7. MATLAB BASED SPEAKER RECOGNITION

7.1 Overview

A MATLAB based program has been developed by the author for speaker recognition. The program is used initially to create a database of known speakers where the speech signals of these speakers are stored as ".wav" files in the database. Then, the framing, Hamming windowing, MFCC feature extraction, and VQ feature matching methods are used to identify unknown speakers. This Chapter describes the details of the program developed.

7.2 The Speaker Recognition Program

The developed program is Graphical User Interface (MENU type) and is shown in Figure 7.1. Full MATLAB code listing of the program is given in Appendix A. The program consists of a main module (speaker.m) and a number of function files called by the main module. The MATLAB code extract used to implement the Graphical User Interface (MENU type) is shown below:

```
options = 13;
sel = 0;
%                         THE MAIN MENU
%                         =============
while sel ~= options,
   sel = menu('NEAR EAST UNIVERSITY SPEAKER RECOGNITION SYSTEM',...
         'Load a new sound file from disk',...
         'Play a sound file from disk',...
         'Display a sound waveform from disk',...
         'Display a sound waveform from database',...
         'Display all sound waveforms in database at the same time',...
         'Speaker recognition',...
         'Display sound power spectrum',...
         'Display sound with and without windowing',...
         'Sound database information',...
         'Display information of a sound file in the database',...
         'Delete sound database',...
         'Help',...
         'Exit');
```

The program consists of 13 GUI (MENU type) items, and each item is described below briefly:

7.2.1 Option 1: Load a New Sound File From Disk

This is the first option of the MENU and it allows the user to add sound files to the database. The files must already exists on a digital medium (e.g. on the hard disk) and must have ".wav" file extensions, duration of the speech file is between 1 and 2 second all the speaker speake the same word "zero". In a typical speaker recognition application a microphone is used to record and save the speech signals as files on a digital medium. These files are just like any other ordinary computer files and can be copied, deleted, renamed and so on. It is important to note that all the recordings should be done in the same environment, and using the same equipment.

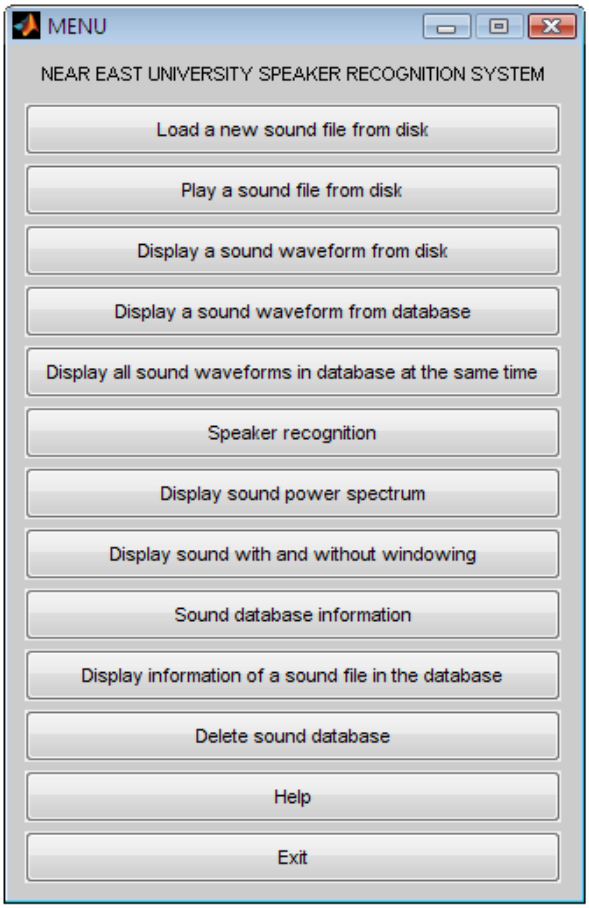

Figure 7.1 Speaker Recognition System Menu

As shown in Figure 7.2, this option enables the user to select the source, and the names of the speech files easily using a Dialog Box.

The user selects the sound file and clicks the Open button. Then the system prompts the user to enter a unique ID number (an integer number) for the sound file for future identification. The sound file selected by the user is saved in the database called SOUNDS.DAT. The filename, the pathname and the ID number of the selected file are all saved in the database for future processing. Figure 7.3 shows how the selected sound file (s2.wav in this example) is saved in the database with ID number of 2.

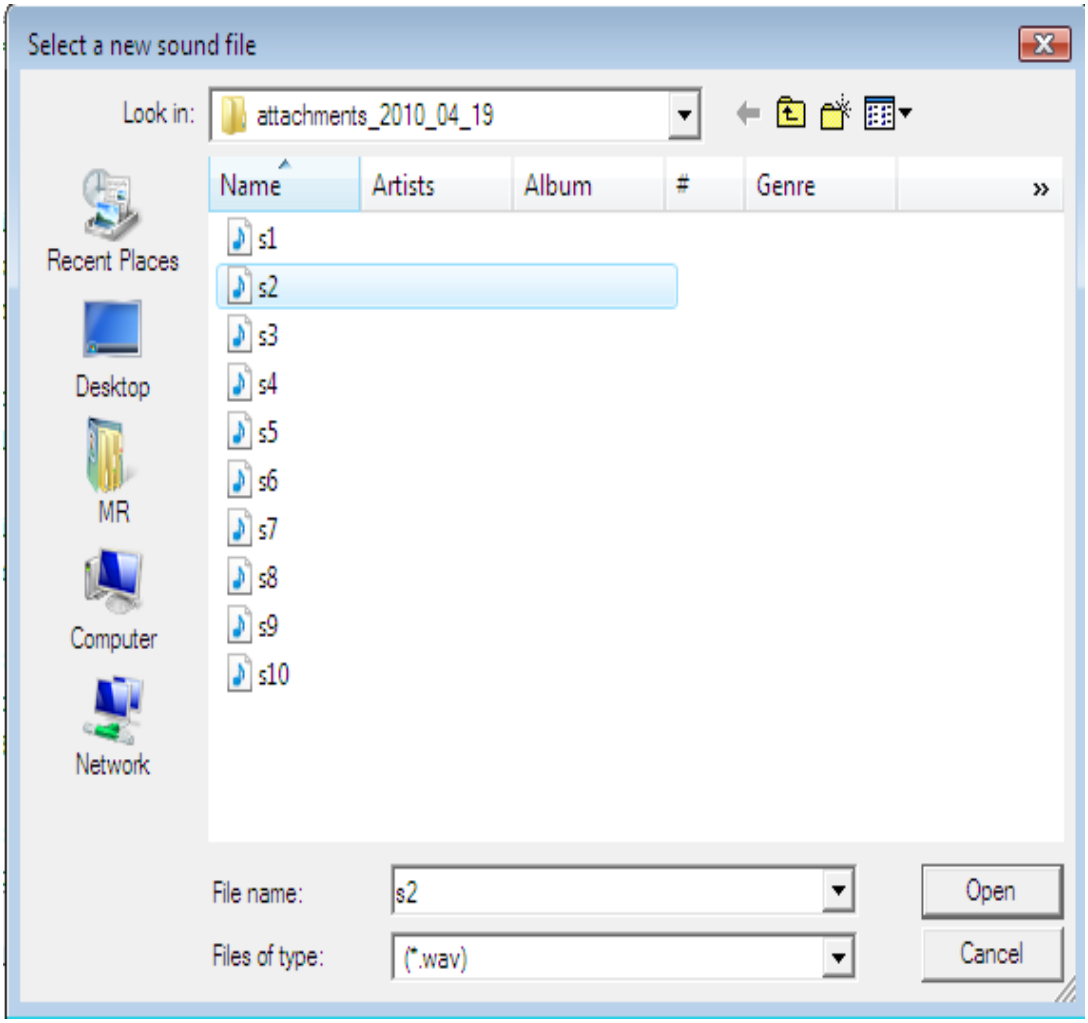

Figure 7.2 Dialog Box for selecting a new sound file

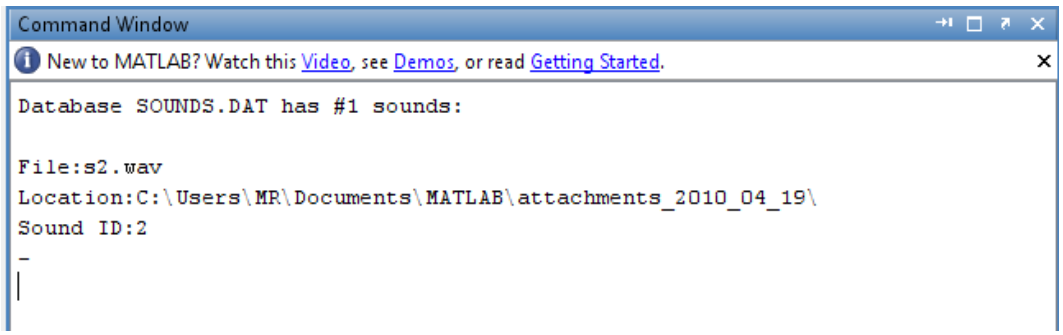

Figure 7.3 File "s2.wav" is saved in the database SOUNDS.DAT with ID number 2

The MATLAB code extract used to implement Option 1 is shown below:

```
% MENU OPTION 1 - LOAD A NEW SOUND FILE FROM DISK
% ================================================

if sel == 1
    [filename,pathname] = uigetfile('*.wav','Select a new sound file');
    [y, Fs, nbits] = wavread(strcat(pathname,filename));
    ID = input('Enter an ID that will be used for recognition:');
    if(exist('C:\Users\MR\Documents\MATLAB\attachments_2010_04_19\
    SOUNDS.DAT') == 2)

      load('C:\Users\MR\Documents\MATLAB\attachments_2010_04_19\
      SOUNDS.DAT','-mat');
      sound_no = sound_no + 1;
      data{sound_no,1} = y;
      data{sound_no,2} = ID;
      data{sound_no,3} = pathname;
      data{sound_no,4} = filename;
      save('C:\Users\MR\Documents\MATLAB\attachments_2010_04_19\
      SOUNDS.DAT','data','sound_no','-append');
      sound(y, Fs);
      msgbox('New sound added to database...');
    else
      sampling_freq = Fs;
      sampling_bits = nbits;
      sound_no = 1;
      data{sound_no,1} = y;
      data{sound_no,2} = ID;
      data{sound_no,3} = pathname;
      data{sound_no,4} = filename;
      save('C:\Users\MR\Documents\MATLAB\attachments_2010_04_19\
      SOUNDS.DAT','data','sound_no','sampling_freq','sampling_bits');
      sound(y, Fs);
      msgbox('New Sound added to database...');
    end
end
```

7.2.2 Option 2: Play a Sound File From Disk

This option allow the user to play a selected sound file on the speakers of the PC. After selecting this option, a Dialog Box as in Figure 7.4 is displayed where the required file can be selected.

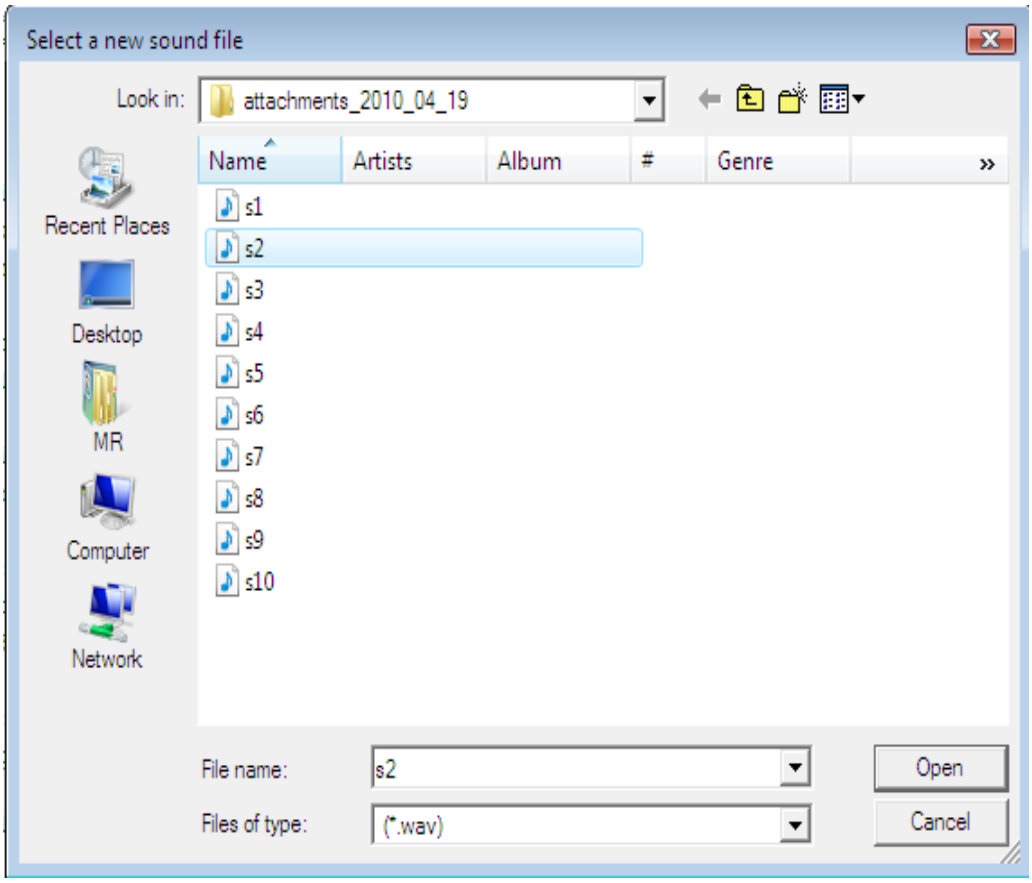

Figure 7.4 Selecting a sound file to play (e.g. s2.wav)

The MATLAB code extract used to implement Option 2 is shown below:

```
% MENU OPTION 2 - PLAY A SOUND FILE FROM DISK
% ============================================

if sel == 2
   [filename,pathname] = uigetfile('*.wav');
   [y, Fs, nbits] = wavread(strcat(pathname,filename));
   wavplay(y,Fs);
end
```

7.2.3 Option 3: Display a Sound Waveform From Disk

This option allows the user to select and display a sound file from a Dialog Box. Figure 7.5 shows a typical output when sound file called "s2.wav" was selected and displayed.

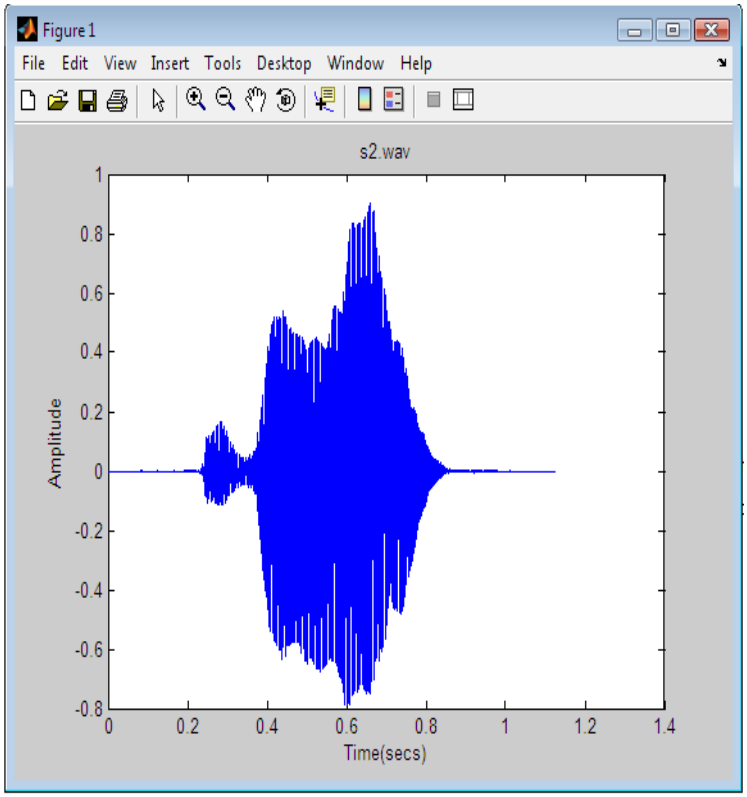

Figure 7.5 The waveform for sound file s2.wav

The MATLAB code extract used to implement Option 3 is shown below:

```
% MENU OPTION 3 - DISPLAY A SOUND WAVEFORM FROM DISK
% ==================================================

if sel == 3
   [filename,pathname] = uigetfile('*.wav');
   [y, Fs, nbits] = wavread(strcat(pathname,filename));
   t = 0:1/Fs:length(y)/Fs-1/Fs;
   plot(t,y)
   xlabel('Time(secs)');
   ylabel('Amplitude');
   title (filename);
end
```

7.2.4 Option 4: Display a Sound Waveform From the Database

This option is similar to option 3, but the sound file to be displayed is selected from the sound database rather than from the disk. As shown in Figure 7.6, the user is first required to enter the ID number of the file to be displayed.

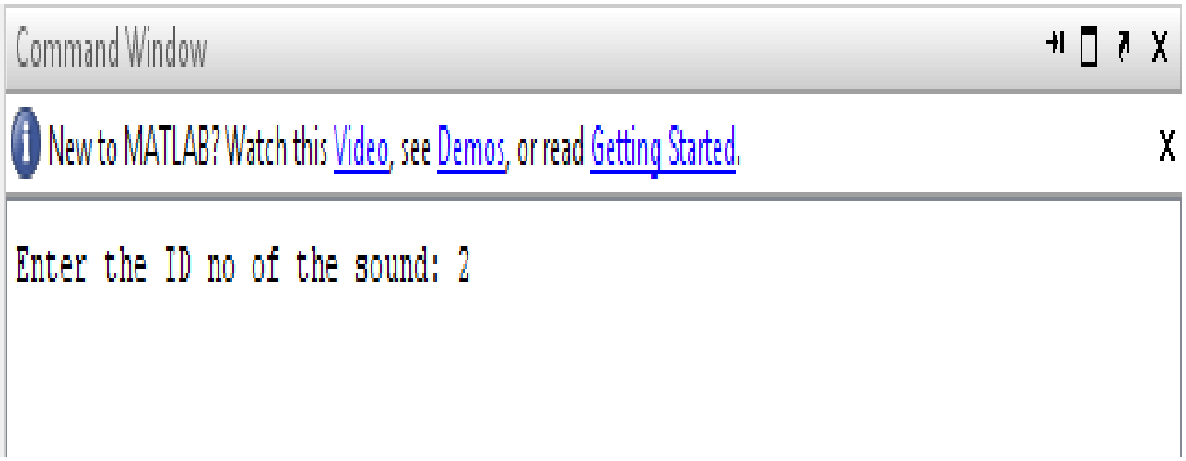

Figure 7.6 Entering ID of the sound file to be displayed

The MATLAB code extract used to implement Option 4 is shown below:

```
% MENU OPTION 4 - DISPLAY A SOUND WAVEFORM FROM DATABASE
% ======================================================

if sel == 4
  IDD = input('Enter the ID no of the sound: ');
  load('C:\Users\MR\Documents\MATLAB\attachments_2010_04_19\
  SOUNDS.DAT','-mat');
       for ii=1:sound_no
             id = data{ii,2};
             if (IDD == id)
                filename = data{ii,4};
                pathname = data{ii,3};
                [y, Fs, nbits] = wavread(strcat(pathname,filename));
                wavplay(y,Fs);
                t = 0:1/Fs:length(y)/Fs-1/Fs;
                plot(t,y);
             end
       end
end
```

7.2.5 Option 5: Display All Sound Waveforms in the Database at the Same Time

This option displays the waveform of all the sound files stored in the database. Figure 7.7 shows an example output when this option is selected (in this example there were only 3 files in the database named "s1.wav", "s2.wav" and s10.wav").

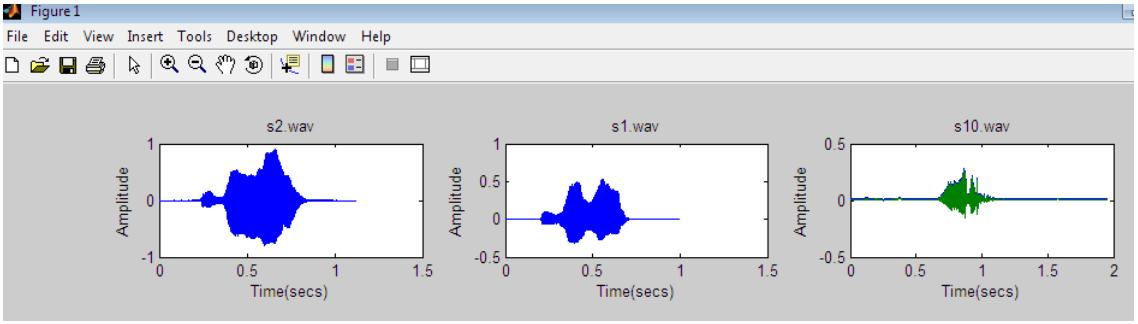

Figure 7.7 Displaying waveforms of all files in the database

The MATLAB code extract used to implement Option 5 is shown below:

```
% MENU OPTION 5-DISPLAY ALL WAVEFORMS IN DATABASE AT THE SAME TIME
% ================================================================

if(sel == 5)
   for ii=1:sound_no
      id = data{ii,2};
      filename = data{ii,4};
      pathname = data{ii,3};
      [y, Fs, nbits] = wavread(strcat(pathname,filename));
      t = 0:1/Fs:length(y)/Fs-1/Fs;
      subplot(4,3,ii); plot(t,y);  xlabel('Time(secs)');
      ylabel('Amplitude'); title (filename);
   end
end
```

7.2.6 Option 6: Speaker Recognition

This is the main option in this thesis which implements the actual speaker recognition process. When the user click this option a Dialog Box as in Figure 7.4 is displayed, prompting the user to select a sound file to be identified. The MFCC algorithm is implemented in this option to extract the features of the sound file. Then these features are compared using Vector

Quantization feature matching algorithm. The identity of the matching speaker is output and displayed by the program.

In Figure 7.8, sound file with ID of 2 (file "s2.wav") was selected for recognition. As can be seen from the figure, at the end of the program the file has been identified and its ID is displayed on the screen. In this example, there were 3 sound files in the database named "s1.wav2, "s2.wav", and "s3.wav". The program displays informative messages during its execution, such as the creation of matrices containing all the frames, application of the Hamming Window and the FFT, and the determination of the MFCC coefficients.

More details about this option are given in later sections of the thesis.

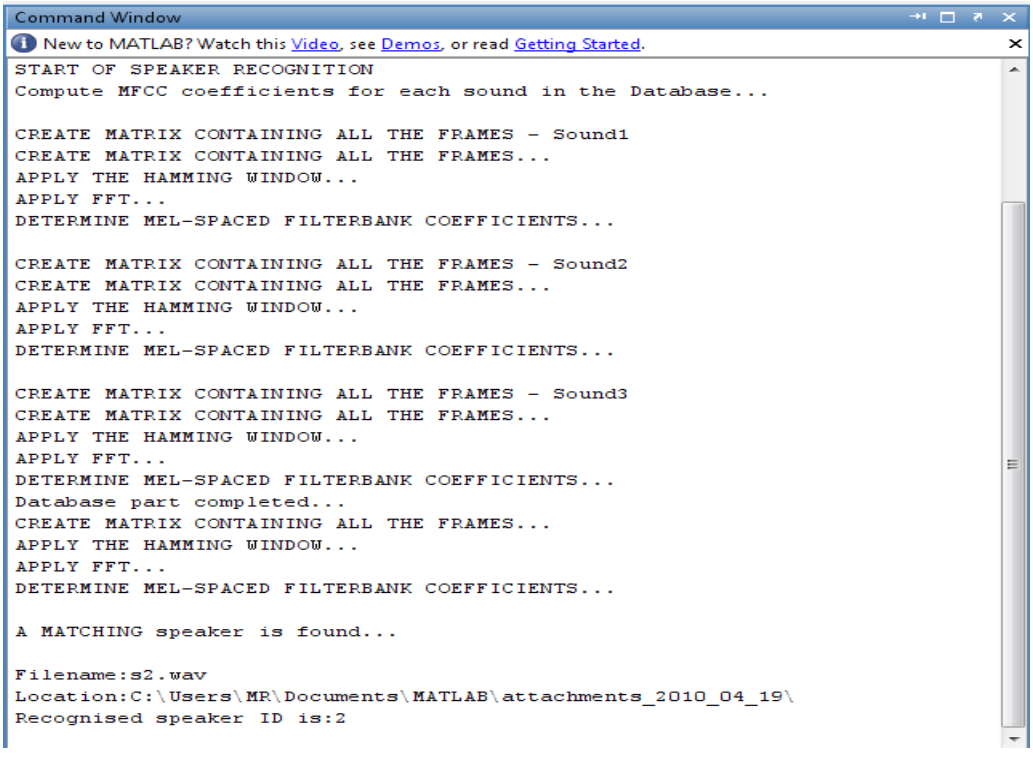

Figure 7.8 The system recognize the selecting file s2.wav

7.2.7 Option 7: Display Sound Power Spectrum

This option displays the power spectrum of a sound file selected by the user. Both the linear and the logarithmic power spectrums are displayed. Figure 7.9 shows a typical output displayed when this option is selected.

The MATLAB code extract used to implement Option 7 is shown below. MATLAB function file "pspectrum.m" is used to implement this option:

```
% MENU OPTION 7 - DISPLAY SOUND POWER SPECTRUM (Linear and Logarithmic)
% ======================================================================

if sel == 7,
   clc;
   load('C:\Users\MR\Documents\MATLAB\attachments_2010_04_19\
   SOUNDS.DAT','-mat');
   [filename,pathname] = uigetfile('*.wav','Select a new sound file');
   [y, Fs, nbits] = wavread(strcat(pathname,filename));
   m = pspectrum(y, Fs);
end
% FUNCTION pspectrum

function r = pspectrum(s, fs)
  l = length(s);
  m = 100;
  n = 256;

  nbFrame = floor((l - n) / m ) + 1;

%
% Create a matrix M containing all the frames
%
  for i = 1:n
    for j = 1:nbFrame
       M(i, j) = s(((j - 1) * m) + i);
    end
 end

%
% Matrix M created. Now apply HAMMING window and store in matrix N.
% Column vectors of N are the original frame vectors transformed by the %
Hamming window filter
%
  h = hamming(n);
  N = diag(h) * M;
```

```
%
% Now apply FFT and create a new matrix M2 where the column
vectors are % the FFTs of the column vectors of N. The elements
of column matrix M2 % contain the frames of the original
% signal, filtered by the Hamming window and transformed with
the FFT. % The elements of M2 are complex numbers and
symmetrical because FFT
% was used to transform the data. Each column in M2 is a power
spectrum % of the original signal

  for i = 1:nbFrame
      M2(:,i) = fft(N(:, i));
  end

  t = n / 2;
  tm = length(s) / fs;
  subplot(121);
  imagesc([0 tm], [0 fs/2], abs(M2(1:t, :)).^2), axis xy;
  title('Power spectrum (Linear)');
  xlabel('Time [s]');
  ylabel('Frequency [Hz]');
  colorbar;

  subplot(122);
  imagesc([0 tm], [0 fs/2], 20*log10(abs(M2(1:t, :)).^2)), axis
xy;
  title('Power spectrum (Logarithmic)');
  xlabel('Time [s]');
  ylabel('Frequency [Hz]');
  colorbar;
```

7.2.8 Option 8: Display Sound With and Without Windowing

The windowing is done to avoid problems due to truncation of the signal. This option displays a sound signal waveform before and after applying Hamming windowing to the sound frames. Figure 7.10 shows a typical output from the program. The Hamming Windows is displayed at the top, followed by the sound frames before and after windowing.

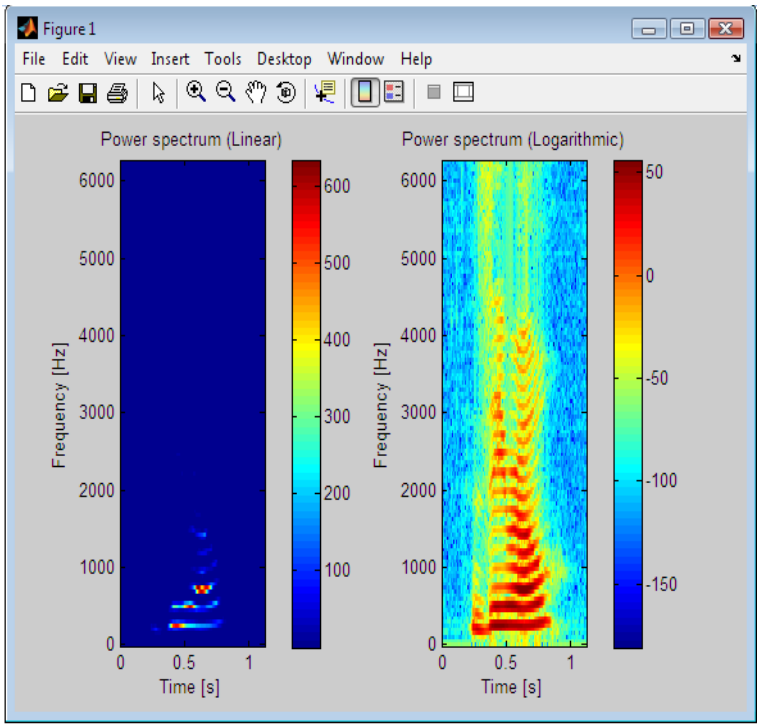

Figure 7.9 Typical Power Spectrum output from the program

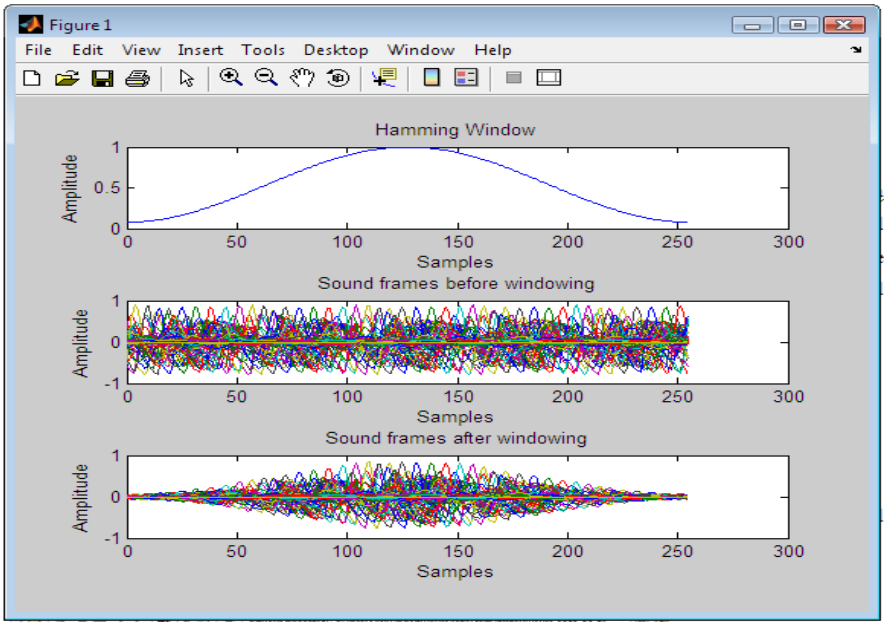

Figure 7.10 Typical output from the program showing the effects of windowing

The MATLAB code extract used to implement Option 8 is shown below. MATLAB function file “compwind.m” is used to implement this option:

```
% MENU OPTION 8 - DISPLAY SOUND WITH AND WITHOUT WINDOWING
% ========================================================

if sel == 8,
   clc;
   load('C:\Users\MR\Documents\MATLAB\attachments_2010_04_19\
   SOUNDS.DAT','-mat');
   [filename,pathname] = uigetfile('*.wav','Select a new sound file');
   [y, Fs, nbits] = wavread(strcat(pathname,filename));
   m = compwind(y, Fs);
end


% FUNCTION compwind
%

function r = compwind(s, fs)
  l = length(s);
  m = 100;
  n = 256;

  nbFrame = floor((l - n) / m ) + 1;

%
% Create a matrix M containg all the frames
%
  for i = 1:n
    for j = 1:nbFrame
       M(i, j) = s(((j - 1) * m) + i);
    end
  end

%
% Matrix M created. Now apply HAMMING window and store in matrix N
% Column vectors of N are the original frame vectors transformed by the %
Hamming window filter
%
  h = hamming(n);
  N = diag(h) * M;

%
% Now plot the window, without window, and with window
%
```

```
  t = (0:n-1);
  subplot(3,1,1);
  plot(t,h);
  title('Hamming Window');
  xlabel('Samples');
  ylabel('Amplitude');

%
% Plot the sound without windowing
%
  subplot(3,1,2);
  plot(t,M);
  title('Sound frames before windowing');
  xlabel('Samples');
  ylabel('Amplitude');

%
% Plot the sound with windowing
%
  subplot(3,1,3);
  plot(t,N);
  title('Sound frames after windowing');
  xlabel('Samples');
  ylabel('Amplitude');
```

7.2.9 Option 9: Sound Database Information

When the user click's this option the details of the sound files stored in the database are displayed. Figure 7.11 shows a typical output from the program.

The MATLAB code extract used to implement Option 9 is shown below.

```
% MENU OPTION 9 - SOUND DATABASE INFORMATION
% ==========================================
%
%
%
if sel == 9,
  if(exist('C:\Users\MR\Documents\MATLAB\
  attachments_2010_04_19\SOUNDS.DAT')== 2)
  load('C:\Users\MR\Documents\MATLAB\
  attachments_2010_04_19\SOUNDS.DAT','-mat');
  clc;
  message = strcat('Database SOUNDS.DAT has # ',
  num2str(sound_no),'  sounds:');
  disp(message);
  disp(' ');
  for ii=1:sound_no
```

```
  message = strcat('File:',data{ii,4});
  disp(message);
  message = strcat('Location:', data{ii,3});
  disp(message);
  message = strcat('Sound ID:',num2str(data{ii,2}));
  disp(message);
  disp('-');
 end
 else
  warndlg('Database SOUNDS.DAT is empty',' Warning ')
 end
end
```

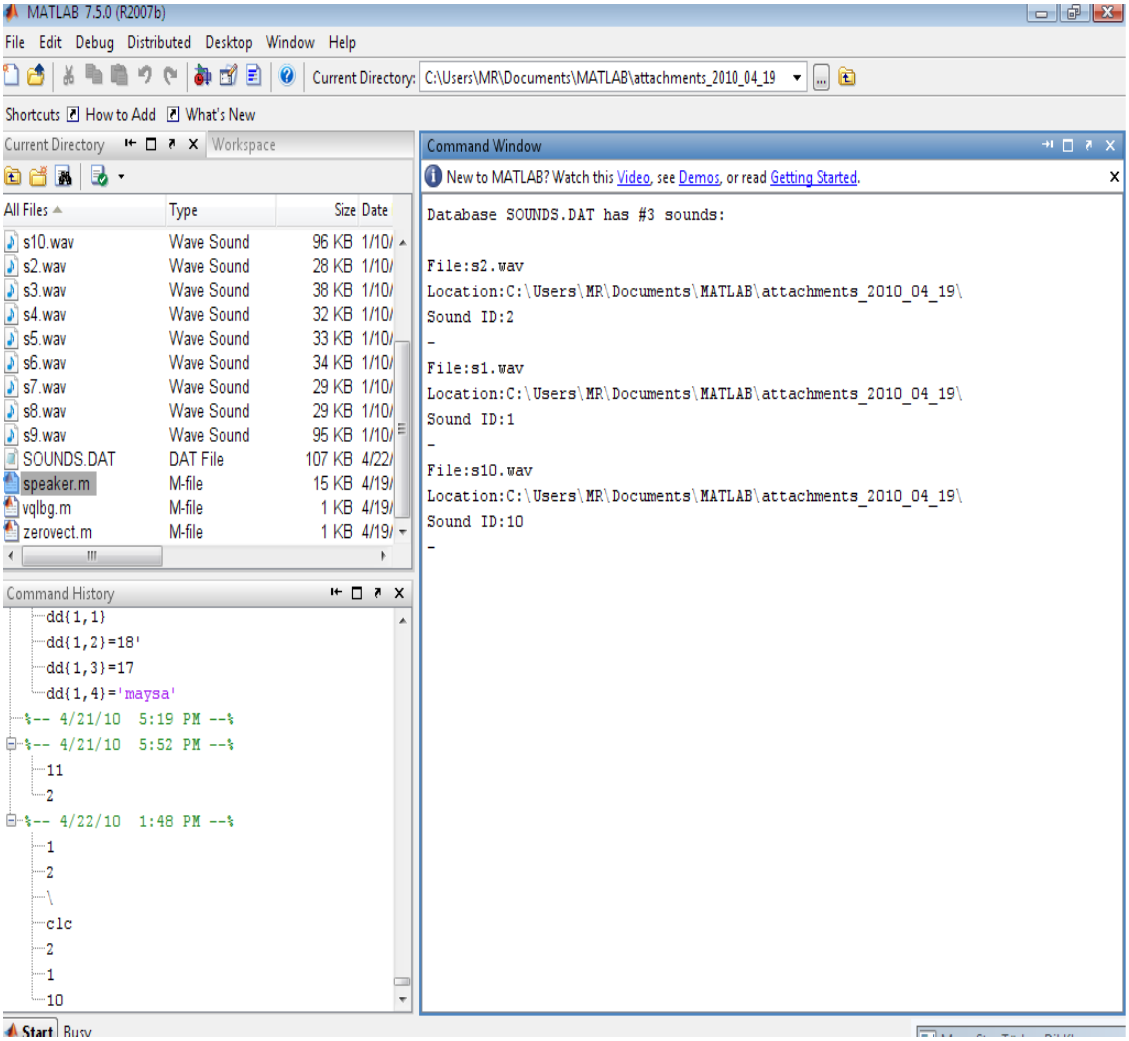

Figure 7.11 Displaying information about the files in the database.

7.2.10 Option 10: Display Information of a Sound File in the Database

This option is similar to option 9 but here the details of only the selected file is displayed on the screen. A file is selected by entering its ID number.

7.2.11 Option 11: Delete Sound Database

This option allows the user to delete the database file SOUNDS.DAT. The user is prompted to confirm the actual deletion process before the database is deleted (see Figure 7.12).

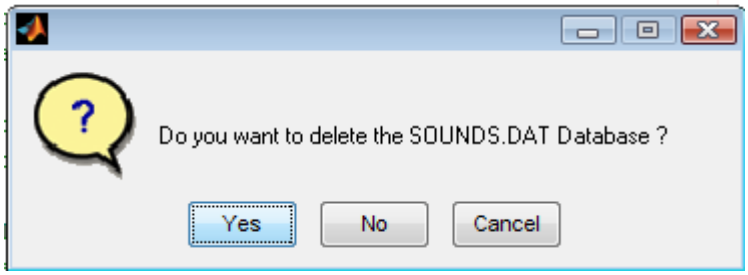

Figure 7.12 Message box after selecting option 11

The MATLAB code extract used to implement Option 11 is shown below.

```
% MENU OPTION 11 - DELETE SOUND DATABASE
% ======================================

if sel == 11
   clc;
   close all;
   if(exist('C:\Users\MR\Documents\MATLAB\
   attachments_2010_04_19\SOUNDS.DAT') == 2)
   button = questdlg('Do you want to delete the SOUNDS.DAT
   Database ?');
   if strcmp(button,'Yes')
     delete('C:\Users\MR\Documents\MATLAB\
     attachments_2010_04_19\SOUNDS.DAT');
     msgbox('SOUNDS.DAT database deleted...');
   end
   else
     warndlg('Database is already empty.', ' Warning')
   end
end
```

7.2.12 Option 12: Help

The help option displays the task of each option in the MENU. Figure 7.13 is the output when this option is selected.

7.2.13 Option 13: Exit

This option is used to terminate the program.

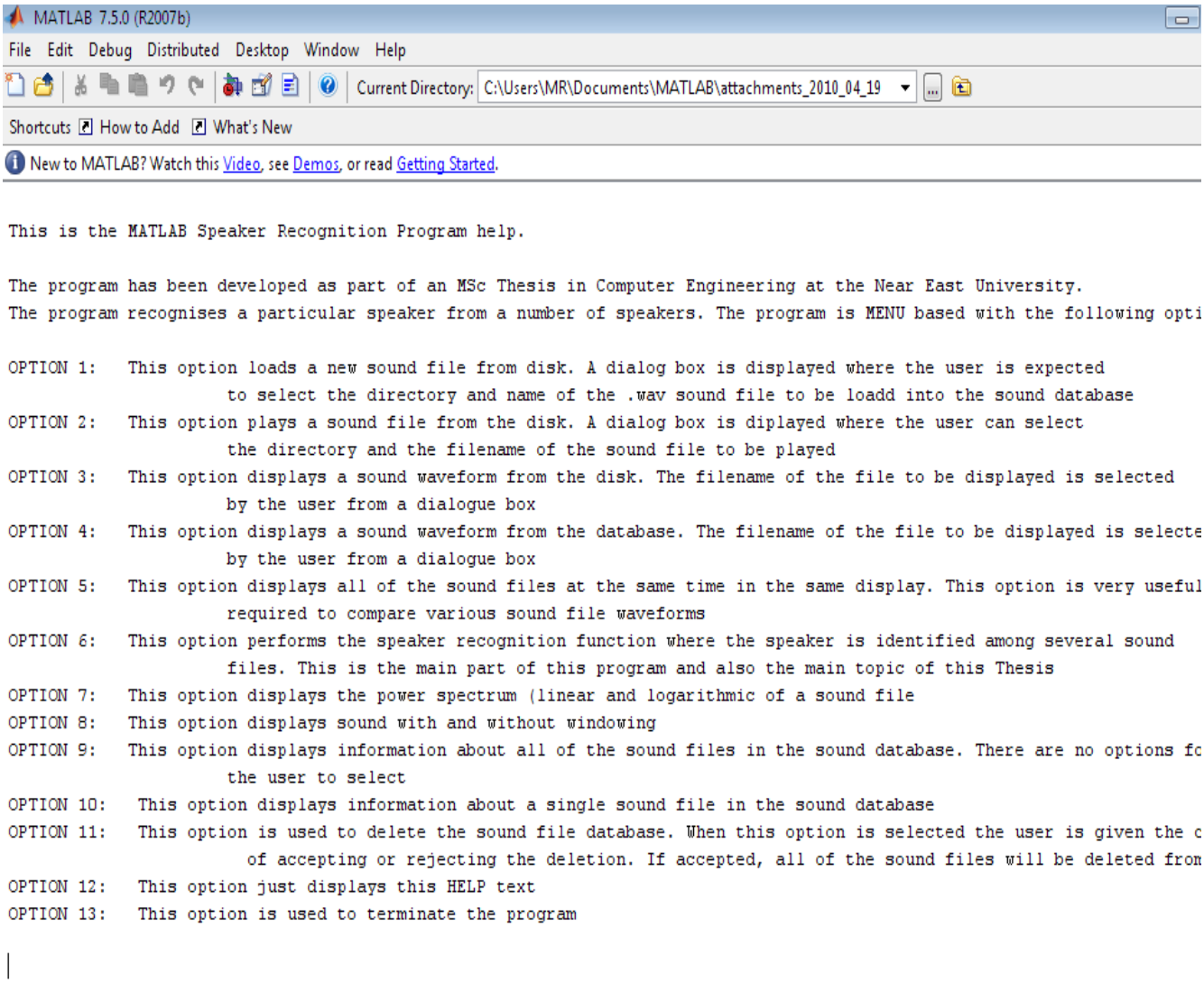

Figure 7.13 Displaying HELP with option 12

7.3 Steps in Speaker Recognition

In this section we will describe the steps used for matlab based speaker recognition system (MENU option 6). The speaker recognition process is carried out in two major phases: Feature Extraction, and Feature Matching. The steps required for each phase are described in the following sections.

7.3.1 Feature extraction

Extracting the features of a sound signal is called the “Feature Extraction”, which is basically the extraction of the fundamental parameters identifying a speech signal. Although there are several methods, the MFCC is the most frequently used method in the speaker recognition and this is the method implemented in the MATLAB program in this thesis. The feature extraction steps are given below:

1-) Frame Blocking (Divide signal into frames)
2-) Windowing (For each frame obtain the amplitude spectrum)
3-) Fast Fourier Transform (FFT) (extract the frequency elements)
4-) Mel- Frequency Warping (convert to mel spectrum)
5-) Cepstrum (take the discrete cosine transform to find the Cepstrum coefficients)

7.3.1.1 Frame blocking

Frame blocking is the process of dividing the sampled signal into specified number of small overlapping frames. Frames aim to model small sections of the signal that is statically stationary. The frame blocking process is applied both for the sounds in the database and for the input signal. The MATLAB code for this process was implemented as a function called “cmatrix.m”. The code extract of frame blocking is given below:

```
function r = Cmatrix(s, fs)
  l = length(s);
  m = 100;
  n = 256;
```

```
  nbFrame = floor((l - n) / m ) + 1;
%
% Create a matrix M containing all the frames
%
  disp('CREATE MATRIX CONTAINING ALL THE FRAMES...');
  for i = 1:n
    for j = 1:nbFrame
       M(i, j) = s(((j - 1) * m) + i);
    end
 end
```

7.3.1.2 Windowing

The second phase is to window all the frames. The aim of the windowing is to remove the discontinuities from the beginning and from the end of the framed signal. Typically, Hamming window is used and the code for this process is given below:

```
% Apply HAMMING window to matrix M and store result in matrix N. Column %
vectors of N are the original frame vectors transformed by the
% Hamming window filter

  disp('APPLY THE HAMMING WINDOW...');
  h = hamming(n);
  N = diag(h) * M;
```

7.3.1.3 Fast Fourier Transform (FFT)

The next step is to take the Fast Fourier transform, the Discrete Fourier transform (or Fast Fourier transform) is performed to find the frequency components of a signal (transform from time domain to frequency domain). The code for this process is given below:

```
% Apply FFT and create a new matrix M2 where the column vectors are the %
FFTs of the column vectors of N. The elements of column matrix M2
% contain the frames of the original signal, filtered by the Hamming
% window and transformed with the FFT. The elements of M2 are complex
% numbers and symmetrical because FFT was used to transform the data.
% Each column in M2 is a power spectrum of the original signal

  disp('APPLY FFT...');
  for i = 1:nbFrame
      M2(:,i) = fft(N(:, i));
  end
```

7.3.1.4 Mel frequency wrapping

Human perception of audio frequencies does not follow a linear scale. The Fourier Transformed signal is passed through a set of band pass filters in order to simplify the spectrum without significant loss of data. The studies show that the mel- frequency scale is a linearly spaced below 1kHz and logarithmically spaced for frequencies above that. Therefore for each tone with actual frequency f(Hz), a subjective pitch is measured on scale called the 'mel' scale. The code for this process is given below:

```
% Determine mel-spaced filterbank

 disp('DETERMINE MEL-SPACED FILTERBANK COEFFICIENTS...');
 m = mel(20, n, fs);
 n2 = 1 + floor(n / 2);
 z = m * abs(M2(1:n2, :)).^2;
```

7.3.1.5 Cepstrum

The final stage is to perform the Discrete Cosine Transform to decorrelate the mel logarithmatic magnitude spectrum to the mel frequency cepstral coefficients MFCC. The cepstrum is the inverse Fourier transform (or the Discrete Cosine Transform) of the frequency spectrum of a signal in a time domain. The code for this process is given below:

```
r = dct(log(z));
```

7.4 Speech Feature Matching

7.4.1 Vector Quantization

Vector quantization (VQ) is a concept, which makes use of the relation among elements in the group to encode groups of data as a whole more efficiently rather than individual elements of data.

A speaker recognition system must able to estimate probability distributions of the computed feature vectors. Storing every single vector that generate from the training mode is impossible,

since these distributions are defined over a high-dimensional space. It is often easier to start by quantizing each feature vector to one of a relatively small number of template vectors, with a process called vector quantization. VQ is a process of taking a large set of feature vectors and producing a smaller set of measure vectors that represents the centroids of the distribution.

The technique of VQ consists of extracting a small number of representative feature vectors as an efficient means of characterizing the speaker specific features. By means of VQ, storing every single vector that we generate from the training is impossible. By using these training data features are clustered to form a codebook for each speaker. In the recognition stage, the data from the tested speaker is compared to the codebook of each speaker and measure the difference. These differences are then use to make the recognition decision.

The K-means algorithm is a way to cluster the training vectors to get feature vectors. In this algorithm the vectors are clustered based on attributes into k partitions. It use the *k* means of data generated from gaussian distributions to cluster the vectors. The objective of the k-means is to minimize total intra-cluster variance, here we use k = 16 and variance 0.001.

7.4.2 Distance measure

In the speaker recognition phase, an unknown speaker's voice is represented by a sequence of feature vector {x1, x2 ….xi), and then it is compared with the codebooks from the database. In order to identify the unknown speaker, this can be done by measuring the distortion distance of two vector sets based on minimizing the Euclidean distance.

The Euclidean distance is the "ordinary" distance between the two points that one would measure with a ruler, which can be proven by repeated application of the Pythagorean Theorem. The Euclidean distance between two points P = (p1, p2…pn) and Q = (q1, q2...qn) is given by:

$$\sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$$

The speaker with the lowest distortion distance is chosen to be identified as the unknown speaker that we are looking for.

The MATLAB code that implements the Euclidean was implemented as a function called "disteu.m", and is shown below:

```
% Euclidean distances between columns of two matrices

function d = disteu(x, y)
  [M, N] = size(x);
  [M2, P] = size(y);
  if (M ~= M2)
    error('Matrix dimensions do not match...');
  end
  d = zeros(N, P);
  for ii=1:N
    for jj = 1:P
        d(ii,jj) = sum((x(:,ii)-y(:,jj)).^2).^0.5;
    end
  end
```

7.4 Results

There are several speech and sound databases on the internet that can be used in the development of speaker recognition systems. Some examples are, TSP is speech database developed by Peter Kabal [46]. This database consists over 1400 utterances spoken by 24 speakers (half male, half female).

Speech Corpus [43] is a database of speech audio files and text transcription in a format that can be used to create acoustic models.

Linquistic data consortiun [45] provides a very wide range of speech and text data for research and commercial users. Oxford Acoustic Phonetic Database [44] contain a large number of words and word combinations in several languages. This database is available on CD-ROM and can be used during the development of speech recognition systems.

The developed system was tested using a database with 10 speech signals, stored in the database with names “s1.wav”, “s2.wav”,……,”s10.wav”. These were short speech files taken from Luigi Rosa database (2010) [13].

As an example, it was attempted to recognize the speech file “s3.wav” among the files in the database. The output from the program is given below:

```
Sound file selected for recognition:
File:s3.wav
Location:C:\Users\MR\Documents\MATLAB\attachments_2010_04_19\

NEAR EAST UNIVERSITY - SPEAKER RECOGNITION SYSTEM
==================================================

START OF SPEAKER RECOGNITION
Compute MFCC coefficients for each sound in the Database...

CREATE MATRIX CONTAINING ALL THE FRAMES - Sound1
CREATE MATRIX CONTAINING ALL THE FRAMES...
APPLY THE HAMMING WINDOW...
APPLY FFT...
DETERMINE MEL-SPACED FILTERBANK COEFFICIENTS...

CREATE MATRIX CONTAINING ALL THE FRAMES - Sound2
CREATE MATRIX CONTAINING ALL THE FRAMES...
APPLY THE HAMMING WINDOW...
APPLY FFT...
DETERMINE MEL-SPACED FILTERBANK COEFFICIENTS...

CREATE MATRIX CONTAINING ALL THE FRAMES - Sound3
CREATE MATRIX CONTAINING ALL THE FRAMES...
APPLY THE HAMMING WINDOW...
APPLY FFT...
DETERMINE MEL-SPACED FILTERBANK COEFFICIENTS...

CREATE MATRIX CONTAINING ALL THE FRAMES - Sound4
CREATE MATRIX CONTAINING ALL THE FRAMES...
APPLY THE HAMMING WINDOW...
APPLY FFT...
DETERMINE MEL-SPACED FILTERBANK COEFFICIENTS...
CREATE MATRIX CONTAINING ALL THE FRAMES - Sound5
CREATE MATRIX CONTAINING ALL THE FRAMES...
APPLY THE HAMMING WINDOW...
APPLY FFT...
DETERMINE MEL-SPACED FILTERBANK COEFFICIENTS...

CREATE MATRIX CONTAINING ALL THE FRAMES - Sound6
CREATE MATRIX CONTAINING ALL THE FRAMES...
APPLY THE HAMMING WINDOW...
APPLY FFT...
DETERMINE MEL-SPACED FILTERBANK COEFFICIENTS...

CREATE MATRIX CONTAINING ALL THE FRAMES - Sound7
CREATE MATRIX CONTAINING ALL THE FRAMES...
```

```
APPLY THE HAMMING WINDOW...
APPLY FFT...
DETERMINE MEL-SPACED FILTERBANK COEFFICIENTS...

CREATE MATRIX CONTAINING ALL THE FRAMES - Sound8
CREATE MATRIX CONTAINING ALL THE FRAMES...
APPLY THE HAMMING WINDOW...
APPLY FFT...
DETERMINE MEL-SPACED FILTERBANK COEFFICIENTS...

CREATE MATRIX CONTAINING ALL THE FRAMES - Sound9
CREATE MATRIX CONTAINING ALL THE FRAMES...
APPLY THE HAMMING WINDOW...
APPLY FFT...
DETERMINE MEL-SPACED FILTERBANK COEFFICIENTS...

CREATE MATRIX CONTAINING ALL THE FRAMES - Sound10
CREATE MATRIX CONTAINING ALL THE FRAMES...
APPLY THE HAMMING WINDOW...
APPLY FFT...
DETERMINE MEL-SPACED FILTERBANK COEFFICIENTS...

Database part completed...
CREATE MATRIX CONTAINING ALL THE FRAMES...
APPLY THE HAMMING WINDOW...
APPLY FFT...
DETERMINE MEL-SPACED FILTERBANK COEFFICIENTS...

A MATCHING speaker is found...

Filename:s3.wav
Location:C:\Users\MR\Documents\MATLAB\attachments_2010_04_19\
Recognised speaker ID is:3
```

It is clear that the correct speaker ("s3.wav") has been identified by the program. It took only a few seconds (3.51 seconds) to identify the speaker on a standard 1.8GHz PC with 1GB memory.

The minimum identification time is 2.38 seconds and the maximum identification time is 3.51 seconds.

7.4.1 Modifying the centroids (codebook size)

It was decided to modify the number of centroids used (i.e. the codebook size) in the algorithm and see its effects on the speaker identification rate. The speaker identification rate is defined as the percentage ratio of the number of recognized speakers to the total number of speakers in the database. Table 7.1 shows the change of the identification rate as the number of centroids (k) is changed from 1 to 64. It is clear that a higher identification rate is obtained with larger

values of k. When k is greater than or equal to 4, the identification rate was 100%. i.e. all the speakers in the database were identified correctly.

7.4.2 Speaker recognition in the presence of noise

The noise clearly influences on the performance of the speaker recognition process, making the classification almost useless at high noise levels. It was decided to investigate the effects of noise as the number of centroids (codebook size) is increased. Gaussian noise with a mean of 0 and at various variance levels was introduced into the noise files and the identification rate was observed.

Table 7.1 Identification rate and number of centroids

Number of centroids (k)	**Database size (number of speakers)**	**Identification rate (%)**
1	10	80
2	10	80
4	10	100
8	10	100
16	10	100
32	10	100
64	10	100

Table 7.2 shows the results for code book sizes of k = 1, 2, 4, 8, 16, 32 and 64, and with the noise variance of 0.001. It is clear that the effects of noise is reduced as the number of centroids is increased and the identification rate was 100% when k was greater than or equal to 16.

Figure 7.14 shows the identification rate as the noise level (variance) was increased from 0.001 to 0.1 for different number of centroids. As expected, the identification rate is very high when the noise is small and large number of centroids are used. As the noise level increases the identification rate falls sharply, even for number of centroids.

Table 7.2 Identification rate with noise variance of 0.001

Number of Centroids (k)	**Database Size (Number of Speakers)**	**Identification Rate (%)**
1	10	%60
2	10	%30
4	10	%50
8	10	%70
16	10	%100
32	10	%100
64	10	%100

Table 7.3 shows the result when Gaussian noise variance change from 0.001 to 0.01 and centroids number 64. The result show that, when the variance is increased the identification rate is decrease.

Table 7.3 Identification rate when number of centroids 64 (k=64) and variance change from 0.001 - 0.01

Variance	**No of speaker**	**Identification rate**
0.001	10	%100
0.002	10	%60
0.003	10	%20
0.004	10	%20
0.005	10	%20
0.006	10	%20
0.007	10	%20
0.008	10	%10
0.009	10	%10
0.01	10	%10

It is interesting to see the sound waveforms with and without the noise added. Figure 7.15 shows the sound waveforms with only a small noise (variance = 0.001). The effect of the noise on the shape of the waveforms seems to be negligible. Figure 7.16 shows the sound waveforms when larger noise (variance = 0.01) is added. In Figure 7.17, a large amount of noise (variance = 0.04) is added to the sound file and the effect of the noise seems to be very clear in this figure.

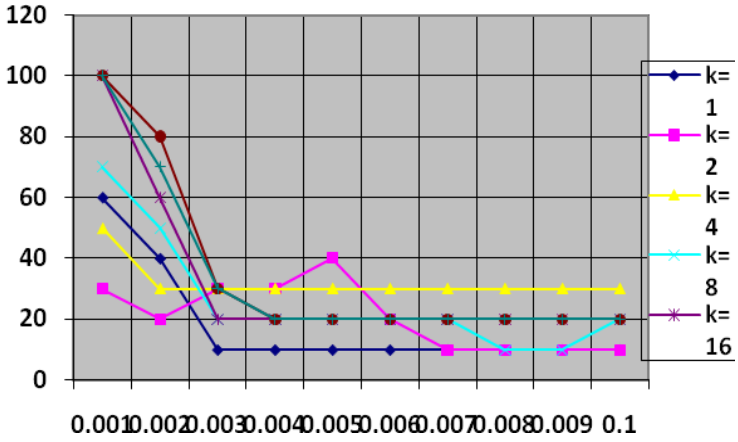

Figure 7.14 Identification rate as the noise level and number of centroids (k) are varied

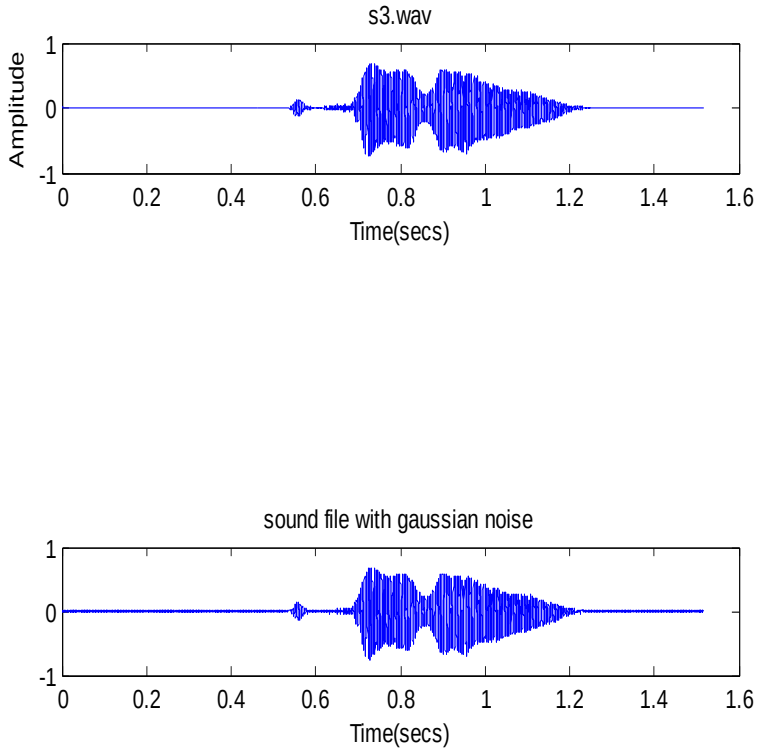

Figure 7.15 Sound waveforms with and without small added noise (variance 0.001)

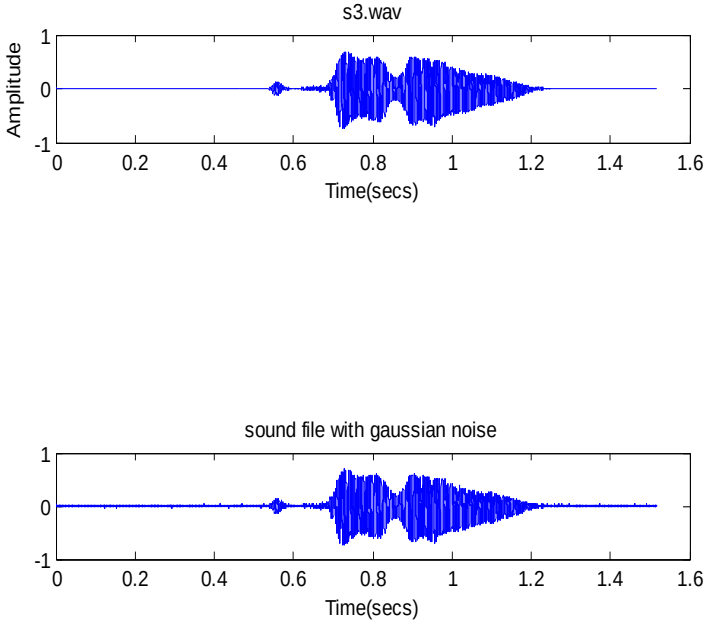

Figure 7.16 Sound waveforms with and without larger added noise (variance 0.01)

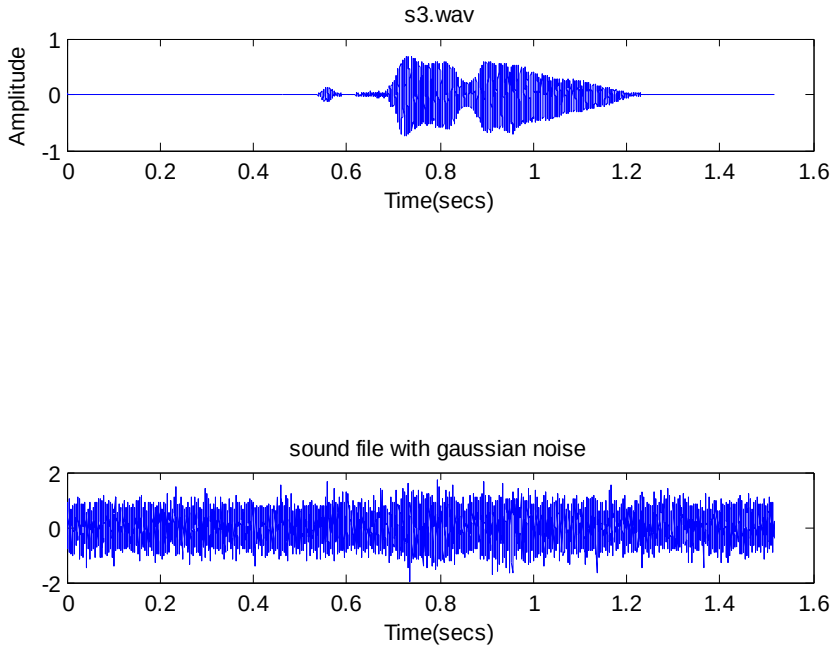

Figure 7.17 Sound waveforms with and without large added noise (variance 0.04).

Table 7.4 shows the results for code book sizes of k = 1, 2, 4, 8, 16, 32 and 64, and with the noise variance of 0.01. It is clear that when the effects of noise is increased the identification rate was small.

Table 7.2 Identification rate with noise variance of 0.01

Number of Centroids (k)	**Database Size (Number of Speakers)**	**Identification Rate (%)**
1	10	%10
2	10	%10
4	10	%40
8	10	%10
16	10	%10
32	10	%10
64	10	%10

7.5 Summary

In this chapter the MATLAB based speaker recognition system and the results are described in detail. The system is Graphical User Interface (MENU type) for this reason each options is described separately with its code taken from the main program (APPENDIX A) and the result obtained by modifying some parameters such a codebook size and the noise variance, the results for this operations is showed in the tables.