Speaker Recognition/Verification
Software for use in Security Applications

B.E. Electronic & Computer
Engineering Project

Student: Matthew Byrne
Id: 05362725

Supervisor: Dr Edward Jones
Co-Supervisor: Dr Martin Glavin
Abstract

Project Area

Speaker recognition is the process of automatically recognising who is speaking based on the characteristics of information included in speech signals. It can be divided into Speaker Identification and Speaker Verification. Speaker identification determines which registered speaker provides a given utterance from a set of known speakers. Speaker verification accepts or rejects the identity claim of a speaker - is the speaker the person they say they are?

Speaker recognition technology makes it possible for the speaker's voice to control access to restricted services, for example, phone access to banking, database services, shopping or voice mail, and access to secure equipment/environments.

Both technologies require users to "enrol" in the system, that is, to give examples of their speech to a system so that it can analyse and store characteristics of their voice patterns.

This thesis will concentrate on development of a Speaker identification system for use in a security application.

Goals

The overall objective of this project is to develop a recognition system that recognises a person based on his/her speech patterns.

Overall Result

A speaker recognition system was developed by creating a database of trained speaker utterances, and testing these against an input speaker utterance to determine the speaker identification.
Acknowledgments

I would like to thank all the staff of the Electronic Engineering Department, especially my supervisor Dr. Edward Jones, and co-supervisor Dr. Martin Glavin for their invaluable words of wisdom and help throughout the year. I would also like to thank the NUI Galway Electronic Engineering department technicians Miles Mehan and Martin Burke for their help over the past 4 years and during my project.

Finally I wish to thank my friends, family and classmates for their help and support during the design, implementation and any aspect of this system.
Declaration of Originality

I declare that this thesis is my original work except where stated

Signature…………………………..   Date…………………………
Contents

Abstract ii
Acknowledgements iii
Declaration of Originality iv

Contents 1

List of figures 3

1. Introduction 5
   1.1 Project Background 5
   1.2 Theory behind Speaker Recognition/Verification software 5
   1.3 Project Aim 7
   1.4 Report Layout 9

2. Technology Background 10
   2.1 MATLAB 10
   2.2 C Programming Language 11
   2.3 Mel Frequency Cepstral Coefficients (MFCCs) 11
   2.4 Artificial Neural Networks (ANNs) 12

3. Research and Development 14
   3.1 Initial work and development 14

4. Front End Processor 15
   4.1 Analysing and creating a database of speaker characteristics 15
   4.2 Endpointing and removing silence 15
   4.3 Overlapping and windowing 16
   4.4 Analysis with MFCC function 17
   4.5 Removing unnecessary data/coefficients 19
   4.6 Adjusting matix for use in ANN 20

5. Neural Network Classifier 22
   5.1 Neural Network Classifier 22
6. Future development

   6.1 VoIP technology and Speaker Recognition  25
   6.2 Real time implementation  27
   6.3 PC GUI  27

7. Discussion  28

8. Conclusion & Results  29

APPENDIX I - TABLE OF REFERENCES  30
APPENDIX II - Code
   Program code  31
   MFCC code  34
# List of figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fig 1.1</td>
<td>Initial flow diagram of project</td>
<td>6</td>
</tr>
<tr>
<td>Fig 1.2</td>
<td>Artificial Neural Network</td>
<td>8</td>
</tr>
<tr>
<td>Fig 2.1</td>
<td>MFCC Triangular filter matrix</td>
<td>12</td>
</tr>
<tr>
<td>Fig 2.2</td>
<td>Flow of the MLP neural network</td>
<td>13</td>
</tr>
<tr>
<td>Fig 4.1</td>
<td>Loop for opening a sequence of files</td>
<td>15</td>
</tr>
<tr>
<td>Fig 4.2</td>
<td>Original signal after removing silence</td>
<td>16</td>
</tr>
<tr>
<td>Fig 4.3</td>
<td>Defining Hamming window</td>
<td>16</td>
</tr>
<tr>
<td>Fig 4.4</td>
<td>1 frame of speech signal</td>
<td>17</td>
</tr>
<tr>
<td>Fig 4.5</td>
<td>Effect of windowing on the signal</td>
<td>17</td>
</tr>
<tr>
<td>Fig 4.6</td>
<td>Calculating MFCC triangular filters</td>
<td>17</td>
</tr>
<tr>
<td>Fig 4.7</td>
<td>Code to compute the filter matrix</td>
<td>18</td>
</tr>
<tr>
<td>Fig 4.8</td>
<td>DCT of a frame</td>
<td>18</td>
</tr>
<tr>
<td>Fig 4.9</td>
<td>Original number of frames Vs. Average overall number of frames</td>
<td>19</td>
</tr>
<tr>
<td>Fig 4.10</td>
<td>Remaining coefficients 1 – 8</td>
<td>20</td>
</tr>
<tr>
<td>Fig 4.11</td>
<td>Matrix containing 80 utterances of a word</td>
<td>21</td>
</tr>
<tr>
<td>Fig 4.12</td>
<td>Speech utterance extracted from the matrix “Train”</td>
<td>22</td>
</tr>
<tr>
<td>Fig 5.1</td>
<td>Multilayer Perceptron neural network overlay</td>
<td>24</td>
</tr>
<tr>
<td>Fig 6.1</td>
<td>Flow diagram of VoIP call to a Phone banking system</td>
<td>26</td>
</tr>
</tbody>
</table>
Glossary

**FFT** – Fast Fourier Transform. FFT is an efficient algorithm to compute the discrete Fourier transform and its inverse.

**DCT** – Discrete Cosine Transform. DCT is a Fourier-related transform similar to the Discrete Fourier Transform, but using only real numbers.

**MFCC** – Mel Frequency Cepstral Coefficients. An acoustical coefficient extracted from a speech waveform to capture phonetically important characteristics of speech.

**ANN** – Artificial Neural Network. ANN is a mathematical model based on biological neural networks (the human brain).

**MLP** - Multilayered Perceptrons (MLPs) are a feed forward network most commonly used for neural networks.

**GUI** – Graphical user interface. GUIs are used as a simple interface between user and software.
Chapter 1 – Introduction

1.1 Project Background

Speech is the ultimate ubiquitous interface. It is the most natural and universal method of communication between people. The aim of speaker recognition is to extend that communication to use within everyday scenarios of security applications, telephone services etc.

In many applications, speech may be the main or only means of transferring information and so would provide a simpler method for authentication. The telephone system provides a familiar network for either delivering instructions, or logging into a system through speech. For telephony based applications, there would be no need for any equipment or networks to be installed as the telephone and mobile phone network would suffice. For non-telephone applications, soundcards and microphones are available as a cheap alternative. The application would only need to be implemented at the receiver end to analyse, characterise and identify the user, hence providing a seamless installation of the system for the user.

1.2 Theory behind Speaker Recognition/Verification software

The aim of a speaker recognition system is to exploit differences in the speakers’ voice quality and style of speech. There are two different methods in speaker Recognition:

- Verification: Determining whether an unknown voice is from one particular enrolled speaker.
- Identification: Associating an unknown voice with one from a set of enrolled speakers.

These systems can be broken down into either text-dependant or text-independent systems. Most systems are based on scenarios where users repeat prompted preset phrases or speak a fixed PIN. These employ what is know as a text-
dependant system. Text-dependant systems can greatly improve accuracy, though such constraints can be hard or impossible to enforce in certain cases. Text-independent systems analyse and use any spoken word to identify the user. Both types of system have a training/enrolment phase, where features relevant to speaker identity are extracted from their utterance of a word and stored for verification against an input utterance in testing. Pattern matching techniques are then used to compare an input utterance with the stored ones.

The first step in the testing stage is to extract features from the input speech, similar to that during training, compare the input speech to all other stored templates and select the most accurately matching template and ID the speaker. The figure below (Fig 1.1) charts this process.

![Initial flow diagram of project](image-url)
1.3 Project Aim

The main aim of this project was to investigate and build a speaker recognition system that recognises a person based on his/her speech patterns, capable of use in security applications.

Such a system could be used, for example, in a security application where the identity of a user is determined based on analysis of a pre-determined (text dependent) or unknown (text independent) phrase. Also, speaker recognition systems differ slightly from speaker verification systems, in particular recognition involves choice of an identity from a set of possible users, while verification involve confirmation of a claimed identity (i.e. a “yes/no” answer).

Such systems typically involve two main components:

- A Front End Processor (FEP) which processes the spoken input and extracts useful “features” from the signal. There are various approaches to the FEP, most of which are based in some way on spectral analysis of the speech signal. Typically a set of 20 cepstral coefficients spaced in a non-uniform mel frequency scale are used to extract these properties. These are thus called, mel-frequency cepstral coefficients (MFCCs). For a speech sample, sampling over a long period of time (half of a second), the signal characteristics change to include all speech sounds being spoken. For this reason short-time spectral analysis is used; the speech signal is broken into approximately 30ms. At 30ms the characteristics of that segment of speech are relatively stationary and can be extracted accurately.

- A Classifier, whose function is to examine the features and make a decision based as to the identity of the speaker. There are also a number of possible approaches for the Classifier including Artificial Neural Networks (ANNs), as well as various statistical methods. Neural networks are pattern-matching devices based on the human brain. They consist of interconnected processing units (neurons). The most widely used neural network classifier is a Multi-layer perceptron (MLP, fig 1.2). In pattern matching applications, the network is trained
by presenting a pattern vector at the input layer and computing the outputs. The output is then compared with some desired output, which is a set of output unit values, which will identify the input pattern. Given enough hidden nodes and enough data, a MLP can approximate virtually any function to any desired accuracy. An important problem for multilayer networks is to determine an optimal number of hidden nodes. [2]

**Web VoIP** functionality may also be added to the tool to enable access to secure information (eg. online banking) over the internet. Various problems arise when considering this implementation, most importantly; the users’ internet connection. Further detail in chapter 3.3

---

Fig 1.2: Artificial Neural Network [3]
1.4 Report Layout

In the subsequent chapters the following will be discussed:

- The technology use throughout the project. This mainly involved working with the MATLAB programming language.
- Chapter 2 discusses the approaches taken towards speaker recognition software; the close approximation to the human auditory system of the MFCC and the training network of artificial neural networks using a MLP system. Other ideas were considered in the earlier stages of development and research such as using the Hidden Markov Model Toolkit (HTK) to manipulate Hidden Markov Models. This approach has been used in similar projects involving pattern recognition.
- Chapter 3, 4 and 5 focus on researching and developing the system including initial investigation into speaker recognition software and how it works, Front end processor and classifier development.
- Chapter 6 describes some possible future development for the project; implementation over the internet using VoIP technology, and a real time implementation of the program.
Chapter 2 – Technology Background

2.1 - MATLAB

MATLAB is a numerical computing programming language created by "The MathWorks". MATLAB® is a high-level technical computing language and interactive environment for algorithm development, data visualization, data analysis, and numeric computation. Using the MATLAB product, you can solve technical computing problems faster than with traditional programming languages, such as C, C++, and Fortran.[4]

You can use MATLAB in a wide range of applications, including signal and image processing, communications, control design, test and measurement, financial modeling and analysis, and computational biology. Add-on toolboxes (collections of special-purpose MATLAB functions, available separately) extend the MATLAB environment to solve particular classes of problems in these application areas.

MATLAB provides a number of features for documenting and sharing your work. You can integrate your MATLAB code with other languages and applications, and distribute your MATLAB algorithms and applications.

Typical uses of MATLAB include:

- High-level language for technical computing
- Development environment for managing code, files, and data
- Interactive tools for iterative exploration, design, and problem solving
- Mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, and numerical integration
- Functions for integrating MATLAB based algorithms with external applications and languages, such as C, C++, Fortran, Java, COM, and Microsoft Excel
2.2 - C Programming language

C has been used successfully for every type of programming problem imaginable from operating systems to spreadsheets to expert systems [5].

For real time implementation and portability to various operating systems and system platforms, the developed code will need to be converted to C programming language. As C is a widely common programming language it brings with it more options for implementation.

2.3 - Mel Frequency Cepstral Coefficients (MFCCs)

In sound processing, MFCC’s are based on the known variation of the human ear’s critical bandwidths with frequency; filters spaced linearly at low frequencies and logarithmically at high frequencies are used to capture phonetically important characteristics of speech. The difference between FFT/DCT and the MFCC is that in the MFCC, the frequency bands are spaced logarithmically on the mel scale, which approximates the human auditory system’s response more closely then the linearly-spaced FFT or DCT frequency bands. Code for setting up the filter bank was acquired online from www.speech-recognition.de[6] and altered to integrate with the system.

Each filter in the MFCC has a magnitude response which is a triangle covering a certain range of frequencies (Fig 2.1). The triangular filter is used to weigh a piece of the spectrum, and then the weighted values are summed together to give the overall filter output. The start of each filter is equal to the centre frequency of the preceding one, and the end of each filter is the centre of the following one (the first filter starts at 0 and the last one ends at 4000).
Suppose you’re dealing with the $12^{\text{th}}$, filter, which has a centre frequency of 1292 Hz. The “lower slope” of the triangle goes from 0 to 1 between 1137 and 1292 Hz, while the “upper slope” of the triangle goes from 1 back down to 0 between 1292 and 1469 Hz.

### 2.4 - Artificial Neural Networks (ANNs)

An Artificial Neural Network, often just called a neural network, is a mathematical model based on biological neural networks (the human brain). It consists of an interconnected group of artificial neurons and processes information using a connectionist approach to information processing. In most cases an ANN is an adaptive system that changes its structure based on external or internal information that flows through the network during the training phase. There are many types of neural networks for various applications; multilayered perceptrons (MLPs) are feedforward networks and approximators. They are the simplest and therefore most commonly used neural network architectures. Fig 2.2 below shows a basic flow diagram of an MLP ANN system.
The MLP is trained in the following way: Using a training set of vectors, they are labelled with “0.9” if they belong to the speaker that it is going to be tested, and with “-0.9” if it is from a different speaker. Thus, MLP is trained for a “0.9” output when the input belongs to the speaker and a “-0.9” output when the input is from a different speaker. Obviously the output will rarely be exactly 0.9 and -0.9, hence some error compensation will be taken into account and the utterance with the closest match, within a range, will be selected as the identified user.

Fig 2.2: Flow of the MLP neural network
Chapter 3 – Research and development

3.1 – Initial work and development

The project initially involved algorithm development and simulation using MATLAB, and using pre-existing publicly available databases of speech. However, the project will also involve the development of a system for real-time operation, using a commercially-available embedded microprocessor development system or a system based on a dedicated Digital Signal Processor (depending on the choice of hardware, only a subset of the functionality of the full system may be implemented).

Initially becoming accustomed to MATLAB and the process of how programs are developed within it needed to be overcome. Facilities such as the mathworks.com website forum and tutorials were helpful to developing a better coding practice.

A text-dependent, speaker identification system would be developed initially, being that it is a simpler system and a good platform to build upon. Input speech data is required for both training and testing. The speech used for initial training and testing was pre-recorded.

The system initially requires the pre-recorded speech file. In latter stages, the system will use recorded audio from multiple users. This will require a microphone for input which will store the signal temporarily while the system analyses it against stored users.

In the final stages of the project, the system will wait for a prompt to start recording. The system will then wait for audio silence to stop analyzing. This audio silence will then be removed from the signal and then be analysis of the signal will continue as usual.
Chapter 4 – Front End Processing

The front-end processing stage is performed in Matlab. Front-end processing is vital as it off-loads the computational data from the rest of the system. The front-end processor is used to extract useful details about the speech data from the input audio file (in this case it extracts MFCCs).

4.1 – Analysing and creating a database of speaker characteristics

For training purposes a database of speakers would be analyzed. Firstly the average number of frames over the database is calculated. Later each utterance will be compressed to this average number for consistency in testing. This involves a loop to cycle through each speech utterance to extract the relevant data (Fig 4.1 below). The loop reads through a simple text file which contains the path address to each utterance on individual lines.

```matlab
ListLoop = fopen('SpeakerList.txt'); %Opens the list of utterance files to be read
while 1
    tline = fgetl(ListLoop); %Reads each line
    %Loop through each line until EOF.
    if ~ischar(tline), break, end

End
```

Fig 4.1: Loop to open a sequence of files

4.2 – Endpointing and removing silence

Endpointing would be implemented (to isolate beginning and end of an utterance) after data is input. Silence at the beginning and end of each utterance must be removed as it is wasted data and would give false readings under testing (Fig 4.2). The average number of frames is then calculated using a simple maths formula.
4.3 – Overlapping and windowing

The loop above (Fig 4.1) is repeated to now extract data from the database of utterances. A 50% overlap of frames is applied for better results. Also a hamming window is used to minimise signal discontinuities at the beginning and end of each frame. The idea is to minimise the spectral distortion by using the window to taper the signal to zero at the beginning and end of each frame. The hamming window is defined by:

\[
W[nT] = \begin{cases} 
0.54 - 0.46 \cos(\frac{2\pi n}{N}) & 0 \leq n \leq N \\
0 & \text{otherwise}
\end{cases}
\]

Where \( n \) is the sample number.
4.4 – Analysis with MFCC function

The signal is then transferred into the frequency domain using the Fast Fourier Transform (FFT), where we represent the short term spectrum with MFCCs. A Discrete cosine transform (DCT) is returned to aid in compression and efficiency (fewer are needed to approximate the signal).

Values of sampling frequency, frame length and the number of channels are passed into the MFCC function. Figure 4.6 demonstrates how the triangular filters for the MFCCs are calculated, and Figure 4.7 shows how they are analysed against the utterance. The signal is analysed against this filterbank. A Discrete Cosine Transform is applied and the result and each subsequent result in the loop are stored to a Matrix.

The triangular filters are defined such that:

\[
w_j(k) = \begin{cases} 
\frac{(k / K) f_s - f_c_{j-1}}{f_c_{j-1} - f_c_{j-1}} & l_j \leq k \leq c_j \\
(f_c_{j+1} - \left(\frac{k}{K}\right) f_s) / f_c_{j+1} - f_c_j & c_j < k \leq u_j \\
0 & \text{elsewhere}
\end{cases}
\]

Fig 4.6: Calculating MFCC triangular filters [8]
Where \( f_{c_{j-1}} \) and \( f_{c_{j+1}} \) are the lower and upper limits of the pass band filter \( j \) with \( f_{c_0} = 0 \) and \( f_{c_j} < f_s / 2 \) for all \( j \), and \( l_j, c_j \) and \( u_j \) are the DFT indices corresponding to the lower, centre and upper limits of the pass band for filter \( j \).

```matlab
% compute matrix of triangle-shaped filter coefficients
W = zeros(nofChannels,Nmax);
for c = 1:nofChannels
    % left ramp
    increment = 1.0/(indexcenterI - indexstartI);
    for I = indexstartI:indexcenterI
        W(c,i) = (I - indexstartI)*increment;
    end %i
    % right ramp
    decrement = 1.0/(indexstopI - indexcenterI);
    for I = indexcenterI:indexstopI
        W(c,i) = 1.0 - ((I - indexcenterI)*decrement);
    end %i
end %c
```

Fig 4.7: Code to compute the triangular filter matrix

![Graph](attachment:image.png)

Fig 4.8: DCT of a frame
4.5 – Removing unnecessary data/coefficients

As shown in Figure 4.8 after approximately 8 - 10 coefficients the signal approximates to zero. These coefficients greater then 8 can be disregarded as they do not provide enough data to characterise the speaker. The number of coefficients may vary, but generally they approximate to zero between 8 and 12 coefficients. To do this we first build a matrix of all frames of that utterance. Each single “MFCC’d” frame (Fig 4.8 above) is stored in a matrix of size (Number of frames)*(No. of coefficients), column by column. Next the matrix must be interpolated to the average number of frames. The original and resulting matrix can be seen in Figure 4.9.

For this utterance the number of frames was less then the average over the whole set of utterance, and hence the number of frames was increased. The interp1 function is used to interpolate the number of frames in the signal to a common average. It requires the current spacing of the Matrix which It is interpolating, and the new spacing of the target “average length” matrix.

Now the utterance matrix can be cut from 19 to 8 coefficients as mentioned above. Removing these coefficients will reduce computational time and improve efficiency which is highly important in the use of real time applications. This is achieved simply by only selecting the first lower 8 coefficients. The resulting matrix is shown below in Figure 4.10. As can be seen, many of the coefficients approximating around zero have been removed.
4.6 – Adjusting matrix for use in ANN

To use this matrix with the MLP ANN each spoken utterance, such as the one above in Figure 4.10, is reconstructed into a single column vector by aligning each column into 1 continuous column of (8 coefficients)*(Number of frames) rows. As the code loops, these columns are passed into a matrix large enough to store each of these vectors for each utterance being used in testing. The Matrix must first be inverted in order to remove the upper 9 to 19 coefficients. A new matrix is created, by selecting only the bottom 8 coefficients. The reshape function is the used to change the matrix from an 8 rows by 35 columns matrix, to a single column vector consisting of the 35 rows of 8 coefficients, stacked upon itself resulting in a vector of length 280. ie. Column 1 is stored as the first 8 rows of the new matrix, column 2 is stored in row 9 -> 16, column 3 is stored in row 17-> 24 etc. As the code loops through its 80 utterances of speech, each 280 length vector will be stored into a large matrix of 280 rows by 80 columns.
The resulting matrix is shown below (Fig 4.11) can be separated into each individual matrix and can be accessed using Train(:, X); where X is the utterance to be accessed (1->80). Eg. Fig 4.12

![Figure 4.11: Matrix containing 80 utterances of a word](image-url)
This and all 80 utterances stored will be used to compare with the input utterance in a neural network which will be described in the next chapter.

Chapter 5 – Neural Network Classifier

5.1 – Neural Network Classifier
The first step in model training was to produce a baseline network, a MLP with 35 frames of 8 element feature vectors for 280 input neurons, and 8 output neurons for the 8 possible speakers to match the input neurons to. Every node in a layer is connected to each node in both the above and below layer. The connections carry weights which show the behaviour of the network and get adjusted during the training process. There are two stages to the operation of the network; the forward pass and back-propagation. In the forward pass, the input pattern vector is passed into the network and the output of
the input node is the components of the input pattern. The input to a node $j$ (seen in Fig 5.1) is given by:

$$input_j = \sum_i w_{ji} out_i \quad [9]$$

Where $w_{ji}$ is the weight (illustrated in Fig 5.1), connecting node $i$ to node $j$ and $out_i$ is the output from the node $i$.

The output of a node $j$ is:

$$output_j = f(input_j) \quad [9]$$

The function $f$ denotes the activation function of each node. This output then sent to all the nodes in the following layer and continues throughout all the layers of the network until the output layer is reached and an output vector is generated. The output signal should indicate which output node is appropriate for the input data. ie. A high output value will appear on the correct node while other incorrect nodes will return low value outputs.
Fig 5.1: Multilayer Perceptron Neural Network overlay [9]
Chapter 6 - Future development

6.1 - VoIP technology and Speaker Recognition

With telecommunications inevitably moving towards an IP address based system, speaker recognition and verification over the internet will soon become a bigger factor in day to day life. VoIP technology was developed to communicate via voice using the IP protocol in place of the existing traditional telephony network. To implement this speaker recognition system in a VoIP network, a fast, low ping internet connection would be required to transfer the audio clearly with minimal packet loss, bit error etc. Packet loss can occur of many reasons. Oversaturated network links, corrupted packets rejected in-transit, faulty networking hardware are some of the causes of packet loss. Problems such as oversaturated network links make testing of a speaker recognition system over VoIP difficult as it is almost impossible to predict when a network will be at its threshold of activity. If the system is tested while the network is lightly loaded the system may work, but under heavier traffic situations, crucial data may be lost, thus giving false readings or no readings at all.

Bit error is the number of bits of the data sent over the network that have been altered by noise (interference). Noise sources can provide random or patterned interference. Patterned interference can be cancelled out effectively.

VoIP technology uses UDP internet protocol instead of the usual TCP internet protocol for the following reasons:
- TCP is too slow for VoIP transmission. TCP requires a handshake and a session between sender and receiver. If a packet is lost or delayed the system will wait a period time for that package to arrive. In a real time implementation of a speaker Recognition system, delay is not acceptable and will reflect poorly on the design of the system.
- UDPs’ task is to deliver data as quickly as possible, and has no control over the order in which the packets arrive at the destination. It is the receivers’ job to sort the order in which the packets rearrange. If packets take too long to transmit, or
get lost in the transmission the receiver will ignore them and wait for the next packet to arrive. In voice transmission it is better to lose a few packets, rather then to wait/hang for those packets to be received.

Companies such as blueface®[10] offer VOIP through our home phones, utilizing the pre-existing telephone system, making implementation of a speaker recognition/verification system simpler. The flow diagram (Fig 6.1) below demonstrates the steps to call to a Phone banking system with speaker recognition via VoIP. As shown, any speaker recognition system would be implemented external to the home users telephone system.

![Diagram](image)

**Fig 6.1: Flow diagram of VoIP call to a Phone banking system**
6.2 – Real time implementation

This includes selection of a suitable development platform based on performance requirements, ease of development etc. Translating the program from MATLAB code to C code would provide a more universal language and hence more flexibility in the speaker recognition software.

Finally the development of a real-time version of the system, and testing/evaluation of the system in a laboratory environment shall be carried out.

6.3 – PC GUI

A program with a graphical user interface will be coded. Users will have the options to train or run the program. Training will require a login from a previously trained user with a sufficient security profile. The training protocol will then be run.

When testing is selected the user will be prompted to speak a given passphrase, and based on the analysis the user will either pass through to the secured area, or be blocked and have to try again.
Chapter 7 – Discussion

Over the course of this project many aspects have been successful, some have caused problems which were overcome, and some were not. Different avenues were looked at during the design process, but a final solution was decided upon. Also, efficient coding and removal of wasteful data is crucial to future development of a real time application.

Some of these aspects include overcoming obstacles such as building a database of characteristics for all 80 utterances. Originally this would require altering the program code for each utterance file. As mentioned in chapter 4 this was overcome by compiling a text file containing the paths of all utterances required and looping through each line of that file, similarly a text file for the output path to store the speaker utterance characteristics was created.

Other challenges such as setting the MFCC function up properly to analyse the data correctly simply took time and research to program correctly.

The various avenues looked at during development include the choice to use MFCCs to approximate analysis similar to the human auditory system, instead of development using the Hidden Markov Model Toolkit. Various approaches to neural networks exist, though given that and MLP based neural network was the most common and applicable forum, it was choose against others.

For better results in implementation and testing, some of the design in coding could be altered/tweaked to produce a faster runtime, or more efficient code. This was taken into account when removing excess, wasteful coefficients from the characteristics of the utterance. When programming the neural network, some parameters could be altered to achieve better results, ie. Hidden nodes.
Chapter 8 – Conclusion & Results

To conclude, in this project a speaker recognition system was designed and developed. Many aspects of the project were successful and an interesting challenge to rise to. This technology looks certain to grow and develop into a seamless means of human-machine communications.

Speaker recognition makes it possible for a speakers’ voice to control access to restricted services. It is being implemented into many call centre, phone banking and other security systems.

The key elements of development in this speaker recognition system are its Mel Frequency Cepstum Coefficients; used as a resemblance to how the human auditory system works, and the MLP artificial neural network used to train the system to select the most speaker whom is speaking.

Though due to time constraints the project did not enter extensive testing, all development outcomes were as expected, and given adequate time, the system would develop towards its functioning goal.
Appendix I - TABLE OF REFERENCES


[4] Dr. Jacek M. Kowalski – Department of Physics, UNT -
http://myuntcourses.com/default.aspx


[9 The Multilayer Perceptron Classifier –
http://www.europa.eu.int/en/comm/eurostat/research/supcom.95/16/result/node7.html


Appendix I - CODE

I - Program Code

%SpeakerListLoop.m

%Constants

Windowing = 128;
TotalFrames = 0;
Count = 0;
NoOfMFCC = 8;

Train = zeros(280, 80);
%Get average number of frames.
ListLoop = fopen('SpeakerList.txt'); %Opens the list of utterance files to be read
while 1
    tline = fgetl(ListLoop); %Reads each line
    %Loop through each line until EOF.
    if ~ischar(tline), break, end
        % disp(tline);
        %Remove Silence from the signal
        NoSilence = resample_ti20(tline, 16, 25, 0.05, 128, 64);

        %Determine the average number of frames used.
        [m,n] = size(NoSilence);
        NoOfFrames = floor(m/Windowing);

        TotalFrames = TotalFrames + NoOfFrames;
        Count = Count +1;
        Avg = floor(TotalFrames/Count);
    end
end
fclose(ListLoop);

Count = 0;
%Run each file through Code2.m
ListLoop = fopen('SpeakerList.txt');
SaveLoop = fopen('Output.txt', 'rt');
while 1
    tline = fgetl(ListLoop);
    %Loop through each line until EOF.
    if ~ischar(tline), break, end
        % disp(tline);
        %Remove Silence from the signal
        NoSilence = resample_ti20(tline, 16, 25, 0.05, 128, 64);
        [m,n] = size(NoSilence);
        NoOfFrames = floor(m/Windowing);

        %Open Code2.m
Code2

% while 1

tline2 = fgetl(SaveLoop);
%Loop through each line until EOF.
if ~ischar(tline2), break, end
% disp(tline2);

outfile = fopen(tline2, 'wb');
InvMFCC = AvgMFCC'; %Invert the matrix
CutMFCC = InvMFCC(1:NoOfMFCC,:); % Remove unused coefficients
ColMFCC = reshape(CutMFCC, (NoOfMFCC*Avg),1); % Matrix is reshaped from 8*35 to 1 column of 280
Count = Count +1;
Train(:, Count) = ColMFCC;
%Write to file
C = fwrite(outfile,CutMFCC', 'double');
fclose(outfile);
end
%Code2.m

%Constants
i = 1;
j = 1;
fsamp = 8000;
framelength = 256;
NoOfChannels = 19;

%Frames per utterance (for use with interp1 function)
IndivLength = 0:1/(floor((m/Windowing)-1)):1;
AvgLength = 0:1/(Avg-1):1;

[m,n] = size(NoSilence);
%timeLength = m/8000; %About 1/2 a second

% nFrames = floor(timeLength/32e-3); %15 Frames of 32e-3 seconds
% N = timeLength * fsamp;

%Defining a Matrix for
MFCC = zeros(floor(m/Windowing), NoOfChannels);

%AllMFCC = zeros(TestRows, Count);
%Overlapping
nFrames = floor((m-Windowing)/(framelength-Windowing));

for frame = 1 : nFrames,

%Overlapping - NoSilence is the input file without silence.
aquiredData = NoSilence(i:i+framelength-1); %256 frames are analysed.
i = i + Windowing; %Increment by 128 for a 50% overlap

% Hamming windowing
windowing = (hamming(framelength).*aquiredData);

% FFT
FFTofData = fft(windowing);
FFTofData = FFTofData(1:end/2); % First 128 frames of the sample

W=melFilterMatrix2(fsamp,framelength,NoOfChannels); %Return filter Matrix from MFCC function
FilteredData = W*(abs(FFTofData)); % Analyse the data against the MFCC

singleMFCC = DCT(FilteredData);

MFCC(frame, :) = DCT(FilteredData); % Load each frame into a Matrix as the loop progresses

%Compressing to a common frame length
AvgMFCC = interp1(IndivLength, MFCC, AvgLength);
end
2 - MFCC Code

```matlab
function W = melFilterMatrix(fs, N, nofChannels)

% for test, use these parameters
% parameters
fs = 8000;
N = 256;
nofChannels = 19;

% compute resolution etc.
df = fs/N; % frequency resolution
Nmax = N/2; % Nyquist frequency index
fmax = fs/2; % Nyquist frequency
melmax = freq2mel(fmax); % maximum mel frequency

% mel frequency increment generating 'nofChannels' filters
melinc = melmax / (nofChannels + 1);

% vector of center frequencies on mel scale
melcenters = (1:nofChannels) .* melinc;

% vector of center frequencies [Hz]
fcenters = mel2freq(melcenters);

% compute bandwidths
% startfreq = [0 , fcenters(1:(nofChannels-1))];
% endfreq = [fcenters(2:nofChannels) , fmax];
% bandwidth = endfreq - startfreq;

% quantize into FFT indices
indexcenter = round(fcenters ./df);

% compute resulting frequencies
% fftfreq = indexcenter.*df;

% compute resulting error
% diff = fcenters - fftfreq;

% compute startfrequency, stopfrequency and bandwidth in indices
indexstart = [1 , indexcenter(1:nofChannels-1)];
indexstop = [indexcenter(2:nofChannels), Nmax];
% idxbw = (indexstop - indexstart) + 1;
% FFT bandwidth = idxbw.*df;

% compute matrix of triangle-shaped filter coefficients
W = zeros(nofChannels, Nmax);
for c = 1:nofChannels
    % left ramp
    increment = 1.0/(indexcenter(c) - indexstart(c));
    for i = indexstart(c):indexcenter(c)
        W(c,i) = (i - indexstart(c))*increment;
    end
end
```

25 Mar. 09 Matthew Byrne 34
%right ramp
decrement = 1.0/(indexstop(c) - indexcenter(c));
for i = indexcenter(c):indexstop(c)
    W(c,i) = 1.0 - ((i - indexcenter(c))*decrement);
end %i
end %c

%normalize melfilter matrix
for j = 1:nofChannels
    W(j,:) = W(j,:)/ sum(W(j,:)) ;
end