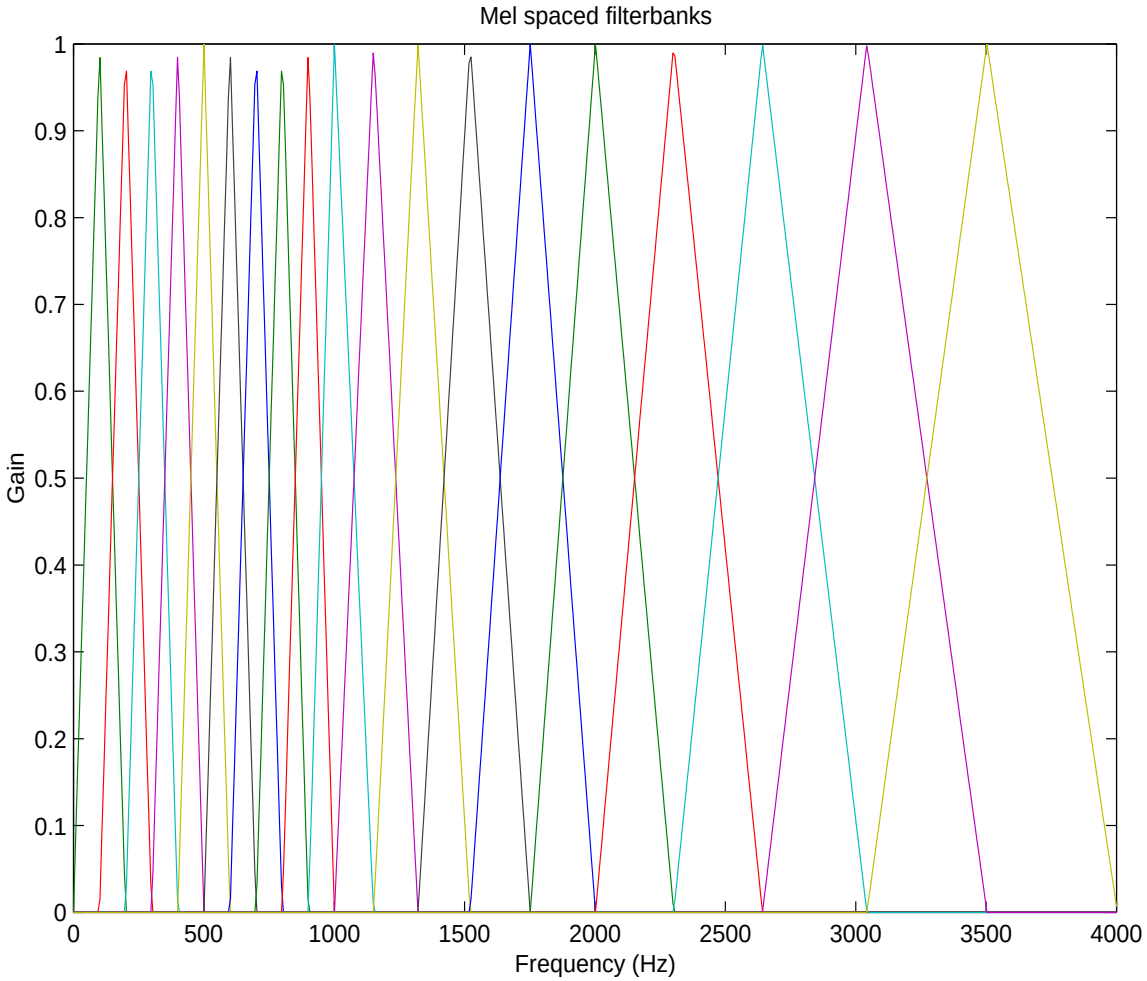

FIGURE 9. Mel frequency spaced filterbanks

3.3.5 Cepstral Coefficients

After this, the power of the triangular filter has to been summarized, and the log value of these sums are taken (this to compress the sum). The next step is to convert the log mel spectrum back to time. The result is called the mel frequency cepstral coefficients (MFCC). The cepstral representation of the speech spectrum provides a good representation of the local spectral properties of the signal for the given frame analysis [8]. Because the mel spectrum coefficients are real numbers, we can convert them to the time domain using the Discrete Cosine Transform (DCT):

$$mfcc(i) = \frac{1}{Nfilters} \sum_{l=1}^{Nfilters} mfb(l) \cos\left(i\left(l - \frac{1}{2}\right) \times \frac{\pi}{Nfilters}\right)$$
$$i = 1, \ldots, Nfilters$$

MFCC calculation (EQ 6)

We will discard the first component from the DCT calculation, since the first component only represent and reflects the average log energy of the speech frame[6].

3.3.6 Summary

By applying the above described procedure for each speech frame a set of mel-frequency cepstral coefficients are computed. This set of coefficients is called an acoustic vector. This data now needs to be “compressed”. By compressing the acoustic vector, we will optimize the verification process. This is described below.

3.4 Feature Matching

3.4.1 Introduction

Our speaker verification system, and the problems with feature matching, is actually a part of a much broader topic in scientific engineering, the subject of pattern recognition. The main goal with pattern recognition is to achieve a classification of objects of interest into one of a number of categories or classes. The objects of interest are generically called patterns and in our case they are sequences of acoustic vectors that are extracted from an input speech using the techniques described in the previous sections. Since the classification procedure in our case is applied on extracted features, it can also be referred to as feature matching.

Their is a different set of feature matching techniques used in speaker verification; Dynamic Time Warping (DTW), Hidden Markov Modeling (HMM), and Vector Quantization (VQ).

3.4.2 Clustering the Training Vectors

We are applying the Vector Quantization technique, since the VQ technique is a technique shown to be effective and has a high accuracy [14]. VQ is a process of mapping vectors from a large vector space to a finite number of regions in that space. Each region is called a cluster and can be represented by its center called a codebook entry or centroid. A full collection of codebook entries are called a codebook.

After the training process the acoustic vectors extracted from input speech of a speaker provide a set of training vectors. As described above, the next important step is to build a speaker-specific VQ codebook for this speaker using those training vectors. There is a well-know algorithm, namely the LBG algorithm [11], for clustering a set of L training vectors into a set of M codebook vectors. The algorithm is formally implemented by the following recursive procedure:

1) Design a 1-vector codebook; this is the centroid of the entire set of training vectors.

2) Double the size of the codebook by splitting each current codebook until you have reached the codebook size you were after following the rule, more entries safer but also slower system.

After making every splitting of the codebook, a nearest neighbor search has to be performed for each training vector. This is achieved by, for every training vector calculating the Euclidean distance between the codebook entry and the training vector. The vector with the smallest Euclidean distance is assigned with that vector. After this, the entry also called the cen-

troid, has to be updated. There are different ways of doing this. The approach used in the current system is to ten times update the centroid and recalculate the nearest neighbor. Another approach is to continue the calculation until the centroids does not move more than a predefined max distance. The figure below shows, in a flow diagram, the detailed steps of the LBG algorithm.

$$Dist(k) = \sum_{i=0}^{NFrames-1} min_{0 \le j \le NEntries-1} \sum_{k=0}^{Entries} |X_i(k) - S_j(k)|$$

Euclidean distance (EQ 7)

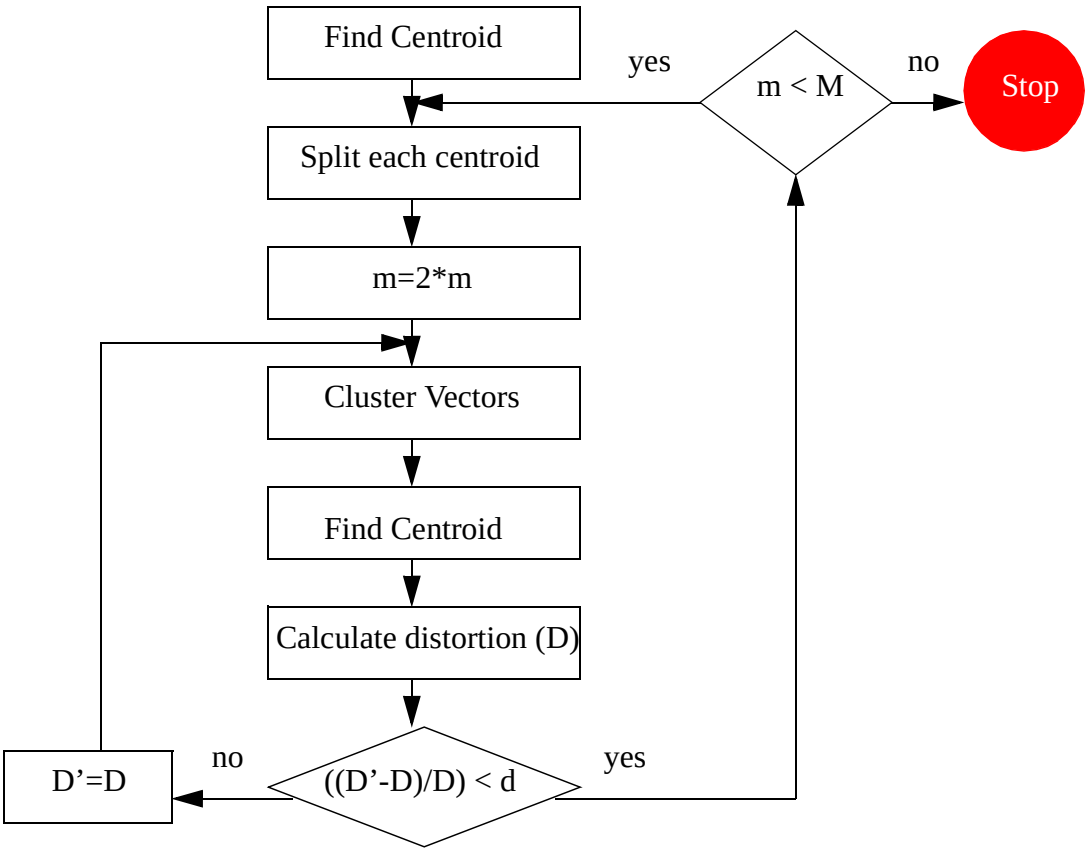

FIGURE 10. Flowchart of the LBG algorithm (derived from [11])

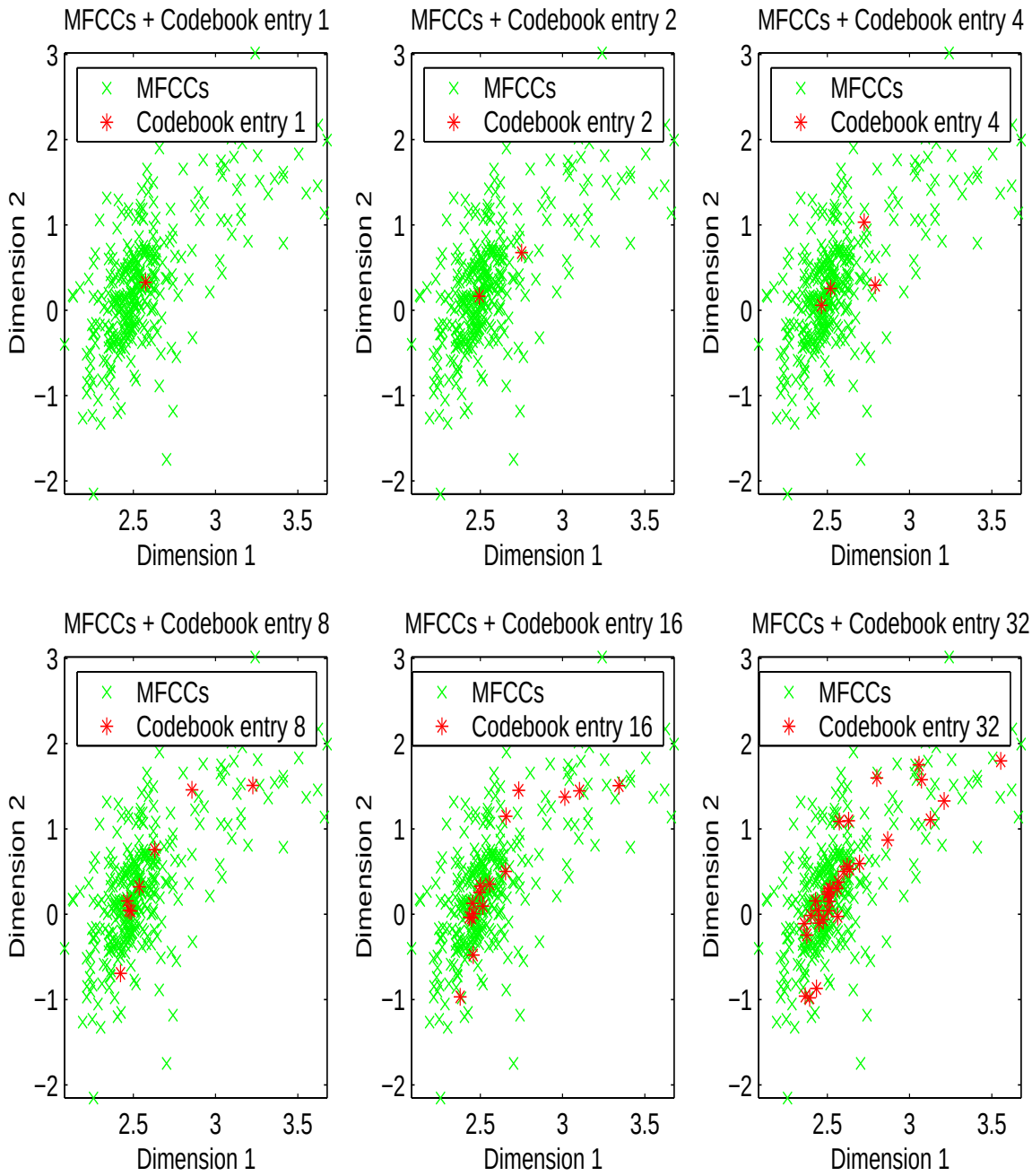

FIGURE 11. MFCC's and 32 entries codebook in two dimensions.

3.5 Verification Part

In the verification part, the distance from a verification speech vector that has gone throw the feature extraction described above is compared with the codebook for the claimed speaker (see figure 2). The distance between the closest verification speech feature vector and an entry of the codebook is called a VQ-distortion.

3.5.1 Threshold

For a speaker recognition system, the recognition feature vectors are compared with all users codebooks for which the system is trained for, and the Euclidean distance is computed. The user codebook that returns the smallest distance is said to be the user. In a speaker verification system, as described in figure two, the user also has to enter his/her user ID together with the speech utterance. In this system, only the distance between the claimed users codebook and the verification feature vectors is calculated. But this is not all. In difference from the recognition system, it is not enough to only take the user with the smallest distance, here we also will calculate a ratio, that is compared with a preset threshold. If the ratio is less than the threshold, the user is said to be a true speaker, otherwise the user is said to be a false speaker or imposter.

3.5.2 Cohort Speakers

To be able to calculate a ratio, in our system we are using the concept of cohort speakers. In our system we are using ten cohort speakers, and for each of the cohort speakers a codebook is calculated. These codebooks are in the verification part utilized for calculating the ratio. The same procedure as described above, with calculating the distance between the claimed speaker and all of the cohort codebooks is completed and the mean value is calculated out of the ten cohort speakers. The ratio then is calculated using the following experimentally derived formula:

$$Ratio = \left(\frac{EuclidDist_{claimant}^{\ 2}}{\frac{Ncohort}{\frac{1}{N_{cohort}} \times \sum_{i=1} EuclidDist_{imposter(i)}}} \right)^4$$

Ratio calculation (EQ 8)

4.0 Implementation in JAVA

After deriving the concept of a Vector Quantization speaker verification system using Mel Frequency Cepstral Coefficients, the next step is to implement the system in Java. For the development of the software, Java2TM version 1.3 was utilized.

4.1 Speech Data

Firstly, according to figure 2, the user is asked to type in a username. Everything that has to do with this speaker will utilize the username. Secondly the speech data has to be recorded. There are some hardware constrains for our system. A soundcard has to be installed on the computer, with a pair of speakers and a microphone attached.

A stream from the soundcard's microphone port is opened, and firstly the background noise is recorded. For two seconds the background noise is recorded and the power of this data is calculated. After this, the real speaker data is recorded for approximately 6 seconds, and the frameing of the data earlier described, is done. The power of these frames is calculated and if the power is less than 1.5 times the background noise power, the frame is discarded.

A good rule of thumb is to record approximately ten times as many speech frames as you want to have entries in your codebook for the training phase, and five times as many speech frames as entries in your codebook for the verification phase. In our system a codebook holding 32 entries is utilized, hence 320 speech frames are recorded for the training phase and 160 speech frames for the verification phase.

The speech is recorded at a sampling rate of 8000 Hz, using 16 - bits resolution.

4.2 Speech Processing

Every speech frame has to go through the above described steps of speech processing, windowing, FFT, Mel-Frequency Wrapping, and MFCC calculation. To achieve some sort of normalization of the MFCC's, a weighting array is applied to the MFCC feature vectors. This has been showed to improve the performance [12]. The figure below shows a unnormalized mean vector of MFCC's and the normalization vector. In our system we are using 18 mel frequency cepstral coefficients.

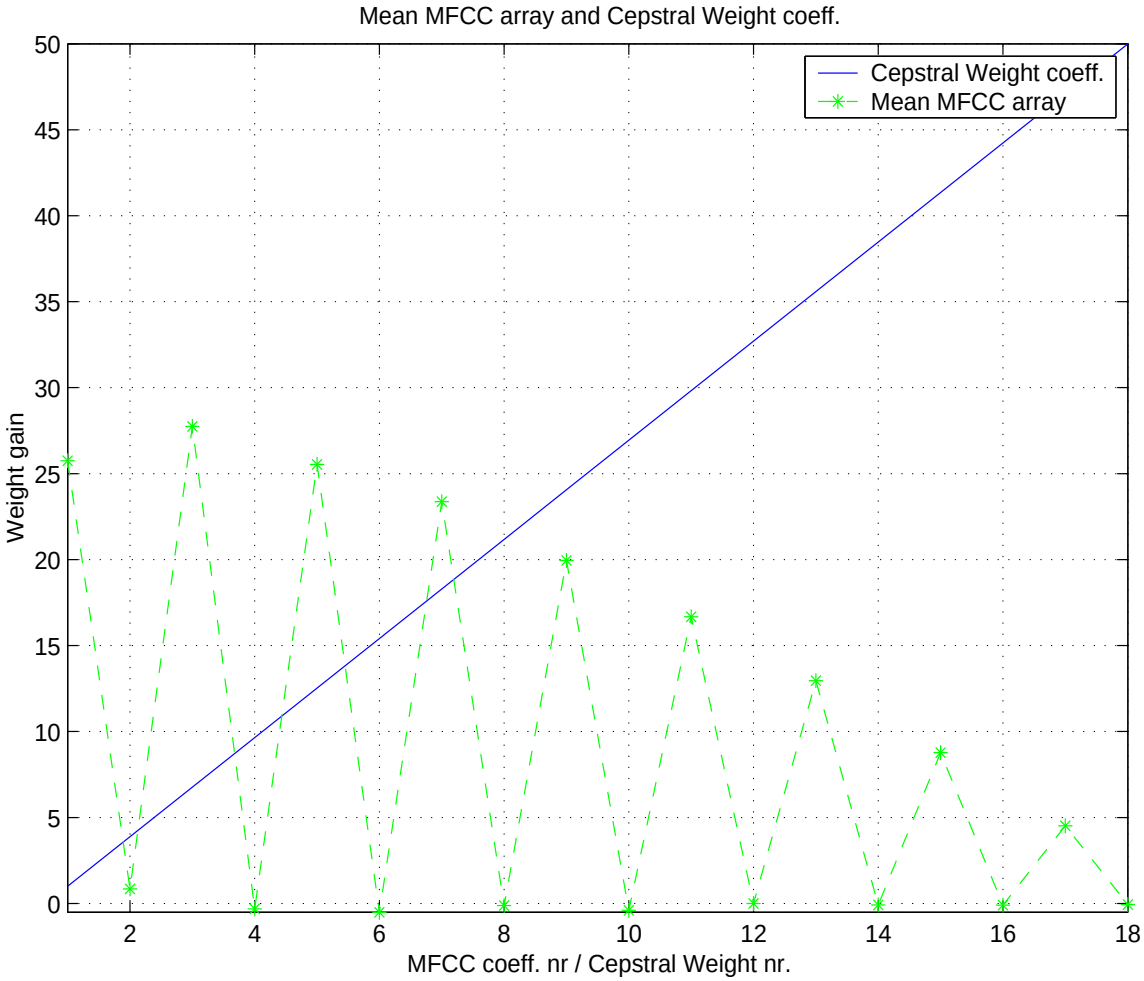

*FIGURE 12. Mean MFCC array and Cepstral Weight coefficients (MFCC's * 10)*

When this is completed, to obtain a fast system, only 10% of the frames are saved for later comparison, this one by using the LBG algorithm described above.

For the verification process of the implementation in Java, the user is asked to type in the username that he/she used when training the system. If the user types in an incorrect username, he will be asked to try again. The user is asked to speak for approximately 4 seconds. During this process 160 speech frames are stored for verification, and the ratio is calculated. If the user is the correct user, he will be given a OK otherwise the user is declined access.

4.3 Simulation and Evaluation

Before the system can be utilized, the algorithm has to be simulated and the system be evaluated. For the simulation and evaluation, the speech database TIMIT is used.

4.3.1 Tests on TIMIT

The TIMIT speech database is a result of joint effort from several American Institutes, where among Massachusetts Institute of Technology (MIT), Stanford Research Institute (SRI), and Texas Instruments (TI) can be found. The corpus speaker distribution found in TIMIT contains of a total of 6300 sentences, 10 sentences spoken by each of 630 speakers from 8 major dialect regions of the United States [5]. TIMIT is recorded under a very good surrounding with a high signal to noise ratio (SNR). The recording of TIMIT is made on with a 16000Hz sampling frequency, 16 - bits resolution. The Signal Processing Laboratory has access to a down sampled to 8000Hz, bandpass filtered noise version of this database, called NTIMIT8. In the simulation and evaluation of the Java algorithms 100 persons were used randomly selected from 3 different areas. The selection of female and male utterances is 50% - 50%.

4.3.2 Error rate

The performance of a speaker verification system [13] is measured in terms of false acceptance rate (FA%) and false rejection rate (FR%):

$$F_A = \left(\frac{I_A}{I_T}\right) \times 100$$

$$F_R = \left(\frac{C_A}{C_T}\right) \times 100$$

False Acceptance, False Rejections (EQ 9)

Where I_A is the number of imposter classified as true speakers, I_T is the total number of speakers, C_R is the number of true speakers classified as imposters, C_T is the total number of speakers. To calculate the total error of a verification system, T_E, the false acceptance rate is added to the false rejection rate giving:

$$T_E = F_A + F_R$$

Total error (EQ 10)

The following picture shows the ratio calculation of our system.

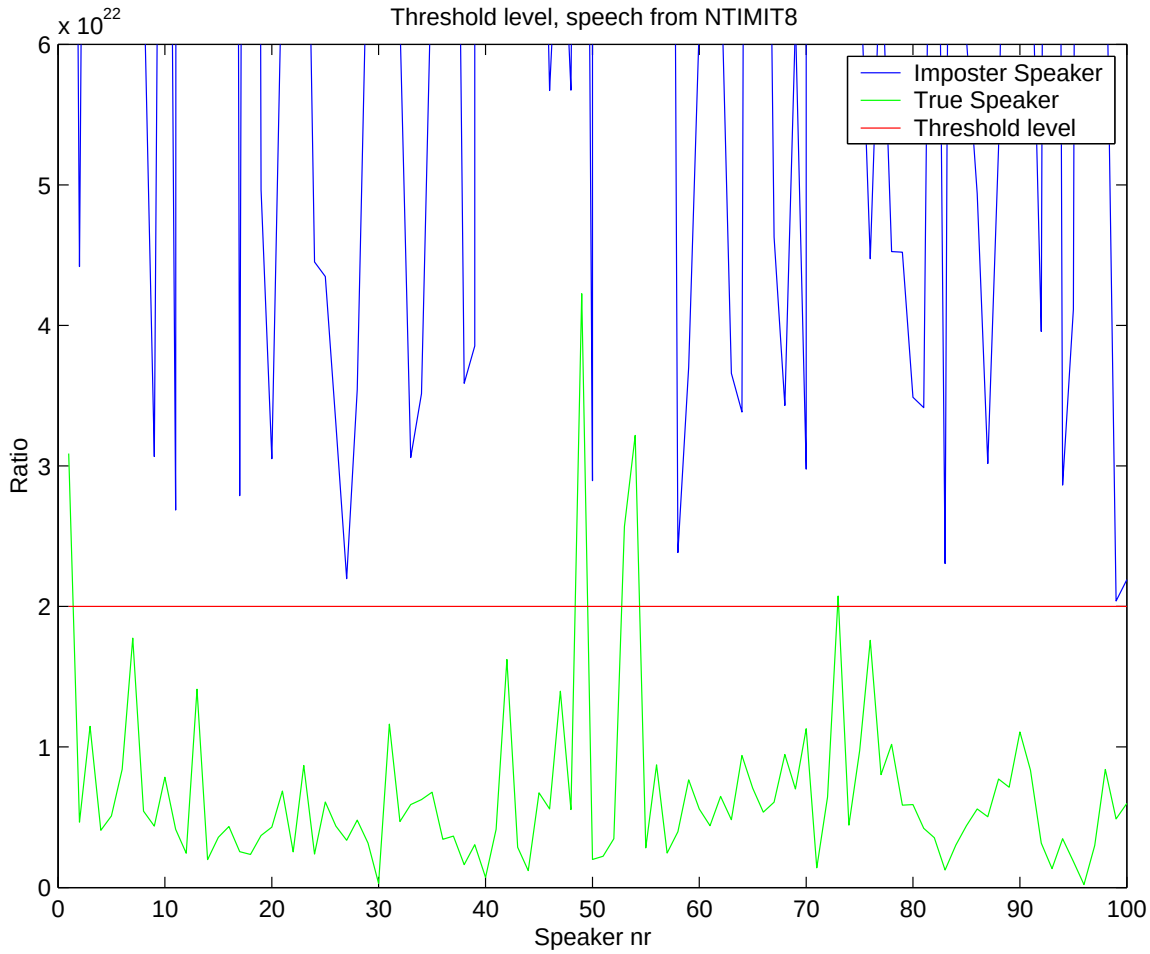

FIGURE 13. Threshold and error rate plot from NTIMIT8

The figure shows on four percent total error, zero percent false acceptance and four percent false rejection.

The number of entries in the codebook, and under what conditions the speech was recorded has a great part in the total error result. The table below shows an approximation of the error rate when using different number of entries in the codebook [14].

TABLE 1. Total Error Rate depending on Codebook Entries

Number of Codebook Entries	Speaker Verification Rate (%)
32	96.0
64	97.7
128	97.9
256	98.1
512	98.2

5.0 Parallel Port Extension

For demonstration purposes and for addressing the problems with accessing hardware through Java, a parallel port module was constructed.

5.1 JAVA and Hardware accessibility

As mentioned earlier, to access any hardware from a web browser based applet, is a big “no no”, since this should become a big security problem. Due to this fact, our software is an application. When dealing with applications, the same hardware constrains are not present. Even when dealing with an application there are many problems present. In the standard version of Java 2, there is not any possibility to access the ports of the computer, the parallel port and the serial port. SUN has constructed a platform dependent toolkit for this access, destroying the platform portability, but this is the only way of accessing the ports [15].

Since we are in need of accessing the sound card from our “live” system, there are also platform dependent problems, hence our software is developed under Microsoft Windows 98, will also run under the other present versions of Microsoft Windows.

To note though is that all the algorithms and test programs are platform independent and can easily be executed under both Microsoft Windows and Linux.

5.2 Hardware module

For demonstration purpose, a hardware module was constructed. This to address the earlier mentioned problems with accessing the hardware through Java and for making it easier when demonstrating the software for the audience.

The hardware module is connected to the parallel port of a PC and will show the decision of the verification process. If the result is a true speaker, four green LED’s will be turned on, for an imposter, four red LED’s will turn on.

Connection diagram can be found in the figures below.

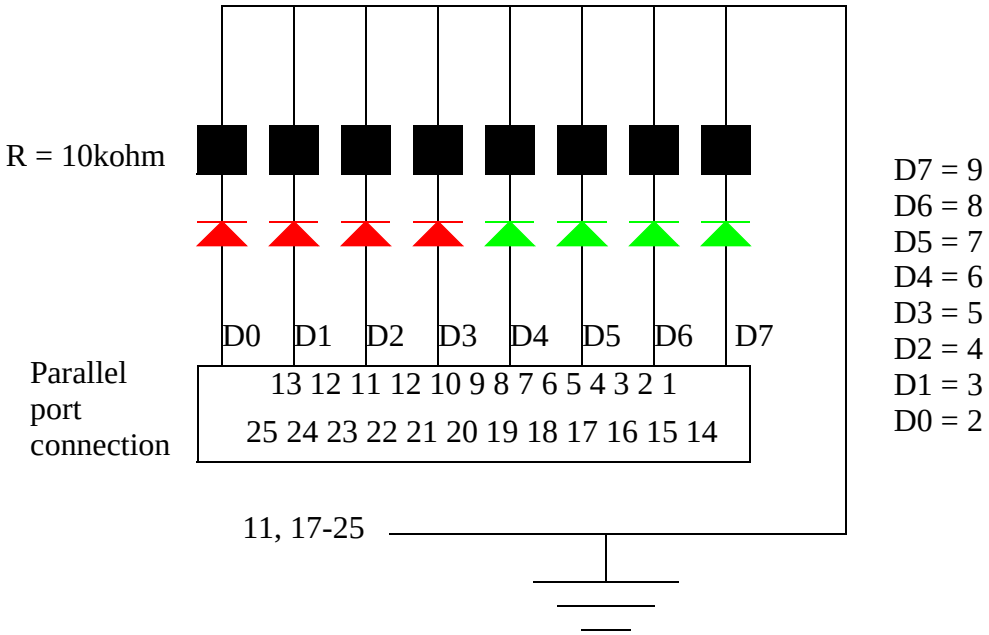

FIGURE 14. Connection diagram

6.0 Conclusion

A speaker verification application has been developed and implemented using Matlab and Java. Since Java is said to be platform independent, some experiments has been conducted that has showed this is not always the case. As soon as you would like to access the hardware through Java you will face problems that will make your software to become platform dependent.

It is interesting to note that all the algorithm developed for the speaker verification system is platform independent, except for those parts accessing the sound card of the users computer, and can easily be executed under both Microsoft Windows and Linux.

Java is a very general tool and the support of porting the software across different computer platforms and networks can be very interesting if an author of a publication can offer the reader to conduct into the research by extending the testing of the algorithms or to participate in the solution of difficult problems by becoming part of a virtual supercomputer.

Testing, simulation and verification of the speaker verification program show a total error rate of four percent.

7.0 Ideas for further studies

A new and very interesting method of feature extraction has been developed at the Signal Processing Laboratory, Griffith University. The so called MACV's [16]. MACV stands for Maximum Autocorrelation Values. Studies has showed that verification system employing both MFCC and MACV features has better performance than systems using MFCC's only.

8.0 References

[1] Holzner, S., “Java Black Book”, Coriolis Group, ISBN 1588800970

[2] Digital, “Writing Portable Applications”, http://wint.decsy.ru/alphaservers/digital/v0000972.htm, (Acc 2001-10-10)

[3] Rabiner, L., Juang, B.H., “Fundamentals of Speech Recognition”, Prentice-Hall Inc., 1993

[4] Atal, B.S., “Automatic recognition of speakers from their voices”, Proc. IEEE, Vol. 64, No. 4, pp. 460-475, April 1976.

[5] The DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus, “TIMIT”, Massachusetts Institute of Technology, Stanford Research Institute and Texas Instruments, 1990

[6] Young, S. et al. “The HTK Book”, Version 3.0, July 2001

[7] Proakis, J.G, Manolakis, D.G, “DSP, principles, algorithms, and applications”, Prentice-Hall Inc. 1996, ISBN 0 13394289 9

[8] Reynolds, D., “A Gaussian Mixture Model Approach to Text-Independent Speaker Identification”, MIT, Technical Report 967, 1995

[9] Nilsson, M., “FFT, Realization and Implementation in FPGA”, Master thesis, Ericsson Microwave Systems AB / Griffith University, 2000 - 2001

[10] Picone J. W., “Signal Modeling Techniques in Speech Recognition”, Proc. of the IEEE, Vol 81, Nr. 9, September 1993, pp. 1215-1247

[11] Linde, Y., Buzo A., Gray, R. M., “An algorithm for vector quantizer design”, IEEE Trans. on Comm., Vol. COM-28, pp. 84-95, Jan. 1980

[12] Soong, F. Rosenberg, A.E., “On the use of Instantaneous and Transitional Spectral Information in Speaker Recognition”, IEEE Trans. Acoustics, Speech, and Signal Processing, Vol 36, No. 6, June 1988

[13] Sanderson, C., “Joint Cohort Normalization in a Multi-Feature Speaker Verification System”, submitted to The 10th IEEE International Conference on Fuzzy Systems, Melbourne, Australia, 2-5 December 2001

[14] Matsui, T., Furui, S., “Comparison of text-independent speaker recognition methods using VQ-distortion and discrete/continuous HMMs”, Acou, Speech, and Signal Processing, 1992. ICASSP-92., Volume: 2 , 1992

[15] SUN, “Java API”, Version 1.3, 2001, http://www.java.sun.com (Acc 2001-10-10)

[16] Wildermoth, B. R., Paliwal, K. K., “Use of voicing and pitch information for speaker recognition”, 8'th Australian International Conference on Speech Science & Technology (SST 2000), Canberra, Australia, 4-7 December 2000

Appendix A. Java Code, simulation

Appendix B. Matlab Code, simulation

Appendix C. Java Code, application