COMMAND BY SPEECH RECOGNITION

ARDIAN SYAH B MOHD YUSOF

This thesis is submitted as partial fulfillment of the requirements for the award of the Bachelor of Electrical Engineering (Hons.) (Electronics)

Faculty of Electrical & Electronics Engineering
Universiti Malaysia Pahang

MAY 2008

UNIVERSITI MALAYSIA PAHANG

BORANG PENGESAHAN STATUS TESIS♦

JUDUL: **COMMAND BY SPEECH RECOGNITION**

SESI PENGAJIAN: **2007/2008**

Saya **ARDIAN SYAH B MOHD YUSOF (851001-01-5745)**
(HURUF BESAR)

mengaku membenarkan tesis (Sarjana Muda/~~Sarjana~~ /~~Doktor Falsafah~~)* ini disimpan di Perpustakaan dengan syarat-syarat kegunaan seperti berikut:

1. Tesis adalah hakmilik Kolej Universiti Kejuruteraan & Teknologi Malaysia.
2. Perpustakaan dibenarkan membuat salinan untuk tujuan pengajian sahaja.
3. Perpustakaan dibenarkan membuat salinan tesis ini sebagai bahan pertukaran antara institusi pengajian tinggi.
4. **Sila tandakan (√)

☐	**SULIT**	(Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972)
☐	**TERHAD**	(Mengandungi maklumat TERHAD yang telah ditentukan oleh organisasi/badan di mana penyelidikan dijalankan)
√	**TIDAK TERHAD**	

Disahkan oleh:

_____ (TANDATANGAN PENULIS)

_____ (TANDATANGAN PENYELIA)

Alamat Tetap:

NO 91B,
Jalan Sungai siput,
81900, Kota Tinggi,
Johor

RAJA MOHD TAUFIKA RAJA ISMAIL
(Nama Penyelia)

Tarikh: **9 MAY 2008**

Tarikh: : **9 MAY 2008**

CATATAN:
* Potong yang tidak berkenaan.
** Jika tesis ini SULIT atau TERHAD, sila lampirkan surat daripada pihak berkuasa/organisasi berkenaan dengan menyatakan sekali tempoh tesis ini perlu dikelaskan sebagai atau TERHAD.
♦ Tesis dimaksudkan sebagai tesis bagi Ijazah doktor Falsafah dan Sarjana secara Penyelidikan, atau disertasi bagi pengajian secara kerja kursus dan penyelidikan, atau Laporan Projek Sarjana Muda (PSM).

"I hereby declare that the scope and quality of this thesis is qualified for the award of the Bachelor Bachelor of Electrical Engineering (Electronics)"

Signature : _____

Name : RAJA MOHD TAUFIKA RAJA ISMAIL

Date : 9 MAY 2008

Signature : _____

Author : ARDIAN SYAH B MOHD YUSOF

Date : 8 MAY 2008

Signature : _____

Author : ARDIAN SYAH B MOHD YUSOF

Date : 9 MAY 2008

DEDICATION

Decicate to my beloved father, mother, brothes and sisters
who always give me a courage to finish this thesis.

Also, to those people who have been supportive through all this time.
Thank you for the kindnes and advices that have been given.

ACKNOWLEDGEMENT

With encouragement and determination, I would like to acknowledge those people who gave all of their effort in helping me to finish this project. This project will not come to the end without considerable ideas and support from them.

Firstly, I would like to thank my project supervisor, Mr. Raja Mohd Taufika Raja Ismail, for providing the guideline with continues advices and feedback throughout the duration of finishing this project.

Secondly, I would also like to give appreciation to all other University Malaysia Pahang staff members especially Faculty of Electrical and Electronics Engineering staffs that I may have called upon for assistance since the genesis of this project. Their opinions and suggestions have helped me in realizing this project. Also not to be forgotten, I would like to thank to all of my friends for their support, valuable opinions and ideas sharing during the progress of this project.

Finally, I would like to thank to all of my family members for their understanding, encouragement and support, towards the completion of my project. Thank you so much! May god bless you all.

ABSTRACT

Speech recognition is a topic that very useful in many applications and enviroments in our daily life. Generally, speech recognizer is a machine which understand humans and their spoken word in some way and can act thereafter. In daily usage, for example, it can be used in a car environment to voice control non-critical operation such as dialing a phone number to ensure a maximum control to the car and enhance the safety. A different aspect of speech recognition is to facilitate for people with functional disability or other kinds of handicap. The system develop for this speech recognizer will be done using MATLAB and for this project, the system will recognize discrete word only and not a sentences or a robust speech. There are two main operations in this speech recognizer which are generating the voiceprint and store it as template in the word bank and recognizing the word spoken by comparing it with the template stored in word bank. The voiceprint is create by extracting it's MEL Frequency Cepstral Coefficient (MFCC) which is the default number of coefficients need to be extracted are 12. To get more accurate result, 20 coefficients will be extracted. For recognizing purpose, Dynamic Time Warping (DTW) method is use. DTW will calculate the distance of two vectors which are the word spoken and the stored voiceprints and it will recognize the word as the same word if the the distance is the lowest or in other words, nearly zero.

ABSTRAK

Pengecaman suara adalah satu topik yang sangat berguna dalam pelbagai kegunaan dan keadaan dalam kehidupan seharian. Secara amnya, pengecam suara adalah satu mesin yang dapat memahami kata-kata manusia dan boleh bertindak balas melaluinya. Dalam penggunaan seharian sebagai contoh, ia boleh digunakan di dalam kereta bagi mengawal operasi yang tidak kritikal seperti mendail nombor telefon bimbit bagi memaksimumkan pengawalan memandu ke atas kereta. Pada aspek yang berlainan pula, teknologi pengecaman suara dpaat membantu orang-orang cacat. Sistem pengecaman suara bagi projek ini telah dibangunkan menggunakan perisisan MATLAB dan untuk projek ini, sistem ini hanya mengecam perkataan diskret sahaja dan bukannya satu ayat penuh. Ada dua operasi utama untuk sistem pengecaman suar aini iaitu menghasilkan ‘cap suara’ dan menyimpannya sebagai templat di dalam bank perkataan dan yang kedua ialah mengecam suara yang disebut dengan membandingkan suara yang disebut dengan ‘cap suara’ yang disimpan di dalam bank perkataan. Penghasilan ‘cap suara’ dilakukan dengan mengekstrak Pemalar Kepstral Frekuensi Mel (MFCC) di mana pemalar yang perlu diekstrak pada asasnya adalah sebanyak 12. Untuk mendapatkan hasil yang lebih jitu, sebanyak 20 pemalar akan diekstrak. Bagi tujuan pengecaman pula, kaedah ‘Dynamic Time warping’ (DTW) akan digunakan. DTW akan mengira perbezaan antara dua vektor iaitu suara yang disebut dan ‘cap suara’ yang disimpan dan ia dapat membua tpengecaman jika perbezaan antara kedua-dua vektor tersebut adalah kecil atau menghampiri sifar.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

cos	-	cosine
DCT	-	Discrete Cosine Transform
DTW	-	Dynamic Time Warping
f	-	frequency
FFT	-	Fast Fourier Transform
GUI	-	Graphical User Interface
GUIDE	-	Graphical User Interface Development Environment
Hz	-	Hertz
kHz	-	kiloHertz
ln	-	Natural logarithma
$mel(f)$	-	Mel frequency
MFCC	-	Mel Frequency Cepstral Coefficient
ms	-	miliseconds
$s(n)$	-	Speech signal
t	-	Seconds
α	-	Alpha
π	-	phi
Δ	-	delta

LIST OF APPENDIXES

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

Speech recognition is a topic that very useful in many applications and enviroments in our daily life. Generally, speech recognizer is a machine which understand humans and their spoken word in some way and can act thereafter. In daily usage, for example, it can be used in a car environment to voice control non-critical operation such as dialing a phone number to ensure a maximum control to the car and enhance the safety. Applying voice control technology or speech recognition technology seems will help a lot in enhancing the safety for certain situation.A different aspect of speech recognition is to facilitate for people with functional disability or other kinds of handicap. To make their daily routine easier, voice control could be helpful. With their voice, they could operate the light switch, turn on/off the electrical aplliances. Incredibly, this leads to the discussion about intelligent homes where these operations can be made available for common man as well as for the handicapped.

The speech signal and all its characteristics can be represented in two different domains,the time and frequency domain. Spectral representation of speech intensity over time is very popular and the most popular one is the sound spectrogram, see Figure 1.1.

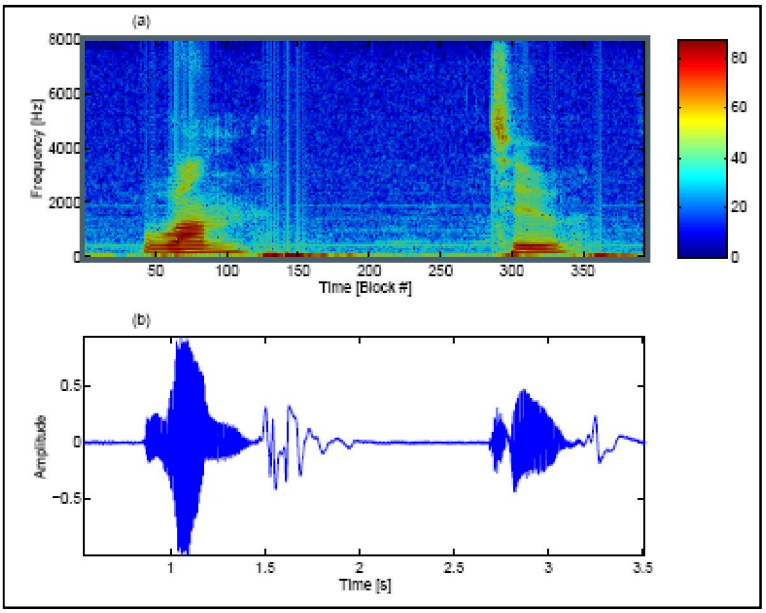


Figure 1.1: Spectral representation of speech intensity (a)Sound Spectrogram
(b) amplitude of the voice signal

The main objective of this final year project is to develop a speech recognition tool based on the MATLAB GUI. This tool will ease and help the researchers, lecturers and the students whom involved in speech recognition study. The implementation of MATLAB software is due to its features which are very useful and easy to use for this project since its already provided certain features that can be used directly for speech recognition analysis. Graphical User Interface (GUI) is create to ensure this tool is user friendly and help the user to perform the analysis faster and easier because they no longer need to write a complex program each time they want to perform some calculations or obtain the graph.

Basically, this speech recognition tool will help the user to record the voice for the analysis purposes. User can choose either to record it with sampling frequency of 8 kHz or 16 kHz. After recording the voice, two graphs will be executed and it will display the voice signal in time domain. One graph will shows the amplitude of the voice signal and the another one will shows its spectrogram.

User can play it back and save the file as .wav file. Besides that, user can also load other wave files which are stored in their computer. Fourier analysis can be done easily with this tool. With one click, user can obtain the voice signal graph in frequency domain which is vital in speech recognition analysis.

This tool offers many advantages to the user because it can help to improve the result of the analysis perform by the user. User no longer need to waste a lot of time just to write a simple program, for example, to obtain the fourier graph for the speech signal. The recording tool provided in this tool also enable the user to record the voice with sampling frequency of 8 kHz compare with the Windows recorder which only enable the user to record the voice with sampling frequency of 16 kHz.

This speech recognition tool is divided into two sections:-

1. Create a GUI to record the voice using microphone and play it back.
2. Create a GUI to perform speech recognition.

1.2 Objectives

1. To create a graphical user interface (GUI) with MATLAB
2. To create a voiceprint for the word spoken
3. To perform speech recognition for commanding electrical devices.

This is the main objective for this project. A GUI will be created as the speech recognizer and it will perform the speech recognition process. The process of recognizing the word spoken is done based on the Dynamic Time Warping (DTW) method/. By using DTW, it will calculate the difference or the distance between the word spoken and and the voiceprint that have been stored in the words bank. If the distance is small, that means the word spoken is equal to the voiceprint and the speech recognizer will recognize it.

1.3 Project scope

The main scope for this project is to create a user friendly GUI using MATLAB to perform the speech signal analysis for the speech recognition purpose. The GUI will consist of several functions and buttons of operation regarding to the speech recognition analysis. User just need to select one of the button to perform certain job. The GUI will display the desire result according to what its task is and have the ability to save or print the result.

1. Recording Tool with Playback Ability.
2. Saving file
3. Perform the speech recognition using MATLAB

1.4 Problem statement

In our daily life, electrical devices are very important in order to improve the quality of our life but most of them are not friendly for those handicapped user.
For that reason, this project is propose in order to create a system that can be attached with electrical devices and command them by using our voice. Those handicapped user can easily operate the devices without touching all the operate button. But, this system is totally not suitable for those who are experience the mute problem.

1.5 Thesis outlines

The Speech Recognition Tool final thesis is a combination of 5 chapters that contains and elaborates specific topics such as the Introduction, Literature Review,

Software Design, Result, Discussion, Conclusion and Further Development that can be applied in this project.

Chapter 1 basically is an introduction of the project. In this chapter, the main idea about the background and objectives of the project will be discussed. The full design and basic concept of the project will be focused in this chapter. The overview of the entire project also will be discussed in this chapter to show proper development of the project.

Chapter 2 is about the literature review and the methodologies for the development of the Speech Recognition Tool. This includes the future project development that can be added in this project.

Chapter 3 will be discussed about the design of the graphical user interface (GUI) using MATLAB. In this chapter, it will explain how to create GUI programmaticaly or using the graphical user interface development environment (GUIDE) provided for MATLAB with version 7.0 and above.

Chapter 4 will discuss the software development. It will shows and sexplain the flow chart that been used to write the coding, developing the process using the MATLAB.

Chapter 5 discusses all the results obtained and the limitation of the project. All discussions are concentrating on the result and performance of the speech recognizer. This chapter also discusses the problem and the recommendation for this project. Commerciliazation value also be discussed in this chapter.

CHAPTER 2

LITERATURE REVIEWS

2.1 Introduction

In this chapter, the basic knowledges and fundamental concept in creating the speech recognizer will be discussed. This speech recognition project is using the MATLAB as the main processor and dynamic programming for the recognition purpose.

2.2 MATLAB

MATLAB is an interactive, matrix-based system for scientific and engineering numeric computation and visualization. User can solve complex numerical problems in a fraction of the time required with a programming language such as Fortan or C. The name MATLAB is derived from MATrix LABoratory. When using MATLAB. The command *help functionname* will give information about a specific function. For example, the command *help fft* will give information about function fast fourier transform and how to use it.

2.2.1 The overview of the MATLAB

The MATLAB high-performance language for technical computing integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Figure 2.1 shows the looks of the MATLAB.

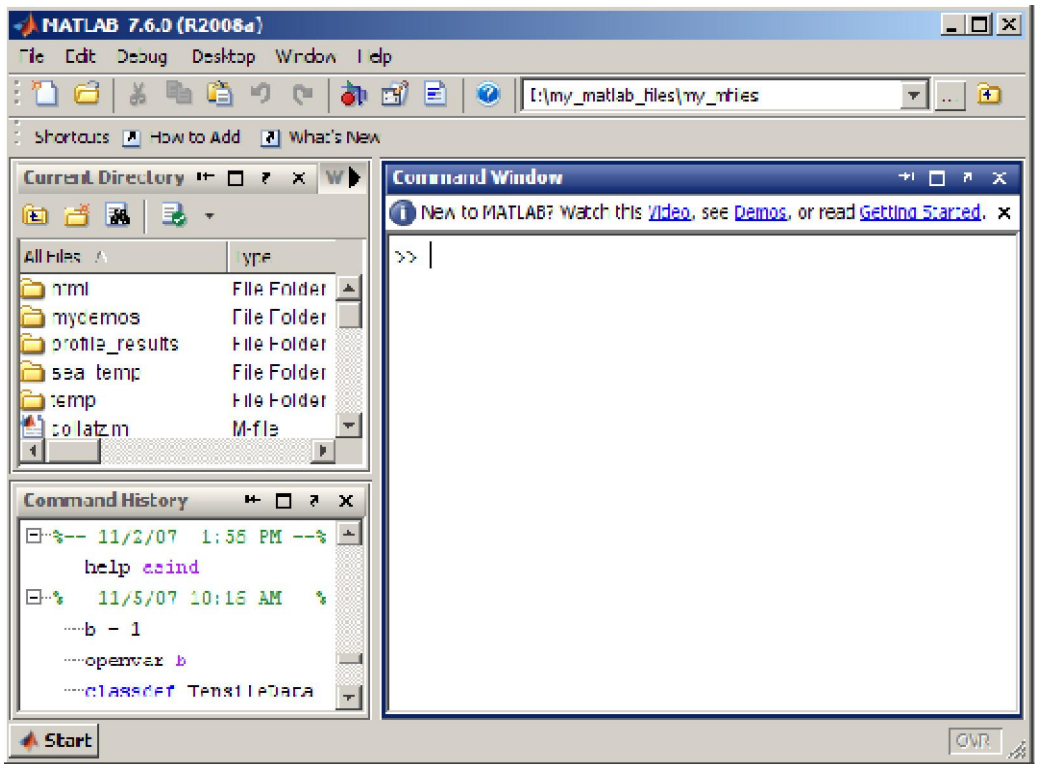


Figure 2.1: MATLAB

Typical uses include:

1. Math and computation
2. Algorithm development
3. Data acquisition
4. Modeling, simulation, and prototyping
5. Data analysis, exploration, and visualization
6. Scientific and engineering graphics
7. Application development, including graphical user interface building

MATLAB has evolved over a period of years with input from many users. In university environments, it is the standard instructional tool for introductory and advanced courses in mathematics, engineering, and science. In industry, MATLAB is the tool of choice for high-productivity research, development, and analysis.

MATLAB features a family of add-on application-specific solutions called *toolboxes*. Very important to most users of MATLAB, toolboxes allow you to *learn* and *apply* specialized technology. Toolboxes are comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular classes of problems. You can add on toolboxes for signal processing, control systems, neural networks, fuzzy logic, wavelets, simulation, and many other areas.

2.2.2 Programming in MATLAB

This section covers those MATLAB product functions that provide conditional program control.

2.2.2.1 If, else, and elseif

The if statement evaluates a logical expression and executes a group of statements when the expression is *true*. The optional elseif and else keywords provide for the execution of alternate groups of statements. An end keyword, which matches the if, terminates the last group of statements. The groups of statements are delineated by the four keywords—no braces or brackets are involved. The MATLAB algorithm for generating a magic square of order *n* involves three different cases: when *n* is odd, when *n* is even but not divisible by 4, or when *n* is divisible by 4. This is described by Figure 2.2:

```
if rem(n,2) ~= 0
        M = odd_magic(n)
elseif rem(n,4) ~= 0
        M = single_even_magic(n)
else
        M = double_even_magic(n)
end
```

Figure 2.2: If, else, and elseif statement

2.2.2.2 Switch and case

The switch statement executes groups of statements based on the value of a variable or expression. The keywords case and otherwise delineate the groups. Only the first matching case is executed. There must always be an end to match the switch. The logic of the magic squares algorithm can also be described by Figure 2.3:

```
switch (rem(n,4)==0) + (rem(n,2)==0)
case 0
M = odd_magic(n)
case 1
M = single_even_magic(n)
case 2
M = double_even_magic(n)
otherwise
error('This is impossible')
end
```

Figure 2.3: Switch and case statement

Unlike the C language switch statement, the MATLAB switch does not fall through. If the first case statement is true, the other case statements do not execute. So, break statements are not required.

2.2.2.3 Try

The general form of a try-catch statement sequence is shown in Figure 2.4. In this sequence the statements between try and catch are executed untilan error occurs.The statements between catch and end are then executed.Use lasterr to see the cause of the error. If an error occurs between catchand end, MATLAB terminates execution unless another try-catch sequencehas been established.

```
try
statement
...
statement
catch
statement
...
statement
end
```

Figure 2.4: Try statement

2.2.3 Creating graphical user interface using GUIDE

GUIDE, the MATLAB graphical user interface development environment, provides a set of tools for creating graphical user interfaces (GUIs). We can also create GUIs programmatically. These tools greatly simplify the process of designing and building GUIs. We can use the GUIDE tools to:

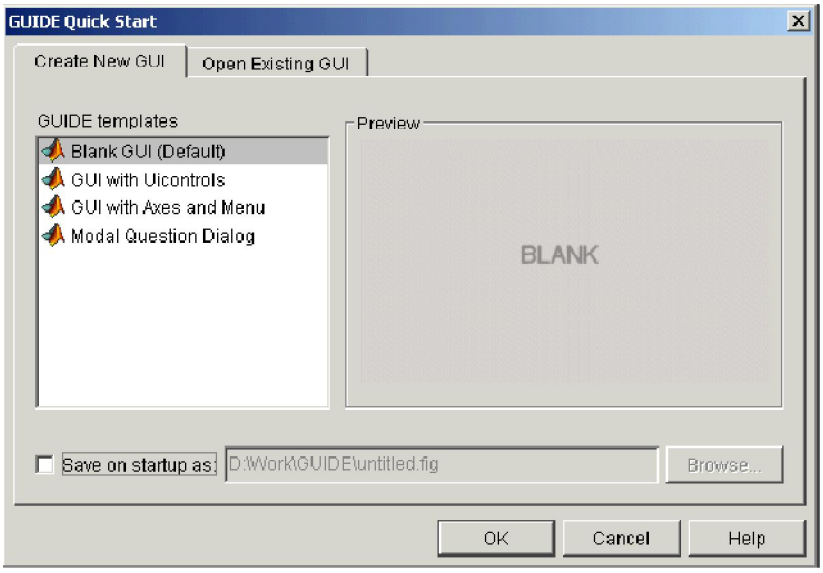
1. Lay out the GUI.
   Using the GUIDE Layout Editor, we can lay out a GUI easily by clicking and dragging GUI components—such as panels, buttons, text fields, sliders, menus, and so on—into the layout area. GUIDE stores the GUI layout in a FIG-file.

2. Program the GUI.

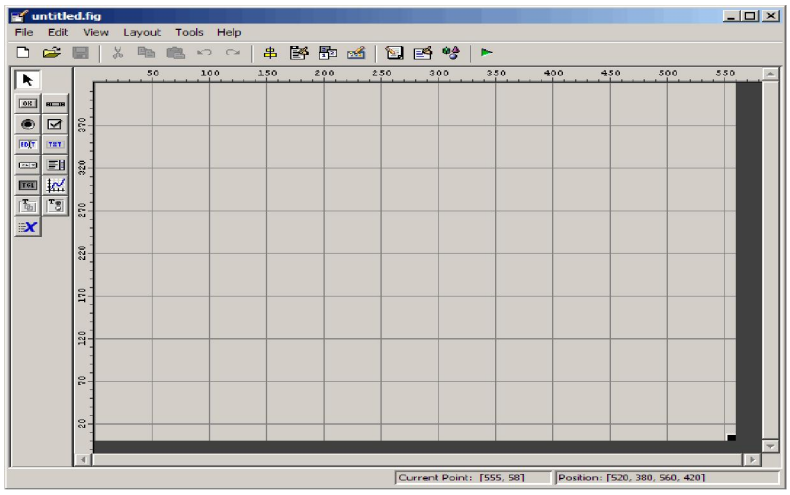
   GUIDE automatically generates an M-file that controls how the GUI operates. The M-file initializes the GUI and contains a framework for the most commonly used callbacks for each component—the commands that execute when a user clicks a GUI component. Using the M-file editor, we can add code to the callbacks to perform the functions we want.

2.2.3.1 Laying out a GUI

Start GUIDE by typing guide at the MATLAB® command prompt. This displays the GUIDE Quick Start dialog box. When open a GUI in GUIDE, it is displayed in the Layout Editor, which is the control panel for all of the GUIDE tools. Figure 2. 5 (b) shows the Layout Editor with a blank GUI template.



(a)



(b)

Figure 2.5: Overview of GUIDE (a) GUIDE quick start page (b) Layout editor

We can lay out our GUI by dragging components, such as panels, push buttons, pop-up menus, or axes, from the component palette, at the left side of the Layout Editor, into the layout area. For example, if you drag a push button into the layout area, it appears as in the Figure 2.6.

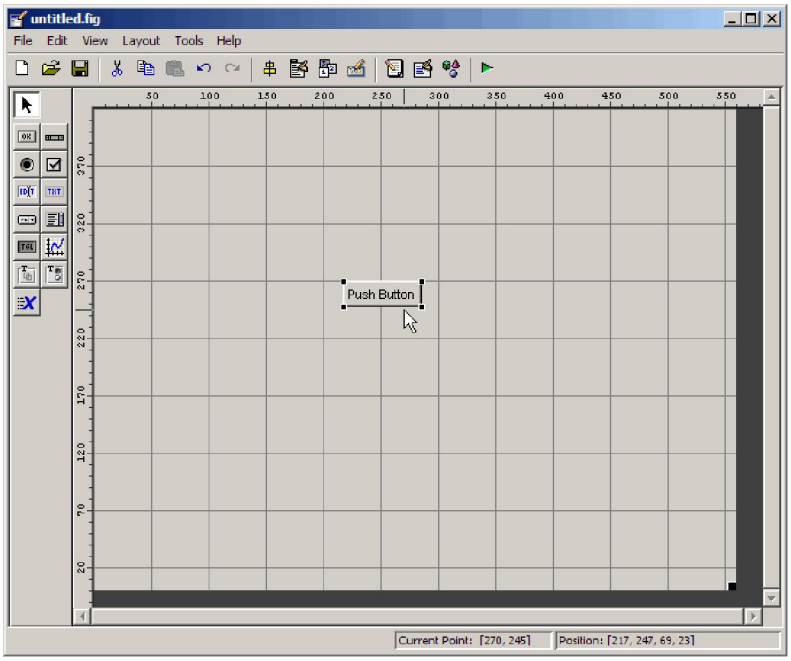


Figure 2.6: Adding the Push Button

We can also use the Layout Editor (along with the Toolbar Editor and Icon Editor) to create menus and toolbars, create and modify tool icons, and set basic properties of the GUI components.

2.2.3.2 Programming a GUI

After laying out the GUI and setting component properties, the next step is to program the GUI. You program the GUI by coding one or more callbacks for each of its components. Callbacks are functions that execute in response to some action by the user. A typical action is clicking a push button. A GUI's callbacks are found in an M-file that GUIDE generates automatically. GUIDE adds templates for the most commonly used callbacks to this M-file, but youmay want to add others. Use the M file Editor to edit this file. The following figure shows the Callback template for a push button.

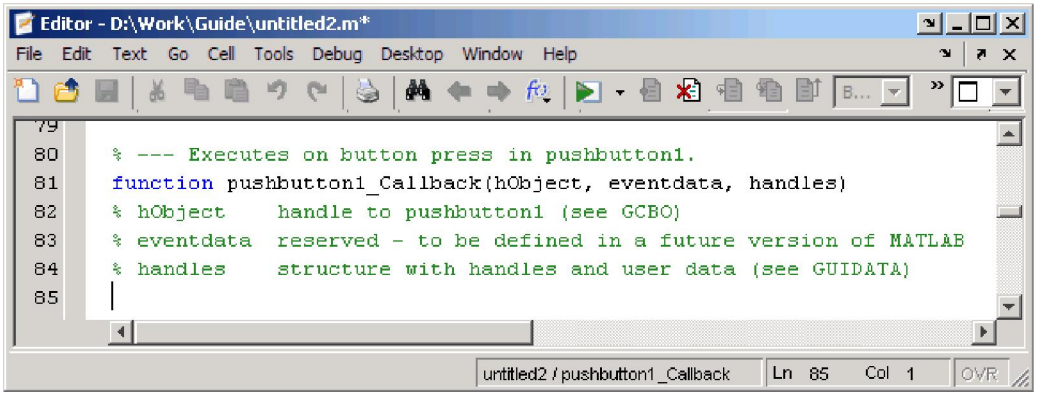


Figure 2.7: Callback template for a push button.

2.3 SPEECH RECOGNITION

Speech recognition (also known as automatic speech recognition or computer speech recognition) converts spoken words to machine-readable input (for example, to the binary code for a string of character codes). The term voice recognition may also be used to refer to speech recognition, but more precisely refers to speaker recognition, which attempts to identify the person speaking, as opposed to what is being said.

2.3.1 SPEECH RECOGNITION METHODOLOGY

Figure 2.8 shows the methodoly of the speech recognition algorithm tha tis been used for this project.

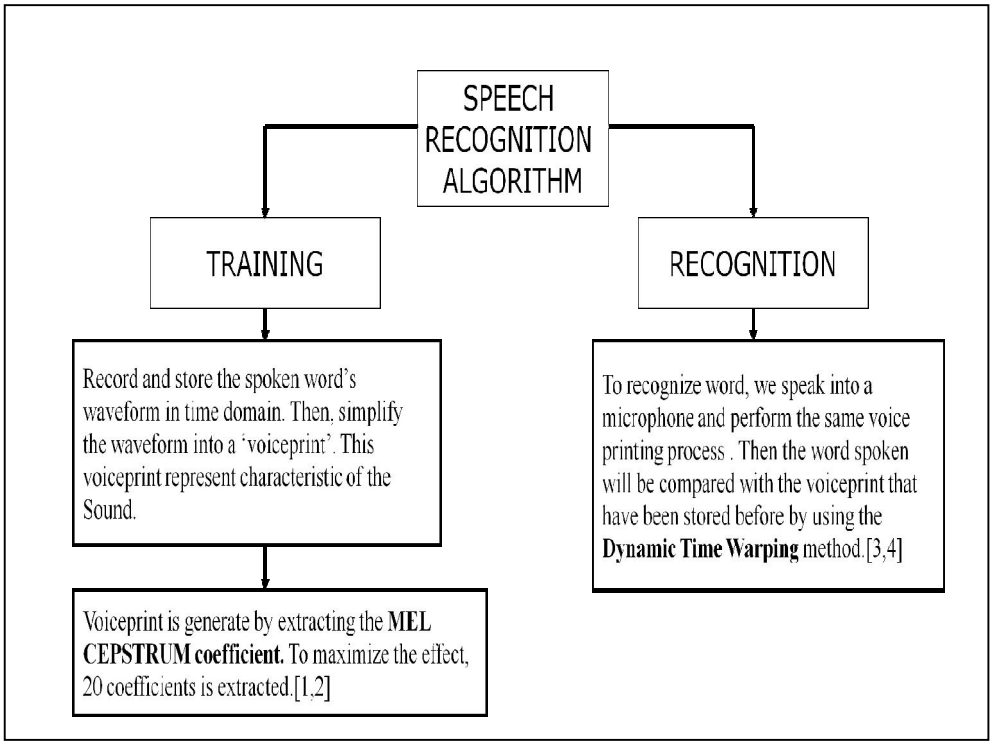


Figure 2.8: Speech recognition algorithm

2.3.2 MEL Frequency Cepstral Coefficient

For speech/speaker recognition, the most commonly used acoustic feature are **mel-scale frequency cepstral coefficient** (**MFCC** for short). The feature takes human perception sensitivity with respect to frequencies into consideration, and thus are best for speech/speaker recognition. Figure 2.9 shows the step on how to calculate the MFCC.

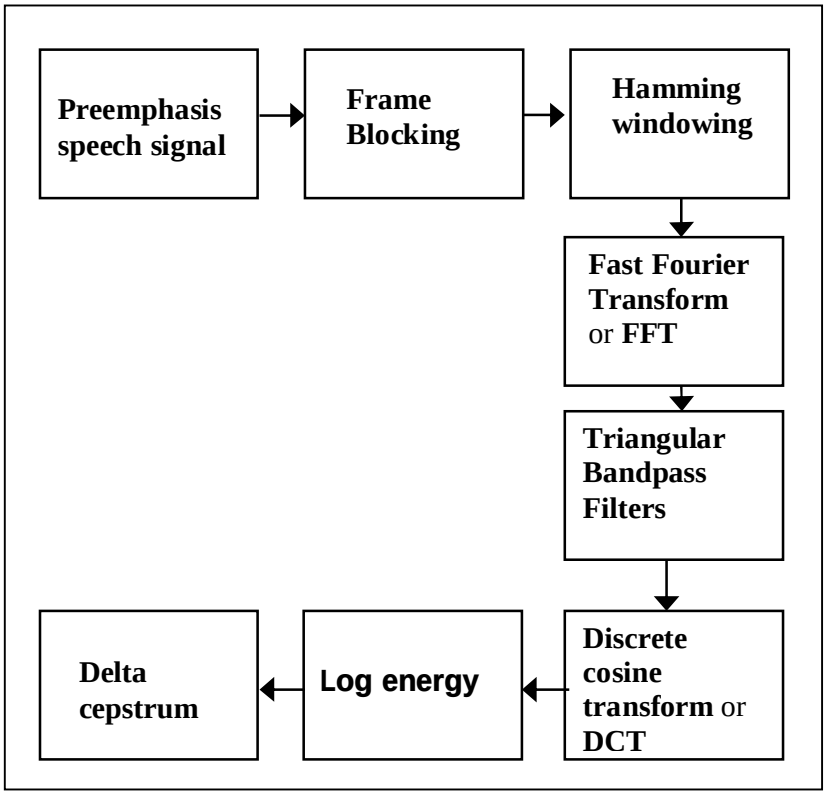


Figure 2.9: Steps to extract MFCC from speech signal

Pre-emphasis is step to send the speech signal s(n) to a high pass filter:

$$2(\ ) = (\ ) - \times (\ - 1)$$

where $s_2(n)$ is the output signal and the value of a is usually between 0.9 to 1.0. The z-transform of the filter is

$$(\ ) = 1 - \times (\ - 1)$$

The goal of pre-emphasis is to compensate the high-frequency part that was suppressed during the sound production mechanism of humans. Moreover, it can also amplify the importance of high-frequency formants. The next example demonstrates the effect of pre-emphasis. The speech after pre-emphasis sounds sharper with a smaller volume as shown in the Figure 2.10.

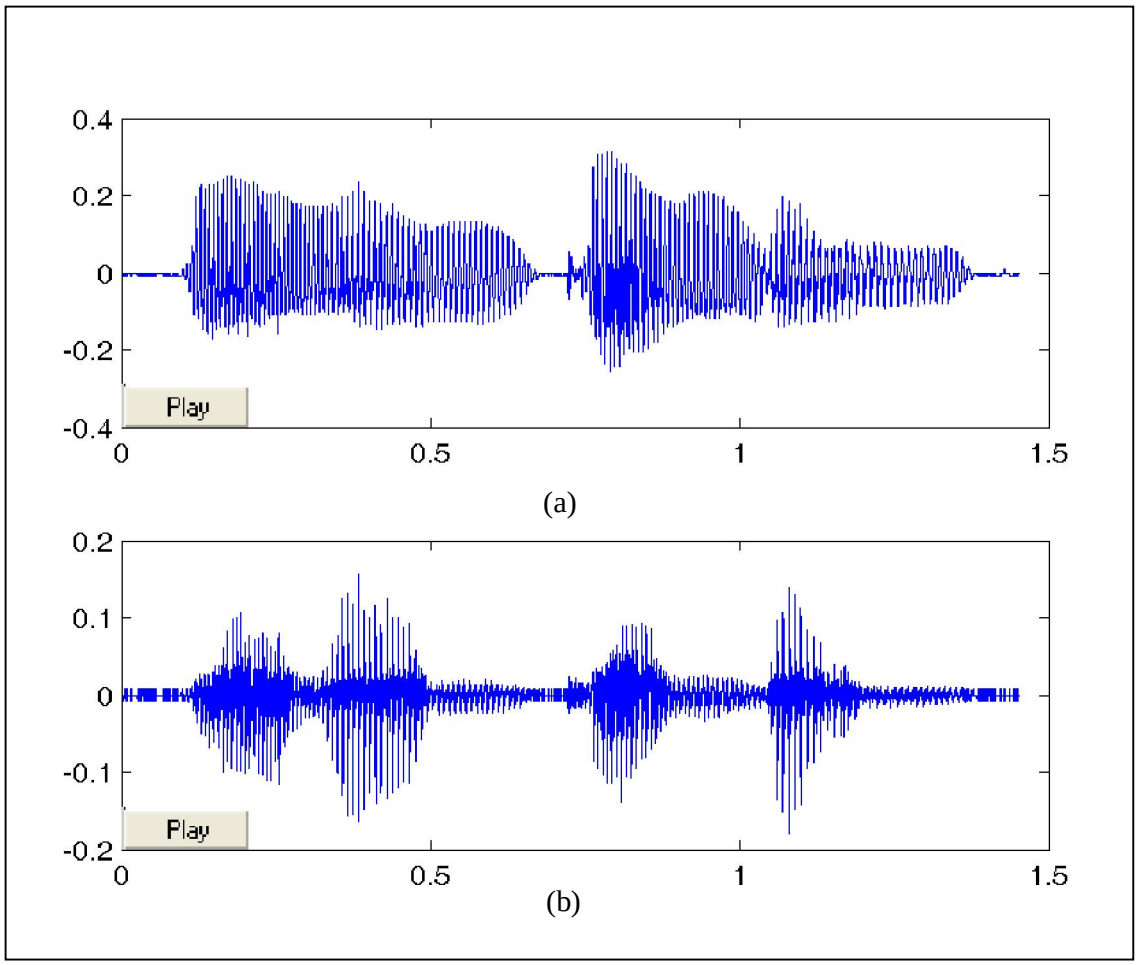


Figure 2.10: Pre-emphasizing the word 'Backward' signal (a) Original signal (b) after pre-emphasizing

Frame blocking: The input speech signal is segmented into frames of 20~30 ms with optional overlap of 1/3~1/2 of the frame size. Usually the frame size in terms of sample points is equal power of two in order to facilitate the use of FFT. If this is not the case, we need to do zero padding to the nearest length of power of

two. If the sampling rate is 16 Hz and the frame size is 320 sample points, then the frame duration is 320/16000*1000 = 20 ms. Additional, if the overlap is 160 points, then the frame rate is 16000/(320-160) = 100 frames per second.

Hamming windowing: Each frame has to be multiplied with a hamming window in order to keep the continuity of the first and the last points in the frame. If the signal in a frame is denoted by s(n), n = 0,…N-1, then the signal after Hamming windowing is s(n)*w(n), where w(n) is the Hamming window defined by:

In practice, the value of α is set to 0.46. MATLAB also provides the command hamming for generating the curve of a Hamming window. Different values of a corresponds to different curves for the Hamming windows shown in Figure 2.11.

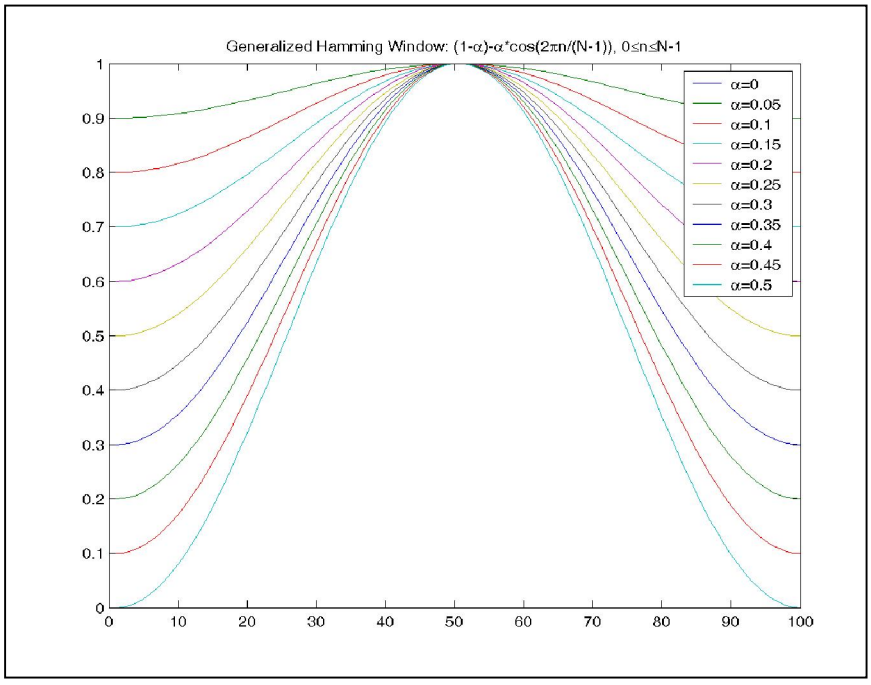


Figure 2.11: Hamming windows with different curves

Fast Fourier Transform or **FFT**: Spectral analysis shows that different timbres in speech signals corresponds to different energy distribution over frequencies. Therefore we usually perform FFT to obtain the magnitude frequency response of each frame.

When we perform FFT on a frame, we assume that the signal within a frame is periodic and continuous on its first and last points. If this is not the case, we can still perform FFT but the incontinuous at the frame's first and last points is likely to introduce undesirable effects in the frequency response. To deal with this problem, we have two strategies:

1. Multiply each frame by a Hamming window to increase its continuity at the first and last points.

2. Take a frame of a variable size such that it always contains a multiple number of the fundamental periods of the speech signal.

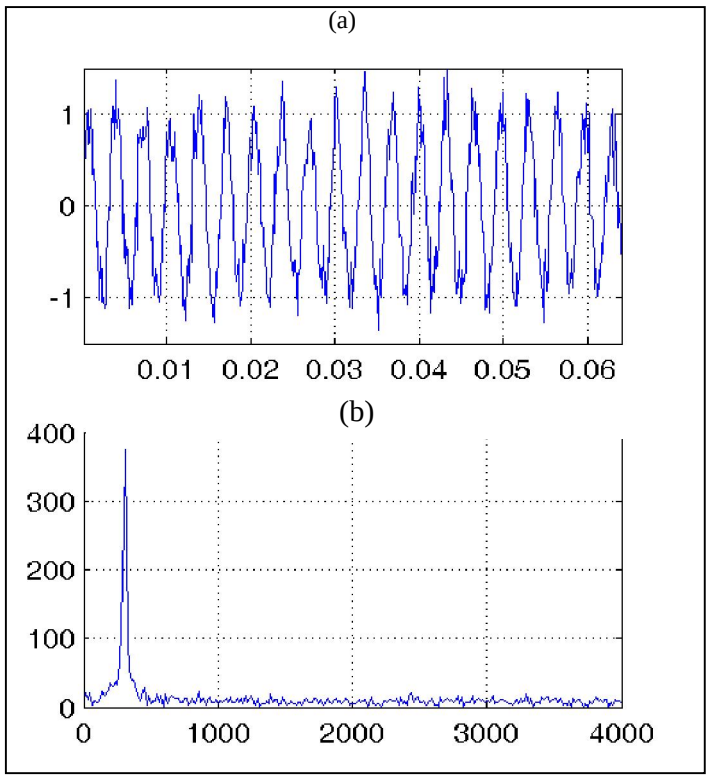


Figure 2.12: FFT of speech signal (a) Original signal (b) FFT signal

Triangular Bandpass Filters: Multiple the magnitude frequency response by a set of 20 triangular bandpass filters to get the log energy of each triangular bandpass filter. The positions of these filters are equally spaced along the Mel frequency, which is related to the common frequency by the following equation:

$$( ) = 1125 \times \quad (1 + \quad /700)$$

Mel-frequency is proportional to the logarithm of the linear frequency, indicating the human's subjective aural perception also goes linearly with the logarithm of the linear frequency. Figure 2.13 plots the relationship between the mel and the linear frequencies.

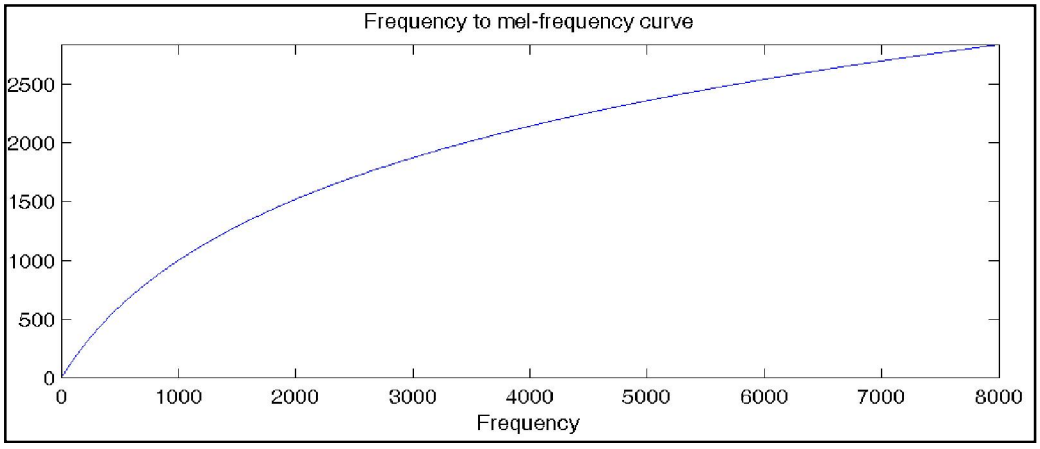


Figure 2.13: Relationship between the mel and the linear frequencies

In practice, two choices for the triangular bandpass filters, as shown in the Figure 2.14.

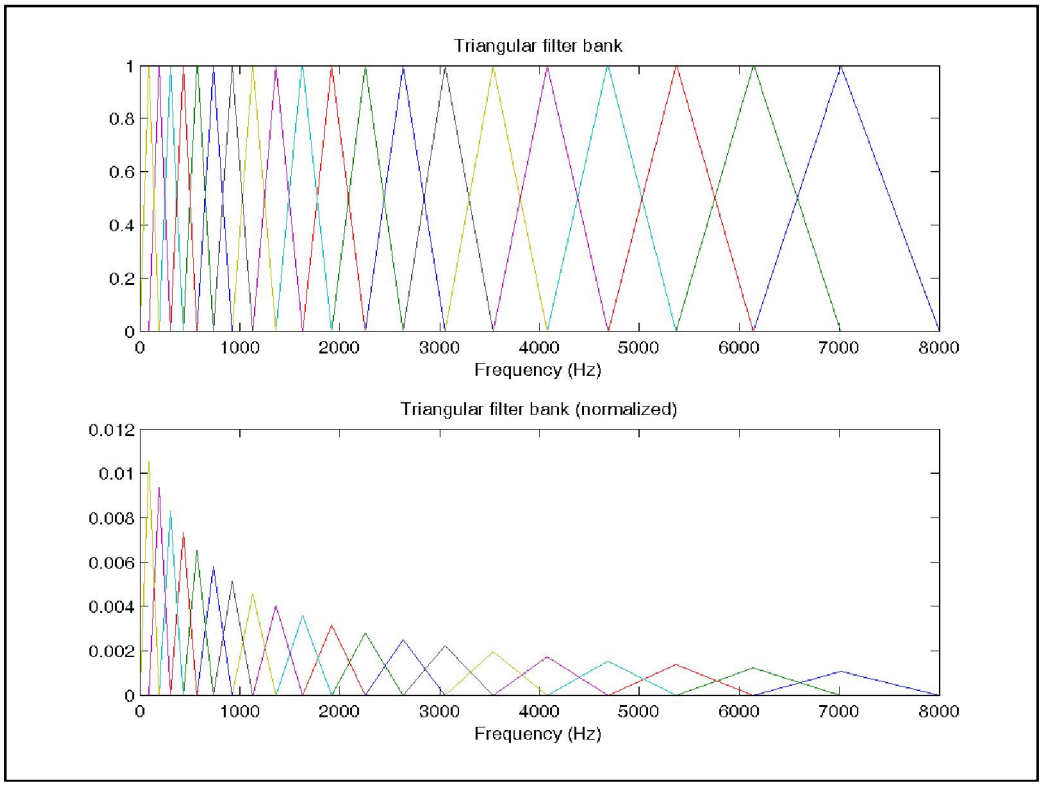


Figure 2.14: Triangular Bandpass Filters

The reasons for using triangular bandpass filters are two fold:

1. Smooth the magnitude spectrum such that the harmonics are flattened in order to reflect the original spectrum. This indicates that, theoretically, the pitch of a speech signal is not reflected in MFCC. As a result, a speech recognition system will behave more or less the same when the input utterances are of the same timbre but with different tones/pitch.
2. Reduce the size of the features involved.

Discrete cosine transform or DCT: In this step, apply DCT on the 20 log energy E_k obtained from the triangular bandpass filters to have L mel-scale cepstral coefficients. The formula for DCT is shown next.

$$= \quad = 1 \quad [\quad \times (\quad - 0.5) \times \quad / \quad] \quad , \quad = 1,2,\ldots,$$

where N is the number of triangular bandpass filters, L is the number of mel-scale cepstral coefficients. Usually we set N=20 and L=12. Since we have performed FFT, DCT transforms the frequency domain into a time-like domain called quefrency domain. The obtained features are similar to cepstrum, thus it is referred to as the mel-scale cepstral coefficients, or MFCC. MFCC alone can be used as the feature for speech recognition. For better performance, add the log energy and perform delta operation, as explained in the next two steps.

Log energy: The energy within a frame is also an important feature that can be easily obtained. Hence we usually add the log energy as the 13rd feature to MFCC. If necessary, we can add some other features at this step, including pitch, zero cross rate, high-order spectrum momentum, and so on.

Delta cepstrum: It is also advantagous to have the time derivatives of (energy+MFCC) as new features, which shows the velocity and acceleration of (energy+MFCC). The equations to compute these features are:

$$( ) = [= - (+)] / [= - 2]$$

The value of M is usually set to 2. If we add the velocity, the feature dimension is 26. If we add both the velocity and the acceleration, the feature dimension is 39. Most of the speech recognition systems on PC use these 39-dimensional features for recognition.

2.3.3 DYNAMIC TIME WARPING

The distance between two point $\mathbf{x} = [x_1, x_2, ..., x_n]$ and $\mathbf{y} = [y_1, y_2, ..., y_n]$ in a n-dimensional space can be computed via the Euclidean distance:

$$(,) = | - | = [(1 - 1)2 + (2 - 2)2 + \ldots + (-)2]1/2$$

However, if the length of **x** is different from **y**, then we cannot use the above formula to compute the distance. Instead, we need a more flexible method that can find the best mapping from elements in **x** to those in **y** in order to compute the distance.

The goal of dynamic time warping (DTW for short) is to find the best mapping with the minimum distance by the use of dynamic programming. The method is called "time warping" since both **x** and **y** are usually vectors of time series and we need to compress or expand in time in order to find the best mapping as shown in Figure 2.16. We shall give the formula for DTW in this section.

Let **t** and **r** be two vectors of lengths m and n, respectively. The goal of DTW is to find a mapping path $\{(p_1, q_1), (p_2, q_2), ..., (p_k, q_k)\}$ such that the distance on this mapping path $S_{i=1}^{k}\ t(p_i) - r(q_i)$ is minimized, with the following constraints:

1. Boundary conditions: $(p_1, q_1) = (1, 1)$, $(p_k, q_k) = (m, n)$. This is a typical example of "anchored beginning" and anchored end".
2. Local constraint: For any given node (i, j) in the path, the possible fan-in nodes are restricted to (i-1, j), (i, j-1), (i-1, j-1). This local constraint guarantees that the mapping path is monotonically non-decreasing in its first and second arguments. Moreover, for any given element in **t**, we should be able to find at least one corresponding element in **r**, and vice versa.

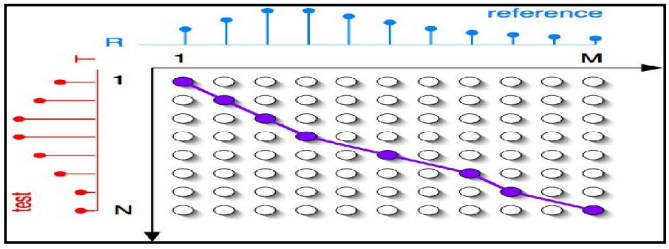


Figure 2.15: Mapping the Dynamic Time Warping

CHAPTER 3

GRAPHICAL USER INTERFACE DESIGN

3.1 Introduction

This chapter will discuss about developing a software for this speech recognizer using MATLAB. A lot of functions provided by MATLAB will be implemented in building this speech recognizer. The software will be a GUI that is being programmed with MATLAB functions which can run a speech recognition system and recognize the word spoken. For this project, it is compulsory to fully understand the way MATLAB works and the language that it use. Knowledge on applying and manipulating functions provided by MATLAB is totally a vital skill to ensure the succes of this project. There are two parts that will be the main concern in this chapter, which are creating the GUI and programming the GUI so that it will work as a speech recognizer.

3.2 Graphical user interface (GUI) with MATLAB

This section will describe and elaborate the details on how to create GUI using the MATLAB. The objective of this section is to give a clear picture on how MATLAB can be used to create a beautiful yet powerful GUI that can ease the user to perform the speech recognition without repeating writing the same programming just to get a result with different inputs. In MATLAB, the GUI can be created

programmatically or using the graphical user interface development environment (GUIDE). Both way have its own advantages and disadvantage but for learning purpose, both methods will be described and explained spesifically.

For this speech recognizer project, there are two GUIs are created which is each GUI will have its own unique function. It is the GUI for recording the voice and GUI for recognize the word spoken. The GUI for recording voice will be created programmatically while for GUI to recognize the word spoken, it is done using the GUIDE.

3.2.1 Creating a GUI programmatically

There are seven objectives need to be fulfilled for this section which are:

1. Describes the GUI to be constructed.
2. Lists the functions that are used in the construction of the GUI
3. Creates the M-file that holds the GUI script and adds help comments to the file.
4. Laying out the GUI by creating the figure and adding the components.
5. Initialize the GUI by performs various initialization chores and generates the data to plot.
6. Programming the GUI by adding code for each component to the GUI M-file to make the GUI work.
7. Runs the final GUI and demonstrates how the components work together.

3.2.1.1 Describing the GUI

Before creating the GUI, it is best to have a clear view on how the GUI will looks alike. It is a good practice to draw a block diagram of the operations that will be involved for this GUI. The block diagrams is shown in Figure 3.1:

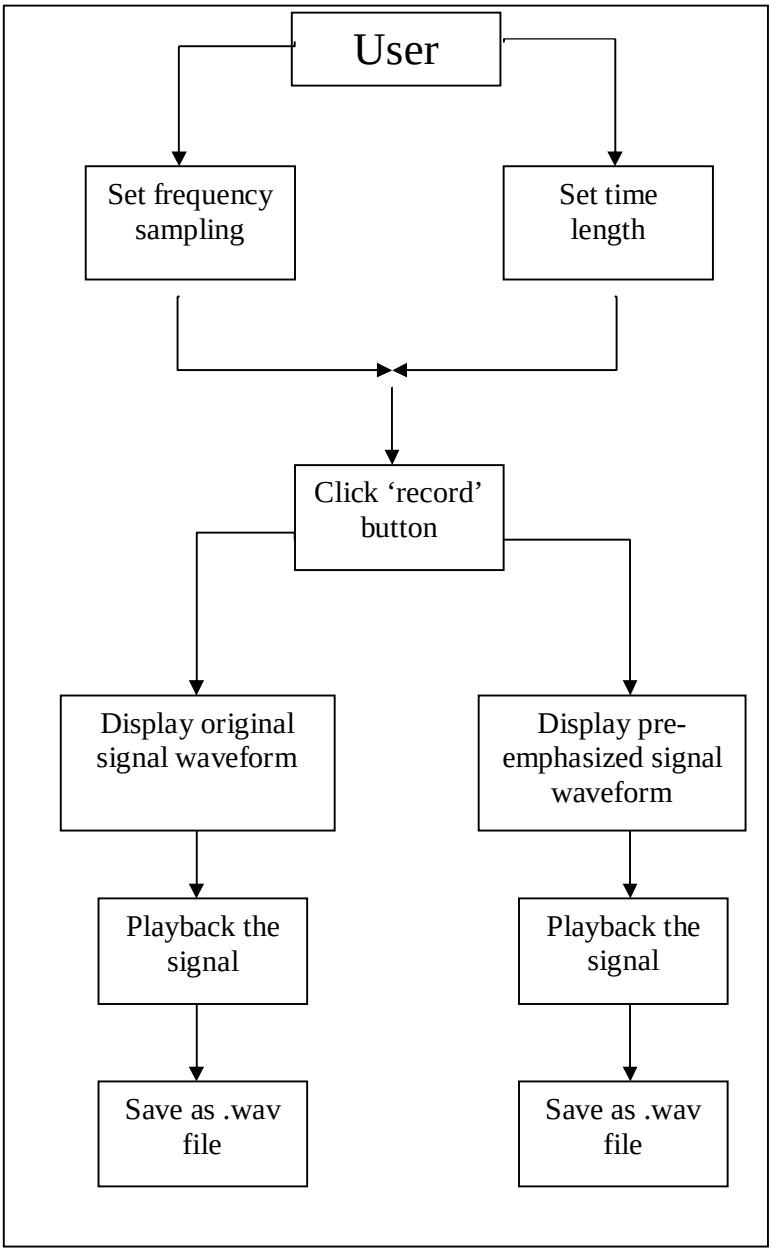


Figure 3.1 : Block diagram of basic operation for recording voice

Block diagram above can give a hint on how to construt the GUI and the elements that are necessary in the GUI. The block diagram above shows that the GUI will contains the record button, a panel to set frequency sampling and time length, two axes that will display two graphs, two play buttons to playback the recorded signal and two save buttons to save the recorded sognal in wave file. Based on this informations, it is always a good start to skctch the looks of the GUI as a guidance as shown in the Figure 3.2.

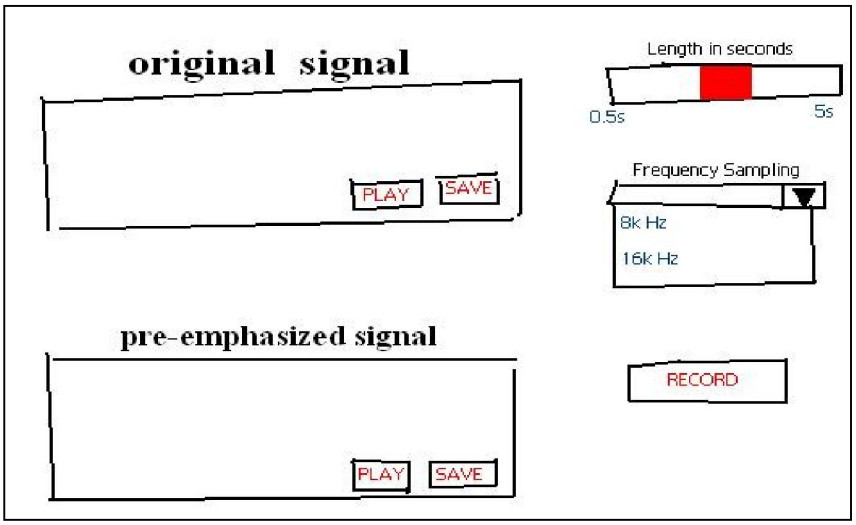


Figure 3.2 : Sketching a GUI

For this GUI, a slider will be used to set the time length, popupmenu listing two different values of frequency sampling and push button for record, play and save button.

3.2.1.2 Functions summary

Table 3.1 below shows the summary of the functions that are used in the construction of this GUI for recording voice using MATLAB.

\\\

Table 3.1 : List of functions to create the GUI

FUNCTION	DESCRIPTIONS
`align`	Align GUI components such as user interface controls and axes.
`axes`	Create axes objects.
`figure`	Create figure objects. A GUI is a figure object.
`movegui`	Move GUI figure to specified location on screen.
`uicontrol`	Create user interface control objects, such as push buttons, static text, and pop-up menus.

3.2.1.3 Creating the GUI M-file

Creating the GUI programmatically will be done using the M-file eitor. The programming is wrote at the M-file editor provided by MATLAB. Hence, it is very important to create a M-file for creating this recording GUI for MATLAb to execute it. To open the M-file, type `edit` at the MATLAB command window.

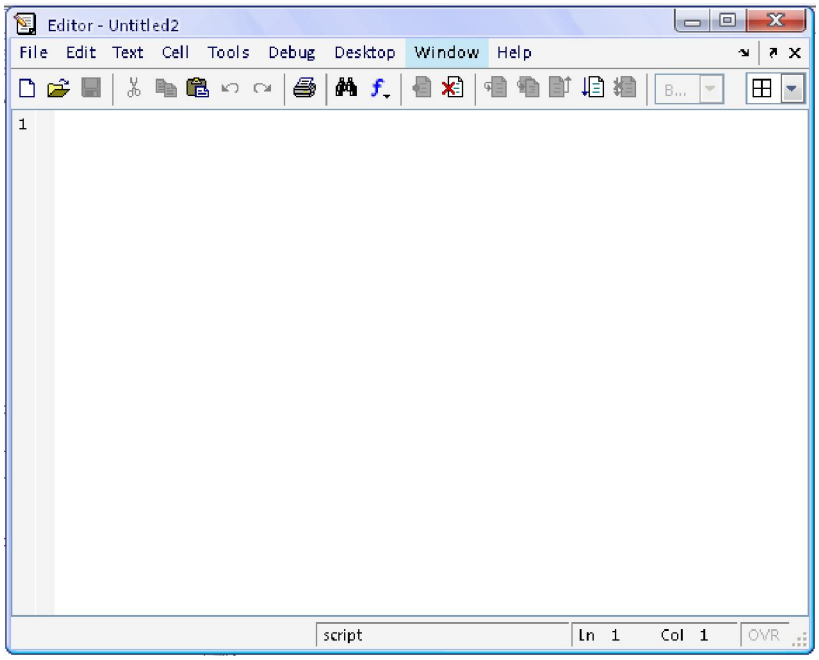


Figure 3.3 : M-file editor

The GUI should be a function so that MATLAB can execute it when the user want to use this GUI. For that purpose, command `function` will be used to make the GUI as a function. For example, command `function recordtool` can be written to tell MATLAB new function is added.Now, the GUI has become a stand-alone executable function that is directly callable from the system command line. 'recordtool' is the name of the GUI and it is up to the user to name it. But since this GUI is create to record the voice, so it will be named as recordtool. To complete this function, an end statement must be added so that the function will work without error. The end statement is needed because the GUI is written using nested functions. End statement is written at the end of the line and a few blank space below the function statement is leaved before adding the end statement. The file now looks like this:

```
function recordtool


end
```

Figure 3.4: Command function

3.2.1.4 Laying out the GUI

In MATLAB, a GUI is a figure. Hence, the first step to create the GUI is to draw the figure and positions it on the screen. To draw the figure, command `figure` will be used. The command `figure` creates figure graphics objects. Figure objects are the individual windows on the screen in which MATLAB displays graphical output. The property of the figure can be define explicitly as the input argument when drawing the figure. The properties are such as its color, size of the figure and the name of the figure. For this recording GUI, the figure will have

the size of 950x600 pixels and is grey in color. The program is written in the M-file that has been created before as shown in Figure 3.5.

```
function recordtool
    width = 950;
    height = 600;

    record_figure = figure('Position',[30 55 width
         height],...
         'NumberTitle','off',...
         'Color',[.8 .8 .8],...
         'Name','recordtool');
end
```

Figure 3.5: Creating figure

The position property specifies the size and the location of the GUI on the screen. In the position property as shown above, the value 30 is the distance of the GUI from left and the value 55 is the distance from bottom. Default units are pixels.

```
Position: [distancefromleft distancefrombottom width
height]
```

Figure 3.6 shows the result generated by the command above and the effect of manipulating its properties.

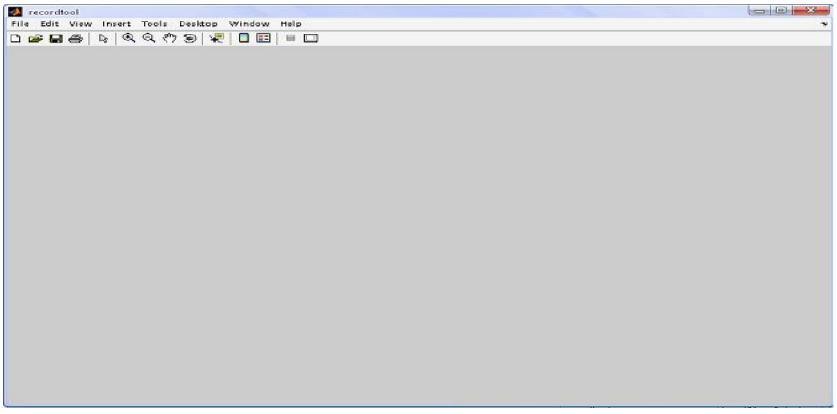


Figure 3.6 : Physical view of figure

This GUI for recording voice has nine components; five push buttons, two axes, one pop-up menu, one slider. Before adding all of this component, it is compulsory to understand the command `uicontrol` since all of the components will be created based on the command `uicontrol` except for the axes.

The command `uicontrol` creates a uicontrol graphics objects (user interface controls), which you use to implement graphical user interfaces. The syntax is:

handle = uicontrol('PropertyName',PropertyValue,...)

handle = uicontrol('PropertyName',PropertyValue,...) creates a uicontrol and assigns the specified properties and values to it. It assigns the default values to any properties you do not specify. The default uicontrol style is a pushbutton. The default parent is the current figure.

MATLAB supports numerous styles of uicontrols, each suited for a different purpose:

1. Check boxes
2. Editable text fields
3. Frames
4. List boxes
5. Pop-up menus
6. Push buttons
7. Radio buttons
8. Sliders
9. Static text
10. labels
11. Toggle buttons

To add the axes, command `axes` is used which will create the axes graphic object. The syntax is:

axes('PropertyName',PropertyValue,...)

where axes('PropertyName',PropertyValue,...) creates an axes object having the specified property values. MATLAB uses default values for any properties that do not explicitly define as arguments.

Statement below is written to add the two axes for this GUI. The program is written in the M-file for this GUI is shown in the Figure 3.7.

```
original_axes = axes('Units', 'normalized',...
     'Position', [0.1 0.57 0.75 0.37],...
     'Xgrid', 'on',...
     'Ygrid', 'on',...
     'Xminortick', 'on');

 preemphasis_axes = axes('Units', 'normalized',...
     'Position', [0.1 0.07 0.75 0.37],...
     'Xgrid', 'on',...
     'Ygrid', 'on',...
     'Xminortick', 'on');
```

Figure 3.7: Statement to add axes

The grid of the axes is controlled by the property argument ‘Xgrid’ for grid at the X axes and ‘Ygrid’ at the Y axes. Argument ‘on’ at the grid properties means that the grid is visible. Figure 3.8 shows the result of this statement.

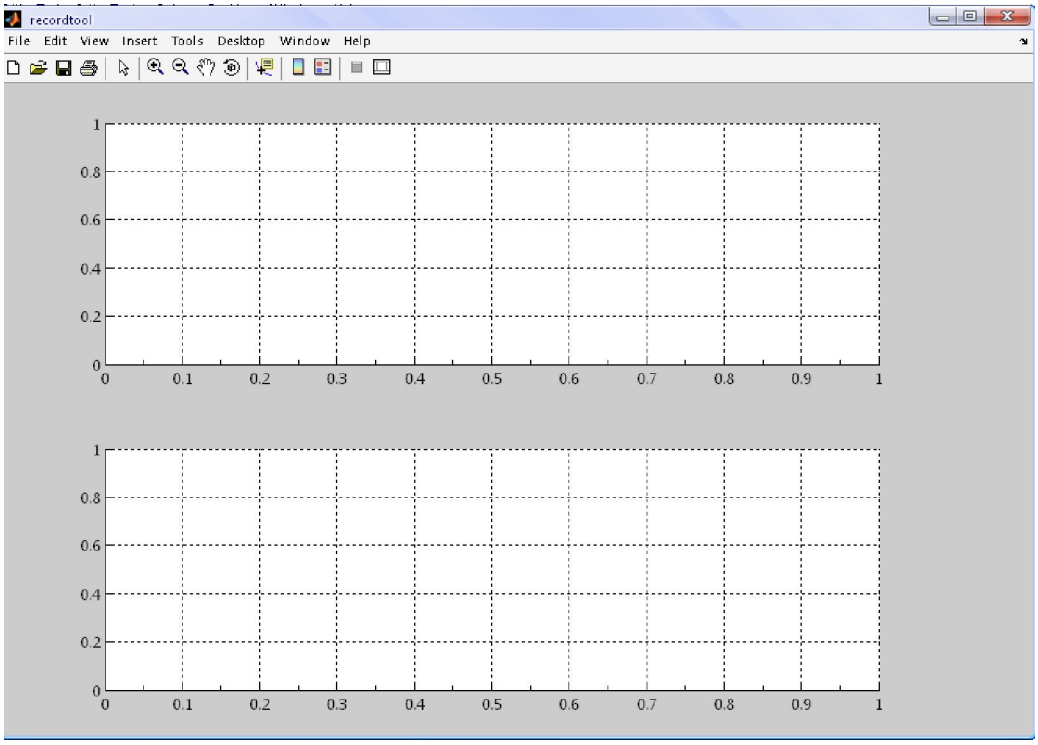


Figure 3.8 : Drawing the axes

'RECORD' push button can be added by writing a statement as shown in Figure 3.9.

```
record_button=uicontrol('Style','pushbutton'
      'Units','normalized',...
      'Position',[850/width 380/height 80/width
       30/height],...
      'ForegroundColor',[0 0.3 0],...
      'FontWeight','bold',...
      'String','RECORD',...
      'Visible','on',...
      'CallBack','recordtool(1)');
```

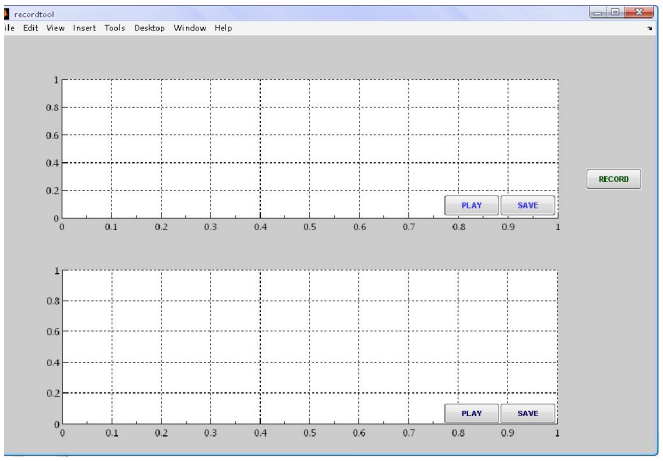
Figure 3.9: Statement to add push button

These statements use the uicontrol function to create the 'record' push button. Each statement uses a series of property/value pairs to define the push button. Table 2 explain more about the property and its description.

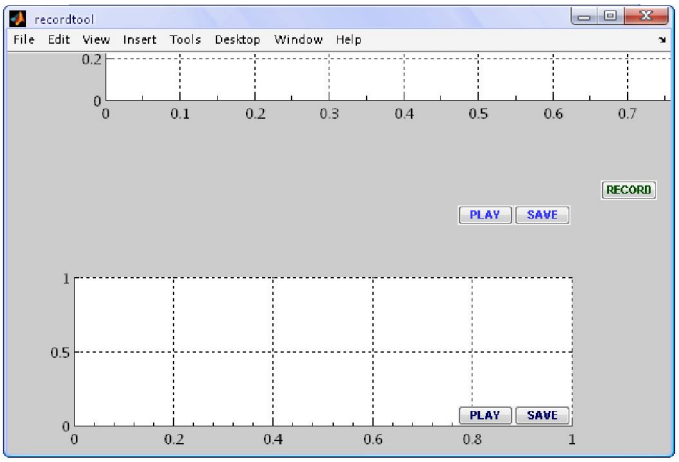
Table 3.2 : Series of property/value pairs

Property Name	**Property Description**	**Property Value**
`Style`	Type of uicontrol object	Value:pushbutton, togglebutton, radiobutton, checkbox, edit, text, slider, listbox, popupmenu Default: pushbutton
`Units`	Units to interpret position vector	Value: pixels, normalized, inches, centimeters, points, characters Default: pixels
`Position`	Size and location of uicontrol	Specify Position as [left bottom width height]
`ForegroundColor`	Color of text	Value: ColorSpec Default: [0 0 0]
`FontWeight`	Weight of text characters	Value: light, normal, demi, bold Default: normal
`String`	For check boxes, editable text, push buttons, radio buttons, static text, and toggle buttons, the text displayed on the object. For list boxes and pop-up menus, the set of entries or items displayed in the object.	Value: string
`Visible`	Uicontrol visibility	Value: on, off Default: on
`Callback`	Control action. A routine that executes whenever you activate the uicontrol object (e.g., when you click on a push button or move a slider).	Value: string or function handle

The value for the position property is divided with the value of the figure size. This is because the units for this push button is normalized. Normalized units causes the components to resize when the GUI is resized. Apply the same statement to create the 'play' and 'save' push button but change the string to 'PLAY" for play push button and 'SAVE' for save push button. The position also must be change to desire location. Figure 3.10 shows the different for using or not using normalized units.



(a)



(b)

Figure 3.10: Effects of normalized unit when resize GUI (a)Units for both axes are normalized Units for upper axes is not normalized

Sliders accept numeric input within a specific range by enabling the user to move a sliding bar. Users move the bar by pressing the mouse button and dragging the pointer over the bar, or by clicking in the trough or on an arrow. The location of the bar indicates a numeric value, which is selected by releasing the mouse button. The minimum, maximum, and current values of the slider can be set. Figure 3.11 shows the statement to add slider in the GUI.

```
time_slider = uicontrol('Style','Slider',...
      'Units','normalized', ...
      'Position',[830/width 530/height 100/width
       20/height],...
      'Min',0.5,'Max',5,'Value',1,...
      'SliderStep',[1/9  1/9],...
      'Callback','recordtool(7)');
```

Figure 3.11: Adding slider to GUI

By setting the value equal to 1, this means the default value for the slider when appear at the GUI is 1. Slidersteps means how many steps tha tare required for the minimum value to reach he maximum value. For this slider, it is create to represent the time length to record the voice and the minimum value is 0.5 seconds and it takes nine steps to reach the maximum value that is 5 seconds. However, the statement above will not display the value of the slider. For that purpose, static text will be used as shown in Figure 3.12. Figure 3.13 shows the result for the statement.

```
slider_text = uicontrol('Style','text',...
'Units','normalized', ...
      'Position',[830/width 555/height 120/width
15/height],...
      'BackgroundColor',[.8 .8 .8],...
      'FontWeight','bold',...
      'String','Length - 1 sec');
```

Figure 3.12: Indicate slider with static text

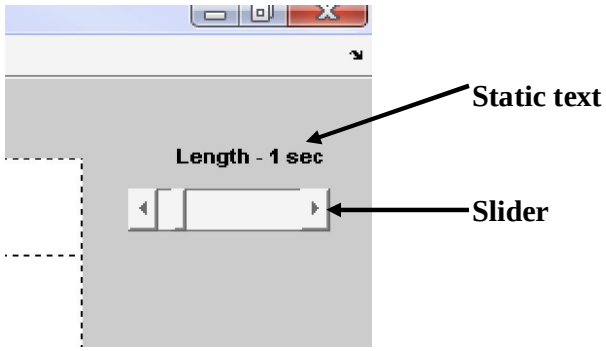


Figure 3.13 : Slider

Pop-up menus open to display a list of choices (defined using the String property) when pressed. When not open, a pop-up menu indicates the current choice. Pop-up menus are useful when you want to provide users with a number of mutually exclusive choices, but do not want to take up the amount of space that a series of radio buttons requires. A value must be specified for the String property as shown in Figure 3.14.

```
popupfreq_button = uicontrol('Style','popupmenu',...
     'Units','normalized',...
     'Position',[830/width 470/height 100/width
      30/height],...
     'ForegroundColor',[0 0 0],...
     'FontWeight','bold',...
     'String',{'8kHz','16kHz'},...
     'Visible','on');
```

Figure 3.14: Statement for adding pop-up menu

This pop-up menu will act as the frequency selector to the user. User can select frequency sampling from this pop-up menu. Two value of frequency sampling are provided which are 8k Hz and 16k Hz. To make the pop-up menu to list both value, the string property is set to 8k Hz and 16k Hz . The first string will be the default value. See Figure 3.15

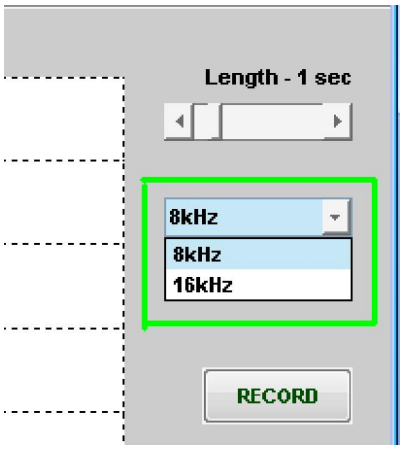


Figure 3.15: Pop-up menu

In order for the GUI to appear at the centre of the screen, the command function `movegui` can be used. The syntax is as follow:

```
movegui(h,'position')
```

`movegui(h,'position')` moves the figure identified by handle h to the specified screen location, preserving the figure's size. The position argument can be any of the following strings: north - top center edge of screen south - bottom center edge of screen east - right center edge of screen west - left center edge of screen northeast - top right corner of screen northwest - top left corner of screen southeast - bottom right corner of screen southwest - bottom left corner center - center of screen onscreen - nearest location with respect to current location that is on screen.

After adding all the components in the GUI, it is time to run the script written at the M-file editor. The script must be saved first and it should be saved at a specific folder with a specific location. To run the script, simply type the name of the GUI that have been turn into function which is `recordtool` at the command window of the MATLAB. Make sure the current directory is specified at the location which the script has been saved. Figure 3.16 below shows the final result of the GUI after adding all the components.

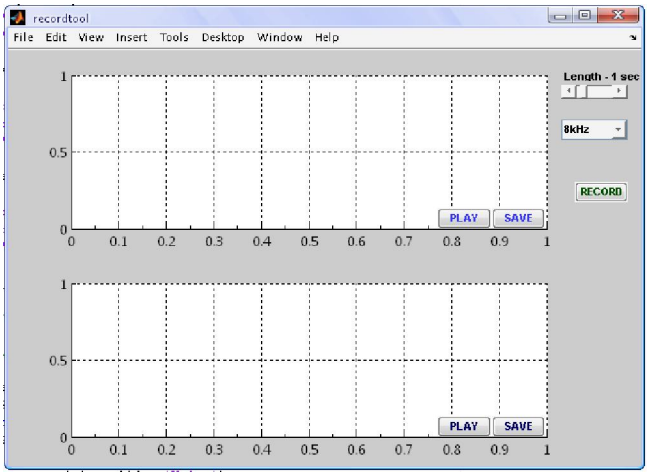


Figure 3.16 : GUI for recording voice

Note that eventhough the GUI has been created but nothing will happen if the user push the push buttonnor select the data set in the pop-up menu. This is because there is no code in the M-file to service the pop-up menu.sliders and the push button. Programming the GUI will be discussed at the next chapter.

3.3 Create GUI using GUIDE

This section shows how to use GUIDE to create the graphical user interface (GUI). GUIDE, the MATLAB graphical user interface development environment, provides a set of tools for creating graphical user interfaces (GUIs). These tools greatly simplify the process of designing and building GUIs. With GUIDE, drawing the figure, adding the components no longer require a hectic or labourous programming but simply drag and drop the components at the layout editor. For this project, GUIDE will be used to create the GUI for the speech recognizer. The GUI for the speech recognizer is shown in Figure 3.17.

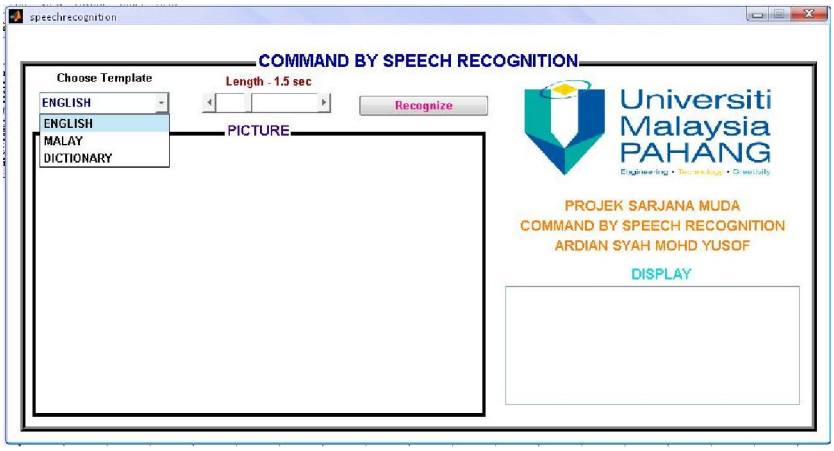


Figure 3.17 : GUI for speech recognizer

The GUI contains push button to run the recognition job, slider for setting the time length and pop-up menu listing three different data sets that correspond to the behaviour of the speech recognizer. The pop-up menu will list three templates for the speech recognizer to recognizer which are english template, malay template and dictionary that will display a picture. From the figure above, the GUI will have a panel to display the word that have been recognized and picture.

3.3.1 GUIDE overview

Before using the GUIDE, it is wise to know the basic things about GUIDE. GUIDE, the MATLAB graphical user interface development environment, provides a set of tools for creating graphical user interfaces (GUIs). These tools simplify the process of laying out and programming GUIs.

Using the GUIDE Layout Editor, you can populate a GUI by clicking and dragging GUI components—such as axes, panels, buttons, text fields, sliders, and so on—into the layout area. From the Layout Editor, GUI can be modified for the component look and feel, align components, set tab order, view a hierarchical list of the component objects, and set GUI options. Figure 3.18 shows the layout editor of the GUIDE and table 3.3 describe the tools in the GUIDE

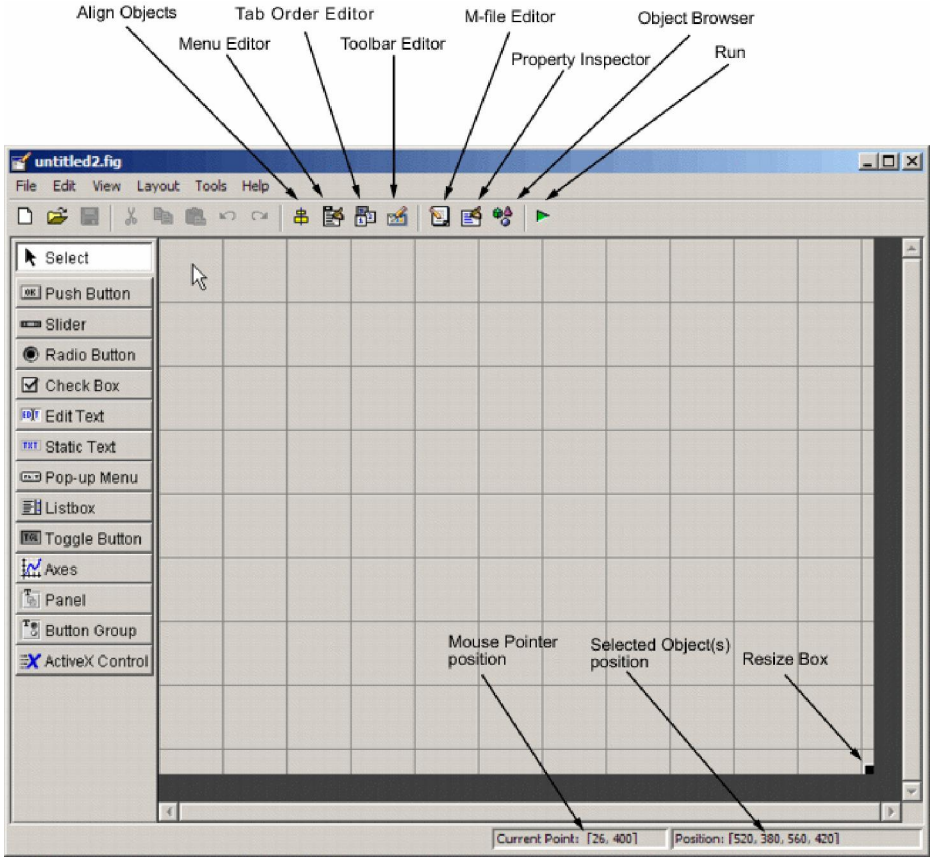


Figure 3.18: GUIDE Layout Editor

Table 3.3 : Tools of layout editor

TOOL	**DESCRIPTIONS**
Layout Editor	Select components from the component palette, at the left side of the Layout Editor, and arrange them in the layout area.
Figure Resize Tab	set the size at which the GUI is initially displayed when you run it.
Menu Editor	Create menus and context, i.e., pop-up, menus.
Align Objects	Align and distribute groups of components. Grids and rulers also enable you to align components on a grid with an optional snap-to-grid capability.
Tab Order Editor	Set the tab and stacking order of the components in your layout.
Toolbar Editor	Create Toolbars containing predefined and custom push buttons and toggle buttons.
Icon Editor	Create and modify icons for tools in a toolbar.
Property Inspector	Set the properties of the components in your layout. It provides a list of all the properties you can set and displays their current values.
Object Browser	Display a hierarchical list of the objects in the GUI.
Run	Save and run the current GUI.
M-File Editor	Display, in your default editor, the M-file associated with the GUI.
Position Readouts	Continuously display the mouse cursor position and the positions of selected objects.

3.3.2 Laying out GUI using GUIDE

To layout the GUI for speech recognizer as shown in Figure 3.19, start the GUIDE from MATLAB. There are many ways to start GUIDE. GUIDE can be started from the:

1. Command line by typing guide
2. **Start** menu by selecting **MATLAB** > **GUIDE (GUI Builder)**
3. **MATLAB File** menu by selecting **New** > **GUI**
4. MATLAB toolbar by clicking the **GUIDE** button 

When starting GUIDE, it displays the GUIDE Quick Start dialog box shown in the following figure.

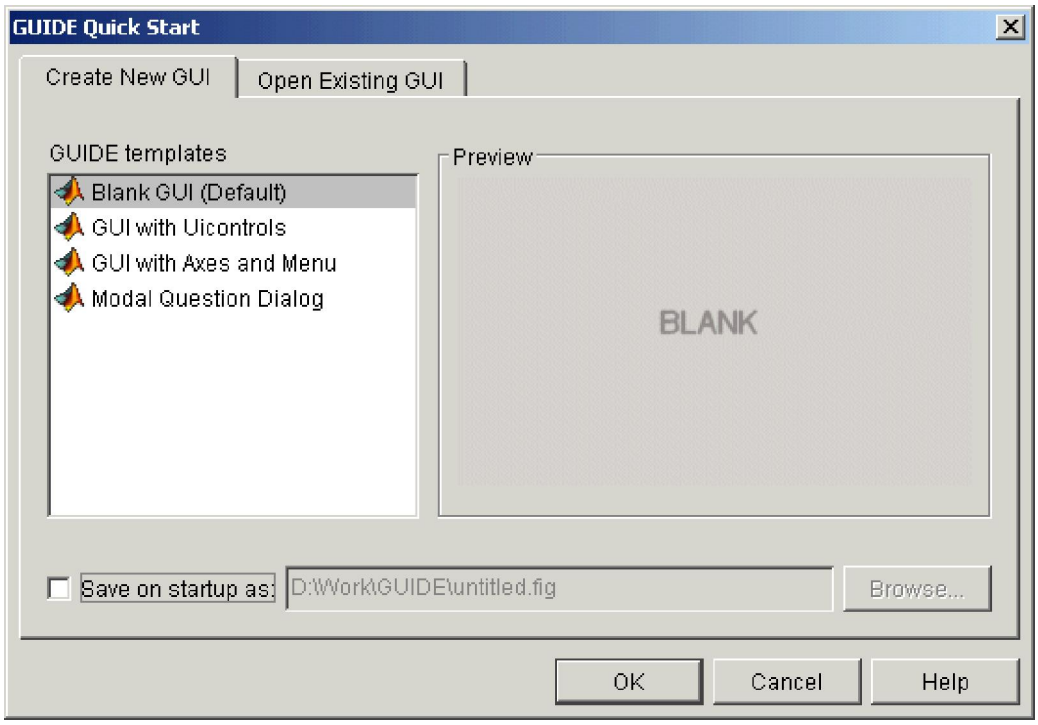


Figure 3.19: GUIDE Quick Start Dialog Box

For this project, blank GUI is selected. The blank GUI template displayed in the Layout Editor is shown in Figure 3.20.

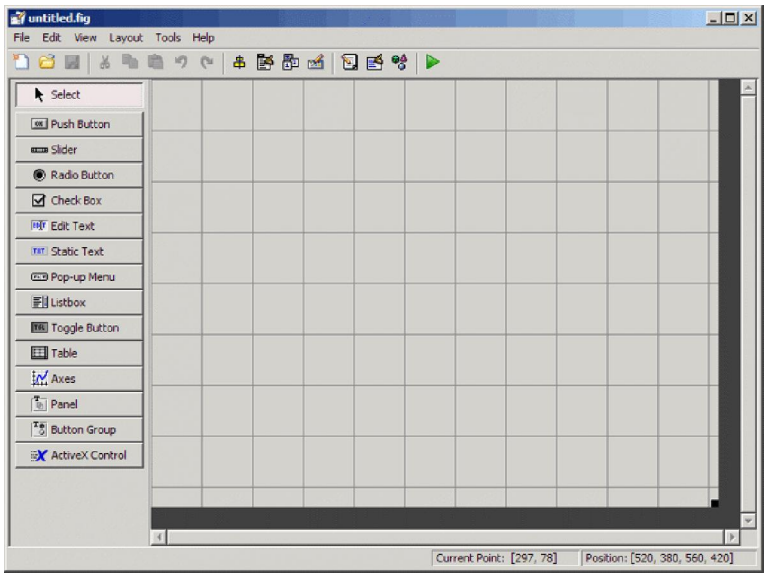


Figure 3.20: GUIDE Blank Template

Size of the GUI can be resize by resizing the grid area in the layout editor. Just click the lower-right corner and drag it until the desire size is accomplish.

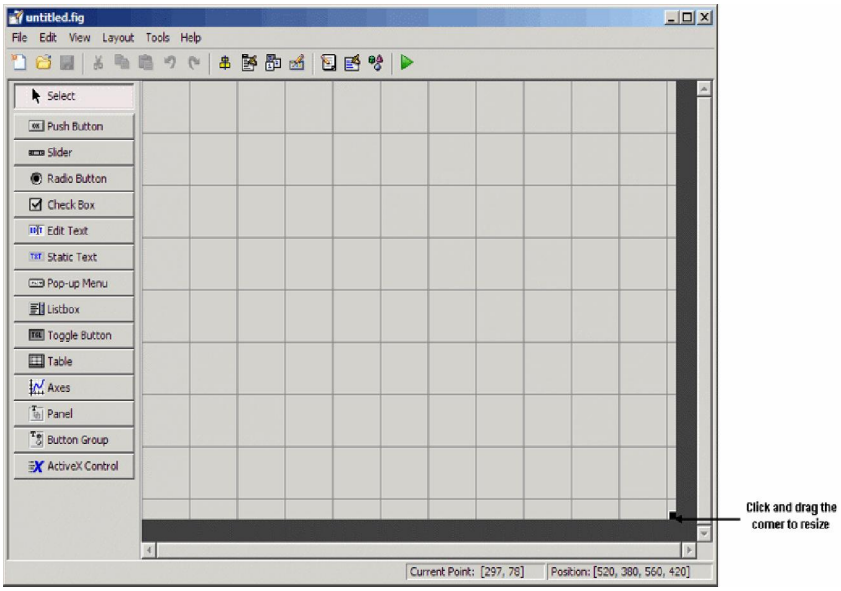


Figure 3.21: Resizing the GUI

3.3.2.1 Adding the components

With GUIDE, it is easy to add the components. To add components, just click the icon at the components palette and drag it to the layout editor. Figure 3.22 shows the components palette.

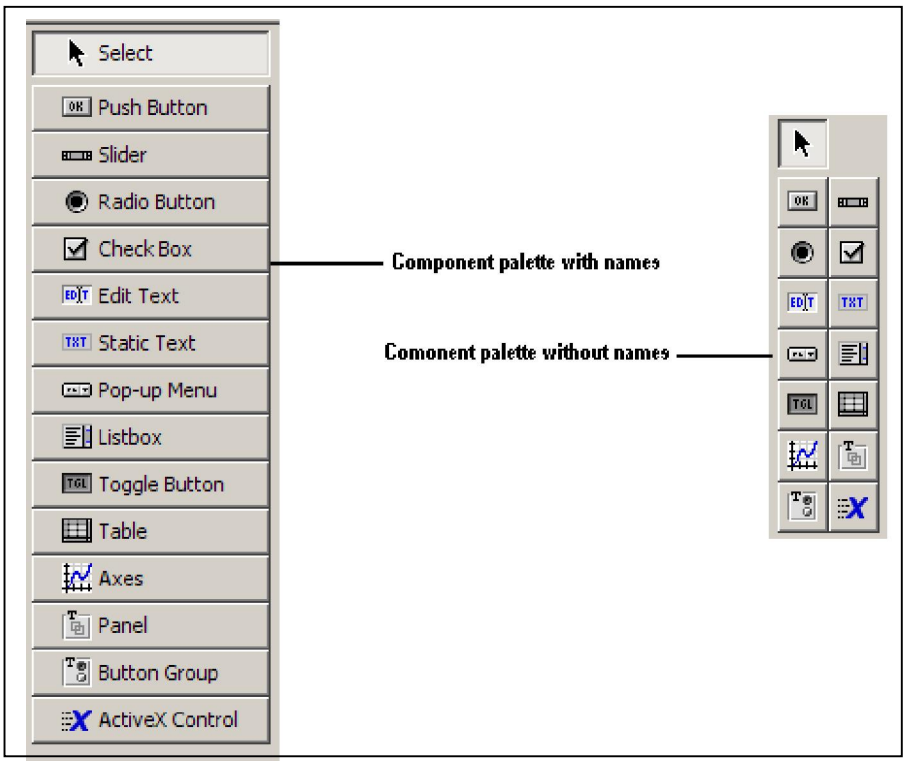


Figure 3.22: Components Palette

When open the Layout Editor, the component palette contains only icons. To display the names of the GUI components, select **Preferences** from the **File** menu, check the box next to **Show names in component palette**, and click **OK**. Table 3.5 explain the function of each components in the components palette.

Table 3.5: Components Description

COMPONENT	ICON	DESCRIPTION
Push Button		Push buttons generate an action when clicked. For example, an **OK** button might apply settings and close a dialog box. When you click a push button, it appears depressed; when you release the mouse button, the push button appears raised.
Slider		Sliders accept numeric input within a specified range by enabling the user to move a sliding bar, which is called a slider or thumb. Users move the slider by clicking the slider and dragging it, by clicking in the trough, or by clicking an arrow. The location of the slider indicates the relative location within the specified range.
Radio Button		Radio buttons are similar to check boxes, but radio buttons are typically mutually exclusive within a group of related radio buttons. That is, when you select one button the previously selected button is deselected. To activate a radio button, click the mouse button on the object. The display indicates the state of the button. Use a button group to manage mutually exclusive radio buttons.
Check Box		Check boxes can generate an action when checked and indicate their state as checked or not checked. Check boxes are useful when providing the user with a number of independent choices, for example, displaying a toolbar.
Edit Text		Edit text components are fields that enable users to enter or modify text strings. Use edit text when you want text as input. Users can enter numbers but you must convert them to their numeric equivalents.
Static Text		Static text controls display lines of text. Static text is typically used to label other controls, provide directions to the user, or indicate values associated with a slider. Users cannot change static text interactively.
Pop-Up Menu		Pop-up menus open to display a list of choices when users click the arrow.
List Box		List boxes display a list of items and enable users to select one or more items.
Toggle Button		Toggle buttons generate an action and indicate whether they are turned on or off. When you click a toggle button, it appears depressed, showing that it is on. When you release the mouse button, the toggle button remains depressed until you click it a second time. When you do so, the button returns to the raised state, showing that it is off. Use a button group to manage mutually exclusive toggle buttons.
Table		Use the table button to create a table component.
Axes		Axes enable your GUI to display graphics such as graphs and images. Like all graphics objects, axes have properties that you can set to control many aspects of its behavior and appearance.
Panel		Panels arrange GUI components into groups. By visually grouping related controls, panels can make the user interface easier to understand. A panel can have a title and various borders.Panel children can be user interface controls and axes as well as button groups and other panels. The position of each component within a panel is interpreted relative to the panel. If you move the panel, its children move with it and maintain their positions on the panel.
Button Group		Button groups are like panels but are used to manage exclusive selection behavior for radio buttons and toggle buttons.

To change its property, simply double click the push button and a new window will appear as shown in Figure 3.23. The new window is the Property Inspector and a lot of properties of the components can be changed here such as the string, tag, callback, font color and etc.

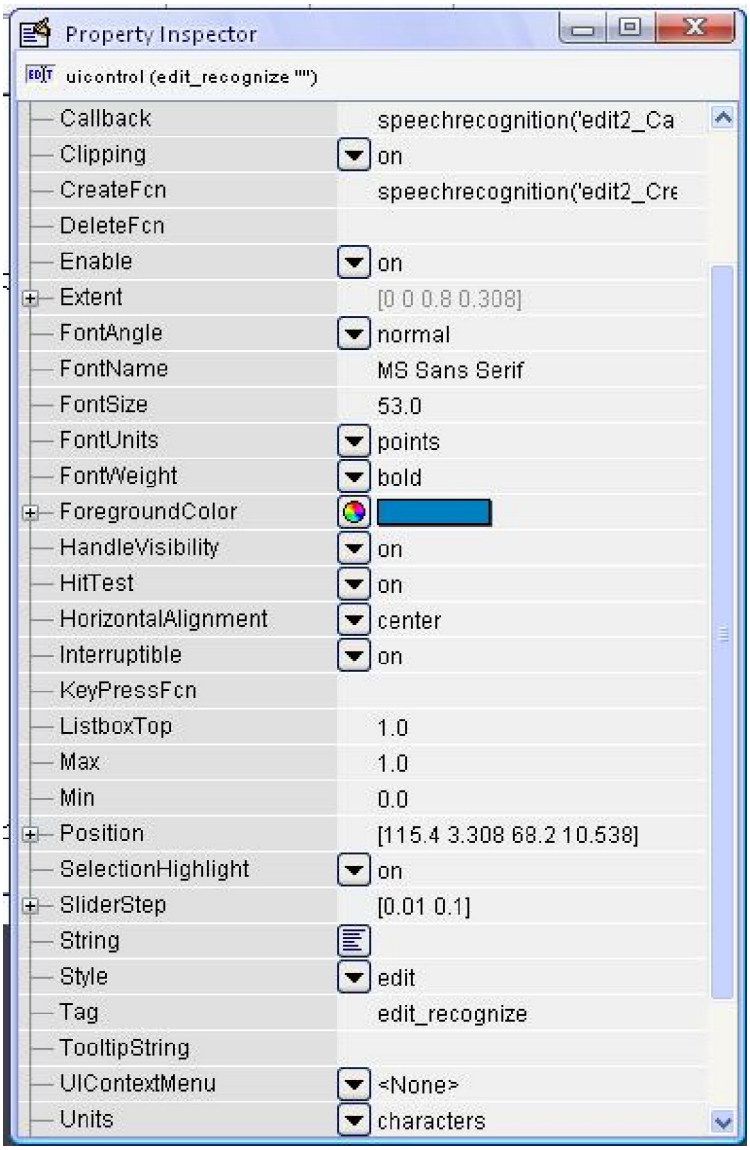


Figure 3.23: Property Inspector

Figure 3.22 shows the GUI for speech recognizer in the layout editor. The GUI has been created by drag and drop the components from the component palette and no programming required. The components in the layout editor are not active.

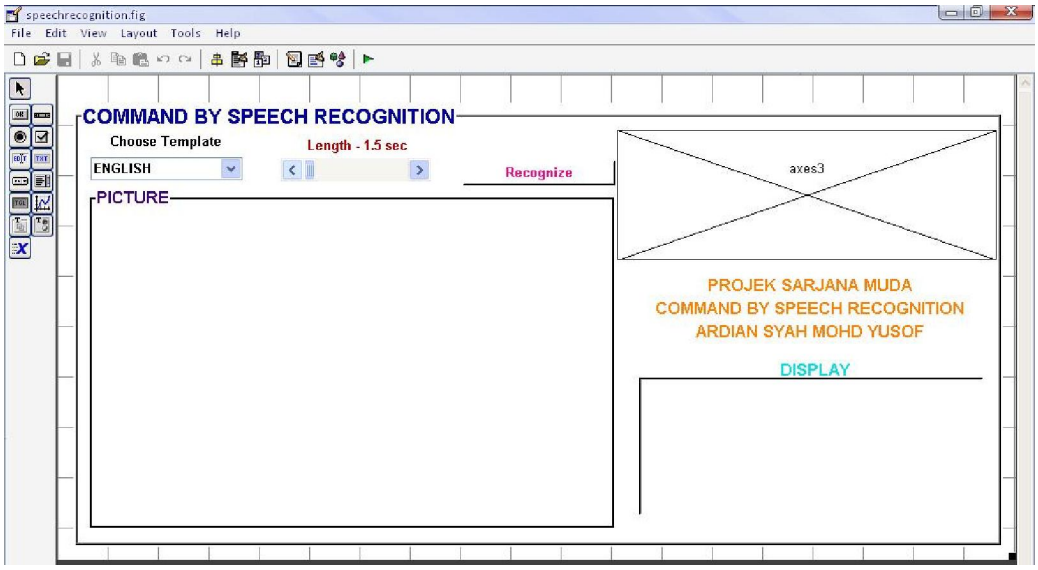


Figure 3.24: GUI for Speech Recognizer in Layout Editor

All of the components properties have been changed using the Property Inspector such as the background color, string, Font weight, font color and the size of the font. For this GUI, editable text is used to display the word that are going to be recognized. Two axes are used to display the UMP logo and picture regarding for dictionary template. To run the GUI, simply click the RUN icon and figure as shown in figure 3.10 will appear. However, this GUI will not act as the speech recognizer since all the components don't have the coding in the M-file to service them specifically. Programming this GUI so that it can react as a speech recognizer will be discussed at the next chapter.

When running or saving this GUI or any GUI created using GUIDE, MATLAB will automatically create an M-file for the GUI. In the M-file, MATLAB already assign all the components it's callback and functions. To run the GUI outside the layout editor, just type the GUI name tha tyou have been saved at the

command window but make sure the current directory is at the same path with the saved file. For this GUI, just type ‘speechrecognition’ at the command window and the GUI will appear as shown in Figure 3.25

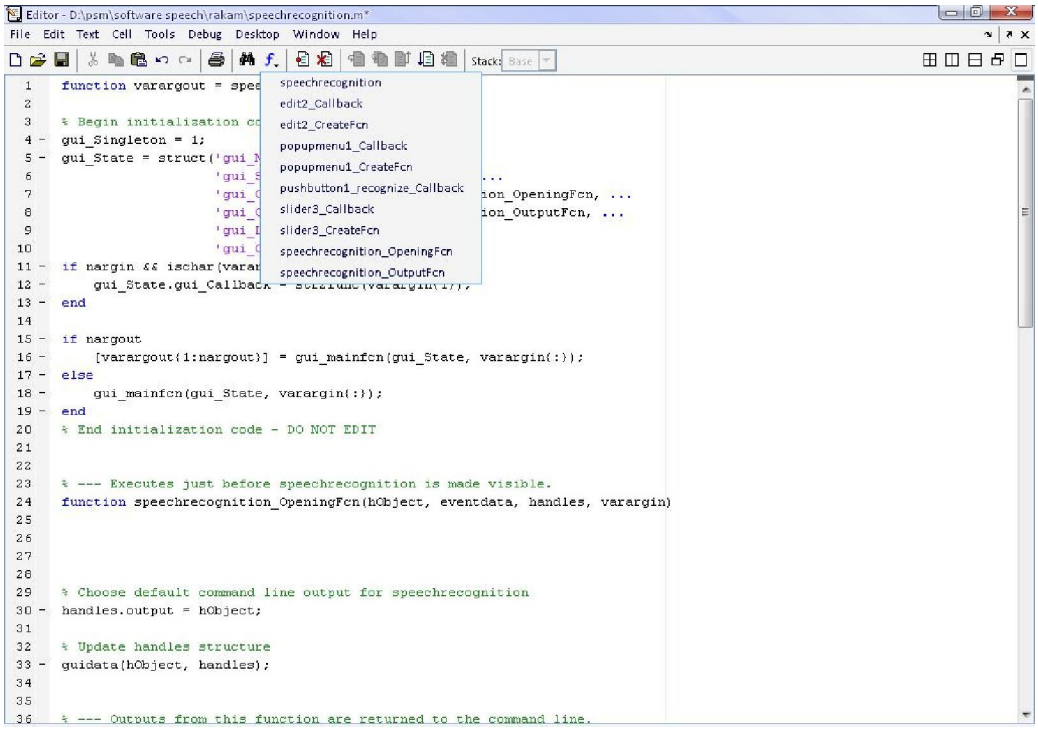


Figure 3.25: M-file Automatically Generated by MATLAB

CHAPTER 4

SOFTWARE DEVELOPMENT

4.1 Introduction

Flowchart is a chart that will show how the system flow step by step. This procedure needed for every system to make sure the system work properly as it desire. Error detection also can be define in the flow chart and it will be easy to recover.

In software development, flow chart have it own role. With the proper flowchart, a programmer will get the basic idea and process to write the program shall less difficult. On the other hand, flowchart will describe overall system function from the start to the end of the system.

4.2 Programming the GUI for recording voice

This section shows the flow chart for programing the GUI made for recording the voice using MATLAB. It consist several flow chart for the components in the GUI that will work to record the voice. GUI that have been created before will not functioning as desire since there is no coding or programming to service the components so that it will act as the recording tool or speech recognizer.

4.2.1 Programming the ‘RECORD’ push button

The ‘record’ push button is the main element in this GUI. When user click the ‘RECORD’ push button, it will cause the MATLAB to initialize the window sound card and start getting data from the microphone. After receiving the data, MATLAB will process the data and plot it to 2D graph. Figure 4.2 shows the whole process when user click the ‘RECORD’ button.

From Figure 4.1, one thing that may caught up some attention is about the pop-up menu. The flow chart shows that to get the frequency sampling for recording the voice, it depends on the value of the pop-up menu, Actually, the value is the number that given to the string listed in the pop-up menu. The first string that being listed will have the value equal to one and so on. Figure 4.1 picture the situation.

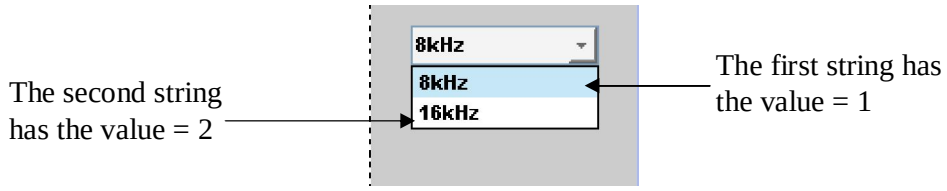


Figure 4.1: Pop-up Menu value

By default, the pop-up menu will have the value equal to one and if the user don’t select anything from the pop-up menu, it will tell this recording tool to set the frequency sampling to 8 kHz.

In order to intialize the sound card, special command from the data acquasition toolbox will be used by MATLAB. The command is `analoginput`. The syntax is as follow:

```
AI = analoginput('adaptor')
```

where `adaptor` is the hardware driver adaptor name. The supported adaptors are advantech, hpe1432, keithley, mcc, nidaq, and winsound. For most computer, the sound card adaptor name is winsound. The function `analoginput` only create the

analog input object for MATLAB only but to get or record the data, another command will be used that is `getdata`. The syntax used to get the data from the sound card:

```
data = getdata(AI);
```

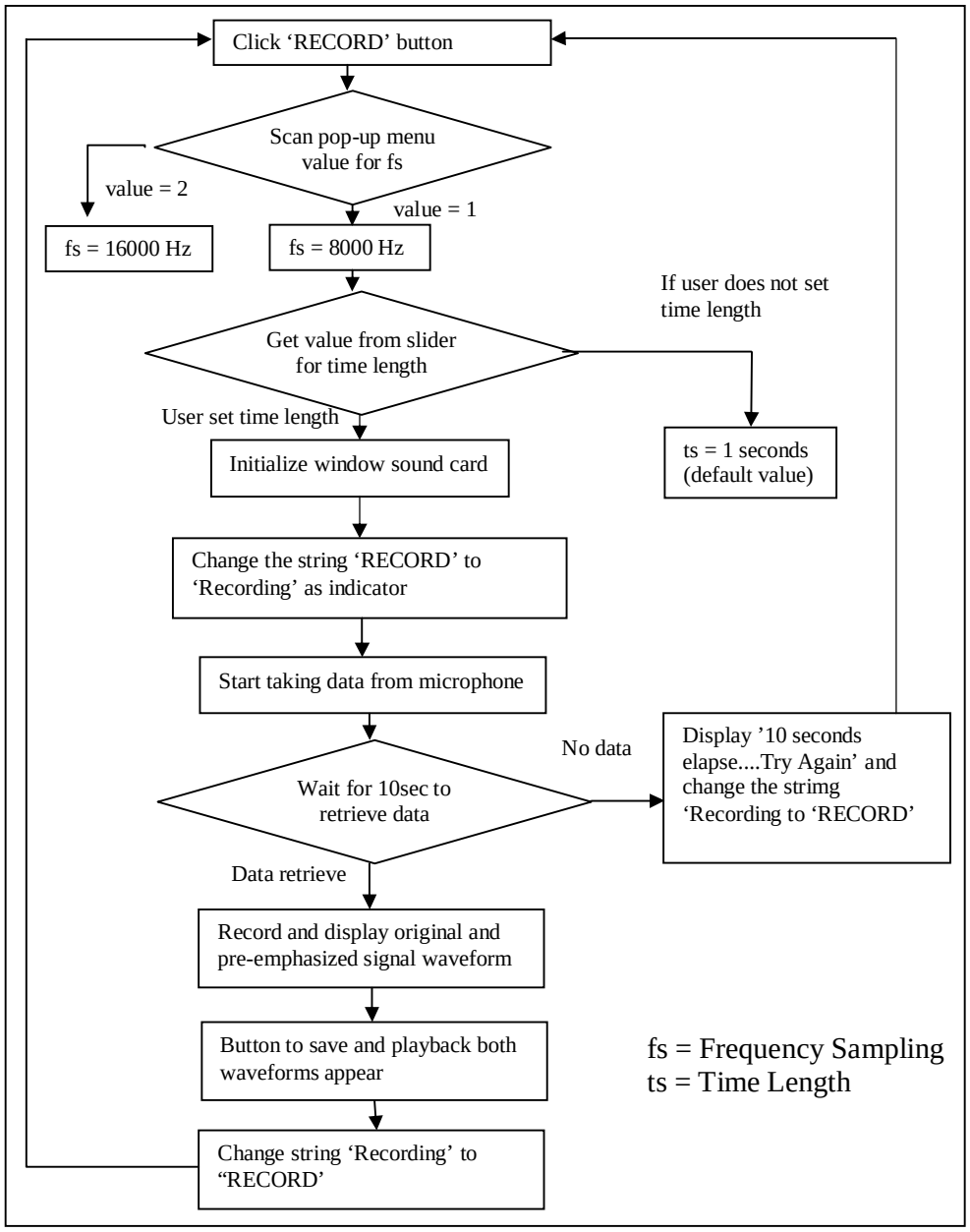


Figure 4.1: Process for recording data

4.2.1.1 Initializing window sound card

Window sound card is a hardware that is installed in the computer/laptop for processing all the files related to audio or sound. MATLAB will utilize the sound card to capture and process the sound or voice signal from the microphone speak by a user when the use ruse this recording tool to record the voice. Figure 4.3 shows the flow chart on how the MATLAB initialize the sound card so that it is compatible with it.

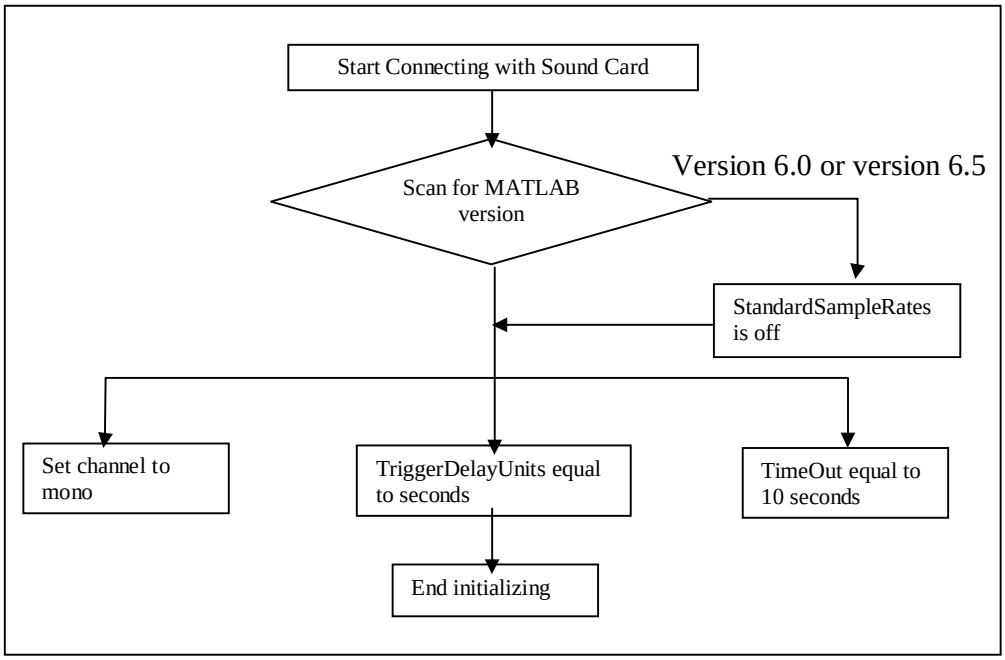


Figure 4.3: Initializing sound card with MATLAB

4.2.2 Programming the static text for slider

Eventhough the slider will change its value whenever the user slide the button at the slider, it doesn't has any indicator to tell that the value have changed. Hence, static text will be used to indicate the changes but it must be programmed first in order for it to become the indicator. The static text is automatically changed according to the value of the slider. The flow chart is shown in Figure 4.4.

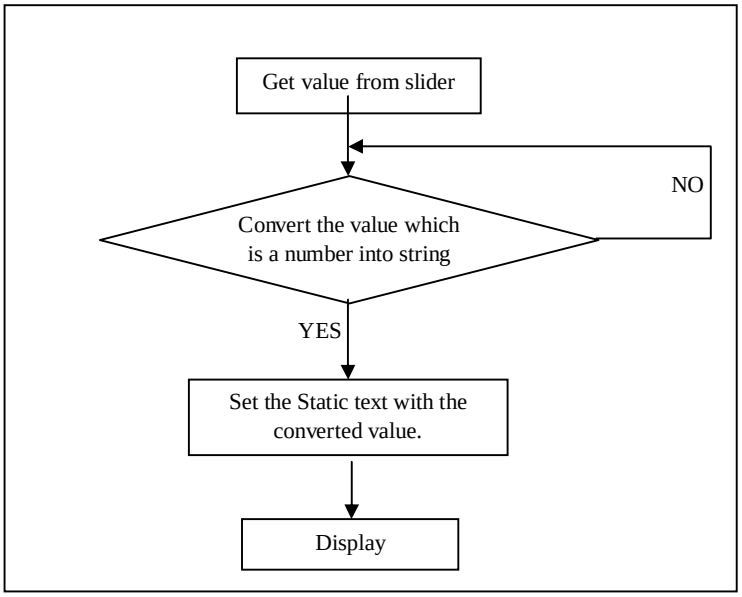
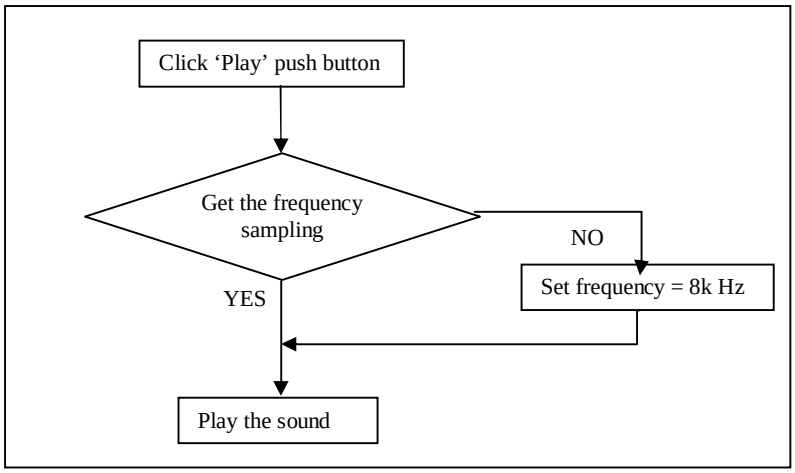


Figure 4.4: Flow Chart for programming the static text

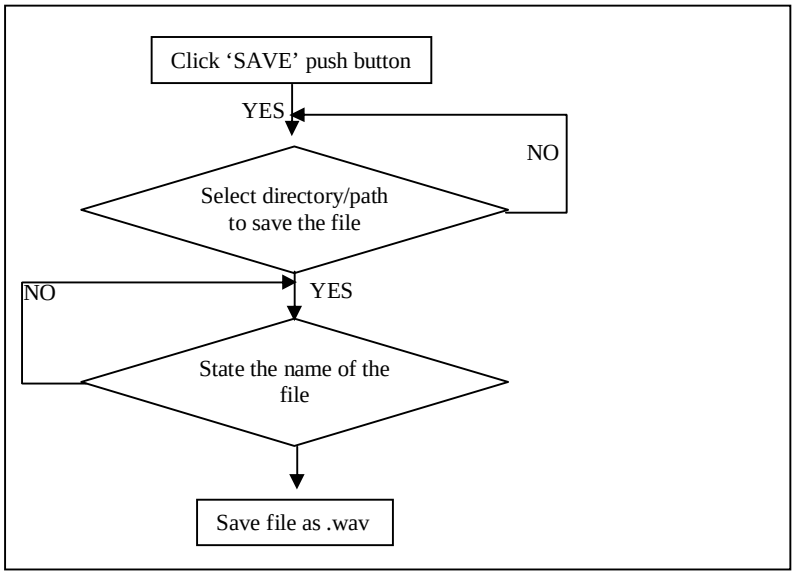
The value from the slider must be converted into string first because the behaviour of the static text that can display string only. If the value which is in numerical form is not converted, then the MATLAB cannot read it as static text. To convert number into string, command `num2str(data)` is used.

4.2.3 Programming the play and save push button

The play and save push button will appear once all the waveform have been displayed. Play push button when click, will playback the recorded voice and the save push button will save the recorded voice as wave file when click. Figures 4.5 shows the flowchart for playing and saving the recorded voice.



(a)



(b)

Figure 4.5: Playing and saving data (a) Flowchart for playing data
(b) Flowchar for saving data

The command `soundsc(y,Fs)` is used by MATLAB to play the audio file. `soundsc(y,Fs)` sends the signal in vector y (with sample frequency Fs) to the speaker on PC and most UNIX platforms. The signal y is scaled to the range before it is played, resulting in a sound that is played as loud as possible without clipping.

To save the recorded file in wave format, MATLAB used command `wavwrite`. The syntax is:

```
wavwrite(y,Fs,'filename')
```

where `wavwrite(y,'filename')` writes the data stored in the variable y to a WAVE file called filename. The data has a sample rate of 8000 Hz and is assumed to be 16-bit. Each column of the data represents a separate channel. Therefore, stereo data should be specified as a matrix with two columns. Amplitude values outside the range [-1,+1] are clipped prior to writing. However, to select the directory or path for saving the file, another command is used which is `uiputfile`. The syntax is:

```
[FileName,PathName] = uiputfile(...)
```

where `[FileName,PathName] = uiputfile(...)` returns the name and path of the file selected in the dialog box. If the user clicks the Cancel button or closes the dialog window, FileName and PathName are set to 0. Figure 4.6 shows the window that appear when the command of uiputfile is executed.

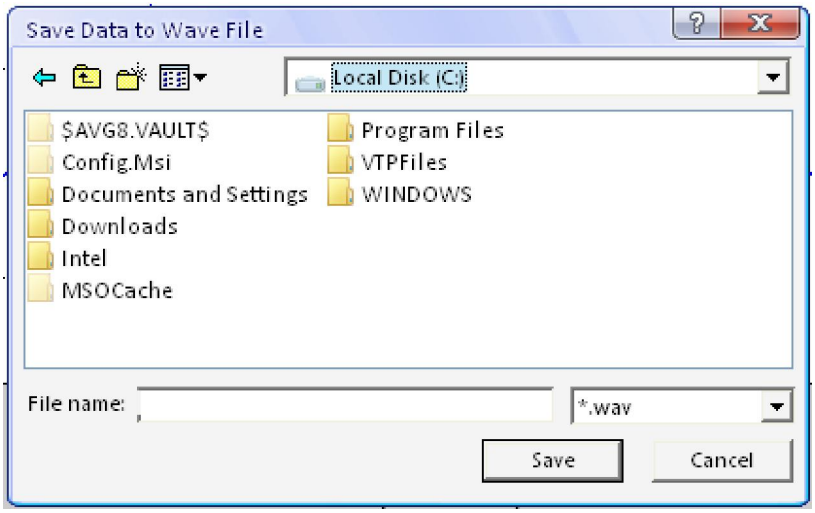


Figure 4.6 : UIPUTFILE window

4.3 Programming the gui for speech recognizer

The GUI tha thas been created for the speech recognizer before this will not functioning unless its components is assigned with the coding that can run the speech recognition program. For this speech recognizer, it have three templates to choose which are the English template, Malay template and Dictionary template. For english template, it will recognize english word only and Malay word will be recognized if the template is the Malay template. For the dictionary template, a new feature is added where it will display the picutre of the item that been spoken. This section will explain the flowchart about programming the GUI so that it will act as the speech recognizer.

4.3.1 Programming the 'RECOGNIZE' push button

When the 'RECOGNIZE' push button is click, it will run the speech recognition function. The speech recognition fucntion is created by writing a lot of coding based on the calculation of Mel Frequency Cepstral Coefficient (MFCC) and Dynamic Time Warping (DTW). Figure 4.7 shows the flowchart of its basic operation.

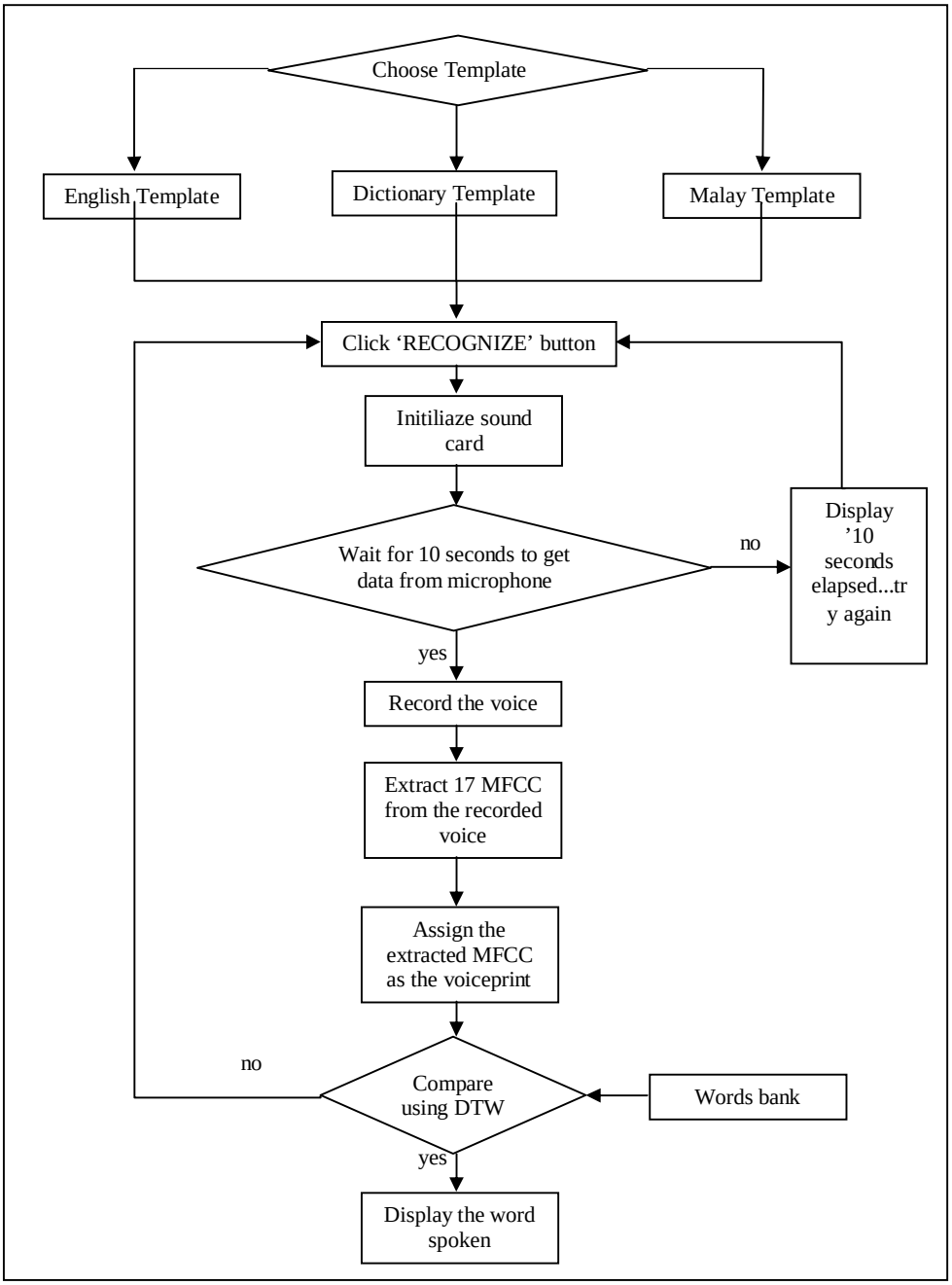


Figure 4.7 : Speech Recognizer flowchart

4.3.2 Creating the template

Template contains numbers of voiceprint of the spoken words. To create a template, ti is necessary to create the voiceprint for the spoken words. Voiceprint is

generate by extracting its Mel Frequency Cepstral Coefficient (MFCC). For this project, ten voiceprints is added for each template means that ten words can be recognized by each template. There are steps to extract MFCC from a voice signal. Figure 4.8 shows the steps:

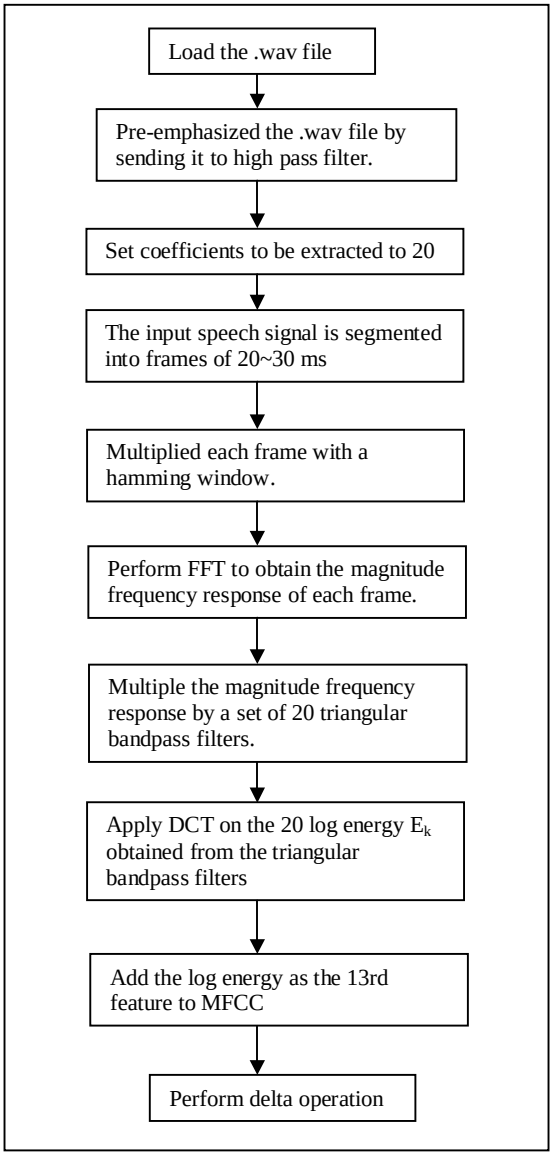


Figure 4.8 : Extracting MFCC from speech signal

After extracting the MFCC from the recorded voice, then it will be save in .mat file. .mat file is an extension of MATLAB file. The advantages saving in .mat is it can allow to save more than parameter into one single file. For exampe, in English

template, there will be ten voiceprints in it. To do so, save all the voiceprint as English_template.mat.

4.3.3 DYNAMIC TIME WARPING

Writing the coding to perform dynamic time warping (DTW) is not an easy task. It requires a good command of mathematical knowledge and dynamic programming. However, by the help of flow chart, it can ease this laborous job. The flowchart is shown in Figure 4.9.

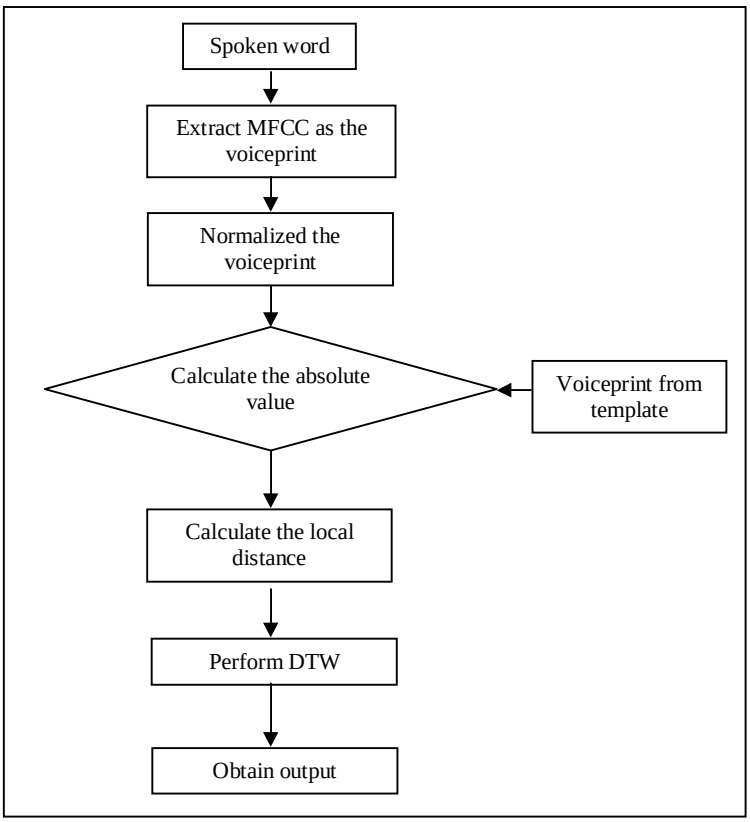


Figure 4.9 : Dynamic Time Warping flow chart

4.3.4 Interfacing MATLAB with hardware

Data from MATLAB can be transfered either using the parallel or serial port to a hardware. The command is `digitalio` and the syntax is as follow:

```
DIO = digitalio('adaptor',ID)
```

`DIO = digitalio('adaptor',ID)` creates the digital I/O object DIO for the specified adaptor and for the hardware device with device identifier ID. ID can be specified as an integer or a string.

Table 4.1: Arguments for digitalio function

'adaptor'	The hardware driver adaptor name. The supported adaptors are advantech, keithley, mcc, nidaq, and parallel.
ID	The hardware device identifier
DIO	The digital I/O object.

CHAPTER 5

RESULT & DISCUSSION

5.1 Introduction

In this chapter, all results and the limitation of the project will be discussed. All discussions will focus on the result obtained and performance of the project.

As a result, this speech recognizer develop using the MATLAB 7.1 as shown in Figure 5.1 has gave a good result in terms of its efficieny for recognizing the word spoken by a user. However, this speech recognizer can only recognize discrete word only and cannot recognize words in a robust speech. This means, if the user want the speech recognizer to recognizer the word that they want to say, the user must say one word only at a time. For example, the speech recognizer can recognize the word ‘off’ if the user speak ‘off’ only but it cannot recognize the word ‘off’ if the user speak a sentence that contain the word ‘off’ such as ‘please turn off the lamp’. There are also several factors that effect the efficiency of this speech recognition and it will be discussed thoroughly.

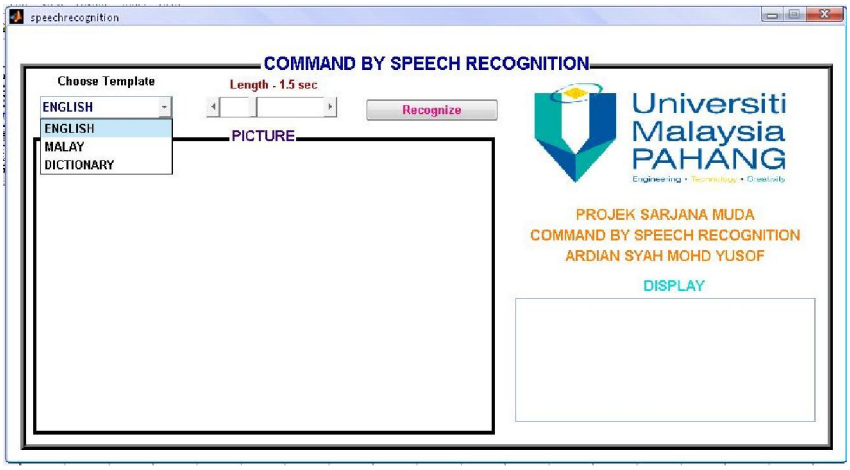


Figure 5.1: Complete GUI for Speech Recognizer

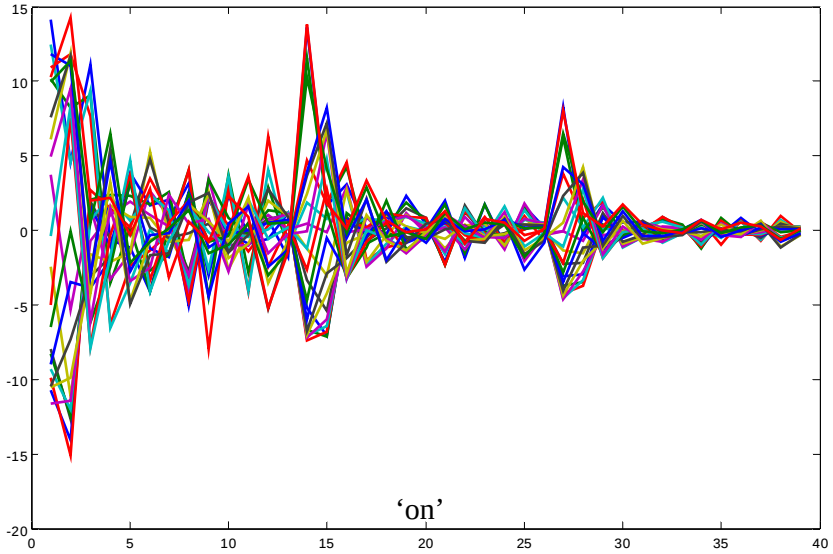
5.2 Voiceprint

Before the speech recognizer can perform, it must have the words bank or template that contain voiceprint of the word. Voiceprint is a mark that represent the word and it have different value for each word. For this project, the voiceprint is generate by extracting the speech signal's Mel Frequency Cepstral Coefficient (MFCC). As mentioned before, this speech recognizer contain three templates which are the english template, Malay template and dictionary template. Table 5.1 list all the words for each template:

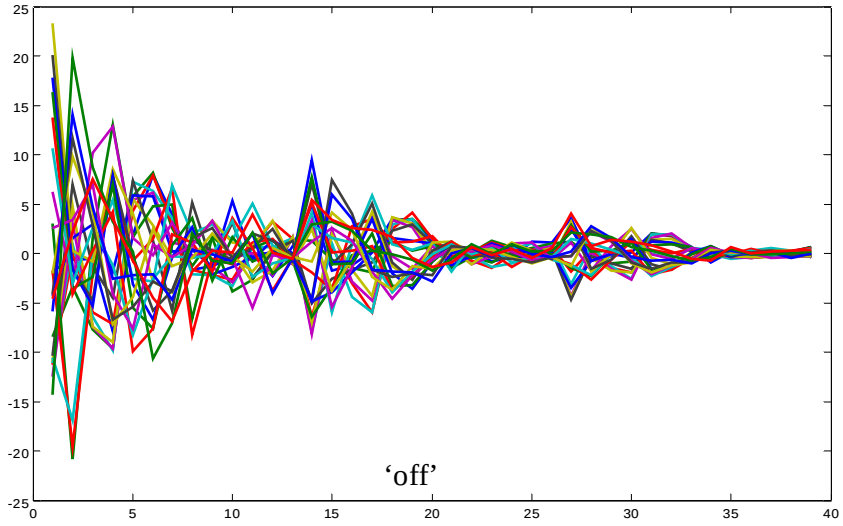
Table 5.1: List of the Words in each template

ENGLISH	**MALAY**	**DICTIONARY**
ON	MULA	KUCING
OFF	KIRI	GAJAH
FORWARD	KANAN	ITIK
BACKWARD	DEPAN	DURIAN
STOP	BELAKANG	BASIKAL
HELLO	BERHENTI	KERETA
APPLE	BUKA	LEMBU
ELEPHANT	TUTUP	KAMBING

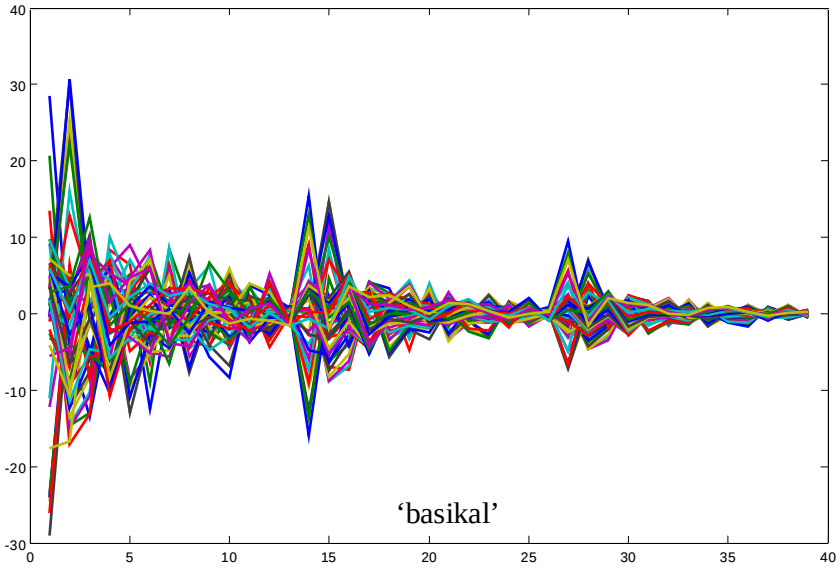
The template does not contain the waveform of the word but it contain the voiceprint of the word. The size of the voiceprint is smaller than the size of the original waveform of the word and that is the reason why voiceprint for each word is use for recognition purpose. Figures below show some of the voiceprints of the words that have been stored in the templates.



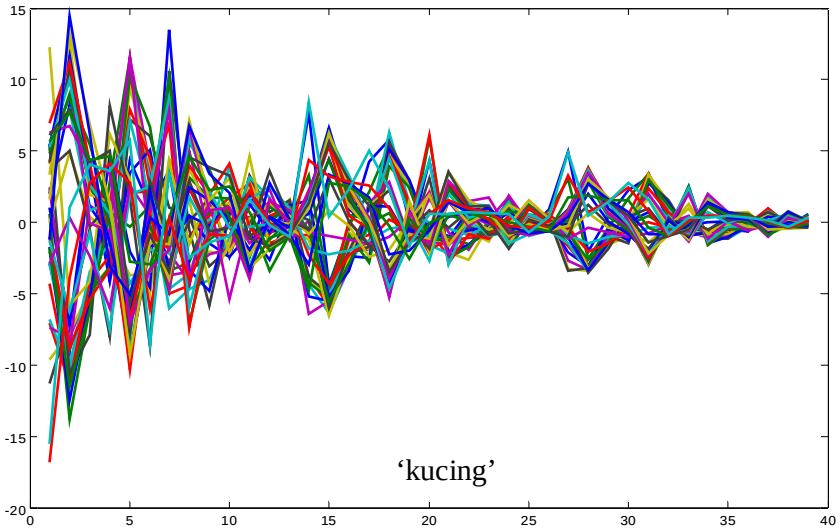
(a)



(b)



(c)



(d)

Figure 5.2: Voiceprints for the word spoken (a) 'on' (b) 'off' (c) 'basikal' (d) 'kucing'

5.3 Performance of the speech recognizer

This speech recognizer cannot recognize the word spoken perfectly. The recognition rate is depends on several factors. The recognition rate totally depends on the way the user spoke and whose voice is used for creating the templates. Table 5.2 shows the recognition rate for this speech recognizer for each template.

Table 5.2: Recognition rate for each template.(a) English template (b) Malay template (c) Dictionary template

WORDS	SPOKEN (n times)	RECOGNIZE (n times)	EFFICIENCY (%)
ON	20	13	65
OFF	20	10	50
FORWARD	20	18	90
BACKWARD	20	17	85
STOP	20	12	60
HELLO	20	16	80
APPLE	20	17	85
ELEPHANT	20	20	100

(a)

WORDS	SPOKEN (n times)	RECOGNIZE (n times)	EFFICIENCY (%)
MULA	20	15	75
KIRI	20	10	50
KANAN	20	11	55
DEPAN	20	17	85
BELAKANG	20	20	100
BERHENTI	20	20	100
BUKA	20	13	65
TUTUP	20	16	80

(b)

WORDS	SPOKEN (n times)	RECOGNIZE (n times)	EFFICIENCY (%)
KUCING	20	15	75
GAJAH	20	10	50
ITIK	20	11	55
DURIAN	20	19	95
BASIKAL	20	20	100
KERETA	20	20	100
LEMBU	20	9	45
KAMBING	20	8	40

(c)

The above results is done by the same male person where is his voice also been used to create the voiceprint. From the result above, it is found that the word

that have more than one consonant can be recognized more than 95% compare to the word that have only one consonant. The performance of this speech recognizer is effected by several factors:

1. User
2. Voiceprint
3. Numbers of voiceprint in the template
4. Environment

User factor is the main problem for this project. This speech recognition will recognize the word based on the template that have been stored in it. Basically, the recognition rate is high for the user which his voice has been used to create the template. But for other persons who his voice is not used to create the template, he need to speak the way the person who create the template spoke. If not, it is quite hard for this speech recognizer to recognize the word eventhough it is the same word. Another issue regarding to the user effect is the gender. If the template is create using the male voice, then it can only recognize the male voice and vice versa. Table 5.3 shows the effect of the user factor towards the efficiency of this speech recognizer.

Table 5.3: User effect to the recognition rate

WORDS / **SPEAKER**	**SPOKEN (n times)**	**TEMPLATE CREATOR(male)**	**OTHER MALE USER**	**FEMALE USER**
ON	20	65%	20%	N/A
OFF	20	50%	20%	N/A
FORWARD	20	90%	35%	N/A
BACKWARD	20	85%	45%	N/A
STOP	20	60%	50%	N/A
HELLO	20	80%	40%	N/A
APPLE	20	85%	30%	N/A
ELEPHANT	20	100%	55%	N/A

The another factor that effect the efficiency of the speech recognizer is the method to create the voiceprint. For this project, the voiceprint is create by extracting the MFCC but there are many method to extract the MFCC from speech recognition. When the voiceprint is made by extracting the basic MFCC only, the recognition rate turn to be so worst that the speech recognition can recognize only

the words tha thave one consonant. Figure 5.3 shows the difference between basic MFCC and improved MFCC.

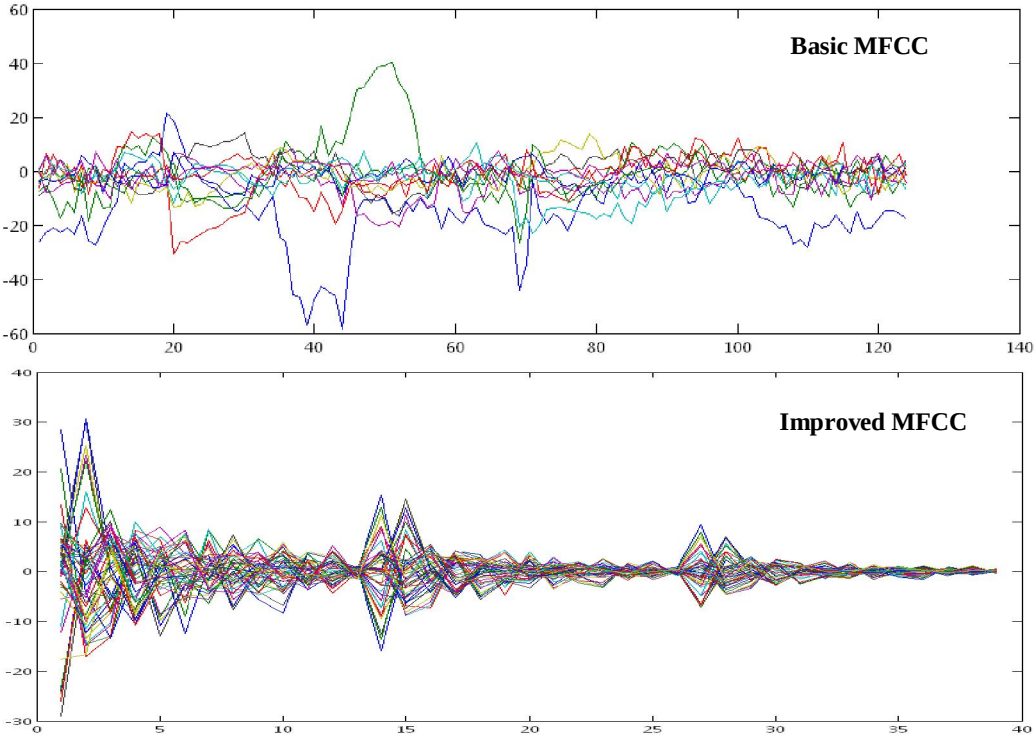


Figure 5.3: The difference between basic and improved MFCC for word 'basikal'

Figure 5.3 shows the difference between the basic and improved MFCC. Improved MFCC is done by adding the the log energy calculation and perform delta operation. This way, it can add more features into the MFCC and increase the recognition rate since it has extra characteristic. As shown in figure 5.6, the graph for basic MFCC has less information compare to the improved MFCC which contain a lot of informations. Table 5.4 tells the huge different for the recognition rate between these two MFCC. The recognition is done using the english template and spoke by a male person. The speech recognizer can recognize not more than 50% for the word that have one consonant but for words more than one consonant, it totally can't recognize it.

Table 5.4:Voiceprint effect to the recognition rate

WORDS	SPOKEN (n times)	Basic MFCC (%)	Improved MFCC (%)
ON	20	45	65
OFF	20	30	50
FORWARD	20	N/A	90
BACKWARD	20	15	85
STOP	20	10	60
HELLO	20	N/A	80
APPLE	20	N/A	85
ELEPHANT	20	N/A	100

The third factor is the number of voiceprint in a template. When all of the words from each template are combined into one template, there is a shocked result regarding to the speech recognizer. Suprisingly, when the number of voiceprint or words in the template is increasing, then the performance is decreasing, This is due to the method of recognition which is using the Dynamic Time Warping (DTW). DTW only calculate the distance or difference between the voiceprint and the spoken word but it will not detect the pattern of the word. So, eventhough the voiceprint may not represent the word ‘off’ for example, but if it have the smallest distance with the spoken word ‘stop’, then the recognizer will recognize the spoken word as ‘off’ and not ‘stop’. Table below shows the result when words in English template and malay template are combined together.

Table 5.5: Number of words in template effect to the recognition

WORDS	SPOKEN (n times)	RECOGNIZE (n times)	EFFICIENCY (%)
ON	20	10	50
OFF	20	7	35
FORWARD	20	15	75
BACKWARD	20	12	60
STOP	20	8	40
HELLO	20	13	65
APPLE	20	15	75
ELEPHANT	20	17	85
MULA	20	10	50
KIRI	20	5	25
KANAN	20	7	35
DEPAN	20	13	65
BELAKANG	20	17	85
BERHENTI	20	19	95
BUKA	20	7	35
TUTUP	20	12	60

The last factor is the environment. In order for this speech recognizer to perform very well, user must use it in a silent room where there is no noise sound surrounding. If the user use this speech recognizer at the busy place or crowded place, then it can't perform well. Since the speech recognizer will capture every sound from the microphone to perform the speech recognition program, then it is best to do it in a silent situation so that only the spoken word is capture by the speech recognizer.

5.4 Conclusion and future recommendation

In conclusion, the speech recognizer that has been develop using MATLAB software can recognize the word spoken by the user but there are several condition must be fulfilled in order to get the best result. The conditions are:

1. This speech recognizer can detect discrete word only and not a robust speech
2. This speech recognition is speaker dependant means that it can give a good performance only for the user who is his voice is used to create the voiceprint.
3. To perform the speech recognition, do it in a silent room or place. Crowded or busy place will effect the performance of the speech recognizer.

There are several recommendations to improve this speech recognizer which can be done in the future. There are:

1. Use Hidden Markov Model (HMM) method to recognize the word. HMM is is a statistical model in which the system being modeled is assumed to be a Markov process with unknown parameters, and the challenge is to determine the hidden parameters from the observable parameters. The extracted model parameters can then be used to perform further analysis, for example for pattern recognition applications. By using HMM, it can boost the performance of the speech recognizer and a large number of words in template will not become a problem anymore since HMM is recognizing by

predicting the pattern and not calculate the distance like Dynamic Time Warping method.

2. Instead of using MATLAB, try using the microcontroller chip such as PIC provided by the Microchip to perform the speech recognition. The coding is written in C language and just embedded it into the microcontroller chip. By using microcontroller chip, the speech recognizer can be a portable device and easy to attach to any electrical devices compare to this speech recognizer which using MATLAB to perform it.

5.5 COST AND COMMERCIALIAZATION VALUE

For this speech recognizer, the cost to develop it for domestic usage is not accountable yet due to several factors which are:

1. This speech recognizer using MATLAB to perform the operation. Because of that, the cost must include the license cost from the MathWork company which can cost more than RM20 000. But compare to the performance of this speech recognizer, that cost is not acceptable.
2. Not every person has the MATLAB software, then this speech recognizer cannot be commercialized.

APPENDIX A

M-file CODING FOR RECORDING GUI

```
function recordtool(a)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%              Projek Sarjana Muda             %
%           ARDIAN SYAH B MOHD YUSOF           %
%                 EA04016- BEE                 %
%          TOOLS FOR SPEECH RECOGNITION        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Type 'recordtool' at the MATLAB command window. Make sure the current
% directory is set at the same place where this file is saved.


global record_figure time_slider popupfreq_button record_button save1_button
save2_button slider_text emph_data recog_text
global load_button play1_button play2_button data R_fs R_samp_len ai
original_axes preemphasis_axes record_text filename

if nargin == 0 % if no input argument, draw the GUI
   a = 0;
end

spec = 'narrowband';
wideband_time = 4e-3;
narrowband_time = 25e-3;
fft_pts      = 2048; % # of points in the FFT

switch a

case 0 % Draw figure

   clear global data  % padam mana2 fail rakaman sebelumnya

   width = 950;
   height = 600;

   record_figure = figure('Position',[30 55 width height],...
      'NumberTitle','off',...
      'Color',[.8 .8 .8],...
      'Name','recordtool');

  record_button = uicontrol('Style','pushbutton',... % "record" button
      'Units','normalized',...
      'Position',[850/width 380/height 80/width 30/height],...
      'ForegroundColor',[0 0.3 0],...
      'FontWeight','bold',...
      'String','RECORD',...
      'Visible','on',...
      'CallBack','recordtool(1)');

  record_text = uicontrol('Style','text',...
      'Units','normalized', ...
```

```
        'Position',[830/width 415/height 110/width 20/height],...
        'BackgroundColor',[.8 .8 .8],...
        'String','      ');

    load_button = uicontrol('Style','pushbutton',... % "load" button
        'Units','normalized',...
        'Position',[850/width 330/height 80/width 30/height],...
        'ForegroundColor',[1 0.5 0.2],...
        'FontWeight','bold',...
        'String','LOAD',...
        'Visible','on',...
        'CallBack','recordtool(6)');
%
    save1_button = uicontrol('Style','pushbutton',... % "save1" button
        'Units','normalized',...
        'Position',[726/width 342/height 80/width 30/height],...
        'ForegroundColor',[0.2 0.2 1],...
        'FontWeight','bold',...
        'String','SAVE',...
        'Visible','on',...
        'CallBack','recordtool(4)');

    play1_button = uicontrol('Style','pushbutton',... % "play1" button
        'Units','normalized',...
        'Position',[645/width 342/height 80/width 30/height],...
        'ForegroundColor',[0.2 0.2 1],...
        'FontWeight','bold',...
        'String','PLAY',...
        'Visible','on',...
        'CallBack','recordtool(2)');

    save2_button = uicontrol('Style','pushbutton',... % "save2" button
        'Units','normalized',...
        'Position',[726/width 42/height 80/width 30/height],...
        'ForegroundColor',[0 0 0.4],...
        'FontWeight','bold',...
        'String','SAVE',...
        'Visible','on',...
        'CallBack','recordtool(5)');

    play2_button = uicontrol('Style','pushbutton',... % "play2" button
        'Units','normalized',...
        'Position',[645/width 42/height 80/width 30/height],...
        'ForegroundColor',[0 0 0.4],...
        'FontWeight','bold',...
        'String','PLAY',...
        'Visible','on',...
        'CallBack','recordtool(3)');

    time_slider = uicontrol('Style','Slider',... % slider untuk menetapkan
tempoh masa rakaman
        'Units','normalized', ...
        'Position',[830/width 530/height 100/width 20/height],...
        'Min',0.5,'Max',5,'Value',1,...
        'SliderStep',[1/9  1/9],...
        'Callback','recordtool(7)');
```

```
  slider_text = uicontrol('Style','text',...  % untuk memaparkan slider
      'Units','normalized', ...
      'Position',[830/width 555/height 120/width 15/height],...
      'BackgroundColor',[.8 .8 .8],...
      'FontWeight','bold',...
      'String','Length - 1 sec');

  popupfreq_button = uicontrol('Style','popupmenu',... % "freq" menu
      'Units','normalized',...
      'Position',[830/width 470/height 100/width 30/height],...
      'ForegroundColor',[0 0 0],...
      'FontWeight','bold',...
      'String',{'8kHz','16kHz'},...
      'Visible','on');

  original_axes = axes('Units', 'normalized',...
     'Position', [0.1 0.57 0.75 0.37],...
     'Xgrid', 'on',...
     'Ygrid', 'on',...
     'Xminortick', 'on');

 preemphasis_axes = axes('Units', 'normalized',...
     'Position', [0.1 0.07 0.75 0.37],...
     'Xgrid', 'on',...
     'Ygrid', 'on',...
     'Xminortick', 'on');

  movegui(record_figure,'center')

    case 1 %rakam

        % menentukan panjang masa & frequency sampling
        val_freq = get(popupfreq_button,'Value');
            if val_freq == 1
                R_fs = 8000;
            elseif val_freq == 2
                R_fs = 16000;
            end
        R_samp_len = get(time_slider,'Value');

        %proses mengenalpasti input & output
        ai = init_sound(R_fs,R_samp_len);
        %R_fs = get(ai, 'SampleRate'); %digunakan jika fequency sampling
tidak ditetapkan.

        % mengambil data dari mikrofon
        nogo=0;

            while not (nogo)
                set(record_button,'String','Recording');
                set(record_text,'String','Talk Now!...');
                start(ai);
                try
                    data = getdata(ai);
```

```
                nogo=1;
            catch
                disp('10 seconds elapsed... try again!');
                stop(ai);
            end
        end
    delete(ai);
    set(record_button,'String','RECORD');
    set(record_text,'String','          ');

    % normalize sound data to 99% of max
    data = 0.99*data/max(abs(data));


    % displays the time graph of the voice signal
    samp_len = length(data)/R_fs;
    t_period = 1/R_fs;
    t = 0:t_period:(samp_len-t_period);
    axes(original_axes)
    plot(t,data)
    grid on
    title('Original signal')
    xlabel('seconds')
    ylabel('amplitude')
    set(play1_button,'visible','on');
    set(save1_button,'visible','on');

    %perform the preemphasis calculation
    a=0.95;
    emph_data = filter([1, -a], 1, data);
    %display preemphasis signal
    axes(preemphasis_axes)
    plot(t,emph_data)
    grid on
    title('Preemphasized signal')
    xlabel('seconds')
    ylabel('amplitude')
    set(play2_button,'visible','on');
    set(save2_button,'visible','on');

case 2  % Play recording for original signal

    % sends an array named z_data to the speakers/headphones
    if length(data) ~= 0
        soundsc(data,R_fs)
    end

case 3 %play recording for emphasized signal
    % sends an array named z_data to the speakers/headphones
    if length(emph_data) ~= 0
        soundsc(emph_data,R_fs)
    end

case 4 %saving original signal as .wav file
    [filename, pathname] = uiputfile('*.wav', 'Save Data to Wave File');
```

```
    if filename ~= 0
        wavwrite(data,R_fs,[pathname filename])
    end

case 5 %saving preemphasized signal as .wav file
    [filename, pathname] = uiputfile('*.wav', 'Save Data to Wave File');
    if filename ~= 0
        wavwrite(emph_data,R_fs,[pathname filename])
    end

case 6 %loading file
    [filename, pathname] = uigetfile('*.wav','Select Data File');

    if filename ~= 0

        cd(pathname);

    %  Get data and sampling rate
        [data,R_fs] = wavread([pathname filename]);
        if min(size(data))>1
            error('Can''t load stereo data')
        end

        % displays the time graph of the voice signal
        samp_len = length(data)/R_fs;
        t_period = 1/R_fs;
        t = 0:t_period:(samp_len-t_period);
        axes(original_axes)
        plot(t,data)
        grid on
        title(['Original signal - ' filename])
        xlabel('seconds')
        ylabel('amplitude')
        set(play1_button,'visible','on');
        set(save1_button,'visible','on');

        %perform the preemphasis calculation
        a=0.95;
        emph_data = filter([1, -a], 1, data);
        %display preemphasis signal
        axes(preemphasis_axes)
        plot(t,emph_data)
        grid on
        title(['Preemphasized signal - ' filename])
        xlabel('seconds')
        ylabel('amplitude')
        set(play2_button,'visible','on');
        set(save2_button,'visible','on');
    end

case 7 % Display time length text

   % Allow the user to set the time length of sample
   num = get(time_slider,'Value');
   set(slider_text,'String',['Length - ' num2str(num) ' sec']
```

APPENDIX B

M-file CODING FOR SPEECH RECOGNIZER GUI

```
function varargout = speechrecognition(varargin)

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @speechrecognition_OpeningFcn, ...
                   'gui_OutputFcn',  @speechrecognition_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before speechrecognition is made visible.
function speechrecognition_OpeningFcn(hObject, eventdata, handles, varargin)

axes(handles.axes3)
RGB=imread('UMP 2.jpg');
image(RGB);
axis off

axes(handles.axes1)
axis off




% Choose default command line output for speechrecognition
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);


% --- Outputs from this function are returned to the command line.
function varargout = speechrecognition_OutputFcn(hObject, eventdata, handles)
% Get default command line output from handles structure
varargout{1} = handles.output;
```

```
function popupmenu1_Callback(hObject, eventdata, handles)


function popupmenu1_CreateFcn(hObject, eventdata, handles)


if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function pushbutton1_recognize_Callback(hObject, eventdata, handles)
%recognition process start here



initialize parallel port
parport=digitalio('parallel','LPT1');
line1=addline(parport,0:4,'out');

temp_type = get(handles.popupmenu1,'Value');

% menentukan panjang masa & frequency sampling

        R_samp_len = get(handles.slider3,'Value');
        R_fs = 8000;
        %proses mendapatkan input & output daripada soundcard
        ai = init_sound(R_fs,R_samp_len);


        % mengambil data dari mikrofon
        nogo=0;

            while not (nogo)
                set(handles.pushbutton1_recognize,'String','Recognizing');
                set(handles.recognition_text,'String','Talk Now!...');
                start(ai);
                try
                    data = getdata(ai);
                    nogo=1;
                catch
                    disp('10 seconds elapsed... try again!');
                    stop(ai);
                end
            end
        delete(ai);
        set(handles.pushbutton1_recognize,'String','Recognize');
        set(handles.recognition_text,'String','          ');

        % normalize sound data to 99% of max
```

```
        data = 0.99*data/max(abs(data));

    %recognition using dtw

[Answer_Distance,Answer_Path_x,Answer_Path_y,Answer_DistanceFrom,Answer_Name]
=Main_DTW(data,R_fs,temp_type);
    set(handles.edit_recognize,'String',Answer_Name);

    if temp_type == 3
        axes(handles.axes1)
        gambar=imread([Answer_Name '.jpg']);
        image(gambar);
        axis off
    elseif temp_type == 1
        if strcmp(Answer_Name,'on')
            putvalue(line1,[1 0 0 0 0]);
        elseif strcmp(Answer_Name,'off')
            putvalue(line1,0);
        elseif strcmp(Answer_Name,'forward')
            putvalue(line1,[1 1 0 0 1]);
        elseif strcmp(Answer_Name,'reverse')
            putvalue(line1,[1 0 1 1 0]);
        end
    end



   guidata(hObject, handles);



if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end




function edit2_Callback(hObject, eventdata, handles)


function edit2_CreateFcn(hObject, eventdata, handles)

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function slider3_Callback(hObject, eventdata, handles)
% Allow the user to set the time length of sample
       num = get(handles.slider3,'Value');
       set(handles.slider_text,'String',['Length - ' num2str(num) ' sec']);

       guidata(hObject, handles);


function slider3_CreateFcn(hObject, eventdata, handles)

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end
```

APPENDIX C

M-file CODING FOR CREATING VOICEPRINT

```
 clear all;
 close all;
 clc;

 Template_MFCC_Features_1=
CMS_Normalization(Feature_Extruction('basikal.wav'));
 Template_MFCC_Features_2= CMS_Normalization(Feature_Extruction('epal.wav'));
 Template_MFCC_Features_3=
CMS_Normalization(Feature_Extruction('gajah.wav'));
 Template_MFCC_Features_4=
CMS_Normalization(Feature_Extruction('harimau.wav'));
 Template_MFCC_Features_5= CMS_Normalization(Feature_Extruction('itik.wav'));
 Template_MFCC_Features_6=
CMS_Normalization(Feature_Extruction('kambing.wav'));
 Template_MFCC_Features_7=
CMS_Normalization(Feature_Extruction('kucing.wav'));
 Template_MFCC_Features_8=
CMS_Normalization(Feature_Extruction('timun.wav'));
 Template_MFCC_Features_9= CMS_Normalization(Feature_Extruction('ayam.wav'));
 Template_MFCC_Features_10=
CMS_Normalization(Feature_Extruction('kuda.wav'));

Template_MFCC_Features_11=CMS_Normalization(Feature_Extruction('lembu.wav'));

Template_MFCC_Features_12=CMS_Normalization(Feature_Extruction('petai.wav'));

Template_MFCC_Features_13=CMS_Normalization(Feature_Extruction('durian.wav'))
;

Template_MFCC_Features_14=CMS_Normalization(Feature_Extruction('pisang.wav'))
;



 %save Templates.mat
 save Templates_dictionary.mat

 clear path
```

APPENDIX D

M-file CODING FOR MFCC

```
%         *******************************************
%         ******     Feature extruction           *******
%         *******************************************
function Featurs= Feature_Extruction(InputWave,Fs)

    % Featurs= Feature_Extruction(InputWave);
    %     Return 20 MFCC feature vectors of InputWave



    if nargin<1
        disp('Error: in Feature_Extruction, no wave file.');
    end
    if isstr(InputWave),
        [InputWave,Fs,NBits] = wavread(InputWave);
    elseif nargin==1
        Fs=8000;
    end



    % ====== Set Parameters.
    %             Frame size: N(ms),Overlapping region is M(ms)
    %             Generally , M = (1/2)*N , which N = 24.
    FrameSize_ms = 24;        % Ex. N=32 = (256/8000)*1000 , each frame has
256 points.
    Overlap_ms = (1/2)*FrameSize_ms;
    FrameSize = round(FrameSize_ms*Fs/1000); % 256
    Overlap  = round(Overlap_ms*Fs/1000);  % 86
    %No_of_Frames= floor((size(InputWave,1)/(FrameSize-Overlap)) - 1)+2;
    % Triangular BandFilter parameters : StartFreq,CenterFreq,StopFreq. (20
Bank filters)
     StartFreq=[1   3   5   7   9   11  13  15  17  19  23  27  31  35  40
46  55  61  70  81];    %Start
     CenterFreq=[3  5   7   9   11  13  15  17  19  21  27  31  35  40  46
55  61  70  81  93];    %Center
     StopFreq=[5    7   9   11  13  15  17  19  21  23  31  35  40  46  55
61  70  81  93  108];   %End

     Threshold = 0.0001;  % for energy test ==> remove fromes with energy
bellow this amount.

    % ====== Step 1: Pre-emphasis.
    InputWave = filter([1, -0.95], 1, InputWave);

    % ====== Step 2: Windowng & overlapping.
    Frame = buffer(InputWave, FrameSize, Overlap);

    normalize_coff = 10;
    energy = sum(Frame.^2)/FrameSize;
    index = find(energy < Threshold);
```

```
    energy(index) = [];
    logEnergy = 10*log10(energy)/normalize_coff;
    Frame(:, index) = [];   % Remove empty frames
    Features = [];
    for i = 1:size(Frame, 2);   % size(Frame, 2)=No_of_Frames

          % ====== Step 3: Hamming window.
          WindowedFrame  = hamming(FrameSize).*Frame(:,i);

          % ====== Step 4: FFT: fast fourier transform.
          %            Using FFT function to calculate.
          %            Compute square of real part and imaginary part.
          FFT_Frame = abs(fft(WindowedFrame));

          % ====== Step 5: Triangular bandpass filter.
          %            Using user defined function
triBandFilter(fftFrame{i}).
          No_of_FilterBanks = 20;      %No_of_FilterBanks means counts of log
spectral magnitude.
          tbfCoef =
TriBandFilter(FFT_Frame,No_of_FilterBanks,StartFreq,CenterFreq,StopFreq);

          % ====== Step 6: Logearithm.
          tbfCoef = log(tbfCoef.^2);

          % ====== Step 7: DCT: Discrete Cosine Transform.
          %            Using DCT to get L order mel-scale-cepstrum
parameters.
          No_of_Features = 12;      % generally No_of_Features is 12.
          Cepstrums = Mel_Cepstrum2(No_of_Features,No_of_FilterBanks,tbfCoef);
          Features = [Features Cepstrums'];

    end;
    Features = [Features; logEnergy];


    %=========compute delta energy and delta cepstrum============
    %Calculate delta cepstrum and delta log energy
    % get 13 order Features.
    Delta_window = 2;
    D_Features = DeltaFeature(Delta_window, Features);

    %=========compute delta-delta energy and delta cepstrum============
    %Calculate delta-delta cepstrum and delta log energy
    %Combine them with previouse features, get 39 order Features.

    %Delta_window = 2;
    %D_d_Features = Delta_DeltaFeature(Delta_window, Features);
    % or
    D_d_Features = DeltaFeature(Delta_window, D_Features);

    %===== Combine cepstrum,delta and delta-delta
     Features = [Features ; D_Features ; D_d_Features];  % 39 features
     %Features = [Features ; D_Features];  % 26 features
```

```
%============================= Sub function ==============================
%=========================================================================

%         ********************************************
%         ******    Triangular Band Filter        *******
%         ********************************************

function tbfCoef = TriBandFilter(fftFrame,P,StartFreq,CenterFreq,StopFreq)
    %The function is triangular bandpass filter.
    for i = 1 : P,
        % Compute the slope of left side of triangular bandpass filter
       for j = StartFreq(i) : CenterFreq(i),
          filtmag(j) = (j-StartFreq(i))/(CenterFreq(i)-StartFreq(i));
       end;
        % Compute the slope of right side of triangular bandpass filter
       for j = CenterFreq(i)+1: StopFreq(i),
          filtmag(j) = 1-(j-CenterFreq(i))/(StopFreq(i)-CenterFreq(i));
       end;
       tbfCoef(i) =
sum(fftFrame(StartFreq(i):StopFreq(i)).*filtmag(StartFreq(i):StopFreq(i))');
    end;

%         ********************************************
%         ******     Mel-scale cepstrums          *******
%         ********************************************

function Cepstrum = Mel_Cepstrum2(L,P,tbfCoef)
    %compute mel-scale cepstrum , L should be 12 at most part.
    for i=1:L,
        coef = cos((pi/P)*i*(linspace(1,P,P)-0.5))';
        Cepstrum(i) = sum(coef.*tbfCoef');
    end;

%         ********************************************
%         ******         Delta cepstrums          *******
%         ********************************************

function D_Featurs = DeltaFeature(delta_window,Featurs)
    % Compute delta cepstrum and delta log energy.
    rows  = size(Featurs,1);
    cols  = size(Featurs,2);
    temp  = [zeros(rows,delta_window) Featurs zeros(rows,delta_window)];
    D_Featurs = zeros(rows,cols);
    denominator = sum([1:delta_window].^2)*2;
    for i = 1+delta_window : cols+delta_window,
       subtrahend = 0;
       minuend    = 0;
       for j = 1 : delta_window,
          subtrahend = subtrahend + temp(:,i+j)*j;
          minuend    = minuend + temp(:,i-j)*(-j);
       end;
       D_Featurs(:,i-delta_window) = (subtrahend - minuend)/denominator;
    end;
    %Featurs = [Featurs ; temp2];
```

```
%         *********************************************
%         ******      Delta-Delta cepstrums       *******
%         *********************************************

function D_d_Features = Delta_DeltaFeature(delta_window,Features)
    % Compute delta delta cepstrum and delta log energy.

    % another way!
    % Features1 = DeltaFeature(delta_window,Features);
    % Features2 = DeltaFeature(delta_window,Features1);
    % Features = [Features ; Features2];

    rows  = size(Features,1);
    cols  = size(Features,2);
    temp1  = [zeros(rows,delta_window) Features zeros(rows,delta_window)];
    temp2  = [zeros(rows,delta_window) Features zeros(rows,delta_window)];
    D_d_Features  = zeros(rows,cols);

    % Rabiner method
    denominator = sum([1:delta_window].^2)*2;
    denominator2 =
delta_window*(delta_window+1)*(2*delta_window+1)*(3*delta_window^2+3*delta_wi
ndow-1)/15;
    for i = 1+delta_window : cols+delta_window,
       subtrahend = 0;
       minuend    = 0;
       subtrahend2 = 0;
       minuend2    = 0;
       for j = 1 : delta_window,
          subtrahend = subtrahend + temp1(:,i+j);
          minuend = minuend + temp1(:,i-j);
          subtrahend2  =  subtrahend2+ j*j*temp2(:,i+j);
          minuend2    = minuend2 + (-j)*(-j)*temp2(:,i-j);
       end;
       temp1(:,i) = subtrahend + minuend + temp1(:,i);
       temp2(:,i) = subtrahend2 + minuend2;
       D_d_Features(:,i-delta_window) = 2*(denominator.*temp1(:,i)-
(2*delta_window+1).*temp2(:,i))/(denominator*denominator-
(2*delta_window+1)*denominator2);
    end;
    % Features = [Features ; temp3];
```

APPENDIX E

M-file CODING FOR DTW

```
% Dynamic Time Warping (DTW)

function
[Answer_Distance,Answer_Path_x,Answer_Path_y,Answer_DistanceFrom,Answer_Name]
=Main_DTW(TestWave,Fs,Template)
 No_Templates=14; % from 0 to 13

Test_MFCC_Features= CMS_Normalization(Feature_Extruction(TestWave,Fs));
% CMS_MFCC=CMS_Normalization(MFCC_Features);
 for i=1:No_Templates
       [Template_MFCC_Features,Template_Name]=SelectNextTemplate(i,Template);
      % Construct the 'local match' scores matrix as the cosine distance
between the featurs
       Local_Distance =
LocalDistance(abs(Template_MFCC_Features),abs(Test_MFCC_Features));

       % Find the lowest-cost path across Local_Distance matrix
       [Path_y,Path_x,Distance] = DTW(Local_Distance);

       % Least cost (final cost) is value in top right corner of Distance
matrix
       Distance_from_Template(i)=Distance(1,size(Distance,2));
       if i>1
           if Distance_from_Template(i)<Answer_DistanceFrom
               Answer_Name=Template_Name;
               Answer_Distance=Distance;
               Answer_Path_x=Path_x;
               Answer_Path_y=Path_y;
               Answer_DistanceFrom=Distance_from_Template(i);
           end
       else
           Answer_Name=Template_Name;
           Answer_Distance=Distance;
           Answer_Path_x=Path_x;
           Answer_Path_y=Path_y;
           Answer_DistanceFrom=Distance_from_Template(i);
       end
 end
```

CALCULATING DTW

```
function [Path_y,Path_x,Distance] = DTW(LocalDistance)
    % [Path_y,Path_x] = DTW(LocalDistance)
    %    Use dynamic programming to find a min-cost path through matrix
LocalDistance.
    %    Return state sequence in Path_y,Path_x
```

```
    [Row,Col] = size(LocalDistance);

    % costs
    Distance = zeros(Row+1, Col+1);
    Distance(Row+1,:) = NaN;
    Distance(:,1) = NaN;
    Distance(Row+1,1) = 0;
    Distance(1:(Row), 2:(Col+1)) = LocalDistance;

    AllPath = zeros(Row,Col);

    for i = Row+1:-1:2;
      for j = 1:Col;
        [SelPath, tb] = min([Distance(i, j), Distance(i, j+1), Distance(i-1,
j)]);
        Distance(i-1,j+1) = Distance(i-1,j+1)+SelPath;
        AllPath(i-1,j) = tb;
      end
    end

    % Traceback from top left for finding Path
    i = 1;
    j = Col;
    Path_y = i;
    Path_x = j;
    while i < Row & j > 1
      tb = AllPath(i,j);
      if (tb == 1)
        i = i+1;
        j = j-1;
      elseif (tb == 2)
        i = i+1;
      elseif (tb == 3)
        j = j-1;
      else
        error;
      end
      Path_y = [i,Path_y];
      Path_x = [j,Path_x];
    end

    Distance = Distance(1:(Row),2:(Col+1));
```

CALCULATING LOCAL DISTANCE

```
function Out2 = LocalDistance(A,B)
    % Out = LocalDistance(A,B)
    %    calculate the local distance between feature matrices A and B.
    %    Using inner product i.e. cos(angle between vectors) between vectors.
    %    A and B have same #rows.
```

```
    Mag_A = sqrt(sum(A.^2));
    Mag_B = sqrt(sum(B.^2));

    Cols_A = size(A,2);
    Cols_B = size(B,2);
    Out = zeros(Cols_A, Cols_B);
    for i = 1:Cols_A
     for j = 1:Cols_B
       % normalized inner product i.e. cos(angle between vectors)
       Out(i,j) = (A(:,i)'*B(:,j))/(Mag_A(i)*Mag_B(j));
     end
    end

    %Out = (A'*B)./(Mag_A'*Mag_B);

    Row=size(Out,1);
    for i=1:fix(Row/2)
        %tmp=M(Row-i+1,:);
        Out2(Row-i+1,:)=Out(i,:);
        Out2(i,:)=Out(Row-i+1,:);%tmp;
    end
    if mod(Row,2)~=0
        Out2(fix(Row/2+1),:)=Out(fix(Row/2+1),:);
    end

   % Use 1-Out2 because DTW will find the *lowest* total cost
    Out2=1-Out2;
```

APPENDIX F

M-file CODING FOR INITIALIZE SOUNDCARD

```
function ai = init_sound(fs,samp_len)
% Function 'init_sound' initializes microphone input for voice
% 'fs' is the sampling rate, 'samp_len' is the time to record
%   in seconds.

v = ver;
name = {v.Name};
ind = find(strcmp(name,'MATLAB'));
if isempty(ind)
    ind = find(strcmp(name,'MATLAB Toolbox'));
end

v_num = str2num(v(ind).Version);

ai = analoginput('winsound');
addchannel(ai, 1);
if (v_num == 6.1) | (v_num == 6.5)
    set(ai, 'StandardSampleRates', 'Off');
end
set(ai, 'SampleRate', fs);
actual_fs = get(ai, 'SampleRate');
set(ai, 'TriggerType', 'software');
set(ai, 'TriggerRepeat', 0);
set(ai, 'TriggerCondition', 'Rising');
set(ai, 'TriggerConditionValue', 0.01);
set(ai, 'TriggerChannel', ai.Channel(1));
set(ai, 'TriggerDelay', -0.1);
set(ai, 'TriggerDelayUnits', 'seconds');
set(ai, 'SamplesPerTrigger', actual_fs*samp_len+1);
set(ai, 'TimeOut', 10);
```

REFERENCES

[1] Huang, X., Acero, A., and Hon, H.-W. (2001), Spoken Language Processing: *A Guide to Theory, Algorithm , and System Development*, Prentice Hall, Upper Saddle River, NJ

.

[2] Reynolds, D. A. (1994), *\Experimental evaluation of features for robust speaker identication"*, IEEE Trans. Speech Audio Process., 2(4): 639{643.

[3] Yiying Zhang ,(Oct 1997), *"A robust and fast endpoint detection algorithm for isolated word recgnition"*IEEE conference.

[4] Kaustubh Kale ,(April 2000), *"Speech synthesis and hardware implementation of speech recognition system using digital signal processing",* I.T.B.H.U dissertation.

[5] H. Sakoe and S. Chiba. *Dynamic Programming Algorithm Optimization for Spoken Word Recognition*, IEEE Transactions on Acoustics, Speech and Signal Processing. ASSP-26(1): 43-49. February 1978.

[6] W. H. Abdulla, D. Chow and G. Sin. *Cross-Words Reference Template for DTW-based Speech Recognition System.* IEEE Technology Conference (TENCON). Bangalore, India, 1: 1-4, 2003.