VOICE RECOGNITION FOR SECURITY PURPOSES

By

NOOR ROHAIZAD NOOR ROZALI

FINAL PROJECT REPORT

Submitted to the Electrical & Electronics Engineering Programme

in Partial Fulfillment of the Requirement for the

Bachelor of Engineering (Hons)

(Electrical & Electronics Engineering)

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

Voice Recognition System for Security Purposes

by

Noor Rohaizad Noor Rozali

A project dissertation submitted to the
Electrical & Electronics Engineering Programme
Universiti Teknologi PETRONAS
in partial fulfilment of the requirement for the
BACHELOR OF ENGINEERING (Hons)
(ELECTRICAL & ELECTRONICS ENGINEERING)

Approved by,

[illegible]

(Dr. Yap Vooi Voon)

UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

JUNE 2007

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

[illegible]

NOOR ROHAIZAD NOOR ROZALI

ABSTRACT

This report presents the implementation of Voice Recognition System for Security Purposes. The scope of study is to understand the existing algorithms and develop a brand new algorithm which is better in terms of its reliability and accuracy. It will be implemented in Matlab environment. The main objective of this project is to develop a new voice recognition system that will have minimum error in identifying the speaker. The system was built by combining a speech recognition system with a speaker recognition system. For speech recognizer, the Dynamic Time Warping is used and for speaker recognizer, the Mel-Frequency Cepstrum Coefficients is used. The results show that the combination of both algorithms has produced a robust Voice Security System.

ACKNOWLEDGEMENTS

First and foremost, the deepest gratitude to The Almighty for His blessing and mercy, I managed to complete this Final Year Project as a requirement of completing a Bachelors of Engineering (Hons) in Electrical & Electronics Engineering.

I would like to give a special thanks to my supervisor, Dr Yap Vooi Voon for all his supports and guidance for the one year which this project had been conducted. Also to all examiners who had thrown to me constructive comments to improve the quality of this project in various directions.

Not forgotten, to all my colleagues and technicians of the department for their helps and encouragements. To my family as they have given me all the necessary support from the beginning.

Thanks to all parties who participated directly and indirectly in completing this project. All your contributions are highly appreciated.

Noor Rohaizad Noor Rozali

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

DTW	Dynamic Time Warping
MFCC	Mel-Frequency Cepstrum Coefficients
VQ	Vector Quantization
NN	Neural Network
HMM	Hidden Markov Model
STFT	Short Time Fourier Transform

CHAPTER 1
INTRODUCTION

1.1 Background of Study

Currently, security, in public places for example, is widely discussed by government security agencies and NGOs. Everybody concerns about it no matter what it is applied to, from the security of a mobile phone up to the security of a bank. A great deal of efforts has been put in to improve security in public places in light of crime on the streets.

A bank manager will try to enhance the security of the bank from robbery by installing alarm system and hiring security company to guard the bank for 24 hours, whereas a school kid will lock his bike to ensure that it can't be taken by someone else. Companies who provide the security also have different ways of enhancing their products, from the materials used to make a lock, up to the technologies used to build an alarm system.

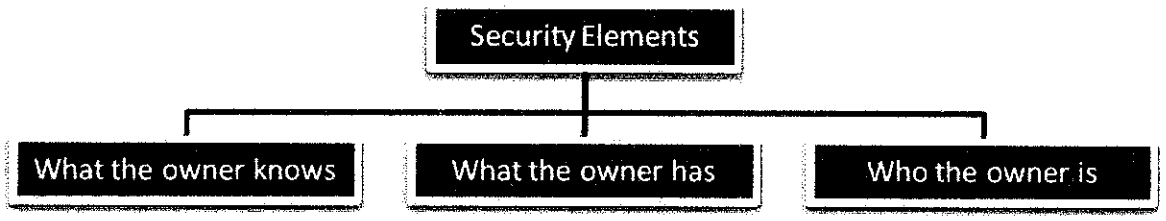


Figure 1: Elements for the Highest Level of Security

Figure 2 shows the elements for the highest level of security. Using voice as the key of a particular security system can provide it with two out of those three elements; what the owner knows like password, and who the owner is like fingerprint.

The uniqueness of one's voice is similar to the uniqueness of one's fingerprint or DNA, where there will be no two people has fingerprints, DNAs or voices which are identical. Therefore, voice can be used as the main key of a security system and it is better than the fingerprint or the DNA as it can provide one more element which is what the owner knows like a password or the answers to a set of private questions.

1.2 Problem Statement

The level of reliability of many security systems nowadays are getting lower as the level of people's knowledge is getting higher. Many solutions have been proposed and used by the irresponsible parties to unlock such a system from the hardware wise up to the software wise. Hence, **better security systems are in heavy demand.** One solution is a robust voice security system.

1.3 Objective

The main objective of this project is to **develop a robust recognition system** which can be used for security purposes. It will be able to identify the owner correctly; by verifying both speech and speaker. It can be achieved by **modifying and improving the existing algorithms**.

CHAPTER 2
LITERATURE REVIEW AND THEORY

2.1 Speech Recognition versus Speaker Recognition

As discussed in the previous chapter, a voice recognition system can provide two important elements to a security system which are the uniqueness and the password. Therefore, the new algorithm should be able to cater for these elements. In order to implement such a system, the combination of both speech and speaker recognition is introduced in this project.

Speech recognition is a process of recognizing the speeches, words or texts spoken by someone correctly regardless of who he is, where, in this case, it will be able to verify whether the password spoken matches the preset password or not.

On the other hand, speaker recognition process is a process of recognizing the owner of the voice or who is saying the password regardless of what he is saying. The password might be right but the system will deny the access if the speaker is not the real owner. Combining these two processes will result in a very reliable security system.

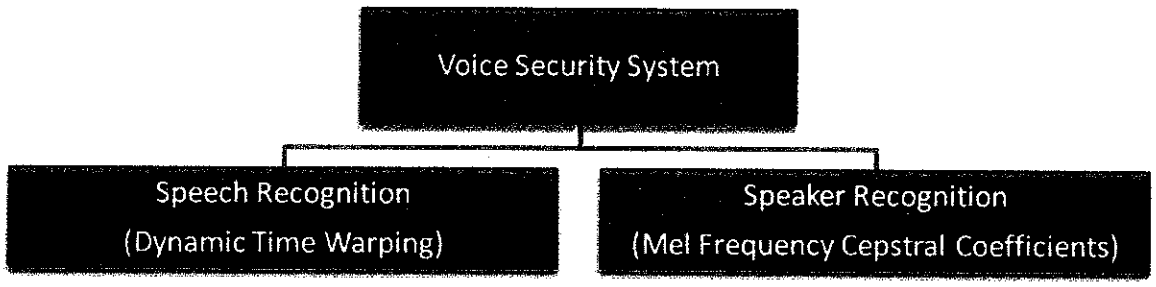


Figure 2: The New Voice Security System

As shown in Figure 3, for the speech recognition, Dynamic Time Warping (DTW) will be used, whereas for the speaker recognition, Mel Frequency Cepstral Coefficients (MFCC) will be used.

2.2 Dynamic Time Warping (DTW)

Dynamic time warping was introduced by Sakoe and Chiba in 1978 [2] in conjunction with dynamic programming techniques for the recognition of isolated words. The DTW algorithm removes timing differences between speech patterns by warping the time axis of one speech pattern until it maximally coincides with the other.

After feature extraction is done, speech patterns can be represented as a sequence of feature vectors:

$$A = a_1, a_2, \ldots, a_i, \ldots, a_K$$
$$B = b_1, b_2, \ldots, b_j, \ldots, b_M$$

Let A be the reference speech pattern and B be the pattern vector to be aligned against A. Figure 3 shows A and B developed against the i and j axes.

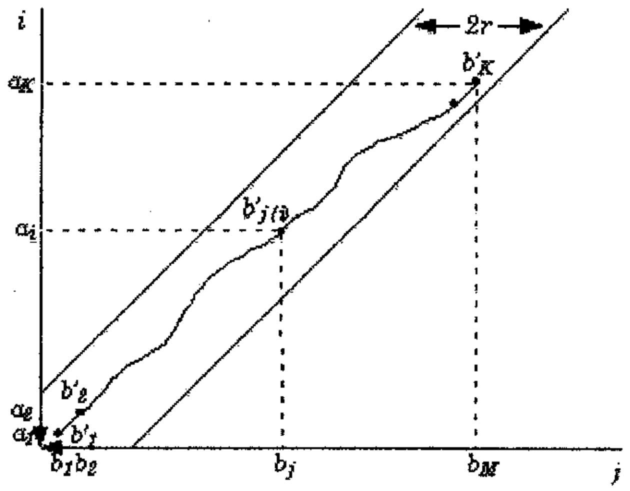


Figure 3: Warping Function and Adjustment Window

Consider a warping function F between the input pattern time j and the reference pattern time i, where:

$$j = j(i)$$

A measure of the difference between the two feature vectors a_i and b_j is the distance:

$$d(i, j) = || a_i - b_j ||$$

When the warping function is applied to B this distance becomes:

$$d(i, j(i)) = || a_i - b'_j ||$$

where b'_j is the jth element of B after the warping function has been applied.

The weighted summation of these distances on the warping function is:

$$E(F) = \sum_{i=1}^{K} d(i, j(i)) \cdot w(i)$$

where $w(i)$ is a nonnegative weighting coefficient.

E reaches a minimal value when the warping function is determined to optimally align the two pattern vectors. The minimum residual distance between A and B is the distance still remaining between them after minimizing the timing differences between them. The time normalized difference is defined as follows:

$$D(A,B) = MinF\left[\frac{E(F)}{\sum_{i=1}^{K} w(i)}\right]$$

Certain restrictions are applied to the warping function to ensure that it approximates the properties of actual time axis fluctuation. It should preserve all the significant linguistic features present in the speech pattern being warped. They are monotonicity and continuity. These can be realized by imposing the following conditions on the warping function E(F).

Monotonic conditions:

$$j(i-1) \leq j(i)$$

Continuity conditions:

$$j(i) - j(i-1) \leq 1$$

Boundary conditions are imposed as follows:

$$j(i) = 1 \text{ and } j(K) = M$$

An adjustment window is implemented such that

$$|i - j(i)| \leq r$$

where r is a positive integer. The adjustment window condition is imposed since the time axis fluctuation does not yield excessive timing differences and therefore the algorithm must do likewise.

The final constraint imposed is the slope constraint condition. The results of this condition is that if $b'_{j(i)}$ moves forward in one direction, m times consecutively, then it must step n times in the diagonal direction before it can step any further in that direction. This ensures a realistic relation between A and B by ensuring that relatively short segments of one are not mapped to relatively long segments of the other. The intensity of slope constraint is measured as follows;

$$P = n/m$$

The warping function slope is more rigidly restricted by increasing *P* but if it is too severe, then time normalization is not effective.

The denominator of the time normalized distance equation can be defined as:

$$N = \sum_{i=1}^{K} w(i)$$

Since *N* is independent of the warping function *F* it can be put out of the bracket in *E(F)* simplifying the equation as follows:

$$D(A,B) = \frac{1}{N} MinF\left[\sum_{i=1}^{K} d(i, j(i)) \cdot w(i)\right]$$

Minimization can be achieved by applying dynamic programming principles. There are two typical weighting coefficient definitions which allow this simplification: symmetric and asymmetric time warping. In symmetric time warping the summation of distances is carried out along a temporarily defined time axis $l = i + j$, whereas in asymmetric time warping the summation is carried out along the *i* axis warping *B* to be of the same size as *A*. In asymmetric time warping the weighting coefficient is defined as:

$$w(i) = j(i) - j(i - 1)$$

When the warping function attempts to step in the direction of the j axis the weighting coefficient reduces to 0, since

$$j(i)=j(i-1)$$

therefore,

$$w(i) = 0$$

and when the warping function steps in the direction of the i axis or the diagonal, then,

$$w(i) = 1$$

then

$$N=K$$

Applying dynamic programming principles to the simplified time normalization equation gives the following algorithm for calculating the minimal value of the summation:

The dynamic programming equation is:

$$g_i(i, j(i)) = \min[g_{i-1}(i-1, j(i-1)) + d(i, j(i)) \cdot w(i)]$$

The time normalized distance is:

$$D(A,B) = \frac{1}{N} g_K(i(K), j(K))$$

The initial condition is:

$$g1(1, 1) = d(1, 1) * w(1) = d(1,1)$$

The dynamic programming equation for *P = 0* is:

$$g(i,j(i)) = \min\begin{cases} g(i-1,j-1)+d(i,j(i)) \\ g(i-1,j)+d(i,j(i)) \end{cases}$$

The permissible paths through which the warping functions may move under this slope constraint are shown below.

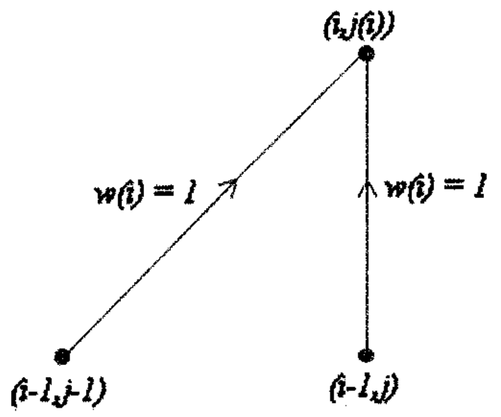


Figure 4: The Permissible Paths for P = 0

The dynamic programming equation for *P = 1* is:

$$g(i,j(i)) = \min\begin{cases} g(i-1,j-2)+(d(i,j-1)+d(i,j))/2 \\ g(i-1,j-1)+d(i,j) \end{cases}$$

The permissible paths through which the warping functions may move under this slope constraint are shown below.

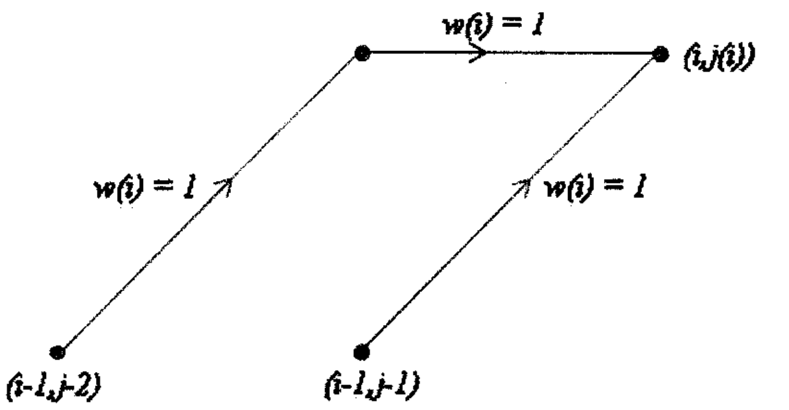


Figure 5: The Permissible Path for P = 1

The dynamic programming equation for $P = 2$ is:

$$g(i, j(i)) = \min \begin{cases} g(i-1, j-1) + d(i, j) \\ g(i-1, j-2) + (d(i, j-1) + d(i, j))/2 \\ g(i-1, j-3) + (d(i, j-2) + d(i, j-1) + d(i, j))/3 \end{cases}$$

The permissible paths through which the warping functions may move under this slope constraint are shown below

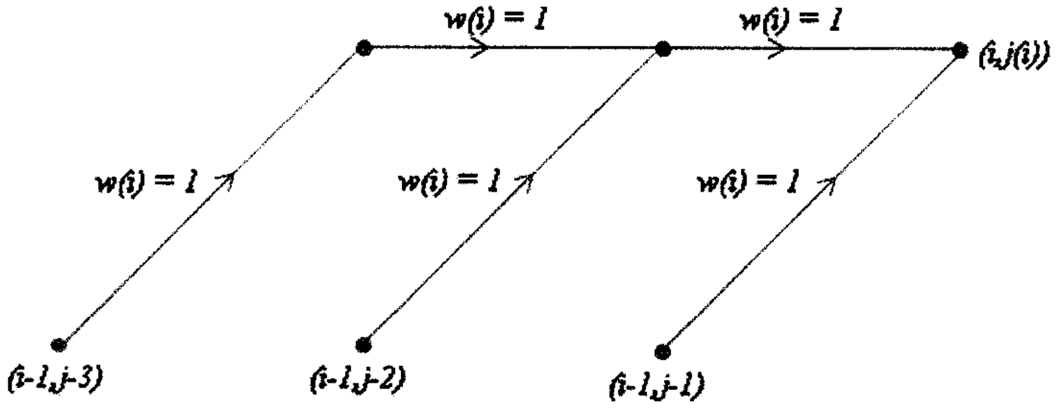


Figure 6: The Permissible Path for P = 2

The entailing result is that, for the initial condition, the first feature vector of *B* is taken as the first feature vector of the warped pattern vector b_1'. Subsequent feature vectors for the warped pattern vector are chosen such that the *nth* feature vector is that feature vector from the input pattern vector *B* closest to the *nth* feature vector of the reference pattern vector.

2.3 Mel-Frequency Cepstrum Coefficients (MFCC)[4]

The purpose of Mel-Frequency Cepstrum Coefficients process is to mimic the behavior of human ears. Figure 3 shows the steps involve in producing the MFCC of a speech signal. Typically, the speech will be recorded at a sampling rate above 10000 Hz. This is to avoid the effects of aliasing in the analog-to-digital conversion. These sampled signals can capture all frequencies up to 5 kHz, which cover most energy of sounds that are generated by humans' voice.

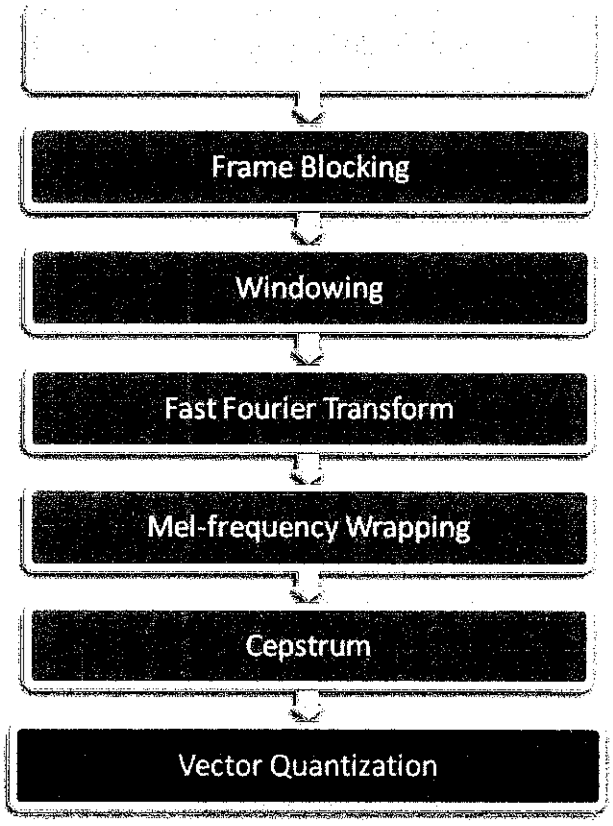


Figure 7: The Flowchart for the MFCC Algorithm [4]

Figure 4 shows the flowchart for MFCC algorithm. The continuous speech signal is firstly blocked into frames of N samples, with adjacent frames being separated by M where $M < N$. The first frame consists of the first N samples. The second frame begins M samples after the first frame, and it is overlapped by $N - M$ samples.

Similarly, the third frame begins $2M$ samples after the first frame and it is overlapped by $N - 2M$ samples. It continues until the speech is accounted for within one or more frames. Typical values for N and M are $N = 256$ which is equivalent to ~ 30 milliseconds windowing and $M \equiv 100$.

Then each individual frame is windowed to minimize the signal discontinuities at the beginning and end of each frame. It is to minimize the spectral distortion by using the window to taper the signal to zero at the beginning and end of each frame. It can be represented as:

$$y_1(n) = x_1(n)w(n)$$

where *w(n)* is typically the Hamming window function:

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

The Fast Fourier Transform will convert each frame of N samples from the time domain into the frequency domain. It is defined as:

$$X_n = \sum_{k=0}^{N-1} x_k e^{-2\pi jkn/N}$$

The result after this step is referred to as spectrum or periodogram. Human perception of the frequency contents of sounds for speech signals does not follow a linear scale. Therefore each tone is measured on a scale called the mel scale. The mel-frequency scale is a linear frequency spacing below 1kHz and a logarithmic spacing above 1kHz.

The pitch of a 1 kHz tone is defined as 1000 mels. The following approximate formula is used to calculate the mels for a given frequency:

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

The mel spectrum is converted back to time by using Discrete Cosine Transform. The result is the MFCC. The cepstral representation of the speech spectrum provides a good representation of the local spectral properties of the signal. The coefficients can be calculated by using the following formula:

$$\tilde{c} = \sum_{k-1}^{K} (\log \tilde{S}_k) \cos\left[n\left(k - \frac{1}{2}\right)\frac{\pi}{K}\right]$$

Finally, the vector quantization takes place to perform pattern recognizing step. 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 is represented by its center called a codeword. The collection of all codewords is called a codebook.

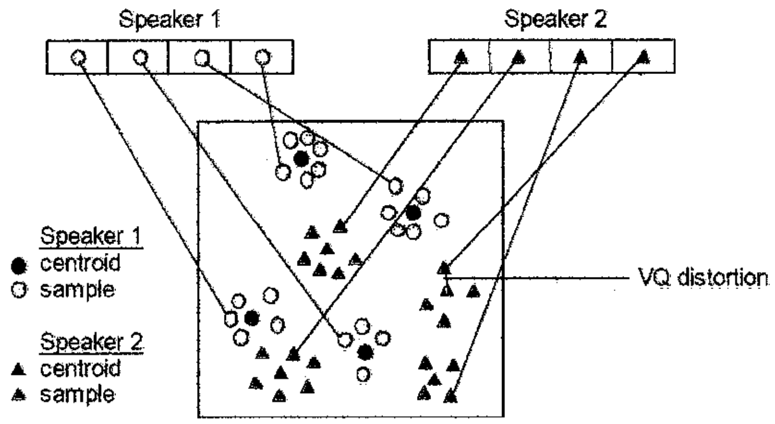


Figure 8: Conceptual Diagram of Vector Quantization

Figure 9 shows a conceptual diagram to illustrate the VQ. The circles refer to the acoustic vectors for speaker 1 while the triangles for speaker 2. In the training phase, a speaker-specific VQ codebook is generated for each known speaker by clustering the training acoustic vectors. The resultant codewords or centroids are shown in Figure 9 by black circles and black triangles for speaker 1 and 2, respectively.

The distance from a vector to the closest codeword of a codebook is called a VQ-distortion. In the recognition phase, an input utterance of an unknown voice is vector-quantized using each trained codebook and the total VQ distortion is computed. The speaker corresponding to the VQ codebook with smallest total distortion will be identified.

CHAPTER 3
METHODOLOGY

3.1 Procedure

The methodology used in this project is shown in Figure 10. The first step taken is doing literature review about voice recognition and security system. This is to get the ideas on what both topics are all about and what are the latest technologies used nowadays. Based on the literature review done, then, the best algorithms for both recognizers are selected.

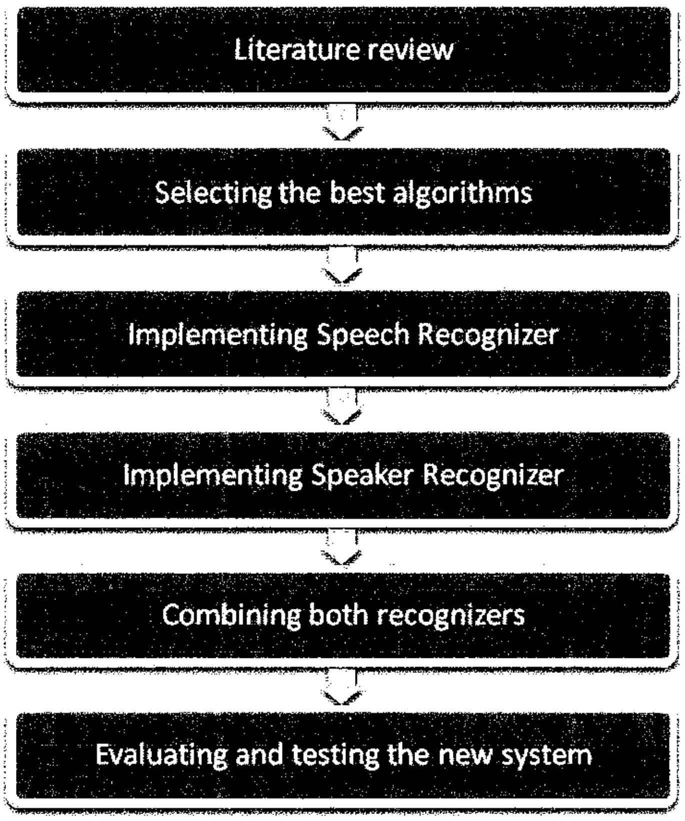

Literature review
Selecting the best algorithms
Implementing Speech Recognizer
Implementing Speaker Recognizer
Combining both recognizers
Evaluating and testing the new system

Figure 9: The Methodology

3.1.1 Selecting the Best Algorithm for Speech Recognizer

There are several established speech recognition algorithms such as Hidden Markov Model (HMM), Neural Network (NN), Dynamic Time Warping (DTW), and Vector Quantization (VQ). This section will concentrate on the selection of the algorithm that will be used by referring to the evaluation done by Rabiner [2]. It was based on average word error rate in recognizing isolated digits in a speaker-independent manner. A training set of consisting of 100 occurrences of each digit by 100 speakers was used. Table 1 summarizes the results of the evaluation.

Table 1: Average Digit Error Rates for Several Speech Recognizers [2]

Recognizer Type	Evaluation Set			
	Original Training	TS2	TS3	TS4
LPC/DTW	0.1	0.2	2.0	1.1
LPC/DTW/VQ	-	3.5	-	-
HMM/VQ	-	3.7	-	-
HMM/CD	0.0	0.2	1.3	1.8
HMM/AR	0.3	1.8	3.4	4.1

where;	LPC/DTW	Conventional template-based recognizer using DTW alignment
	LPC/DTW/VQ	Conventional recognizer with VQ of the feature vectors
	HMM/VQ	HMM recognizer with M = 64 codebook
	HMM/CD	HMM recognizer using continuous density model
	HMM/AR	HMM recognizer using autoregressive observation density
	TS2	The same 100 speakers as were used in the training
	TS3	A new set of 100 speakers (50 males, 50 females)
	TS4	Another new set of 100 speakers (50 males, 50 females)

Based on the results of the evaluation, using a VQ degrades the performance of the word recognizer. The performances of the conventional template-based recognizer using DTW alignment and the HMM recognizer using continuous density model are comparable. It is therefore, VQ can be removed from the list.

NN can be removed as well due to the source used while processing it is larger than others. It is not pragmatic because the algorithm will be implemented as an embedded system where the source and speed are limited.

The remaining algorithms are DTW and HMM. It is the best if both can be implemented as they can be compared later. In this project, however, due to the time constrain factor, DTW has been chosen as the selected algorithm. The details about this algorithm are discussed in 2.2.

3.1.2 Implementing Speech Recognizer

The next stage was implementing the selected algorithm in Matlab environment. Matlab is used as the platform because of its features where it has the capability in signal processing and it is a universal platform which can be linked or integrated with other systems.

For example, it has Link for Code Composer Studio Development Tools which allows the programmer to use MATLAB functions to communicate with Code Composer Studio® and with information stored in memory and registers on a target. This is very useful when an embedded system is desired.

Some reviews were done on the DTW implementation in Matlab via the Net and books. After trying and comparing all available sources, one's from the Lab for Recognition and Organization of Speech and Audio (LabROSA), Colombia University, was successfully built [3]. Routines in m-file for the speech recognition using DTW approach are provided. They are:

- **simmx.m**
  A utility to calculate the full local-match matrix by calculating the distance between every pair of frames from the sample and template signals.

- **dp.m**

  The implementation of the simple dynamic programming algorithm that allows three steps, (1,1), (0,1) and (1,0), with equal weights.

- **dpfast.m**

  The faster version of dp.m that uses a MEX routine (dpcore.c) to execute the non-vectorizable inner loop.

- **dpcore.c**

  The C source for the MEX routine that speeds up dpfast.m.

The implementation was done by referring on the example given which explains how to integrate all these routines. Below are the basic steps of the DTW algorithm as shown in the example:

1. Load two speech waveforms of the same utterance

```
>> [d1,sr] = wavread('test1.wav');
>> [d2,sr] = wavread('test2.wav');
```

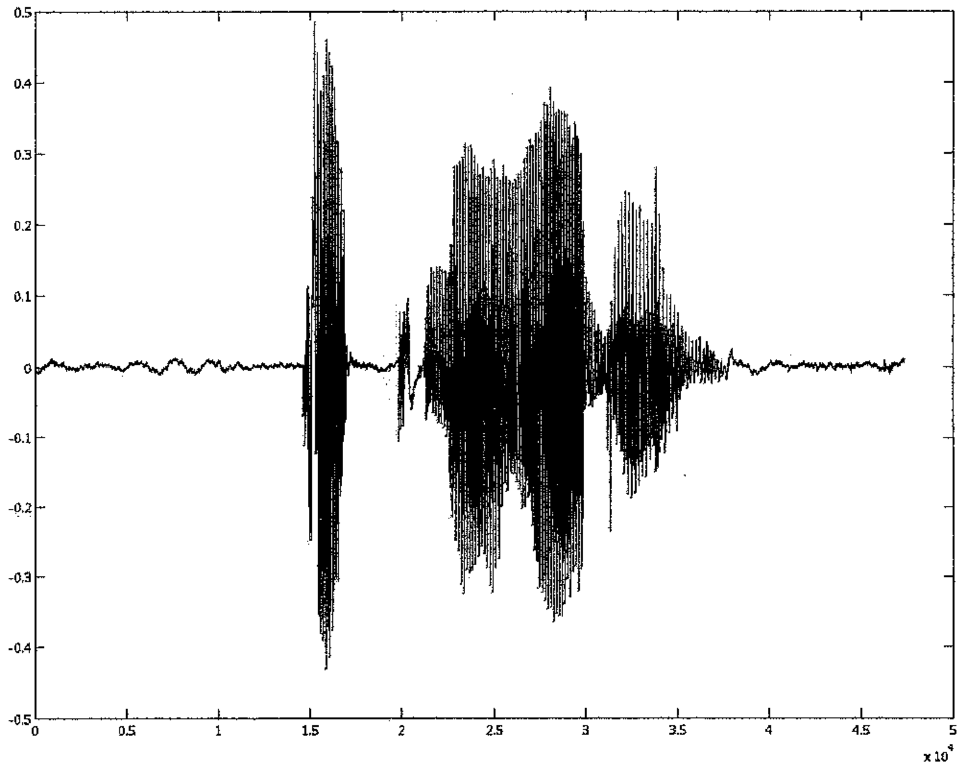


Figure 10: Speech Waveform for *test1.wav*

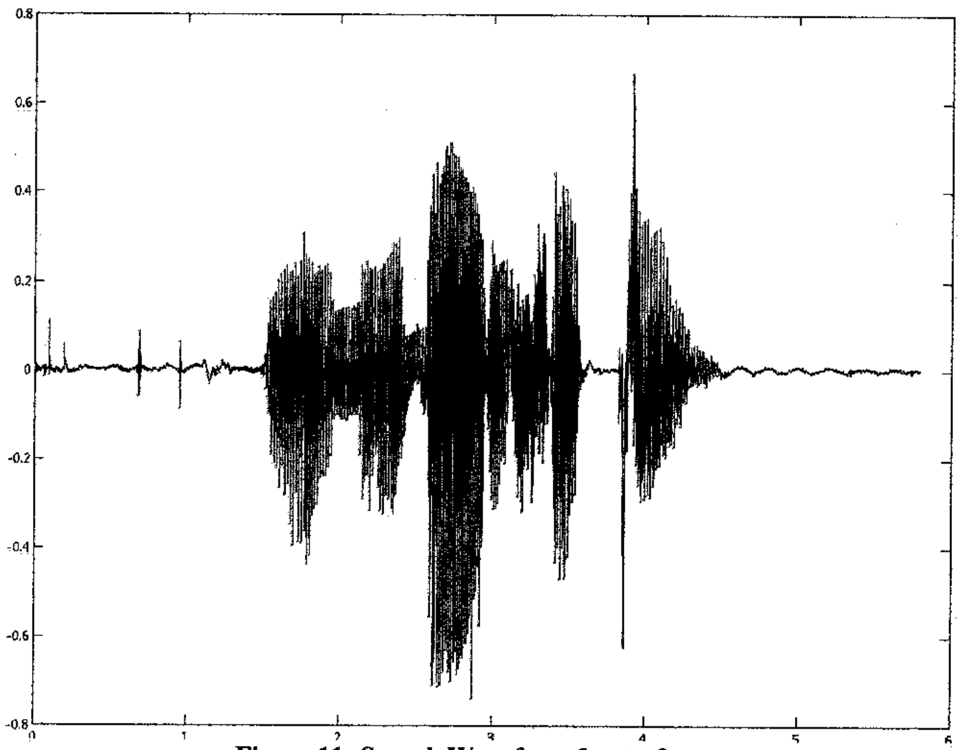


Figure 11: Speech Waveform for *test2.wav*

2. Calculate STFT features for both sounds (25% window overlap)

```
>> D1 = specgram(d1,512,sr,512,384);
>> D2 = specgram(d2,512,sr,512,384);
```

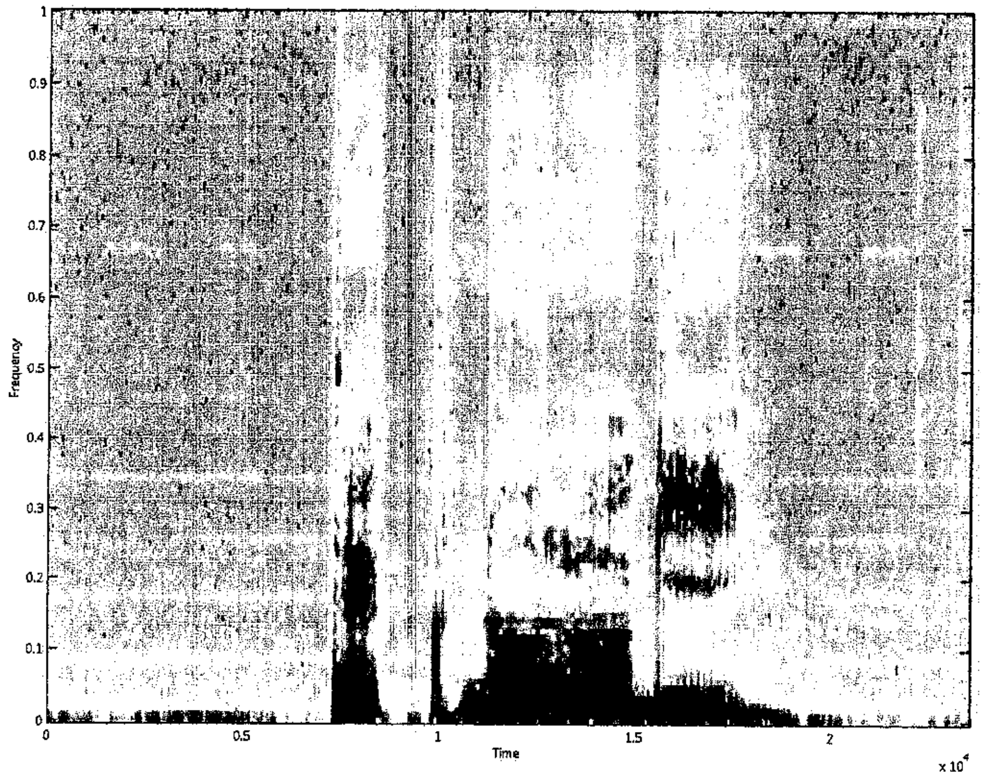


Figure 12: Spectogram for test1.wav

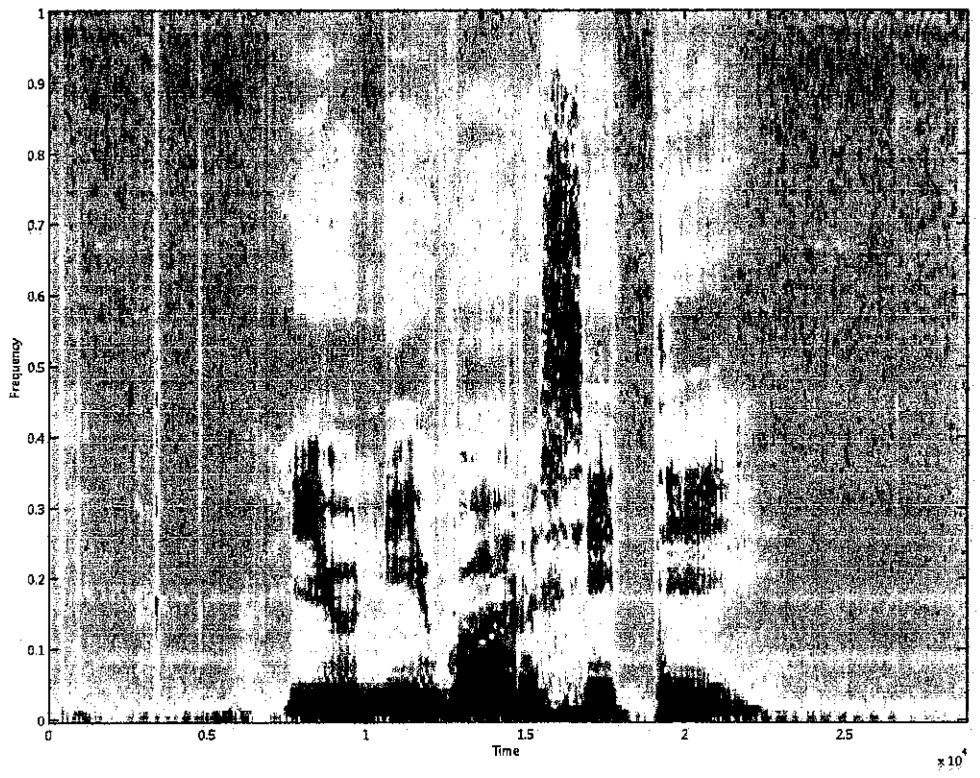


Figure 13: Spectogram for *test2.wav*

3. Construct the local match scores matrix between the STFT magnitudes

```
>> SM = simmx(abs(D1),abs(D2));
```

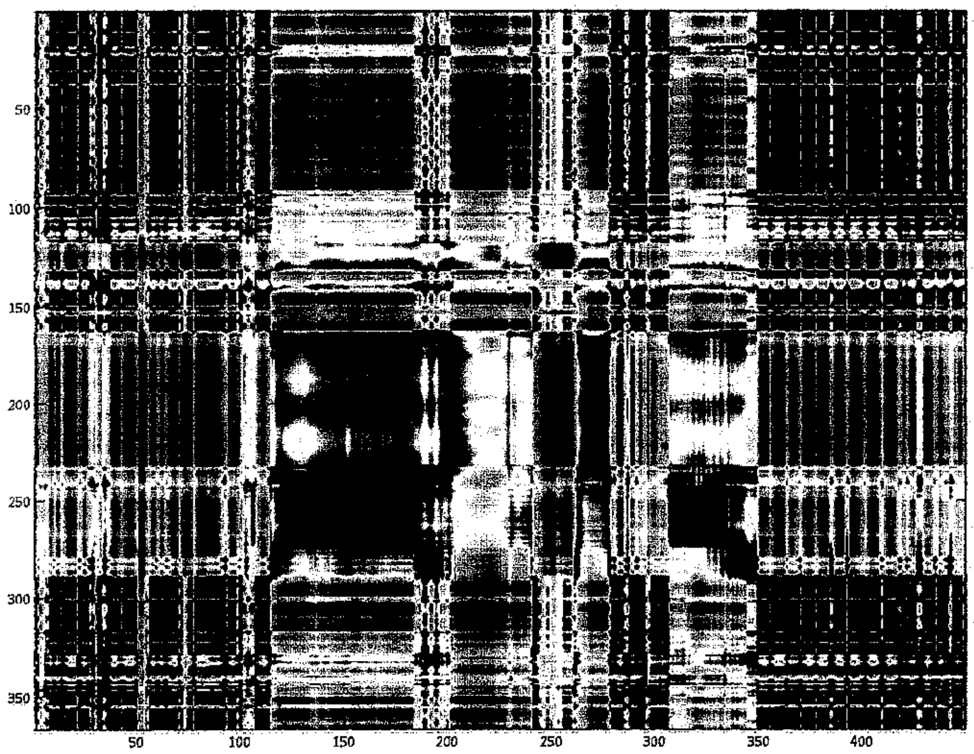


Figure 14: Local Match Score Matrix

4. Use dynamic programming to find the lowest-cost path

```
>> [p,q,C] = dp(1-SM);
```

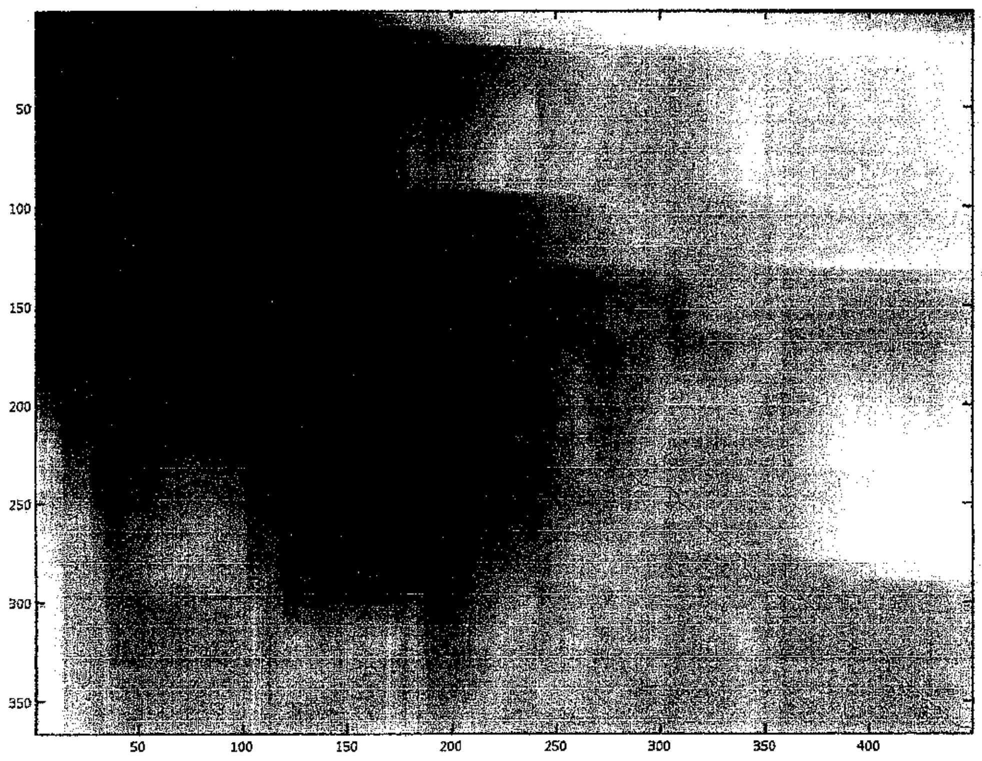


Figure 15: The Lowest-Cost Path

5. Calculate the cost of minimum-cost alignment of the two

```
>> C(size(C,1),size(C,2))
```

6. Compare the C values for each template to find the best match template.

After the basic operation was understood, a graphical user interface was created. Most of the programming parts were done by referring to the Matlab Help Documentations. The resultant system will be discussed in the next chapter.

3.1.3 Selecting the Best Algorithm for Speaker Recognizer

Speech recognition algorithms have been studied for decades. There are several kinds of parametric representations for the acoustic signals. Among them the Mel-Frequency Cepstrum Coefficients (MFCC) is the most widely used [6-8]. There are many reported works on MFCC, especially on the improvement of the recognition accuracy [9-11].

Based on those facts, the MFCC algorithm has been selected as the one that will be used in this project. It will be integrated with the VQ approach to do the feature matching task. This is because experiments done by [12] prove that the accuracy of 98% was achieved for speaker recognition using VQ.

3.1.4 Implementing Speaker Recognizer

For speaker recognizer using MFCC, a similar approach was used but the routines are shared in the Matlab Central website. They are combined in such a way that a speaker recognition system will work as discussed in 2.3. The routines used are:

- **MFCC_BasedCodebook.m**

  This function takes the speech sample and the sampling frequency as input and generates the codebook.
- **mfcc.m**

  This function computes the MFCC output of the sampled input signal.
- **melfilterbank.m**

  This computes the mel filter banks.
- **euclid_dist.m**

  This calculates the Euclidean distance between two column vectors.
- **vector_quant.m**

  This function performs the VQ to find the minimum distance.

3.1.5 Combining Both Recognizers

After both recognizers were found working, they were then combined so that they can work in a single system. This is the hardest part among others because they have to be the same in programming wise. Many debugging problems were faced and again with the help of Matlab Help Documentation, they were overcome.

3.1.6 Evaluating and Testing the System

Before this, the system was tested separately. As it is envisaged that it must be able to perform both recognitions, it was evaluated. The objective of evaluating the system was to find the optimum values for the important coefficients such as the threshold values.

Finally, it was tested in various conditions to prove that it can provide a more reliable security system. The results of the evaluations and tests will be discussed in the next chapter.

3.2 Software and Tools

The software and tools used in this project are:

1. Matlab
2. Computer
3. Microphone
4. Speaker

CHAPTER 4
RESULTS AND DISCUSSION

This system was implemented based on the subroutine functions available at [3]. The Matlab code is provided in Appendix A. It first calculates the Short Time Fourier Transform (STFT) features for the trained and test sounds.

```
Train     = specgram(a,512,sr,512,384)
Test      = specgram(b,512,sr,512,384)
```

It then constructs the 'local match' scores matrix as the cosine distance between the STFT magnitudes using the following function:

```
SM = simmx(abs(train),abs(test))
```

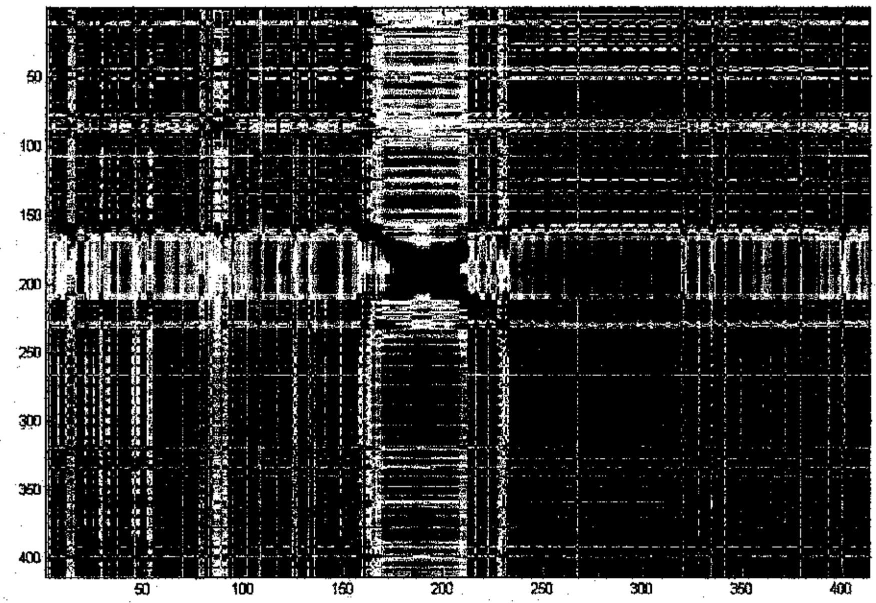


Figure 16: The Local Match Scores Matrix

A dark stripe can be seen down the leading diagonal which represents high similarity values. Dynamic programming is used to find the lowest-cost path between the opposite corners of the cost matrix by using the following function.

[p,q,C] = dp(1-SM)

The value of C will be given by the following syntax and it will be compared between different templates. The template which gives the minimum value will be selected as the most likelihood template for the testing speech.

C(size(C,1),size(C,2))

The value of C can also be limited between a certain ranges to improve the accuracy of the system. However, evaluations and tests need to be done so that it won't affect the reliability of the system.

The following evaluation was done to verify the above statement. The value of C is analyzed for different words as well as the same words.

Table 2: The results for the evaluation to verify the value of C

Training	Testing				
	"one"	"two"	"three"	"four"	"five"
"one"	7.77×10^{-15}	53.33	59.19	62.58	61.18
"two"	53.33	5.99×10^{-15}	51.23	66.93	67.96
"three"	59.19	51.23	6.11×10^{-15}	76.31	75.62
"four"	62.58	66.93	76.31	2.99×10^{-15}	72.38
"five"	61.18	67.96	75.62	72.38	8.88×10^{-15}

Based on the results summarized in the table, it can be seen that the value of C for the same words spoken will be smaller than for different words. The word that has the smallest value will be selected.

The following figures show the lowest-cost paths for template 'one':

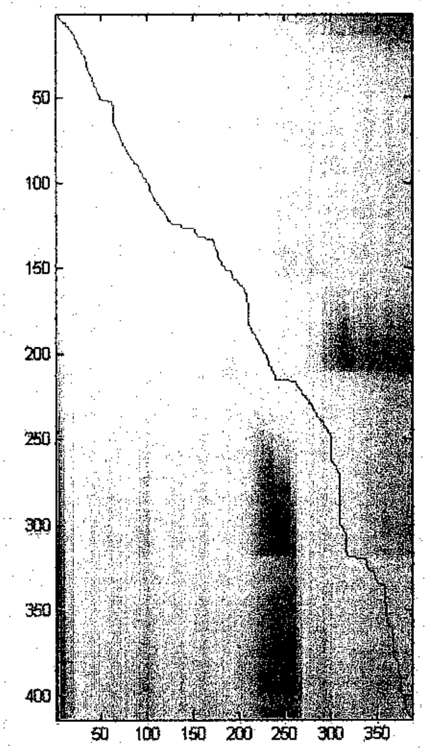


Figure 17: Test with 'two'

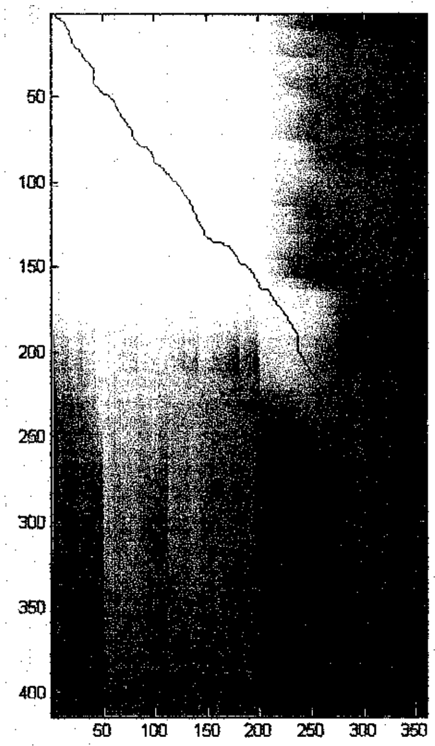


Figure 18: Test with 'three'

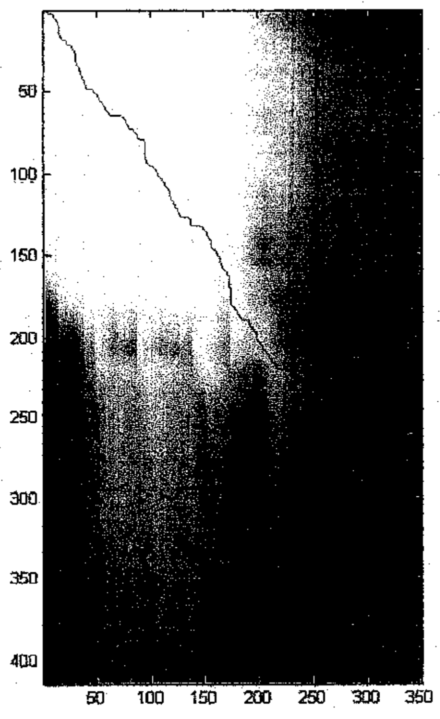


Figure 19: Test with 'four'

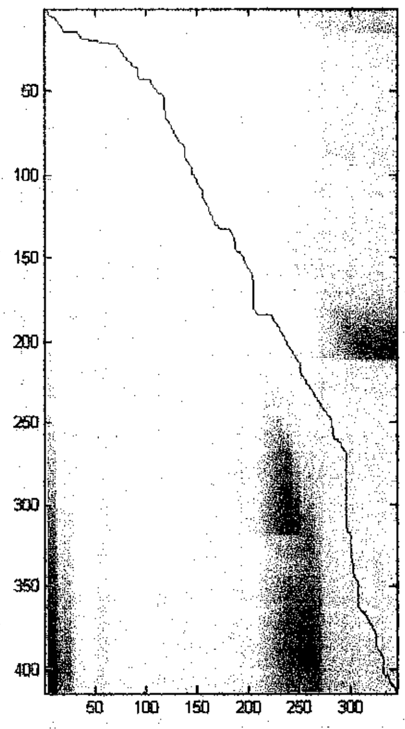


Figure 20: Test with 'five'

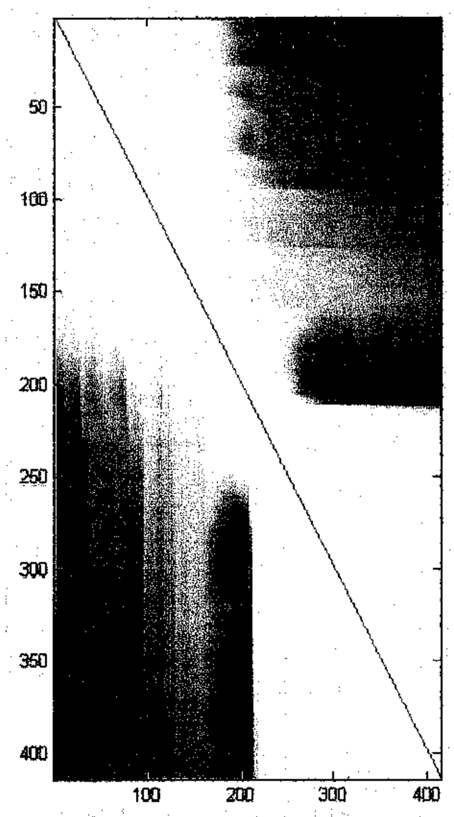


Figure 21: Test with 'one'

Based on the figures, it can be seen that the same word (Figure 30) creates the minimum lowest-cost path among others. This illustrates how the DTW works to perform the speech recognition task.

The speaker recognizer which applies MFCC algorithm was added to or combined with the speech recognizer in one system. The interface is the same. Instead of running only the speech recognizer, it now can perform speaker recognition process as well. It is implemented based on [5] as discussed in 2.3. The resultant codebook from the system is shown below:

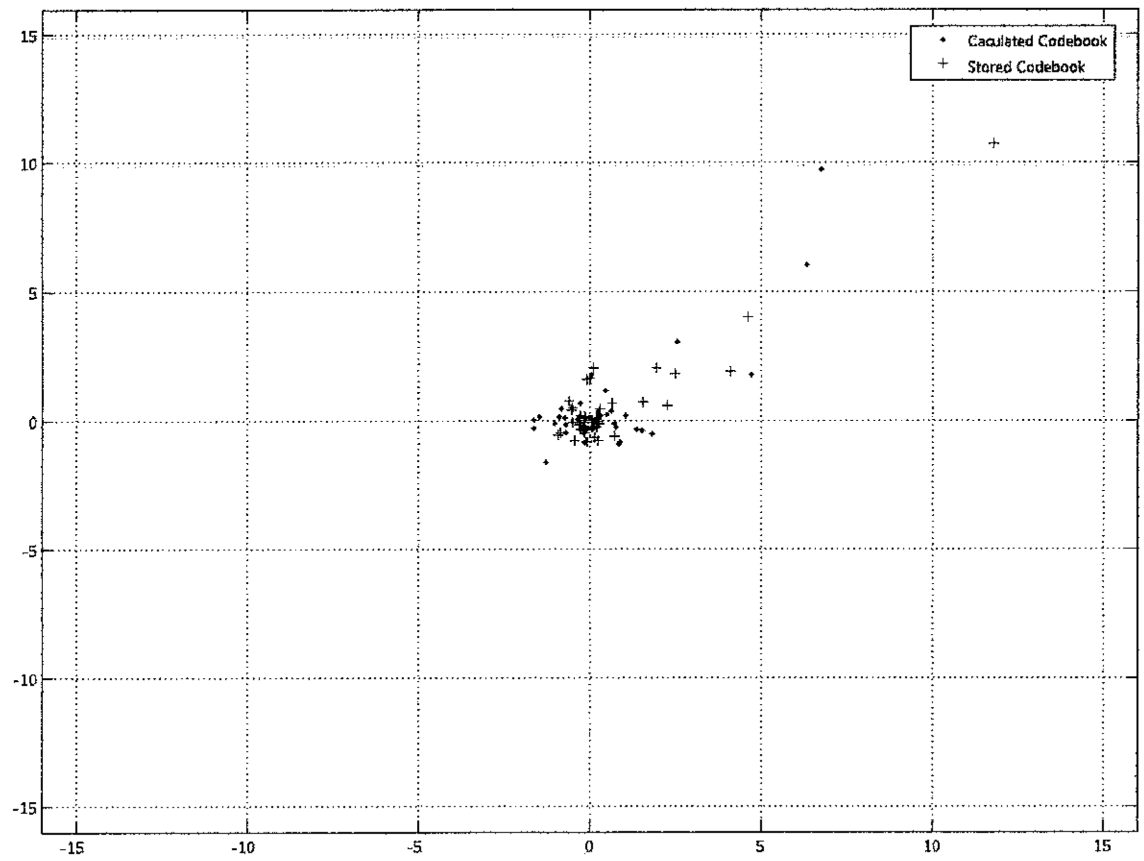
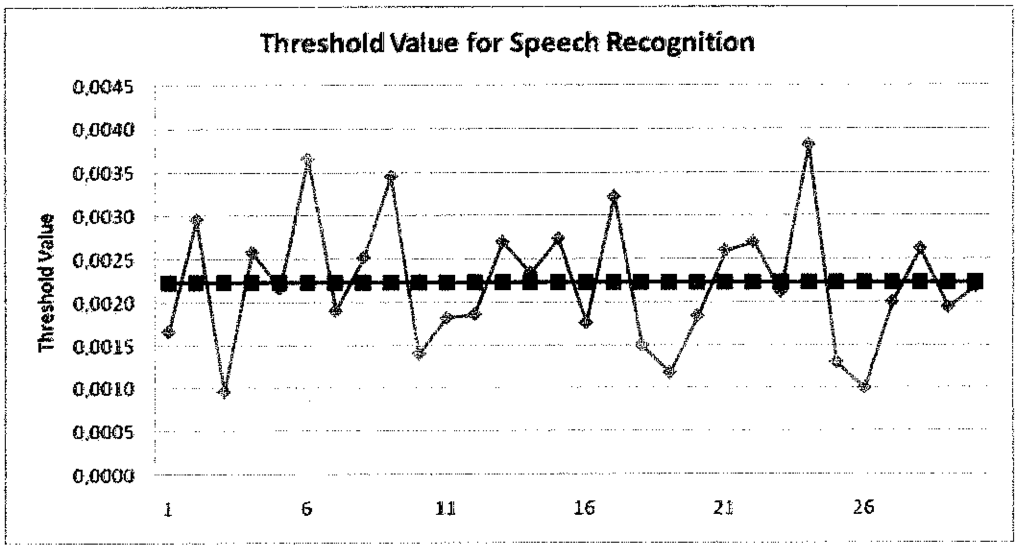


Figure 22: The Codebook

This speaker recognition was created by combining the subroutines shared in Matlab File Exchange website. It is combined in such a way that it can operate with the speech recognition to create a better voice recognition system.

The new system was evaluated and tested to improve its accuracy. The key of this system is the setting for the threshold values. Each recognizer has its own threshold value which must be set with an appropriate value. If the value is too big, anybody can access the system, on the other hand, if it is too small, even the owner can't.

The values were determined by experiment. A sample of 32 students' voice was used to do this experiment. The results obtained are shown below:



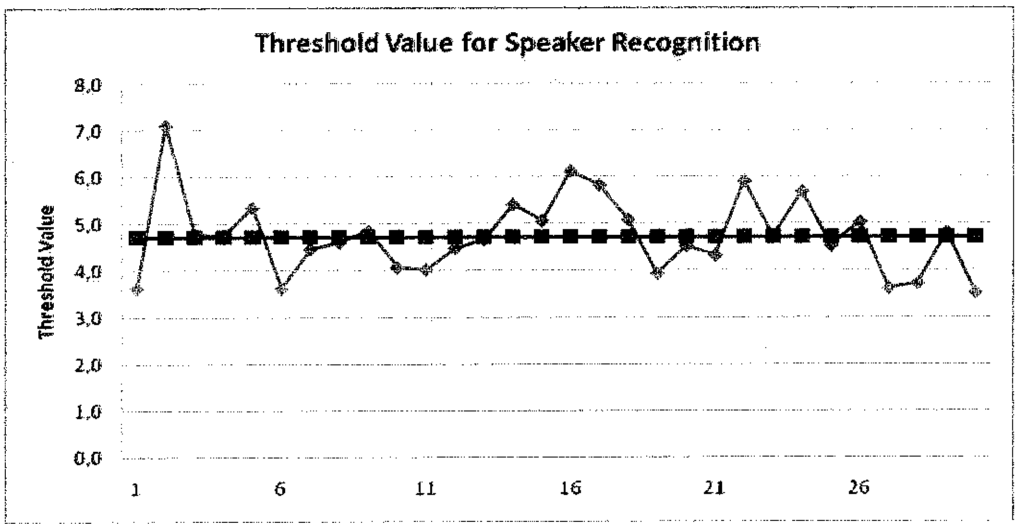


Figure 23: The Results for the Threshold Values Evaluation

Each student tested (word: "petronas") the system for 5 times and the average value was recorded. Based on the results, the threshold value for speech recognizer is 0.0022 and for the speaker recognizer it is 4.7. The system was then tested by using these values to evaluate the performance of the system in terms of its accuracy.

Based on the results (refer Appendix B) of 32 samples experiment, for randomly selected speaker, either the word spoken was true or not, it is 100% accurate. For the same speaker and the same word, the accuracy is only 84.38%. This means 15.62% possibilities that the owner can't access the system.

Many factors can affect the performance of the system. One of them is the microphone. Different microphone will result in different values of the threshold values. In this project, the microphone is designed and manufactured by Altec Lensing. The different is due to different technology and components used. Another one is the surrounding factor. The system was tested in a quite room. The accuracy can drop to about 40% if the room contains noise. The solution to overcome these problems is discussed in 5.2.

CHAPTER 5
CONCLUSION AND RECOMMENDATION

5.1 Conclusion

It can be concluded that the objective mentioned in 1.3 is met. The new voice security system was successfully implemented in Matlab environment. It is done by combining speech recognizer and speaker recognizer in a system. The accuracy level based on the experiment is 84.38% and it is depending on some external factors such as microphone used and surrounding noise.

5.2 Recommendation

The main part of the voice security system was implemented. It is recommended to increase the accuracy level of the system. As stated in 4.3, the only problem is noise. Therefore, a noise cancellation process should be placed before the front-end process. It can be done using hardware or/and software approach. This will produce a robust voice security system.

It is also recommended that the system is implemented as an embedded system. It can be done as the algorithm is deeply understood. The only thing is to understand and think of how it can work as an embedded system and what are the available platforms in the market. It may take time but the performance will be better than the software approach as used in this project.

REFERENCES

[1] http://www.nst.com.my/Current_News/nst/Thursday/National/20061026094454/Article/index_html, Date View: 30 October 2006

[2] Sakoe, H., and Chiba, S., "Dynamic Programming Algorithm Optimization For Spoken Word Recognition", *IEEE trans. on Acoustics, Speech and Signal Processing* vol. 26, no. 1, pp. 43-49, February 1978.

[3] http://www.ee.columbia.edu/~dpwe/resources/matlab/dtw/
Date View: September 2006

[4] Minh N. Do, "*An Automatic Speaker Recognition System*"

[5] Sirko Molau, Michael Pitz, Ralf Schliitel, Hermann Ney, "Computing MFCC on the Power Spectrum", *IEEE trans. on Acoustics, Speech and Signal Processing,* April 2001.

[6] L. Rabiner and Biing-Hwang Juang, Fundamentals of Speech and Recognition, Prentice Hall PTR, c1993

[7] Joseph W. Picone, "Signal Modeling Techniques in Speech Recognition", Proceedings of the IEEE, vol. 81, No. 9, pages 1215-1247, 1993.

[8] Steven B. Davis and Paul Mermelstein, "Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences", IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. ASSP-28, No. 4, August 1980.

[9] Qi Li, Frank K, Soong and Olivier Siohan, "A High-Performance Auditory Feature for Robust Speech Recognition", 6th International Conference on Spoken Language Processing, Beijing, October 2000.

[10] Jia Lei and Xu Bo, "Including detailed information feature in MFCC for large vocabulary continuous speech recognition", Acoustics, Speech, and Signal Processing, 2002. Proceedings. (ICASSP '02). IEEE International Conference on, Volume 1, 2002 Pages 805 - 808

[11] Phadke, S.; Limaye, R.; Verma, S.; Subramanian, K., "On design and implementation of an embedded automatic speech recognition system", VLSI Design, 2004. Proceedings. 17th International Conference on, 2004 Pages: 127- 132.

[12] F. K. Soong, A. E. Rosenberg, L. R. Rabiner, B. H. Juang, " A Vector Quantization Approach to Speaker Recognition" AT&T Bell Laboratories
Murray Hill, New Jersey 07974

APPENDIX A

vr.m

```
clc;
clear;
close all;


% Initial setup %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
samplingfrequency=22050;
samplingbits=16;
duration=2;

% Main menu %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
option=0;
possibility=5;
while option~=possibility,
    option=menu('SPEECH RECOGNIZER',...
        '                    TRAINING                  ',...
        'TESTING',...
        'DATABASE',...
        'DELETE',...
        'EXIT');


    % Teaching %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    if option==1
        if (exist('database.dat')==2)
            load('database.dat','-mat');
            teaching = audiorecorder(samplingfrequency,samplingbits,1);
            record(teaching,duration);
            while (isrecording(teaching)==1)
            end
            y = getaudiodata(teaching, 'uint8');
            name = input('Save as:','s');
            sound_number = sound_number+1;
            data{sound_number,1} = y;
            data{sound_number,2} = name;
            save('database.dat','data','sound_number','-append');

            m=1;
            n=5;
            No_pts=12948;               % No of points to be recorded
            Fs=22050;                   %sampling rate

              codebook = MFCC_BasedCodebook(y, Fs);
            save codebook;

            warndlg('DONE! The password has been added in our
database','Teaching progress...');

        else
            teaching = audiorecorder(samplingfrequency,samplingbits,1);
            record(teaching,duration);
            while (isrecording(teaching)==1)
            end
            y = getaudiodata(teaching, 'uint8');
            name = input('Save as:','s');
            sound_number        = 1;
            data{sound_number,1} = y;
            data{sound_number,2} = name;

save('database.dat','data','sound_number','samplingfrequency','samplingbits');
```

```
            m=1;
            n=5;
            No_pts=12948;                 % No of points to be recorded
            Fs=22050;                     %sampling rate

             codebook = MFCC_BasedCodebook(y, Fs);
            save codebook;


            warndlg('DONE! The password has been added in our
database','Teaching progress...');

        end
    end

   % Testing %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    if option==2
        if (exist('database.dat')==2)
            load('database.dat','-mat');
            load('codebook.mat');
            testing = audiorecorder(samplingfrequency,samplingbits,1);
            record(testing,duration);
            while (isrecording(testing)==1)
            end
            y = getaudiodata(testing, 'uint8');

            % RECOGNITION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            recognize = zeros(sound_number,1);
            D1 = specgram(y,512,samplingfrequency,512,384);
            for ii=1:sound_number
                D2                 =
specgram(data{ii,1},512,samplingfrequency,512,384);
                SM                 = simmx(abs(D1),abs(D2));
                [p,q,C]            = dpfast(1-SM);
                c_value            = C(size(C,1),size(C,2));
                recognize(ii)      = c_value;
            end

           [min_value,min_index] = min(recognize);

            No_pts=12948;                % No of points to be recorded
            Fs=22050;                   %sampling rate
            Speech = y;

            v = mfcc(Speech, Fs);         % Compute MFCC's
            d = euclid_dist(v, codebook);   % compute the distance of the
currently recorded speech with file in the database
            dist = sum(min(d,[],2)) / size(d,1);
            display (dist);
            display (min(recognize));

              if ( min(recognize) <= 0.004 && dist <= 4.00 )
                selected = data{min_index,2};
                message = strcat('Recognized: ',num2str(selected));
                msgbox(message,'Result','warn');
            else
                warndlg('WRONG','Warning')
            end

        else
            warndlg('Database is empty','Warning')
        end
    end
    % Database %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    if option==3
        if (exist('database.dat')==2)
            load('database.dat','-mat');
            disp('Reading database...');
            for ii=1:sound_number
                message=strcat(num2str(data{ii,2}));
                disp('__________')
                disp(message);
            end
        else
            warndlg('Database is empty','Warning')
        end
    end

    % Delete database %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    if option==4
        clc;
        close all;
        if (exist('database.dat')==2)
            button = questdlg('Do you really want to delete the
database?','Warning');
            if strcmp(button,'Yes')
                delete('database.dat');
                delete('codebook.mat');

                warndlg('Database has been deleted','Warning');
            end
        else
            warndlg('Database is empty','Warning')
        end
    end

end
```

simmx.m

```
function M = simmx(A,B)
% M = simmx(A,B)
%.   calculate a sim matrix between specgram-like feature matrices A and B.
%    size(M) = [size(A,2) size(B,2)]; A and B have same #rows.
% 2003-03-15 dpwe@ee.columbia.edu

EA = sqrt(sum(A.^2));
EB = sqrt(sum(B.^2));

%ncA = size(A,2);
%ncB = size(B,2);
%M = zeros(ncA, ncB);
%for i = 1:ncA
%  for j = 1:ncB
%    % normalized inner product i.e. cos(angle between vectors)
%    M(i,j) = (A(:,i)'*B(:,j))/(EA(i)*EB(j));
%  end
%end

% this is 10x faster
M = (A'*B)./(EA'*EB);
```

dpfast.m

```
function [p,q,D,sc] = dpfast(M,C,T,G)
% [p,q,D,sc] = dpfast(M,C,T,G)
%    Use dynamic programming to find a min-cost path through matrix M.
%    Return state sequence in p,q; full min cost matrix as D and
%    local costs along best path in sc.
%    This version gives the same results as dp.m, but uses dpcore.mex
%    to run ~200x faster.
%    C is a step matrix, with rows (i step, j step, cost factor)
%    Default is [1 1 1.0;0 1 1.0;1 0 1.0];
%    Another good one is [1 1 1;1 0 1;0 1 1;1 2 2;2 1 2]
%    T selects traceback origin: 0 is to any edge; 1 is top right (default);
%    T > 1 finds path to min of anti-diagonal T points away from top-right.
%    Optional G defines length of 'gulleys' for T=0 mode; default 0.5
%    (i.e. accept path to only 50% of edge nearest top-right)
% 2003-04-04,2005-04-04 dpwe@ee.columbia.edu $Header:
/Users/dpwe/projects/dtw/RCS/dpfast.m,v 1.3 2005/04/05 14:46:38 dpwe Exp $

if nargin < 2
  % Default step / cost matrix
  C = [1 1 1.0;0 1 1.0;1 0 1.0];
end

if nargin < 3
  % Default: path to top-right
  T = 1;
end

if nargin < 4
  % how big are gulleys?
  G = 0.5;  % half the extent
end

[r,c] = size(M);

% Core cumulative cost calculation coded as mex
[D,phi] = dpcore(M,C);

p = [];
q = [];

%% Traceback from top left?
%i = r;
%j = c;

if T == 0
  % Traceback from lowest cost "to edge" (gulleys)
  TE = D(r,:);
  RE = D(:,c);
  % eliminate points not in gulleys
  TE(1:round((1-G)*c)) = max(max(D));
  RE(1:round((1-G)*r)) = max(max(D));
  if (min(TE) < min(RE))
    i = r;
    j = max(find(TE==min(TE)));
  else
    i = max(find(RE==min(RE)));
    j = c;
  end
else
  % Traceback from min of antidiagonal
  %stepback = floor(0.1*c);
  stepback = T;
  slice = diag(fliplr(D),-(r-stepback));
  [mm,ii] = min(slice);
  i = r - stepback + ii;
  j = c + 1 - ii;
end

p=i;
q=j;

sc = M(p,q);

while i > 1 & j > 1
%  disp(['i=',num2str(i),' j=',num2str(j)]);
  tb = phi(i,j);
  i = i - C(tb,1);
  j = j - C(tb,2);
  p = [i,p];
  q = [j,q];
  sc = [M(i,j),sc];
end
```

MFCC BasedCodebook.m

```
function codebook = MFCC_BasedCodebook(Speech, Fs);
% The function takes the Speech sample and the sampling frequency as input
% and generated the codebook after vector quantization of the modified mfcc
output

%  train_dir : string name of directory contains all train sound files
%  n         : number of train files in traindir
%  codebook : trained VQ codebooks, code{i} for i-th speaker

% The number of centeriods
CentN = 16;
% The MODIFIED mfcc algorithm
MFCC_Signal = mfcc(Speech, Fs);
% vector quatization for database
codebook = vector_quant(MFCC_Signal, CentN);
```

mfcc.m

```
function  v= mfcc(s,fs)

% The function computed the mfcc output of the input signal s sampled at fs
%    s:    No of points
%    fs:   Sampling rate


N=256;                        % size of each frame
M=156;                        % overlap size
nof=40;                       % number of filters
len=80;                       % The number of times for loop is to be run
a(1:N,1:len)=0;               % framing the signal with overlap
% initialization of the first chunk
a(:,1)=s(1:N);
for j=2:len
    % extracts frames from the input speech vector
    a(:,j)=s((N-M)*j+1:(N-M)*j+N);
end;
% change 1. kaiser window is used in stead of hamming window
% computes the kaiser window coefficients
h= kaiser(N, 5);                   % windowing
% applies the hamming window to each frame
for j=1:len;
   b(:,j)= a(:,j).* h;
end
% computes the mel filter bank coeffs
 m=melfilterbank(nof,N,fs);              % normailising to mel freq
% The computation of the cepstrum coefficients
for j=1:len
    y(:,j)=fft(b(:,j));                 % calculating fft
    n2 = 1 + floor(N/2);             % adjust the dimensions of the vector y for mel
filter banks
    % The absolute of the fft is considered instead of computing the square
    % of the fft
    ms = m * abs(y(1:n2,j));  % applies the mel filter bank
    v(:,j)=dct(log(ms));                          % converting back to time domain
end
v(1,:)=[];
```

melfilterbank.m

```
function m = melfilterbank(p, n, fs)
% The function computed the mel filter banks for robust speaker recognition
% The filter spectrum is such that the passband area remains the same yet
% the pasband frequencies decrease and the power increases, to emphasis the
% higher frequency components

% p   number of filters in filterbank
% n   length of fft
% fs  sample rate in Hz

f0 = 700/fs;
fn2 = floor(n/2);

lr = log(1 + 0.5/f0) / (p+1);
% convert to fft bin numbers with 0 for DC term
bl = n * (f0 * (exp([0 1 p p+1] * lr) - 1));

b1 = floor(bl(1)) + 1;
b2 = ceil(bl(2));
b3 = floor(bl(3));
b4 = min(fn2, ceil(bl(4))) - 1;

pf = log(1 + (b1:b4)/n/f0) / lr;
fp = floor(pf);
pm = pf - fp;

r = [fp(b2:b4) 1+fp(1:b3)];
c = [b2:b4 1:b3] + 1;
v = 2 * [1-pm(b2:b4) pm(1:b3)];

m = sparse(r, c, v, p, 1+fn2);
```

euclid_dist.m

```
function d = euclid_dist(x,y)

% x, y:   Two matrices whose each column is an a vector data.
%   d:    Element d(i,j) will be the Euclidean distance between two
%         column vectors X(:,i) and Y(:,j)
%
% The Euclidean distance D between two vectors X and Y is:
%  D = sum((x-y).^2).^0.5

% The dimentions of the matrix x is returned in M,N
[M, N] = size(x);
% The dimensions of the matrix y is returned in M2 and P
[M2, P] = size(y);

d = zeros(N, P);
% Checks if the siz miss matched
if (N < P)
    copies = zeros(1,P);
    % Implements the formula for computing euclidean distanc
    for n = 1:N
        d(n,:) = sum((x(:, n+copies) - y) .^2, 1);
    end
else
    copies = zeros(1,N);
    % Implements the formula for computing euclidean distanc
    for p = 1:P
        d(:,p) = sum((x - y(:, p+copies)) .^2, 1)';
    end
end
% the final step of computing the eulid distance Z = sqrt(X^2 + Y^2)
d = d.^0.5;
```

vector_quant.m

```
function b = vector_quant(v,no_centroids)


% v:            speech vectors ,mel scaled.
% no_codebook:  no.of centroids reqd. Here k=16.
% b:            Codebook generated
warning off MATLAB:divideByZero
no_update=5 ;                     %no of updates
c=mean(v,2)  ;                    % finding initial codebook
e = 0.01 ;                        % splitting parameter
c(:,1)=c(:,1) + c(:,1)*e ;        % splitting the codebook into 2.
c(:,2)=c(:,1) - c(:,1)*e;
for up1=1:no_update;
    d=euclid_dist(v,c);               % calculating the  euclidean distance.
    [m,id]=min(d,[],2);               % finding the minimum distance.
    [rows,cols]=size(c);

    for j=1:cols;

        c(:,j)=mean(v(:,find(id==j)),2);  % finding the centroid of the new
cluster.

    end;
end    %update end
n=1;n=n*2;
```

```
while cols < no_centroids  ;                    % updating the code book to get
the reqd. no.

    for i = 1:cols ;                    % of code vectors.

        c(:,i)=c(:,i) + c(:,i)*e;
        c(:,i+n)=c(:,i) - c(:,i)*e;

    end;

    for up2=no_update  %update2
        d=euclid_dist(v,c);             % calculating the  euclidean distance

        [m,i]=min(d,[],2);              % finding the minimum distance

        [rows,cols] = size(c);

        for j=1:cols;

            c(:,j)=mean(v(:,find(i==j)),2);  % finding the centroid of the new
cluster

        end;
    end %end update2
    n=n*2;
end
b=c;
```

APPENDIX B

Table 2: The Results for the Performance Evaluation

No. of Samples	Randomly Selected Speakers										Same Speaker Same Word
	Different Words					Same Word					
	1	2	3	4	5	1	2	3	4	5	
1	No	No	No	No	No	No	No	No	No	No	Yes
2	No	No	No	No	No	No	No	No	No	No	Yes
3	No	No	No	No	No	No	No	No	No	No	Yes
4	No	No	No	No	No	No	No	No	No	No	Yes
5	No	No	No	No	No	No	No	No	No	No	Yes
6	No	No	No	No	No	No	No	No	No	No	Yes
7	No	No	No	No	No	No	No	No	No	No	Yes
8	No	No	No	No	No	No	No	No	No	No	Yes
9	No	No	No	No	No	No	No	No	No	No	Yes
10	No	No	No	No	No	No	No	No	No	No	No
11	No	No	No	No	No	No	No	No	No	No	No
12	No	No	No	No	No	No	No	No	No	No	Yes
13	No	No	No	No	No	No	No	No	No	No	No
14	No	No	No	No	No	No	No	No	No	No	Yes
15	No	No	No	No	No	No	No	No	No	No	Yes
16	No	No	No	No	No	No	No	No	No	No	Yes
17	No	No	No	No	No	No	No	No	No	No	Yes
18	No	No	No	No	No	No	No	No	No	No	Yes
19	No	No	No	No	No	No	No	No	No	No	Yes
20	No	No	No	No	No	No	No	No	No	No	No
21	No	No	No	No	No	No	No	No	No	No	Yes
22	No	No	No	No	No	No	No	No	No	No	Yes
23	No	No	No	No	No	No	No	No	No	No	Yes
24	No	No	No	No	No	No	No	No	No	No	Yes
25	No	No	No	No	No	No	No	No	No	No	Yes
26	No	No	No	No	No	No	No	No	No	No	Yes
27	No	No	No	No	No	No	No	No	No	No	Yes
28	No	No	No	No	No	No	No	No	No	No	Yes
29	No	No	No	No	No	No	No	No	No	No	Yes
30	No	No	No	No	No	No	No	No	No	No	Yes
31	No	No	No	No	No	No	No	No	No	No	No
32	No	No	No	No	No	No	No	No	No	No	Yes

This experiment was done on 32 persons. Each of them stored their templates in the database. The first experiment was to test the system by different speech and different speaker. The second one was to test the system by the same speech but different speaker. Finally, it was the same speaker and the same speech. The results are discussed in Chapter 4.