# DECLARATION

### ACKNOWLEDGEMENTS

We would like to express our sincere gratitude to our guide **D.SRINU** , **Associate Professor**, Department of Electronics and Communication Engineering, for his excellent guidance and invaluable support, which helped me accomplish the B.Tech (ECE) degree and prepared me to achieve more life goals in the future. His total support of my dissertation and countless contributions to my technical and professional development made for a truly enjoyable and fruitful experience. Special thanks are dedicated for the discussions we had on almost every working day during my project period and for reviewing my dissertation.

We are very much grateful to my Project Coordinator**, Dr. G. Amarnath**, **Associate Professor,** Department of ECE, MLRITM, Dundigal, Hyderabad, who has not only shown utmost patience, but was fertile in suggestions, vigilant in directions of error and has been infinitely helpful.

We are extremely grateful to **Dr. Srinivas Bachu**, **Associate Professor & HOD-ECE**, MLRITM, Dundigal, Hyderabad, for the moral support and encouragement given in completing my project work.

We wish to express deepest gratitude and thanks to **Dr. K. Venkateswara Reddy, Principal,** MLRITM, for his constant support and encouragement in providing all the facilities in the college to do the project work.

We would also like to thank all our faculties, administrative staff and management of MLRITM, who helped me to completing the mini project.

On a more personal note, I thank my **beloved parents and friends** for their moral support during the course of our project.

# ABSTRACT

This project presents an energy-efficient architecture to extract Mel-Frequency Cepstrum Coefficients (MFCCs) for real-time speech recognition systems. Based on the algorithmic property of MFCC feature extraction, the architecture is designed with floating-point arithmetic units to cover a wide dynamic range with a small bit-width. The dataflow of MFCC extraction is tailored to minimize the computation time. As a result, the energy consumption is considerably reduced compared with previous MFCC extraction systems

**Keywords:** Floating-point operations, hardware optimization, MEL-frequency cepstrum coefficients (MFCCs), speech recognition.

|  |  |  |  |
| --- | --- | --- | --- |
|  | | | **Page No.** |
| *Certificate* | | | *ii* |
| *Acknowledgements* | | | *iii* |
| *Table of Contents* | | | *iv-vi* |
| *List of Figures* | | | *vii-viii* |
| *List of Tables* | | | *ix* |
| *Abstract* | | | *x* |
| **CHAPTER 1: INTRODUCTION** | | |  |
|  | 1.1 | Introduction | 1 |
|  | 1.2 | Fundamentals of speech recognition | 2 |
|  | 1.3 | Objectives | 3 |
|  | 1.4 | Problem statement | 5 |
|  | 1.5 | Methodologies | 8 |
|  | 1.6 | Research gaps | 9 |
|  | 1.7 | Motivation | 10 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CHAPTER 2: LITERATURE SURVEY** | | | | 20-21 |
| **CHAPTER 3: PROPOSED WORK** | | | |  |
|  | 3.1 | Proposed Architecture | | 22 |
|  | 3.2 | DDMFCC | | 23 |
|  |  | 3.2.1 | DDMFCC | 23 |
|  |  | 3.2.2 | Pre processing | 24 |
|  |  | 3.2.3 | Framing and Windowing | 24 |
|  |  | 3.2.4 | Fast Fourier Transform | 25 |
|  |  | 3.2.5 | Mel scaled filter bank | 26 |
|  |  | 3.2.6 | Classification process |  |
|  |  | 3.2.7 | Cepstrum |  |
|  | 3.3 | Classification process | | 26 |
|  | 3.4 | Training HMMs on statistical data | | 26 |
|  | 3.5 | Width Calculation | | 27 |
|  | 3.6 | Two element Specifications | | 28 |
|  | 3.7 | Four element Specifications | | 29 |
| **CHAPTER-4: SOFTWARE IMPLEMENTATION** | | | |  |
|  | 4.1 | Background on MATLAB | | 30 |
|  | 4.2 | Numerical computing MATLAB | | 31 |
|  | 4.3 | History | | 31 |
|  | 4.4 | Interfacing with other languages | | 32 |
|  | 4.5 | Technical computing | | 33 |

|  |  |
| --- | --- |
| **CHAPTER-5: RESULTS** | 43 |
| **CHAPTER-6: CONCLUSION** | 44 |
| **CHAPTER-7: REFERENCES** |  |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Figure Number** | **Description** | **Page**  **Number** |
| **Figure 1.1** | Linear frequency versus Mel frequency scale | 4 |
| **Figure 3.1** | Modified architecture of MFCC | 11 |
| **Figure 3.2** | DDMFCC Flow diagram | 13 |
| **Figure 3.3** | Framing and windowing | 14 |
| **Figure 3.4** | Filter banks | 16 |
| **Figure 3.5** | Mel Cepstrum Coefficients | 16 |
| **Figure 5.1** | Graphical user interface | 22 |
| **Figure 5.2** | The plot of speaker two | 23 |
| **Figure 5.3** | Frequency representation of speaker two | 24 |
| **Figure 5.4** | Plot of coefficients | 25 |
| **Figure 5.5** | Recognition of speaker | 26 |
| **Figure 5.6** | Recognition of Unauthorized speaker | 26 |

### Introduction

**Chapter 1**

## INTRODUCTION

Increasingly, as they are applied in real world applications, speech recognition systems must operate in situations where it is not possible to control the acoustic environment. This may result in a serious mismatch between the training and test conditions, which often causes a dramatic degradation in performance of these systems. The aim of the work presented in this thesis is to make automatic speech recognition systems robust to these environmental differences.

Speech can be characterized by a slowly changing spectral envelope. This spectral envelope is perceived by humans and translated into words and their associated meaning. Automatic speech recognition attempts to emulate part of this task, that of mapping the spectral envelope into a series of words. There are many problems associated with this process. Not all people speak the same. The spectral envelope will vary due to regional accents and differences in the individual, for example whether male or female and their height.

Furthermore, an individual speaker may utter a given sentence in a wide variety of ways. This may be due to speaker stress, such as when shouting, or by the speaker deliberately emphasizing words to alter the meaning. Humans have little difficulty coping with all these variations, however, the design of an automated system that mimics this process is a major challenge.

Most state-of-the-art speech recognition systems use statistical approaches to cope with the many variations. The performance of these recognition systems has gradually increased over the years to a stage where today, on an unlimited vocabulary speaker independent task, performance figures of over 90% word accuracy can be obtained, and on Smaller vocabulary tasks this figure rises to above 95%. At this level of performance the systems are usable; however the majority of them are both trained and tested in identical, quiet environments.

In practical systems, it is rarely quiet and there is usually little control over the acoustic environment. There may be differences in both the background Noise, for example fans running or cars passing by, and the channel conditions, that is the microphone or telephone channel being used.

As the background noise changes and the channel conditions vary, so the speech spectra are altered and there is a resulting, often dramatic, deterioration in the performance of the system.

### Fundamental Steps In Speech Processing

Broadly speaking, the field of speech recognition is interested in addressing three (not necessarily compatible) objectives : (a) the improvement of the perceptual quality of noisy speech, (b) the immunization of speech encoders against input noise and

(c) the improvement of the performance of speech recognition systems in the presence of noise.This thesis investigates the first of these. In our context, the speech enhancement problem concerns the estimation of "clean" (de-noised) speech J (t) 6om noisy speech z (t).

Speech recognition has applications in a wide variety of speech communication contexts where the quality or the intelligibility of speech has been degraded by the presence of background noise. For example, cellular radio telephone systems are plagued not only by background noise but also by channel noise. Public telephones from environmental disturbances of t heir location. Air-ground communication systems are corrupted with cockpit noise. Moreover the hearing impaired require an increase of between 2.5 and 12 dB signal-to noise ratio to achieve similar speech discrimination capabilities to those of normal hearing [9]. These problems call for the use of speech enhancement.

Researchers have been working on devising an efficient way to extract clean speech from noisy speech for the last 30 years. Two broad divisions of speech enhancement techniques are non-parametric and paramedic mode1 based approaches [IO]. One of the pop& digital signal processing (DSP) non-paramedic techniques for speech enhancement is spectral subtraction. In 1979, Lim and Oppenheim presented an overview of contemporary speech enhancement techniques. They inferred that spectral subtraction was the most efficient in enhancing speech corrupted by uncorrelated additive noise. The spectral subtraction method estimates the Fourier transform of the dean signal by removing an estimate of the power spectral density of the noise signal. The basic advantage of this approaches the implementation simplicity and low computational complexity . One major drawback of this technique is the annoying no stationary "musical noise which is the residual noise consisting of narrow- band signals with the varying amplitudes and frequencies .

### Feature Extraction

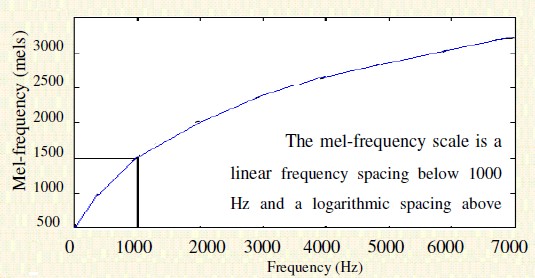
Feature extraction is a special form of dimensionality reduction, and this is done by extracting a specific feature from the speech, these features carry the characteristics of the speech which are different from one speaker to another, so these features will play the major role. At a high level, most speech feature extraction methods fall into the following two categories: modelling the human voice production system or modelling the peripheral auditory system. For the first approach, one of the most popular features is a group of cepstral coefficients derived from linear prediction known as the linear prediction cepstral coefficients (LPCCs). The LPCC feature extraction utilizes an all- pole filter to model the human vocal tract with speech formants captured by the poles of the all-pole filter. The narrow band (e.g., up to 4 kHz) LPCC features work well in a clean environment.

For the second approach, there are two groups of features, based on either Fourier transforms (FTs) or auditory-based Transforms(ATs). Representative for the first group are the Mel frequency cepstral coefficients (MFCCs), where a fast Fourier transform (FFT) is applied to generate the spectrum in the linear scale, and then a bank of band- pass filters is placed along a Mel frequency scale on top of the FFT output. Alternatively, the FFT output is warped to a Mel or Bark scale and then a bank of band- pass filters is placed linearly on top of the warped FFT output.

In the second group, where the auditory-based transform is defined as an invertible time–frequency transform. The output from this kind of transform can be in any kind of frequency scale (e.g., linear, Bark, ERB, etc.). Therefore, there is no need to place the bandpass filter in a Mel scale as in the MFCC or warp the frequency distributions

.The MFCC features in the first group are one of the most popular features for speech and speaker recognition. Like the LPCC features, the MFCC features perform well in clean environments but not in adverse environments or mismatched training and testing conditions.

MFCC’s are based on the Mel scale which is a heuristically derived perceptual scale. The Mel (from Melody) scale provides the relationship between perceived frequency or pitch, of a pure tone as a function of its acoustic frequency. In the Mel scale, to capture the phonetically important characteristics of speech of frequency *F* in Hz, a subjective pitch is measured in units known as *mel*. The reference point between this scale and normal frequency measurement is defined by equating a 1000 Hz tone, 40 dB

frequency scale is shown in Figure 1.6. The scale of Figure 1.6 is linear frequency spacing below 1000 Hz and logarithmic spacing above 1000Hz

**Figure 1.1: Linear frequency versus Mel frequency scale**

After the introduction of MFCC by Davis and Mermelstein, numerous variations and improvements of the original idea have been proposed. The variations differ mainly in the number of filters, the shape of the filters, the way the filters are spaced, the bandwidth of the filters, and the manner in which the power spectrum is warped. Also, it has been proposed to vary the frequency range of interest, the selection of the actual coefficient subset and the number of MFCC that are employed in the classification. The MFCC have been found to be an important result used in one step of modern ASR systems. Whilst the MFCC may differ depending on the method used to calculate the MFCC, the outcome is that the MFCC can be used flexibly in conjunction with the other steps in the SR process.

The most popular MFCC implementations are:

* MFCC FB-20 – introduced by Davis and Mermelstein,
* HTK MFCC FB-24 – from the Cambridge HMM Toolkit (HTK) described by Young
* MFCC FB-40 – from the MATLAB Auditory Toolbox developed by Slaney
* HFCC-E FB-29 (Human Factor Cepstral Coefficients) – proposed by Skowronski and Harris.

The endpoints of each one of the triangular filters are determined by the centre frequencies of adjacent filters, and therefore, the bandwidth of these filters is not an independent variable. More precisely, the bandwidths of the filters are determined by the spacing between the centre frequencies of the adjacent filters, which is a function of the sampling rate of the signal and the number of the filters in the filter bank. Therefore, for a given sampling frequency, increasing the number of filters results in a bandwidth decrease for each filter.

This does not provide an explanation for the choice of the shape of the filters, the overlap between them, the number of filters, nor does it explain how the overall design can be adapted for sampling frequencies different than the 10 kHz.

The Cambridge Hidden Markov Models (HMM) Toolkit (HTK) described another implementation of the MFCC that is now widely used. The designation HTK MFCC FB-24 reflects the number of filters (*M* = 24) recommended by HTK for an 8 kHz signal bandwidth. In the implementation of HTK, similar to the original approach of Davis and Mermelstein, a filter bank of equal height filters is assumed.

### Applications

Speech recognition technologies such as Alexa, Cortana, Google Assistant and Siri are changing the way people interact with their devices, homes, cars, and jobs. The technology allows us to talk to a computer or device that interprets what we’re saying in order to respond to our question or command.

* In-car systems.
* Health care.
* Military.
* Speech recognition digital assistants.
* In banking.

### Objectives

* MFCC algorithm have better accuracy and the errors are very less compared to the LPCC algorithm.
* Look up Tables are introduced for reducing the multiplications
* We provide a floating point algorithm to cover large dynamic range with small bit width.
* MAC units are required to reduce the calculations.
* Hidden markov model is used for better classification.
* ANN Classification is used for testing the given signal.

### Problem Statement

* The Feature extraction of speech using LPCC algorithm have more errors and less accuracy.
* In general process FFT consists of many multiplications due to this the system processing speed is very low .
* The feature extraction using MFCC with Fixed point cannot cover wide range with small bit width.

### Methodologies

To overcome the problem of LPCC algorithm we are using MFCC algorithm along with two classification process. One for training the input speaker and the other for testing the input speaker which increases the precision. Using floating point representation can cover wide range with small bit width.

### Research Gaps

The existing paper proposed the extraction with fixed point representation and the classifier used in existing paper was GMM classifier with this classifier the extraction was not accurate. In our model we are using floating point representation along HMM classifier for better accuracy.

### Motivation

Now a days we can see face recognition and finger print recognition in most number of places but there is another recognition that is speech recognition. The speech recognition is also another best way for recognition. Although the process is difficult but the accuracy is high in speech recognition. The only way to get better accuracy for this recognition is using a better feature extraction technique that is MFCC extraction method.

### Outline Thesis

This thesis is organized as follows.

In Chapter 1, The Introduction, problem statement, objective ,methodology. In Chapter 2, Study of literature.

In Chapter 3, Project study.

In Chapter 4, Software implementation for Speech Processing. In Chapter 5, Simulation Results.

In Chapter 6, Conclusion and Future Scope.

### Introduction

**Chapter 2**

## LITERATURE SURVEY

Feature extraction is a special form of dimensionality reduction, and this is done by extracting a specific feature from the speech, these features carry the characteristics of the speech which are different from one speaker to another, so these features will play the major role. At a high level, most speech feature extraction methods fall into the following two categories: modeling the human voice production system or modeling the peripheral auditory system.

For the first approach, one of the most popular features is a group of cepstral coefficients derived from linear prediction known as the linear prediction cepstral coefficients (LPCCs). The LPCC feature extraction utilizes an all-pole filter to model the human vocal tract with speech formants captured by the poles of the all-pole filter. The narrow band (e.g., up to 4 kHz) LPCC features work well in a clean environment. MFCC’s are based on the Mel scale which is a heuristically derived perceptual scale. The Mel (from Melody) scale provides the relationship between perceived frequency or pitch, of a pure tone as a function of its acoustic frequency. In the Mel scale, to capture the phonetically important characteristics of speech of frequency *F* in Hz, a subjective pitch is measured in units known as MEL.

Speech recognition is the process of automatically recognizing the spoken words of person based on information in speech signal. Recognition technique makes it possible to the speaker’s voice to be used in verifying their identity and control access to services such as voice dialing, banking by telephone, telephone shopping, database access services, information service, voice mail, security control for the confidential information areas, and remote access to Computers.

The acoustical parameters of spoken signal used in recognition tasks have been popularly studied and investigated, and being able to be categorized into two types of processing domain: First group is spectral based parameters and another is dynamic time series. The most popular spectral based parameter used in recognition approach is the Mel Frequency Cepstral Coefficients called MFCC [2,3].

Due to its advantage of less complexity in implementation of feature extraction algorithm, only sixteen coefficients of MFCC corresponding to the Mel scale frequencies of speech Cepstrum are extracted from spoken word samples

All extracted MFCC samples are then statistically analyzed for principal components, at least two dimensions minimally required in further recognition performance evaluation.

### Literature Survey

1. **Fundamental of speech recognition**

Speech recognition from speech helps us in improving the effectiveness of human- machine interaction. This paper presents a method to identify suitable features in DWT domain and improve good accuracy. In this work, 7 speeches (Berlin Database) are recognized using Support Vector Machine (SVM) classifier. Linear Predictive Cepstral Coefficients(LPCC), shimmer, spectral roll off, spectral flux, spectral centroid, pitch, short time energy and Harmonic to Noise Ratio (HNR) are considered as features. The obtained average accuracy is 82.14 % Earlier work done on emotion recognition using DWT coefficients yielded an accuracy of 63.63 % and 68.5% for 4 speeches on Berlin and Malayalam databases respectively. The proposed algorithm shows a significant increase in accuracy of about 15% to 20% for 7 speeches on Berlin database.

### A Modified MFCC Feature Extraction Technique For Robust Speaker Recognition

In Speaker Recognition (SR) system, feature extraction is one of the crucial steps where the particular speaker related information are extracted. The state of the art algorithm for this purpose is Mel Frequency Cepstral Coefficient (MFCC), and its complementary feature, Inverted Mel Frequency Cepstral Coefficient (IMFCC). MFCC is based on mel scale and IMFCC is based on inverted mel (imel) scale. In this paper, another complementary set of features are proposed which is also based on mel-imel scale, and the filtering operation makes these set of features different from MFCC and IMFCC. The Classification is done by using Gaussian Mixture Model (GMM) .

### Improvement of MFCC Feature Extraction Accuracy Using PCA Speech Recognition

For speech recognition system, Mel Frequency Cepstral Coefficients (MFCC) becomes a popular feature extraction method but it has various weaknesses especially about the accuracy level and the high of result feature dimension of the extraction method. This paper presents the combination of MFCC feature extraction method with Principal Component Analysis (PCA) to improve the accuracy in Indonesian speech recognition system. By combining MFCC and PCA, it was expected to increase the accuracy system

added with delta coefficients formed matrix data that later would be reduced using PCA. PCA method in the process of data reduction was designed to be two versions. Then the result of PCA reduction data was processed to the classification process using K-Nearest Neighbour (KNN) method. Composing the data was formed from 140 speech data that were recorded from 28 speakers. The research findings showed that adding PCA method version 1 could reduce the feature dimension from 26 to 12 by the same accuracy of speech recognition with the conventional MFCC method without PCA, that is 86.43%.

### Hardware Implementation Of Real-Time Speech Recognition System Using TMS320C6713 DSP

Continuous, real-time speech recognition is required for various mobile and hands-free applications. In this paper, hardware implementation of real-time speech recognition system is proposed using two approaches and their performances are evaluated. The first approach uses Mel Filter Banks with Mel Frequency Cepstrum Coefficients (MFCC) as feature input . The features extracted from input speech are fed to multi- class Support Vector Machine (SVM) classifier for recognition. The proposed recognition systems are implemented on a Texas Instruments TMS320C6713 floating point digital signal processor for recognizing isolated digits (0-9) . It is observed that the program memory required for MFCC feature extraction is 44.2%.

### Automatic Segmentation And Labeling Of Speech

Here the speech databases fully segmented and labeled using linguistic and speech research. And then they investigate an automatic approach to segmentation of labeled speech and labelling and segmentation of speech when only the orthographic transcription of speech is available The technique is based on a phone recognition system based on a gamma distribution phone duration models and a spectral model based on five different structures for phone models of varying contextual dependencies The alignment of speech with a given phone sequence is performed as d very constrained phone recognition task with the phonic model based only on the given phone sequence. When only orthographic description is pronged.

**Chapter 3**

## PROPOSED WORK

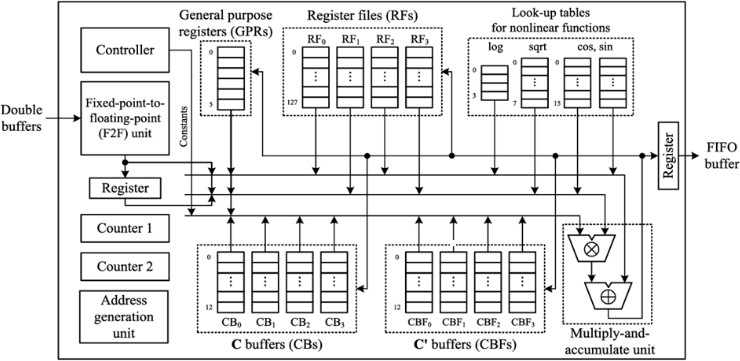
Among diverse human-device interfaces, speech recognition has widely been used in the last decade, and its importance becomes higher as the era of the Internet of Things comes close to reality . Due to the prevalence of energy-limited devices, energy- efficient architecture is inevitably demanded to lengthen the device life. The demand for low-energy architecture leads to the speech recognition system being implemented with dedicated hardware units . A speech recognition system consists of two processes:

1) feature extraction and 2) classification . The feature extraction process picks the characteristics of a sound frame, and a word is selected in the classification process by analyzing the extracted features. This brief mainly focuses on the hardware design of feature extraction. The most widely known feature extraction is based on the mel- frequency cepstrum coefficients (MFCCs), as MFCC-based systems are usually associated with hh recognition accuracy . In MFCC extraction was implemented with an optimized recognition program running on a low-power reduced instruction set computer processor platform. To reduce energy consumption further, dedicated architectures have been proposed in and constructed with fixed-point operations. The previous architectures, however, have not fully considered the arithmetic property of the MFCC extraction algorithm. This brief presents a new energy-efficient architecture for MFCC extraction. Investigating the algorithmic property of MFCC extraction, we renovate the previous architecture with optimization techniques to reduce both hardware complexity and computation time. As a result, the energy consumption is remarkably reduced compared with the previous architectures.

### Proposed Architecture

This approach is completely different from that have utilized a separate hardware unit for each operation. The proposed architecture is described with setting N to 256, M to 13, and L to 32. For sound signals sampled with 16 bits at 16 kHz, in addition, the bit- widths of F and E in the floating-point representation are determined to 6 and 7 bits, respectively. By analyzing the dataflow of the modified MFCC algorithm, we propose a new MFCC extraction system implementable with a small hardware cost. The overall architecture of the proposed system is shown in Fig 3.2, which consists of a multiply- and accumulate (MAC) unit, an address generation unit, a controller, memories, and counters. Though the proposed architecture has one MAC unit, it is sufficient to process

the entire MFCC extraction in real time. The constraint for real-time processing is that the MFCC vectors of a frame should be computed in a time limit corresponding to a half frame. Accordingly, a frame should be processed in 8 ms. The total number of MAC operations in the modified MFCC extraction is ∼15k to 2M operations can be performed.



**Fig 3.1 Modified architecture of MFCC**

Though the proposed architecture has one MAC unit, it is sufficient to process the entire MFCC extraction in real time. The constraint for real-time processing is that the MFCC vectors of a frame should be computed in a time limit corresponding to a half frame. Accordingly, a frame should be processed in 8 ms. The total number of MAC operations in the modified MFCC extraction is ∼15k to 2M operations can be performed.

For floating-point operations, the fixed-point-to-floating-point unit is included to convert the sound data loaded from the double buffers to the floating-point representation. The MAC unit processes floating-point multiplication and accumulation in serial. Each operator consists of small fixed-point adders and multipliers, and the resultant fraction F is normalized to ensure that 1 ≤ F < 2 if F != 0.

In terms of memory size, the proposed memory structure is more efficient than those of previous works, since it is shared with several processes.

To access an entity of a memory, the corresponding address is computed by employing counters. To fetch data for a MAC operation, each counter is increased by a certain amount. The proposed architecture utilizes two counters to generate two addresses needed to access two memories simultaneously.

### Floating Point System

Since many operations used in the MFCC algorithm depend on complex functions, such as square and logarithmic functions, their outputs are associated with a large dynamic range. Compared with the fixed-point representation, the floating-point representation can cover such a large dynamic range with a much smaller number of bits. In addition, the operation bit-width can be reduced further, grounded on the property that the resulting feature vectors are influenced by the order of magnitude of interim values. For these reasons, a floating-point representation is employed in this brief to implement the modified MFCC extraction algorithm described above.

### DDMFCC

DDMFCC refers to Delta Mel Frequency Cepstral Coefficients. It is obtained as the second derivative of MFCC (Mel Frequency Cepstral Coefficients). MFCC is one of the most important features, which is required among various kinds of speech applications. It shows high accuracy results for clean speech. They can be regarded as the "standard" features in speaker as well as speech recognition. However, experiments show that the parameterization of the MFC coefficients is best for discriminating speakers from the one usually used for speech recognition applications ***.***It is the most common algorithm that is used for speaker recognition system. The MFCC feature vectors that were extracted did not accurately capture the transitional characteristics of the speech signal which contains the speaker specific information. It is possible to obtain more detailed speech features by using a derivation on the MFCC acoustic vectors. This approach permits the computation of the DMFCCs, as the first order derivatives of the MFCC. Then, the DDMFCCs are derived from DMFCC, being the second order derivatives of MFCCs. The speech features which are the time derivatives of the spectrum-based speech features are known as dynamic speech features.

### DDMFCC Algorithm

This is the block diagram for the feature extraction processes applying DDMFCC \



Input Signal

Framing and windowing

FFT

MEL-Scaled Filter bank



DCT

1st Derivative

LOG

2nd Derivative

MFCC OUTPUT

**Figure 3.2: DDMFCC Flow diagram**

### Pre-processing

To enhance the accuracy and efficiency of the extraction processes, speech signals are normally pre-processed before features are extracted. Speech signal pre-processing covers digital filtering and speech signal detection.

In general, the digitized speech waveform has a high dynamic range and suffers from additive noise. In order to reduce this range, pre-emphasis is applied. This pre-emphasis is done by using a first-order FIR high-pass filter.

In the time domain, with input *x*[*n*] and 0.9 ≤ a ≤ 1.0, the filter equation

*y*[*n*]=*x*[*n*]−a *.x*[*n*−1] ...(3.1)

And the transfer function of the FIR filter in z-domain is:

*H*(*Z*) = 1−α.*z*−1 , 0.9 ≤ α≤1.0 ...(3.2)

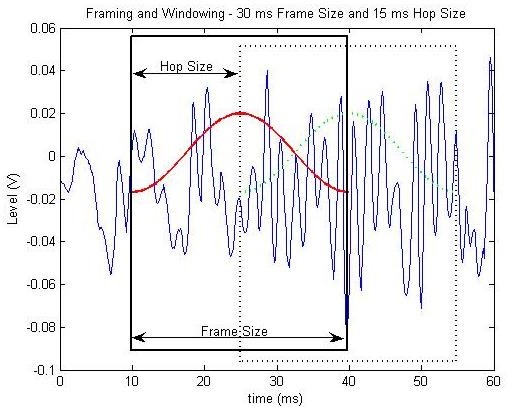
Where α is the pre-emphasis parameter

The pre-emphasizer is implemented as a fixed coefficient filter or as an adaptive one, where the coefficient a is adjusted with time according to the auto-correlation values of the speech. The aim of this stage is to boost the amount of energy in the high frequencies. The drop in energy across frequencies (which is called spectral tilt) is caused by the nature of the glottal pulse. Boosting the high frequency energy makes

information from these higher formants available to the acoustic model. The pre- emphasis filter is applied on the input signal before windowing.

### Framing and Windowing

In figure 3.2, First the signal is split up into several frames such that we are analyzing each frame in the short time instead of analyzing the entire signal at once, at the range (10-30) ms the speech signal is for the most part stationary .Also an overlapping is applied to frames. This is called the Hop Size. In most cases half of the frame size is used for the hop size. The reason for this is because on each individual frame, a hamming window is applied which will get rid of some of the information at the beginning and end of each frame. Overlapping will then reincorporate this information back into our extracted features.



**Figure 3.3: Framing and windowing**

Windowing is performed to avoid unnatural discontinuities in the speech segment and distortion in the underlying spectrum**.** In speaker recognition, the most commonly used window shape is the hamming window **.**It gradually attenuates the amplitude at both ends of extraction interval to prevent an abrupt change at the endpoints. It produces the convolution for the Fourier transform of the window function and the speech spectrum. The hamming window WH(n) , defined as :-

𝑊𝐻

(𝑛) = 0.54 − 0.46 cos ( 2𝑛𝜋 ) (3.3)

𝑁−1

### Fast Fourier Transform

To convert the signal from time domain to frequency domain preparing to the next stage (Mel frequency wrapping). The basis of performing fourier transform is to convert the convolution of the glottal pulse and the vocal tract impulse response in the time domain into multiplication in the frequency domain .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.

### Mel Scaled Filter Bank

The speech signal consists of tones with different frequencies. For each tone with an actual Frequency, f, measured in Hz, a subjective pitch is measured on the ‘Mel’ scale. The mel-frequency scale is a linear frequency spacing below 1000Hz and a logarithmic spacing above 1000Hz.

We can use the following formula to compute the mels for a given frequency f in Mel(f)=2595\*log10(1+f/700) (3.4)

One approach to simulating the subjective spectrum is to use a filter bank, one filter for each desired Mel frequency component. In figure 3.3, the filter bank has a triangular band pass frequency response. Mel-Frequency analysis of speech is based on human perception experiments **.** Human ears, for frequencies lower than 1 kHz, hears tones with a linear scale instead of logarithmic scale for the frequencies higher than 1 kHz. The information carried by low frequency components of the speech signal is more important compared to the high frequency components. In order to place more emphasize on the low frequency components, mel scaling is performed. Mel filter banks are non-uniformly spaced on the frequency axis, so we have more filters in the low frequency regions and less number of filters in high frequency regions.

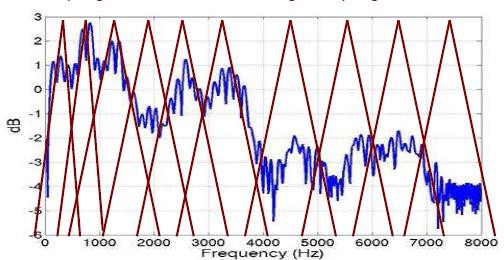
S(l)=∑S(K) M(K) ….(3.5)

Where,

S(l) : Mel spectrum. S(K) : Original spectrum. M(K) : Mel filterbank.

l=0, 1,. , L-1 , Where L is the total number of mel filterbanks.

N/2 = Half FFT size



### Cepstrum

**Figure 3.4: Filter banks**

In the final step, the log mel spectrum has to be converted back to time. The result is called the mel frequency cepstrum coefficients (MFCCs). The cepstral representation of the speech spectrum provides a good representation of the local spectral properties of the signal for the given frame analysis. Because the mel spectrum coefficients are real numbers (and so are their logarithms), they may be converted to the time domain using the Discrete Cosine Transform (DCT).As shown in figure3.4 cepstrum is obtained by taking the logarithm and DCT of mel filter bank output.

DCT

LOG

MEL

Cepstrum

**Figure 3.5: Mel Cepstrum Coefficients**

It is known that the logarithm has the effect of changing multiplication into addition. Therefore we can simply convert the multiplication of the magnitude of the fourier transform into addition. Then, by taking the inverse FFT or DCT of the logarithm of the magnitude spectrum, the glottal pulse and the impulse response can be separated. The IFFT needs complex arithmetic than DCT. The DCT implements the same function as the FT more efficiently by taking advantage of the redundancy in a real signal. The DCT is more efficient computationally.

The MFCCs may be calculated using this equation

𝑐 = ∑𝑘 (𝑙𝑜𝑔 𝑆

)[n(k-1)𝜋] (3.6)

𝑛 𝑘=1

𝑘 2 𝑘

Where n=1,2,….K

The number of mel cepstrum coefficients, K, is typically chosen as (10-15). The first component, *c*~0 , is excluded from the DCT since it represents the mean value of the input signal which carries little speaker specific information. Since the log power spectrum is real and symmetric, inverse FFT reduces to a Discrete Cosine Transform (DCT). By applying the procedure described above, for each speech frame of about 30 ms with overlap, a set of mel-frequency cepstrum coefficients is computed. This set of coefficients is called an acoustic vector.

### Classification Process

The process of estimating the VQ codebook involves division of the observed data into clusters. The centroid of each cluster becomes the codeword representing that cluster. The set of all centroids constitutes the VQ codebook. If the Cartesian distance measure is used, then the centroid simply represents the mean vector calculated from all vectors belonging to the given cluster. The best known VQ codebook generation algorithms used in speaker verification/recognition tasks include: the K-means algorithm, the Linde Buzo Gray (LBG) algorithm, the Kohonen’s self organizing map (KSOM) and Fuzzy C-means. In these algorithms the process of finding an optimal codebook is guided by minimization of the average distortion function (objective or cost function) representing an average total sum of distances between the original vectors and the codeword’s. It is also called the quantization error.

M is a feature modeling and classification algorithm widely used in speech based pattern recognition, since it can smoothly approximate a wide variety of density distributions. Adapted GMMs known as UBM-GMM and MAP-GMM are further enhanced speaker verification outcomes. The introduction of the adapted GMM algorithms has increased computational efficiency and strengthened the speaker verification optimization process. The Expectation Maximization (EM) algorithm is most commonly used to iteratively derive class models. The EM algorithm is initialized with a speaker model and estimates a new model at the end of algorithm iterations.The Hidden Markov Model (HMM) is created using continuous probability measures of GMM. HMM is used for text-dependent speaker recognition. In HMM, time-dependent parameters are observation symbols which are created by VQ codebook labels. The main assumption of HMM is that the current state depends on the previous state. In the

training phase, state transition probability distribution, observation symbol probability distribution and initial state probabilities are estimated for each speaker as a speaker model. The probability of observations for a given speaker model is calculated for speaker recognition.

### Training HMMs On Statistical Data

Having derived expressions for the effects of noise on the various parameters of the feature vector, it is necessary to obtain expressions for estimating the parameters of the corrupted speech HMMs. Instead of having observations, as in the case of standard HMM training, only statistics about the speech and noise are available. It is therefore necessary to modify the standard HMM training algorithms to allow for the training of models on statistical data. For this work a ML estimate will be used. The standard re- estimation formula of the new mean of mixture *Mm* of state *sj* of the corrupted-speech model can be expressed in terms of the expected values of observations instead of summations.

∑{𝐿𝑗𝑚(𝑐)𝑂𝑐(𝑐)}

𝜇𝑗𝑚 =

∑{𝐿

𝑗𝑚

……(3.7)

(𝑐)}

Similar expressions for the variance and mixture weights in terms of expected values may be obtained. As HMMs are used to model the training data there are no longer explicit observations at each time interval*.* Instead there are PDFs describing the observations for a particular state.

**CHAPTER 4**

## SOFTWARE SPECIFICATIONS

### Background On MATLAB

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include the following:

* + - Math and computation
    - Algorithm development
    - Data acquisition
    - Modeling, simulation, and prototyping
    - Data analysis, exploration, and visualization
    - Scientific and engineering graphics
    - Application development, including graphical user interface building MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. This allows formulating solutions to many technical computing problems, especially those involving matrix representations, in a fraction of the time it would take to write a program in a scalar non-interactive language such as C or Fortran. The name MATLAB stands for matrix laboratory. MATLAB was written originally to provide easy access to matrix software developed by the LINPACK and EISPACK projects. Today, MATLAB engines incorporate the LAPACK and BLAS libraries, constituting the state of the art in software for matrix computation. In university environments, MATLAB is the standard computational tool for introductory and advanced courses in mathematics, engineering, and science. In industry, MATLAB is the computational tool of choice for research, development, and analysis. MATLAB is complemented by a family of application specific solutions called toolboxes.

### [Numerical Computing](http://en.wikipedia.org/wiki/Numerical_analysis) MATLAB

MATLAB (matrix laboratory) is a [numerical computing](http://en.wikipedia.org/wiki/Numerical_analysis) environment and [fourth-](http://en.wikipedia.org/wiki/Fourth-generation_programming_language) [generation programming language.](http://en.wikipedia.org/wiki/Fourth-generation_programming_language) Developed by [MathWorks](http://en.wikipedia.org/wiki/MathWorks), MATLAB allows [matrix](http://en.wikipedia.org/wiki/Matrix_(mathematics)) manipulations, plotting of [functions](http://en.wikipedia.org/wiki/Function_(mathematics)) and data, implementation of [algorithms](http://en.wikipedia.org/wiki/Algorithm), creation of [user interfaces](http://en.wikipedia.org/wiki/User_interface), and interfacing with programs written in other languages, including [C](http://en.wikipedia.org/wiki/C_(programming_language)), [C++](http://en.wikipedia.org/wiki/C%2B%2B), and [Fortran.](http://en.wikipedia.org/wiki/Fortran)

Although MATLAB is intended primarily for numerical computing, an optional toolbox uses the [MuPAD](http://en.wikipedia.org/wiki/MuPAD) [symbolic engine](http://en.wikipedia.org/wiki/Computer_algebra_system), allowing access to [symbolic computing](http://en.wikipedia.org/wiki/Symbolic_computing) capabilities. An additional package, [Simulink](http://en.wikipedia.org/wiki/Simulink), adds graphical multi-domain simulation and [Model-Based Design](http://en.wikipedia.org/wiki/Model_based_design) for [dynamic](http://en.wikipedia.org/wiki/Dynamical_system) and [embedded systems](http://en.wikipedia.org/wiki/Embedded_systems).

In 2004, MATLAB had around one million users across industry and academia. MATLAB users come from various backgrounds of [engineering](http://en.wikipedia.org/wiki/Engineering), [science](http://en.wikipedia.org/wiki/Science), and [economics.](http://en.wikipedia.org/wiki/Economics) MATLAB is widely used in academic and research institutions as well as industrial enterprises.

### History

MATLAB was created in the late 1970s by [Cleve Moler](http://en.wikipedia.org/wiki/Cleve_Moler), the chairman of the [computer](http://en.wikipedia.org/wiki/Computer_science) [science](http://en.wikipedia.org/wiki/Computer_science) department at the [University of New Mexico](http://en.wikipedia.org/wiki/University_of_New_Mexico). He designed it to give his students access to [LINPACK](http://en.wikipedia.org/wiki/LINPACK) and [EISPACK](http://en.wikipedia.org/wiki/EISPACK) without having to learn [Fortran.](http://en.wikipedia.org/wiki/Fortran) It soon spread to other universities and found a strong audience within the [applied mathematics](http://en.wikipedia.org/wiki/Applied_mathematics) community. [Jack Little](http://en.wikipedia.org/wiki/John_N._Little), an engineer, was exposed to it during a visit Moler made to [Stanford University](http://en.wikipedia.org/wiki/Stanford_University) in 1983. Recognizing its commercial potential, he joined with Moler and Steve Bangert. They rewrote MATLAB in [C](http://en.wikipedia.org/wiki/C_(programming_language)) and founded [MathWorks](http://en.wikipedia.org/wiki/MathWorks) in 1984 to continue its development.

These rewritten libraries were known as JACKPAC. In 2000, MATLAB was rewritten to use a newer set of libraries for matrix manipulation, [LAPACK](http://en.wikipedia.org/wiki/LAPACK).

MATLAB was first adopted by [control design engineers](http://en.wikipedia.org/wiki/Control_engineering), Little's specialty, but quickly spread to many other domains. It is now also used in education, in particular the teaching of [linear algebra](http://en.wikipedia.org/wiki/Linear_algebra) and [numerical analysis](http://en.wikipedia.org/wiki/Numerical_analysis), and is popular amongst scientists involved with [image processing.](http://en.wikipedia.org/wiki/Image_processing)

### Interfacing With Other Languages

MATLAB can call functions and subroutines written in the [C programming language](http://en.wikipedia.org/wiki/C_(programming_language)) or [Fortran.](http://en.wikipedia.org/wiki/Fortran) A wrapper function is created allowing MATLAB data types to be passed and returned. The dynamically loadable object files created by compiling such functions are termed "[MEX-files](http://en.wikipedia.org/wiki/MEX_file)" (for MATLAB executable).Libraries written in [Java](http://en.wikipedia.org/wiki/Java_(programming_language)), [ActiveX](http://en.wikipedia.org/wiki/ActiveX) or [.NET](http://en.wikipedia.org/wiki/.NET_Framework) can be directly called from MATLAB and many MATLAB libraries (for example [XML](http://en.wikipedia.org/wiki/XML) or [SQL](http://en.wikipedia.org/wiki/SQL) support) are implemented as wrappers around Java or ActiveX libraries.

Calling MATLAB from Java is more complicated, but can be done with MATLAB extension, which is sold separately by [MathWorks](http://en.wikipedia.org/wiki/MathWorks), or using an undocumented

mechanism called JMI (Java-to-Matlab Interface), which should not be confused with the unrelated [Java Metadata Interface](http://en.wikipedia.org/wiki/Java_Metadata_Interface) that is also called JMI.

As alternatives to the [MuPAD](http://en.wikipedia.org/wiki/MuPAD) based Symbolic Math Toolbox available from MathWorks, MATLAB can be connected to [Maple](http://en.wikipedia.org/wiki/Maple_(software)) or [Mathematica.](http://en.wikipedia.org/wiki/Mathematica)

MATLAB has a direct node with [modeFRONTIER,](http://www.modefrontier.com/) a multidisciplinary and multi- objective optimization and design environment, written to allow coupling to almost any computer aided engineering (CAE) tool. Once obtained a certain result using Matlab, data can be transferred and stored in a [modeFRONTIER](http://www.modefrontier.com/) workflow and viceversa.

### Technical Computing

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. MATLAB features a family of application-specific solutions called toolboxes.

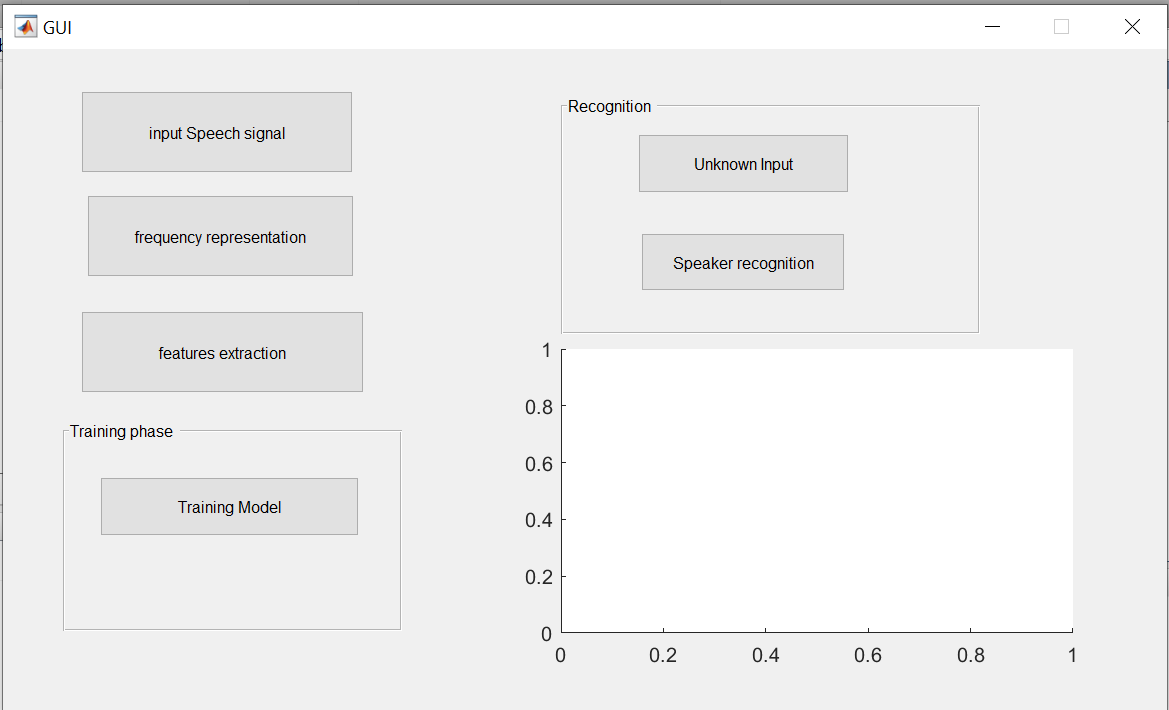
Very important to most users of MATLAB, toolboxes allow you to learn and apply specialized technology. Toolboxes are comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular classes of problems. Areas in which toolboxes are available include signal processing, control systems, neural networks, fuzzy logic, wavelets, simulation, and many others.

**Chapter 5**

## RESULTS

### The Graphical User Interface

A graphical user interface (GUI) is a pictorial interface to a program. A good GUI can make programs easier to use by providing them with a consistent appearance and with intuitive controls like pushbuttons, list boxes, sliders, menus, and so forth. The GUI should behave in an understandable and predictable manner, so that a user knows what to expect when he or she performs an action. For example, when a mouse click occurs on a pushbutton, the GUI should initiate the action described on the label of the button. This chapter introduces the basic elements of the MATLAB GUIs. The chapter does not contain a complete description of components or GUI features, but it does provide the basics required to create functional GUIs for your programs.



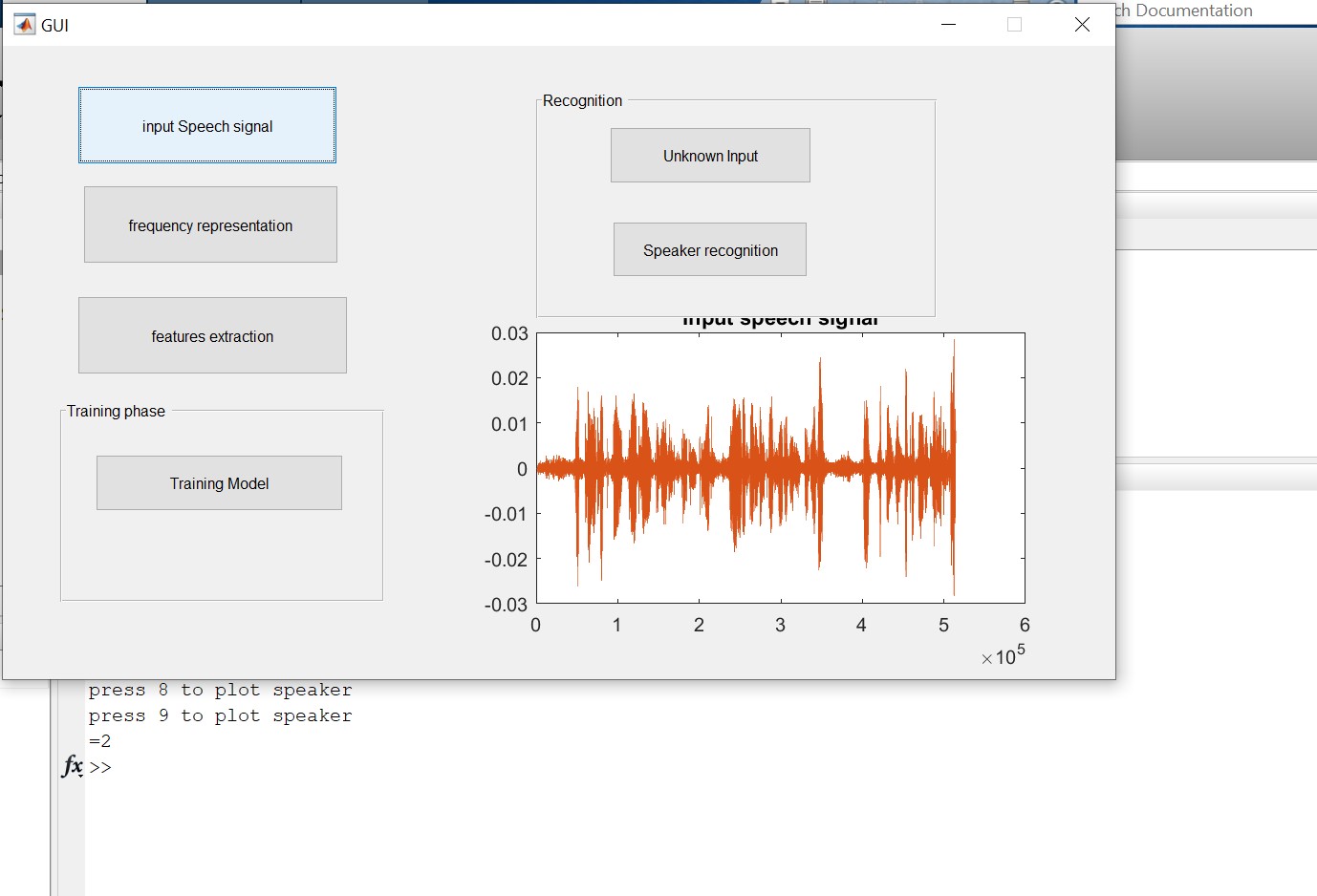
**Fig 5.1 Graphical user interface.**

Each item on a MATLAB GUI (pushbuttons, labels, edit boxes, etc.) is a graphical component. The types of components include graphical controls (pushbuttons, edit boxes, lists, sliders, etc.), static elements (frames and text strings), menus, and axes. Graphical controls and static elements are created by the function uicontrol, and menus are created by the functions uimenu and uicontextmenu. Axes, which are used to display graphical data, are created by the function axes.

### Simulation Process

Train & Test : Dividing the samples into training set and testing set. Training set is a standard sample, and the testing set is the samples from different person in different environment. The author will train the HMM model based on the training sample set and test the system based on the testing set.

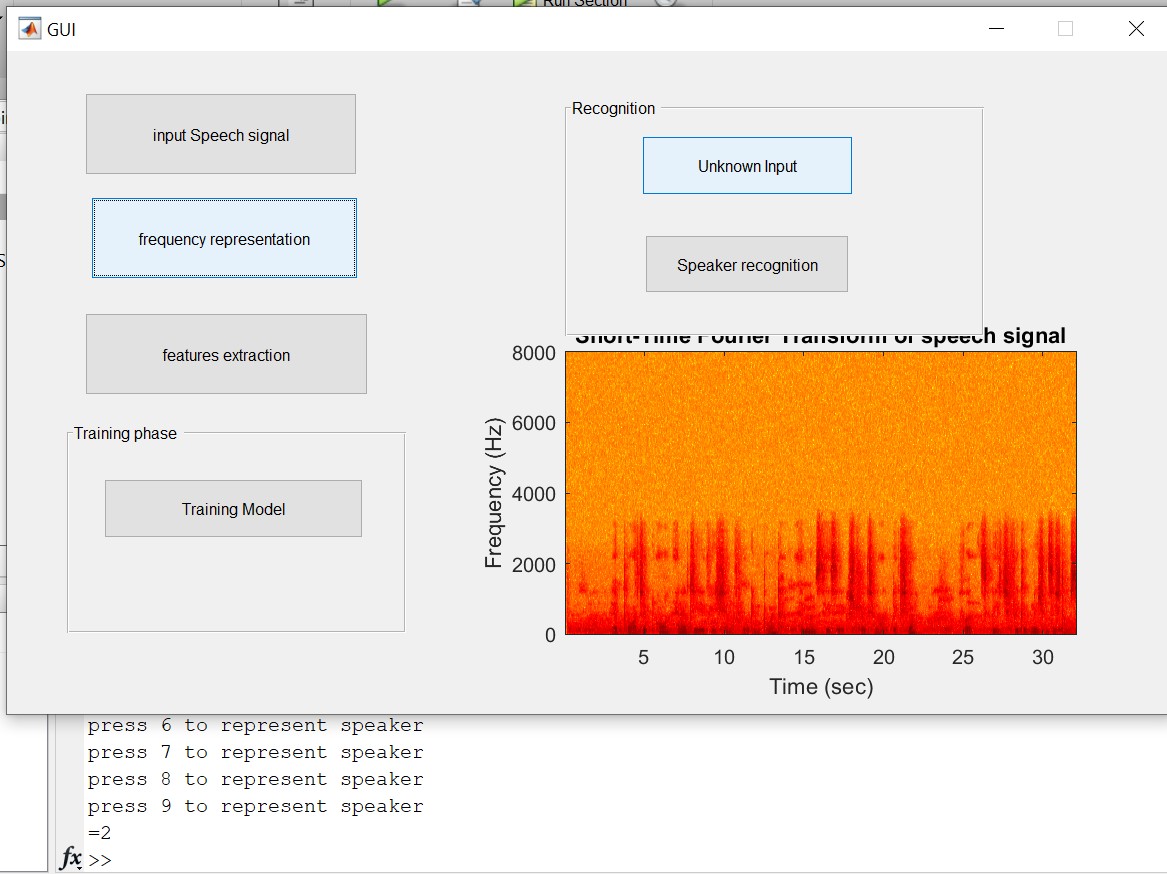
**Input speech**: The input speech signal is given in the format of (.wav). The WAV is an audio file format that stores wave form data. What makes the WAV different from other audio formats is it's uncompressed – making it much larger than something like an mp3. It's a raw audio file capable of saving recordings using different bitrates .The speech signal given for input is “Energy Efficient Floating Point MFCC Extraction Architecture For Speech Recognition Systems”. The recorded signal is not a noise free signal.



**Fig 5.2 The plot for a speaker two**

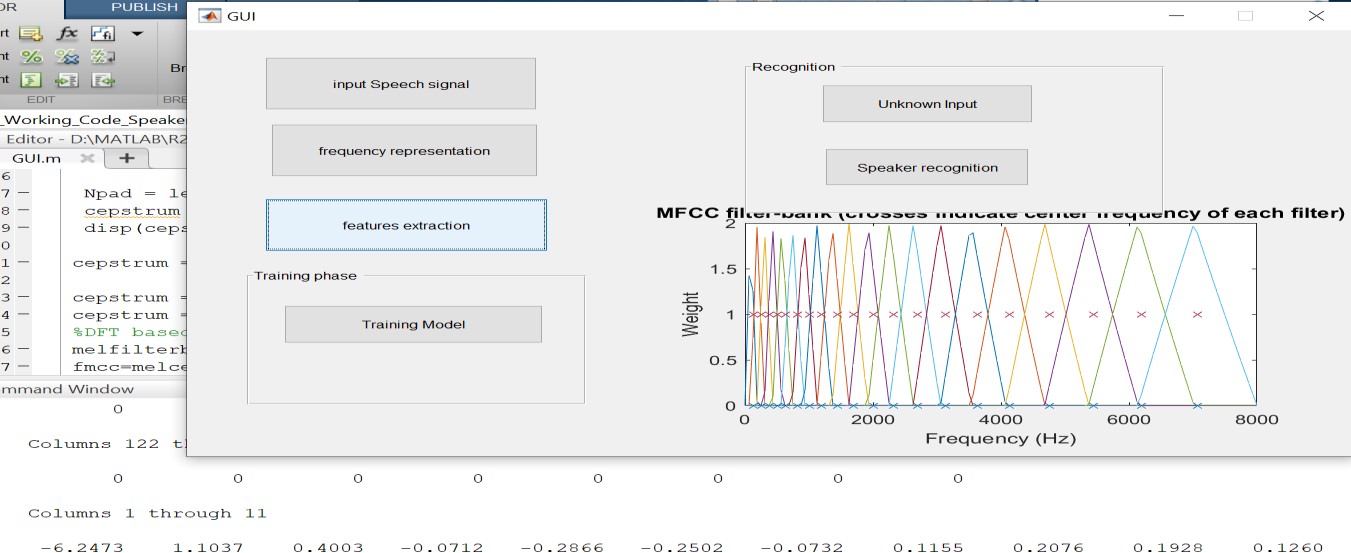
**Frequency Representation:** In the input speech signal function we convert the speech signal into frames. These frames must undergo windowing process of blackmann window. These signal is converted to frequency domain using short time fourier transform and fast fourier transform. The difference is the short time fourier transform

converts only the small signals into speech signal and the FFT can convert only the large signals so we are using to forms. The axis is used for plotting the frequency converted signal of one speaker.



**Fig 5.3 Frequency representation of speaker two.**

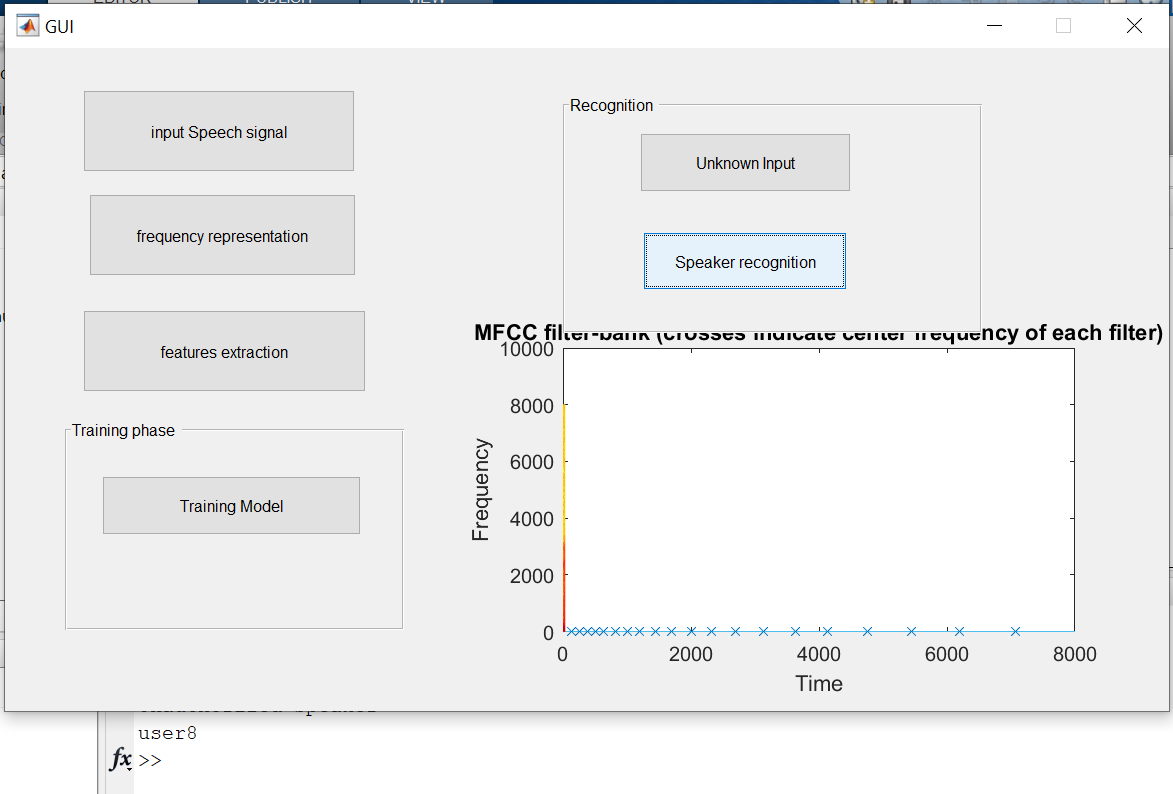
**Feature Extraction**: This is the main process where we are converting the speech signal into some coefficients for training and testing the data. Here we use MFCC process for frequency extraction. The output is represented in mel filter bank format. Here we are extracting two features are MFCC coefficients, the other is logarithmic coefficient.



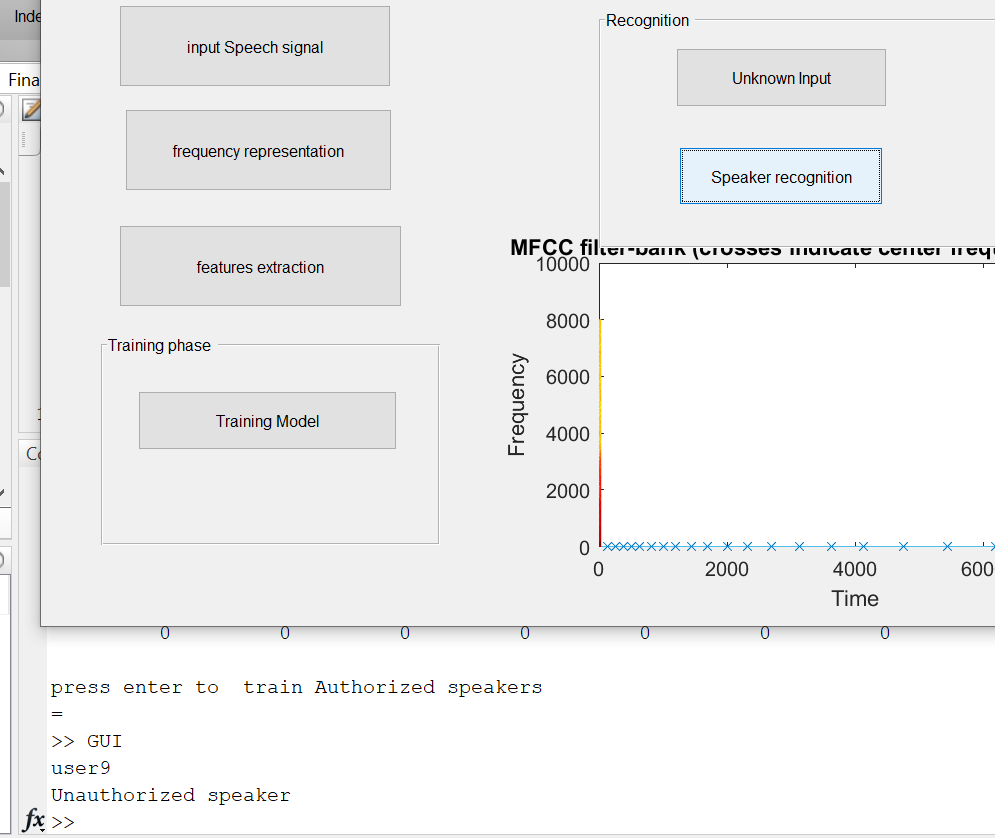
**Fig 5.4 Plot of coefficients.**

**Training :** The training is performed using HMM model. Training data “teaches” an algorithm to recognize patterns in a dataset. More specifically, training data is the dataset you use to train your algorithm or model so it can accurately predict your outcome.

**Testing :** The algorithm can make every gaussian distribution of the HMM model as one kind, The parameter of Mean is the position in Characteristic parameters space of different sample Characteristic parameters, which is the most important parameter in HMM, The variance is the density of data distribution, the weight is the number of data. After performing these two processes the model is ready for identification. Here total 9 speech signals are trained from those 9 signals if any one signal is given for test in testing process then we get the out as the given user signal for test. If any other signal which is not from trained signal then we get the output as “UNAUTHORISED SPEAKER”.



**Fig 5.5 Recognition of speaker**



**Fig 5.6 Recognition of unauthorized speaker.**

* 1. **Conclusion**

**Chapter 6**

## CONCLUSION

An energy-efficient MFCC extraction architecture has been presented for speech recognition. The MFCC extraction algorithm is modified to minimize computation time without degrading the recognition accuracy noticeably. In addition, the proposed architecture employs floating-point arithmetic operations to minimize the operation bit- width and the total size of LUTs, while have relied on fixed-point operators. The effectiveness of energy consumption makes the proposed architecture a promising solution for energy-limited speech recognition systems.

### Future Scope

Feature extraction is the first crucial component in automatic speech processing. Generally speaking, successful front-end features should carry enough discriminative information for enhancements, and it should fit well with the back-end modelling, and be robust with respect to the changes of acoustic environments. As a part of the project work the MFCC feature extraction were analyzed for better understanding of the work. To extend this work for better performances of auto segmentation process in detail analyzes of speech over various cestrum scales like bark scale and its robustness over different noise conditions will be evaluated.

## REFERENCES

1. N.-V. Vu, J. Whittington, H. Ye, and J. Devlin, “implementation of the MFCC front-end for low-cost speech recognition systems,” in proc. ISCAS, may/jun. 2010, pp. 2334–2337.
2. P. Elkan, T. Allen, and S. F. Quigley, “FPGA implementation for gym-based speaker identification,” int. J. Recon fig. Compute., Vol. 2011, no. 3, pp. 1–8, jan. 2011, art. Id 420369
3. R. Ramos-lara, M. López-garcía, E. Cantó-navarro, and L. Puente-rodriguez, “real- time speaker verification system implemented on reconfigurable hardware,” J. Signal process. Syst., Vol. 71, no. 2, pp. 89–103, may 2013
4. W. Han, C.-F. Chan, C.-S. Choy, and K.-P. Pun, “an efficient MFCC extraction method in speech recognition,” in proc. IEEE ISCAS, may 2006, pp. 145–148.
5. D. A. Sunderland, R. A. Strauch, S. S. Wharfield, H. T. Peterson, and C. R. Cole, “CMOS/SOS frequency synthesizer LSI circuit for spread spectrum communications,” IEEE J. Solid-state circuits, vol. 19, no. 4, pp. 497–506, aug. 1984.
6. Qi.Li, “An auditory-based feature extraction algorithm for Robust speaker Identification under mismatched conditions,” IEEE Transactions on audio, Speech, and language processing, Vol.19, No. 6, August 2011.
7. Tze Fen Li, Shui-Ching Chang, “ Speech recognition of mandarin syllables using both linear predict coding cepstra and Mel frequency cepstra,”2009.
8. Zhou H., and Aziz A., (1998) “Simultaneous PTL buffer insertion and sizing for minimizing Elmore delay,” in Proc. Int. Workshop Logic Synth., pp. 162
9. Lalitha V, and Kathiravan S, “A Review of Manchester, Miller, and FM0 Encoding Techniques”, Smart Computing Review, vol. 4, no. 6, pp.481-490 December 2014.6.

V. Hemalatha, P. Srividhya.

1. Hung.Y.C, Kuo.M.M, Tung.C.-K., and S.-H. Shieh, (2009) “High speed CMOS chip design for Manchester and Miller encoder,” in Proc. Intel. Inf. Hiding Multimedia Signal Process., pp. 538–541.

**Appendix-I**

## Source Code

function varargout = GUI(varargin) gui\_Singleton = 1;

gui\_State = struct('gui\_Name', mfilename, ... 'gui\_Singleton', gui\_Singleton, ... 'gui\_OpeningFcn', @GUI\_OpeningFcn, ... 'gui\_OutputFcn', @GUI\_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 GUI is made visible.

function GUI\_OpeningFcn(hObject, eventdata, handles, varargin)

% This function has no output args, see OutputFcn.

% hObject handle to figure

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% varargin command line arguments to GUI (see VARARGIN)

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

% Update handles structure

guidata(hObject, handles);

% UIWAIT makes GUI wait for user response (see UIRESUME)

% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line. function varargout = GUI\_OutputFcn(hObject, eventdata, handles)

% varargout cell array for returning output args (see VARARGOUT);

% hObject handle to figure

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure varargout{1} = handles.output;

% --- Executes on button press in pushbutton1.

function pushbutton1\_Callback(hObject, eventdata, handles)

% hObject handle to pushbutton1 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA) axes(handles.axes1);

Fs=16000;

disp('press 1 to plot speaker ') disp('press 2 to plot speaker ') disp('press 3 to plot speaker ') disp('press 4 to plot speaker ') disp('press 5 to plot speaker ') u=input('=');

if u == 1

y = audioread('user1.wav'); elseif u == 2

y = audioread('user2.wav'); elseif u == 3

y = audioread('user3.wav');

elseif u == 4

y = audioread('user4.wav'); elseif u == 5

end

[speech\_length temp] = size(y); frame\_length = 256;

number\_of\_frames = floor(speech\_length / frame\_length); numberoffeatures = 12;

plot(y);

title ('Input speech signal');

% --- Executes on button press in pushbutton2.

function pushbutton2\_Callback(hObject, eventdata, handles)

% hObject handle to pushbutton2 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA) axes(handles.axes1);

Fs=16000;

disp('press 1 to represent speaker ') disp('press 2 to represent speaker ') disp('press 3 to represent speaker ') disp('press 4 to represent speaker ') disp('press 5 to represent speaker ') u=input('=');

if u == 1

y = audioread('user1.wav'); elseif u == 2

y = audioread('user2.wav'); elseif u == 3

y = audioread('user3.wav'); elseif u == 4

y = audioread('user4.wav'); elseif u == 5

y = audioread('user5.wav'); end

[speech\_length temp] = size(y); frame\_length = 256;

number\_of\_frames = floor(speech\_length / frame\_length); numberoffeatures = 12;

grid;

Nfft=512;

window = blackman(Nfft);

%wvtool(blackman(window)) Fs=16000;

specgram(y(:,1),Nfft,Fs,window,round(3/4\*length(window))); ylabel('Frequency (Hz)');

xlabel('Time (sec)');

title('Short-Time Fourier Transform of speech signal')

% --- Executes on button press in pushbutton3.

function pushbutton3\_Callback(hObject, eventdata, handles)

% hObject handle to pushbutton3 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA) axes(handles.axes1);

Fs=16000;

disp('press enter to extract speakers parameters') y = audioread('user1.wav');

[speech\_length temp] = size(y) frame\_length = 256;

number\_of\_frames = floor(speech\_length / frame\_length); numberoffeatures = 12;

grid;

Nfft=512;

window = blackman(Nfft);

Fs=16000;

specgram(y(:,1),Nfft,Fs,window,round(3/4\*length(window))); for i=1:number\_of\_frames

begin\_sample = 1+(i-1)\*frame\_length; end\_sample = i\*frame\_length;

s = y(begin\_sample:end\_sample ); s = s(:);

sw = s.\*hamming(frame\_length); dft\_rc\_temp = log ( abs( fft( sw ) ) ); dft\_rc = real( ifft( dft\_rc\_temp) );

dft\_rc = dft\_rc(1:floor(frame\_length/2 + 1)); dft\_rc = dft\_rc(1:numberoffeatures); dftcepstrumstra(i,:)=dft\_rc';

cepstrum = dftcepstrumstra(i,:); Nfft = frame\_length;

lengthOfPositivePart = floor(Nfft/2 + 1);

Npad = lengthOfPositivePart - length(cepstrum); cepstrum = [cepstrum zeros(1,Npad)]; disp(cepstrum)

cepstrum = fft(cepstrum); cepstrum = exp(real(cepstrum)); cepstrum = 20\*log10(cepstrum);

%DFT based mel-cepstrumstra melfilterbankorder = 20;

fmcc=melcepst(s,Fs,'Mta0',numberoffeatures,melfilterbankorder,frame\_length,frame\_ length,0,0.5);

dftmelcepstrumstra(i,:)=fmcc; melcepstrum = fmcc;

melcepstrum = [melcepstrum zeros(1,melfilterbankorder-length(melcepstrum))]; melcepstrum = exp(melcepstrum);

melcepstrum = 20\*log10(melcepstrum) - 20; filterAmplitudes=full(melbankm(melfilterbankorder ,frame\_length,Fs)); peak = max(filterAmplitudes');

for index = 1:length(peak)

filterCenter(index) = find(filterAmplitudes(index,:)==peak(index)); end

end cepstrum=cepstrum;

save('C:\S\param1.mat','cepstrum');

%user1 feature extaction. y = audioread('user2.wav');

[speech\_length temp] = size(y); frame\_length = 256;

number\_of\_frames = floor(speech\_length / frame\_length); numberoffeatures = 12;

grid;

Nfft=512;

window = blackman(Nfft); Fs=16000;

specgram(y(:,1),Nfft,Fs,window,round(3/4\*length(window))); for i=1:number\_of\_frames

begin\_sample = 1+(i-1)\*frame\_length; end\_sample = i\*frame\_length;

s = y(begin\_sample:end\_sample ); s = s(:);

sw = s.\*hamming(frame\_length); dft\_rc\_temp = log ( abs( fft( sw ) ) ); dft\_rc = real( ifft( dft\_rc\_temp) );

dft\_rc = dft\_rc(1:floor(frame\_length/2 + 1)); dft\_rc = dft\_rc(1:numberoffeatures); dftcepstrumstra(i,:)=dft\_rc';

cepstrum = dftcepstrumstra(i,:); Nfft = frame\_length;

lengthOfPositivePart = floor(Nfft/2 + 1);

Npad = lengthOfPositivePart - length(cepstrum); cepstrum = [cepstrum zeros(1,Npad)]; disp(cepstrum)

cepstrum = fft(cepstrum); cepstrum = exp(real(cepstrum)); cepstrum = 20\*log10(cepstrum);

%DFT based mel-cepstrumstra melfilterbankorder = 20;

fmcc=melcepst(s,Fs,'Mta0',numberoffeatures,melfilterbankorder,frame\_length,frame\_ length,0,0.5);

dftmelcepstrumstra(i,:)=fmcc; melcepstrum = fmcc;

melcepstrum = [melcepstrum zeros(1,melfilterbankorder-length(melcepstrum))]; melcepstrum = exp(melcepstrum);

melcepstrum = 20\*log10(melcepstrum) - 20; filterAmplitudes=full(melbankm(melfilterbankorder ,frame\_length,Fs)); peak = max(filterAmplitudes');

for index = 1:length(peak)

filterCenter(index) = find(filterAmplitudes(index,:)==peak(index)); end

end cepstrum1=cepstrum;

save('C:\S\param2.mat','cepstrum1'); y = audioread('user3.wav'); [speech\_length temp] = size(y); frame\_length = 256;

number\_of\_frames = floor(speech\_length / frame\_length); numberoffeatures = 12;

grid;

Nfft=512;

window = blackman(Nfft); Fs=16000;

specgram(y(:,1),Nfft,Fs,window,round(3/4\*length(window))); for i=1:number\_of\_frames

begin\_sample = 1+(i-1)\*frame\_length; end\_sample = i\*frame\_length;

s = y(begin\_sample:end\_sample ); s = s(:);

sw = s.\*hamming(frame\_length); dft\_rc\_temp = log ( abs( fft( sw ) ) ); dft\_rc = real( ifft( dft\_rc\_temp) );

dft\_rc = dft\_rc(1:floor(frame\_length/2 + 1)); dft\_rc = dft\_rc(1:numberoffeatures); dftcepstrumstra(i,:)=dft\_rc';

cepstrum = dftcepstrumstra(i,:); Nfft = frame\_length;

lengthOfPositivePart = floor(Nfft/2 + 1);

Npad = lengthOfPositivePart - length(cepstrum); cepstrum = [cepstrum zeros(1,Npad)]; disp(cepstrum)

cepstrum = fft(cepstrum); cepstrum = exp(real(cepstrum)); cepstrum = 20\*log10(cepstrum);

%DFT based mel-cepstrumstra melfilterbankorder = 20;

fmcc=melcepst(s,Fs,'Mta0',numberoffeatures,melfilterbankorder,frame\_length,frame\_ length,0,0.5);

dftmelcepstrumstra(i,:)=fmcc; melcepstrum = fmcc;

melcepstrum = [melcepstrum zeros(1,melfilterbankorder-length(melcepstrum))]; melcepstrum = exp(melcepstrum);

melcepstrum = 20\*log10(melcepstrum) - 20; filterAmplitudes=full(melbankm(melfilterbankorder ,frame\_length,Fs)); peak = max(filterAmplitudes');

for index = 1:length(peak)

filterCenter(index) = find(filterAmplitudes(index,:)==peak(index)); end

end cepstrum2=cepstrum;

save('C:\S\param3.mat','cepstrum2'); y = audioread('user2.wav'); [speech\_length temp] = size(y); frame\_length = 256;

number\_of\_frames = floor(speech\_length / frame\_length); numberoffeatures = 12;

grid;

Nfft=512;

window = blackman(Nfft); Fs=16000;

specgram(y(1:1079,1),Nfft,Fs,window,round(3/4\*length(window))); for i=1:number\_of\_frames

begin\_sample = 1+(i-1)\*frame\_length; end\_sample = i\*frame\_length;

s = y(begin\_sample:end\_sample ); s = s(:);

sw = s.\*hamming(frame\_length);

dft\_rc\_temp = log ( abs( fft( sw ) ) ); dft\_rc = real( ifft( dft\_rc\_temp) );

dft\_rc = dft\_rc(1:floor(frame\_length/2 + 1)); dft\_rc = dft\_rc(1:numberoffeatures); dftcepstrumstra(i,:)=dft\_rc';

cepstrum = dftcepstrumstra(i,:); Nfft = frame\_length;

lengthOfPositivePart = floor(Nfft/2 + 1);

Npad = lengthOfPositivePart - length(cepstrum); cepstrum = [cepstrum zeros(1,Npad)]; disp(cepstrum)

cepstrum = fft(cepstrum);

cepstrum = exp(real(cepstrum)); cepstrum = 20\*log10(cepstrum);

%DFT based mel-cepstrumstra melfilterbankorder = 20;

fmcc=melcepst(s,Fs,'Mta0',numberoffeatures,melfilterbankorder,frame\_length,frame\_ length,0,0.5);

dftmelcepstrumstra(i,:)=fmcc; melcepstrum = fmcc;

melcepstrum = [melcepstrum zeros(1,melfilterbankorder-length(melcepstrum))]; melcepstrum = exp(melcepstrum);

melcepstrum = 20\*log10(melcepstrum) - 20; filterAmplitudes=full(melbankm(melfilterbankorder ,frame\_length,Fs)); peak = max(filterAmplitudes');

for index = 1:length(peak)

filterCenter(index) = find(filterAmplitudes(index,:)==peak(index)); end

end

xaxis\_in\_Hz = (0:128)\*Fs/frame\_length; plot(xaxis\_in\_Hz,filterAmplitudes'); hold on HzScale = filterCenter \* Fs / frame\_length; plot(HzScale,ones(1,length(filterCenter)),'x'); plot(HzScale,zeros(1,length(filterCenter)),'x'); xlabel('Frequency (Hz)');

ylabel('Weight');

title('MFCC filter-bank (crosses indicate center frequency of each filter)');

% % Hint: place code in OpeningFcn to populate axes1

% --- Executes on button press in pushbutton5.

function pushbutton5\_Callback(hObject, eventdata, handles)

% handles structure with handles and user data (see GUIDATA) Fs=16000;

[filename, pathname] = uigetfile('\*.wav;\*.tif;\*.jpg;\*.pgm','Pick an M-file'); y = audioread(filename);

[y,fs1] = audioread(filename); sound(y,fs1);

% sound(audioIn,fs) [speech\_length temp] = size(y); frame\_length = 256;

number\_of\_frames = floor(speech\_length / frame\_length); numberoffeatures = 12;

% --- Executes on button press in pushbutton6.

function pushbutton6\_Callback(hObject, eventdata, handles)

% hObject handle to pushbutton6 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA) load('C:\S\param1.mat');

load('C:\S\param2.mat'); load('C:\S\param3.mat'); load('C:\S\param4.mat'); load('C:\S\param5.mat'); Fs=16000;

[filename, pathname] = uigetfile('\*.wav;\*.tif;\*.jpg;\*.pgm','Pick an M-file'); y = audioread(filename);

[speech\_length temp] = size(y); frame\_length = 256;

number\_of\_frames = floor(speech\_length / frame\_length); numberoffeatures = 12;

Q = 7;

O = 25;

D = zeros(9,3200); c = 0;

for i = [1:7] for j = [1:20] c = c+1;

if D >0

L = coeff\_extraction(filename); k1 = 32\*(c-1) + 1;

k2 = 32\*c;

D( k1:k2,1:9) = L(1:32, 1:9);

end end end

C = vqlbg(D,32); Nfft=512;

window = blackman(Nfft); Fs=16000;

specgram(y(:,1),Nfft,Fs,window,round(3/4\*length(window))); for i=1:number\_of\_frames

begin\_sample = 1+(i-1)\*frame\_length; end\_sample = i\*frame\_length;

s = y(begin\_sample:end\_sample ); s = s(:);

sw = s.\*hamming(frame\_length);

if C >= Q

[min\_value, obs\_seq\_rc, obs\_seq\_singlerow, obs\_seq\_lpcform] = vector\_quantize(C, D);

[states\_seq\_rc, states\_seq\_singlerow] = uniform\_seg(); end

dft\_rc\_temp = log ( abs( fft( sw ) ) ); dft\_rc = real( ifft( dft\_rc\_temp) );

dft\_rc = dft\_rc(1:floor(frame\_length/2 + 1)); dft\_rc = dft\_rc(1:numberoffeatures); dftcepstrumstra(i,:)=dft\_rc';

scepstrum = dftcepstrumstra(i,:); Nfft = frame\_length;

lengthOfPositivePart = floor(Nfft/2 + 1);

Npad = lengthOfPositivePart - length(scepstrum); scepstrum = [scepstrum zeros(1,Npad)]; scepstrum = fft(scepstrum);

scepstrum = exp(real(scepstrum)); scepstrum = 20\*log10(scepstrum);

%DFT based mel-cepstrumstra melfilterbankorder = 20;

fmcc=melcepst(s,Fs,'Mta0',numberoffeatures,melfilterbankorder,frame\_length,frame\_ length,0,0.5);

dftmelcepstrumstra(i,:)=fmcc; melcepstrum = fmcc;

melcepstrum = [melcepstrum zeros(1,melfilterbankorder-length(melcepstrum))]; melcepstrum = exp(melcepstrum);

melcepstrum = 20\*log10(melcepstrum) - 20; filterAmplitudes=full(melbankm(melfilterbankorder ,frame\_length,Fs)); peak = max(filterAmplitudes');

for index = 1:length(peak)

filterCenter(index) = find(filterAmplitudes(index,:)==peak(index)); for k=1:Q,

if cepstrum(1,k) == scepstrum (1,k) tt=1;

elseif cepstrum1(1,k) == scepstrum (1,k) tt=2;

elseif cepstrum2(1,k) == scepstrum (1,k) tt=3;

elseif cepstrum3(1,k) == scepstrum (1,k) tt=4;

elseif cepstrum4(1,k) == scepstrum (1,k) tt=5;

elseif cepstrum5(1,k) == scepstrum (1,k) else

tt=0;

end end end end

if tt==1

disp('user1'); elseif tt==2

disp('user2'); elseif tt==3

disp('user3'); elseif tt==4

disp('user4'); elseif tt==5

disp('user5'); else

disp('Unauthorized speaker');

end

% --- Executes on button press in pushbutton4.

function pushbutton4\_Callback(hObject, eventdata, handles)

% hObject handle to pushbutton4 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

load('C:\S\param1.mat'); load('C:\S\param2.mat'); load('C:\S\param3.mat'); load('C:\S\param4.mat'); load('C:\S\param5.mat');

disp('press enter to train Authorized speakers') u=input('=');

filename1 = audioread('speaker Sai Ganesh.wav'); filename2 = audioread('speaker Nithesh Kumar.wav'); filename3 = audioread('speaker Prudhviraj Reddy.wav'); filename4 = audioread('speaker Sahith.wav');

filename5 = audioread('speaker Ravi Kumar Sir.wav'); filename6 = audioread('speaker Avinash.wav'); filename7 = audioread('speaker Venkat.wav'); filename8 = audioread('speaker Raghavendra.wav'); filename9 = audioread('speaker Rajya Lakshmi.wav');

Q = 7;

O = 32;

D = zeros(9,3200); c = 0;

for i = [1:7] for j = [1:20] c = c+1;

if D >0

L1 = coeff\_extraction(filename1); L2 = coeff\_extraction(filename2); L3 = coeff\_extraction(filename3); L4 = coeff\_extraction(filename4); L5 = coeff\_extraction(filename5); k1 = 32\*(c-1) + 1;

k2 = 32\*c;

D1( k1:k2,1:9) = L1(1:32, 1:9);

D2( k1:k2,1:9) = L2(1:32, 1:9);

D3( k1:k2,1:9) = L3(1:32, 1:9);

D4( k1:k2,1:9) = L4(1:32, 1:9);

D5( k1:k2,1:9) = L5(1:32, 1:9);

end end end

C = vqlbg(D,32); if C >= Q

[min\_value, B2, A3, obs\_seq\_lpcform] = vector\_quantize(C1, D1); [A2, states\_seq\_singlerow] = uniform\_seg();

A2(6,1) = 0;

A2(6,6) = 1;

[R0,C0] = find(B2==0);

for i=1:size(R0,1) B2(R0(i),C0(i)) = 0.0001;

end

[A3,B3] = hmmtrain(number, A2, B2,'TOLERANCE', 0.001, 'MAXITERATIONS', 100);

[R0,C0] = find(B2==0);

for i=1:size(R0,1) B2(R0(i),C0(i)) = 0.0001;

end

[min\_value, B2, A3, obs\_seq\_lpcform] = vector\_quantize(C2, D2); [A2, states\_seq\_singlerow] = uniform\_seg();

A2(6,1) = 0;

A2(6,6) = 1;

[R0,C0] = find(B2==0);

for i=1:size(R0,1) B2(R0(i),C0(i)) = 0.0001;

end

[A3,B3] = hmmtrain(number, A2, B2,'TOLERANCE', 0.001, 'MAXITERATIONS', 100);

[R0,C0] = find(B2==0);

for i=1:size(R0,1) B2(R0(i),C0(i)) = 0.0001;

end

[min\_value, B2, A3, obs\_seq\_lpcform] = vector\_quantize(C3, D3); [A2, states\_seq\_singlerow] = uniform\_seg();

A2(6,1) = 0;

A2(6,6) = 1;

[R0,C0] = find(B2==0);

for i=1:size(R0,1) B2(R0(i),C0(i)) = 0.0001;

end

[A3,B3] = hmmtrain(number, A2, B2,'TOLERANCE', 0.001, 'MAXITERATIONS', 100);

[R0,C0] = find(B2==0);

for i=1:size(R0,1) B2(R0(i),C0(i)) = 0.0001;

end

[min\_value, B2, A3, obs\_seq\_lpcform] = vector\_quantize(C4, D4); [A2, states\_seq\_singlerow] = uniform\_seg();

A2(6,1) = 0;

A2(6,6) = 1;

[R0,C0] = find(B2==0);

for i=1:size(R0,1) B2(R0(i),C0(i)) = 0.0001;

end

[A3,B3] = hmmtrain(number, A2, B2,'TOLERANCE', 0.001, 'MAXITERATIONS', 100);

[R0,C0] = find(B2==0);

for i=1:size(R0,1) B2(R0(i),C0(i)) = 0.0001;

end

[min\_value, B2, A3, obs\_seq\_lpcform] = vector\_quantize(C5, D5); [A2, states\_seq\_singlerow] = uniform\_seg();

A2(6,1) = 0;

A2(6,6) = 1;

[R0,C0] = find(B2==0);

for i=1:size(R0,1) B2(R0(i),C0(i)) = 0.0001;

end

[A3,B3] = hmmtrain(number, A2, B2,'TOLERANCE', 0.001, 'MAXITERATIONS', 100);

[R0,C0] = find(B2==0);

for i=1:size(R0,1) B2(R0(i),C0(i)) = 0.0001;

end end