Speech Recognition System

Project Work

P- 672

SUBMITTED BY:

ASHWINI .M

ASHWINYA .R

NASREEN .B

RAMYA .P

UNDER THE GUIDANCE OF

Mrs. S. Devaki B.E., M.S.,

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF

BACHELOR OF ENGINEERING IN COMPUTER SCIENCE AND ENGINEERING

OF THE BHARATHIAR UNIVERSITY, COIMBATORE.

P- 672

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Kumaraguru College of Technology

(Affiliated to Bharathiar University)

COIMBATORE - 641 006.

Certificate

CERTIFICATE

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
KUMARAGURU COLLEGE OF TECHNOLOGY
(Affiliated to Bharathiar University, Coimbatore)

THIS IS TO CERTIFY THAT PROJECT WORK ENTITLED

SPEECH RECOGNITION SYSTEM

is a bonafide record of work done by

ASHWINI M.	**9827K0165**
ASHWINYA R.	**9827K0166**
NASREEN B.	**9827K0189**
RAMYA P.	**9827K0207**

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF
BACHELOR OF ENGINEERING
DURING THE ACADEMIC YEAR 2001-02

[illegible] 13/3/02
Head of the Department

[illegible] 13/3/2002.
Guide

Submitted for the University Examination held on _________

[illegible] 14/3
Internal Examiner

[illegible]
External Examiner

Place: Coimbatore
Date:

SKYSOFT TECHNOLOGIES INDIA (P) LTD
Red Rose Towers III Floor, 424 - D.B. Road,
R.S. Puram, Coimbatore - 641 002. Tel : 477319, 471452
E-mail marketing@skysoftindia.com
web www.skysoftindia.com

SKY SOFT
SKYSOFT

Date : 11-03-2002

To Whomsoever It May Concern

This is to certify that the project titled "**SPEECH RECOGNITION SYSTEM**" in **Turbo C++/ MATLAB** is a bonafide project work done by the following final year B.E Computer Science students of Kumaraguru College of Technology :

ASHWINI.M [Reg. No.: 9827K0165]
ASHWINYA.R [Reg. No.: 9827K0166]
NASREEN.B [Reg. No.: 9827K0189]
RAMYA.P [Reg. No.: 9827K0207]

This work was undertaken during the period, December 2001 to February 2002 at **Skysoft Technologies India (P) Ltd., Coimbatore.**

They were found to be hardworking, dedicated and sincere during the period of work.

[illegible] 11/03/02
Project Manager

• Software Development • Software Skills Development

Acknowledgement

The exhilaration achieved upon the successful completion of any task should be definitely shared with the people behind the venture. This project is an amalgam of study and experience of many people without whose help this project would not have taken shape.

At the onset, we take this opportunity to thank the management of our college for having provided us excellent facilities to work with.We express our deep gratitude to **Dr.K.K.Padmanabhan** B.Sc (Engg), M.Tech, Ph.D., Principal, Kumaraguru College of Technology, for ushering us in the path of triumph.

We are greatly indebted to **Prof. Dr. S. Thangaswamy** B.E.(Hons),Ph.D., Head of Department of Computer Science & Engineering and our guide, **Mrs.S.Devaki** B.E, M.S., for helping us to complete the project by giving timely guidance and constant encouragement throughout our endeavor.

We extend our sincere thanks and ineffable gratitude to Skysoft Technologies Ltd, for their benevolent attitude and stupendous persuasiveness to push up this project to its position of success.

Last but not the least, we thank all our teaching and non-teaching staff without whom our project would not have been a success.

Synopsis

Our project concentrates on using human voice to interact with a computer system, which is obviously easier and an ever-ready facility.

Our objectives are to:

- make the computer recognize isolated words (IWR).
- make it recognize users based on their voice (AVR).

The crux of the project is ***Speech Recognition***. Speech recognition is the process by which a computer (or other type of machine) identifies spoken words. Basically, it means talking to the computer, and having it correctly recognize what we are saying.

The ***Isolated Word Recognition(IWR)*** module is a speaker dependent system having a computer room speaking environment. The system is capable of recognizing words in a particular dialect uttered by a single speaker. This is implemented using Turbo C++.

The ***Automatic Voice Recognition(AVR)*** module recognizes users with their voice as the key. The goal of this module is to build a simple, yet complete and representative automatic speaker recognition system. Due to the limited space, we will only test our system on a very small (but already non-trivial) speech database consisting of eight speakers. MATLAB 5.3 has been used for implementation.

Speaker recognition is the process of automatically recognizing who is speaking on the basis of individual information included in speech waves. This technique makes it possible to use the speaker's voice to verify their identity and control access to services such as voice dialing, banking by telephone, database access services, information services, voice mail, security control for confidential information areas, and remote access to computers.

Contents

Prologue

Prologue

WHY THIS PROJECT?

Man has definitely persevered in computerizing things that were dreary. Through computerization, he has not only relieved himself of the drudgery involved but also has improved the efficiency factor.

Upgradation is a magic word in the field of computer science and man has thought of every possible way those faculties of men and machine could be used to simplify things. Primarily, the faculty of speech has been used in a variety of ways to enable man to interact with his computer better.

Isolated Word Recognition is one type of automatic speech recognition system. We have implemented a minimal IWR system that recognizes words like 'one', 'two', etc. IWR systems may be classified as speaker dependent recognition and speaker independent recognition systems. IWR is the first step in achieving speech-to-action and speech-to-text systems.

With the increasing use of personal electronics systems that store private information, and electronic commerce that is responsible for personal financial transactions, there is a potential danger of malicious use of systems by unwanted people. It is therefore important to develop fraud-proof identification and recognition systems to increase security in these highly susceptible areas. In order to solve the problem of unwanted access to system, we propose the AVR system.

COMPANY PROFILE

SKYSOFT TECHNOLOGIES INDIA (P) LTD.

Overview:

Skysoft Technologies India (P) Ltd., is a leading software development organization, formed in the year 1998 with an aim to emerge as a forerunner in the field of Information Technology related services with an accepted brand image synonymous with quality.
Skysoft develops software for people and people for software

Vision:

Skysoft dreams to develop into a center of excellence for the dissemination of professional knowledge by inculcating an understanding of the modern organizational environment developing the basic ingredients of business solutions in terms of strategies, techniques and essential skills using the latest developments in IT.

Objectives:

Software Development:

Expertise in designing and developing business solutions at organizational and functional levels including development of operational strategies and software solutions.

We have provided software solutions in the following segments:

- Supply chain management system
- Customer relationship management
- Processing & Mfg. Industries
- Garment Exports/Production
- Pump Industries
- Financial Mgt. Package

Software Skills Development:

There is a wide gap between the requirements and expectations of the industry-corporate and the mass produced by Academia." No experience-No work" –is the current job scenario. Skysoft bridges the gap, and facilitates the students to undergo live projects by providing hands-on experience on developing human resources for the IT industry.

Every corporate looks around to pick up the cream of the talent available in the job market, because it is this kind of talent that makes up the solid warfront for every company's competitive strategies.

The training methodologies used are aimed at maximizing the training effectiveness and professional to suit the job requirements in the industry. The division is backed by the vast developing experience of the software development team in familiarizing the trainees with the actual convention which is an added advantage and an edge over others because of the practical experience, which gives immense confidence to students once they start working in an organization.

In addition, we are training people on various languages to equip them for undergoing live project training.

Skysoft provides professional training for trainees wanting to pursue a challenging career in the IT industry by offering latest state of-the-art technology in order to meet the ever increasing demand for IT professionals the world over, especially in the developed countries. The well-trained computer professional finds absolutely no difficulty in being placed globally by giant corporations keeping pace with the changing trend.

Skysoft provides need based customized solution to corporate, industries meeting industrial standards and delivery schedule.

Software Requirements Specification

Software Requirements Specification

PURPOSE

The purpose of Speech recognition is to formulate a method of providing speech related services to users. Recent advances in speech recognition technology coupled with the advent of modern operating systems and high-powered affordable personal computers, have culminated in the first speech recognition systems that can be deployed to a wide community of users.

SCOPE

The scope of this project will be to provide two facilities:

1.Enable users to recognize isolated words like 'one', 'two' etc.

2.Enable users to be recognized based on their voice.

The primary objective is Voice recognition that, we have been able to achieve on a small scale. Enhancement would be considered in our future work.

GENERAL CONSTRAINTS

- ✓ The program is intended to run on a Microsoft operating system only (Windows 98, 2000).
- ✓ The program is application-based and is not intended for web-based application due to its large size because of the use of audio files.

OVERVIEW

This document contains the description of the initial specification of Speech recognition system design.

PRODUCT PERSPECTIVE

This project will be compound of two modules: Isolated word recognition, Automatic Voice recognition.

USER CHARACTERISTICS

Anybody conversant with the platform and GUI of Windows 9x can use the system .Particularly, users working with voice systems are benefited.

SPECIFIC SOFTWARE REQUIREMENTS

The requirements of Speech recognition include real-time aspects. The microphone and speakers should be chosen so that external noise and disturbances do not affect the quality of the product. The details are provided in the Hardware section.

External Interface Requirements

- ✓ *User Interface*: Accomplished via mouse input and keyboard input.
- ✓ *Hardware Interaction*: The product needs a system with the following hardware configuration: a PC with at least a Pentium processor, 32 MB of RAM, and a 640x480 screen resolution with a minimum of 256 color, a Multimedia kit with good quality microphone and speakers. Hard disk space is of 4MB and above is ideal.
- ✓ *Software Interaction*: The product requires MATLAB 5.3 and Turbo C++.

Performance Requirements

- ✓ The application will be run from a floppy disk on a local machine.
- ✓ The application will use multimedia microphone and speakers to record and deliver output for the IWR and AVR modules.
- ✓ The program targets users who need an introduction to speech recognition.

Input Data Requirements

- ✓ The project shall be able to acquire voice data and convert them into phonemes and visemes.
- ✓ The product shall be able to handle data inputs only from 8 users
- ✓ The product shall be able to detect voices only when good quality microphones are used in a noise-free room.
- ✓ The product shall allow user inputs to manipulate the display in the IWR module

Output Data Requirements

The Speech recognition project shall be able to handle user displays that will display the following information:

The results of

- Pre-emphasis
- End point location
- Windowing
- Auto correlation
- Cepstrum analysis
- Weighted cepstrum analysis

are displayed on a graphics screen .Also, the recognized word is displayed.

MATLAB 5.3 provides screens for training and testing user voices and storing in the codebook(database).

This document has been used for the making of the project

System Requirements

System Requirements

PRODUCT DEFINITION

Problem Statement:

This system

1. should recognize isolated words like 'one','two' etc
2. should recognize a trained voice

Hence the name " **Speech recognition** ".

Processing Environment:

Hardware Specification:

Processor	Pentium II 550 Mhz
Floppy Disk Drive	1.44 MB
CD-ROM Drive	52X
Hard Disk	40GB
Mouse	Microsoft compatible PS/2 mouse
Keyboard	Windows keyboard
System RAM	128 MB
Monitor	Color monitor for graphical output
Microphone	FM microphone
Speakers	60W
Sound Blaster	32 kit

Hardware

Sound Cards

Because speech requires a relatively low bandwidth, just about any medium-high quality 16-bit sound card will get the job done. We must have sound enabled in the kernel, and we must have correct drivers installed.

Sound cards with the 'cleanest' A/D (analog to digital) conversions are recommended, but most often the clarity of the digital sample is more dependent on the microphone quality and even more dependent on the environmental noise. Electrical "noise" from monitors, PCI slots, hard-drives, etc. are usually nothing compared to audible noise from the computer fans, squeaking chairs, or heavy breathing.

Some ASR software packages may require a specific sound card. It's usually a good idea to stay away from specific hardware requirements, because it limits many of the possible future options and decisions. We'll have to weigh the benefits and costs if we are considering packages that require specific hardware to function properly.

Microphones

A quality microphone is key when utilizing ASR. In most cases, a desktop microphone just won't do the job. They tend to pick up more ambient noise that gives ASR programs a hard time.

Hand held microphones are also not the best choice because they can be cumbersome to pick up all the time. While they do limit the amount of ambient noise, they are most useful in applications that require changing speakers often, or when speaking to the recognizer isn't done frequently (when wearing a headset isn't an option).

The best choice, and by far the most common is the headset style. It allows the ambient noise to be minimized, while allowing we to have the microphone at the tip of the tongue all the time. Headsets are available without earphones and with earphones (mono or stereo).

It is essential to turn up the microphone volume for optimum results. If the ASR software includes auto-adjustment programs, use them instead, as they are optimized for their particular recognition system.

Computers/Processors

AVR applications can be heavily dependent on processing speed. This is because a large amount of digital filtering and signal processing can take place in AVR.

With CPU intensive software, usually, faster is better. More memory is, of course, good. It is possible to do some SR with 100MHz and 16M RAM, but for fast processing (large dictionaries, complex recognition schemes, or high sample rates), a minimum of a 400MHz and 128M RAM is better. Because of the processing required, most software packages list their minimum requirements.

Software Specifications:

Languages Used : MATLAB 5.3, Turbo C++

Operating System : Windows 9x

Operating environment:

The language chosen for IWR is Turbo C++. The operating system used is Windows 98. The reason for choosing Turbo C++ is because of its capability handling complicated tasks easily. Turbo C++ has good support for system level functions. We have chosen MATLAB 5.3 for AVR as the module calls for bright, user-friendly interface and graphical outputs. MATLAB 5.3 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 is an interactive system whose basic data element is an array that does not require dimensioning. This allows you to solve many technical computing problems, especially those with matrix and vector formulations, in a fraction of the time it would take to write a program in a scalar non-interactive language such as C or Fortran.

PROJECT PLAN

Isolated Word Recognition System:

Speech recognition is a conversion from an acoustic waveform to a written equivalent of the message information. The nature of the speech recognition problem is highly dependent upon the constraints placed on the speaker, speaking situation and message content.

Automatic speech recognition (ASR) can be classified as continuous speech recognition (CSR) and isolated word recognition (IWR). Another classification of ASR systems can be speech-to-text systems and speech understanding systems. Further CSR and IWR systems may be classified as speaker dependent recognition and speaker ndependent recognition systems. In the former, the system is capable of recognizing peech in a particular dialect uttered by a single speaker. Independent recognition is very ifficult to achieve as it requires a large amount of training. The voice of different eople differs based on their sex, context, pitch and duration.

P-762

utomatic Voice Recognition System:

The goal of this system is to build a simple, yet complete and representative itomatic speaker recognition system. Due to the limited space, we will only test our stem on a very small (but already non-trivial) speech database. There can be eight eakers. All speakers utter the same single digit "*zero*" once in a *training session* and ce in a *testing session* later on. The vocabulary of digit is used very often in testing eaker recognition because of its applicability to many security applications. For mple, users have to speak a PIN (Personal Identification Number) in order to gain ess to the laboratory door, or users have to speak their credit card number over the phone line. By checking the voice characteristics of the input utterance, the system is to add a level of security.

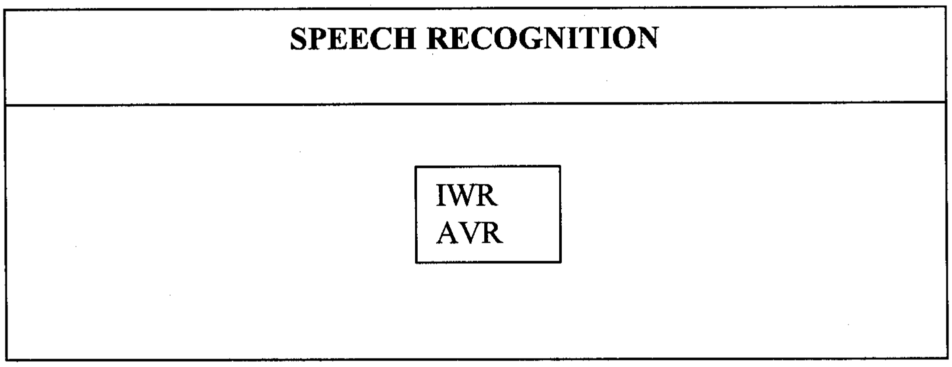
System Design

System design

EXTERNAL INTERFACES

Main Screen

The main screen has two option buttons for the two applications.



The system has two screens for a user-friendly interface. From the main screen ,the user can choose the application he wants to work with.

The screens are

- Isolated Word Recognition Screen
- Automatic Voice Recognition Screen

Isolated Word Recognition

The output screens are graphical in nature.

Automatic Voice Recognition

```
        1. Add New User
        2. Improve User DataBase
        3. Identify User Utterance
        4. Exit
Enter your choice:1
```

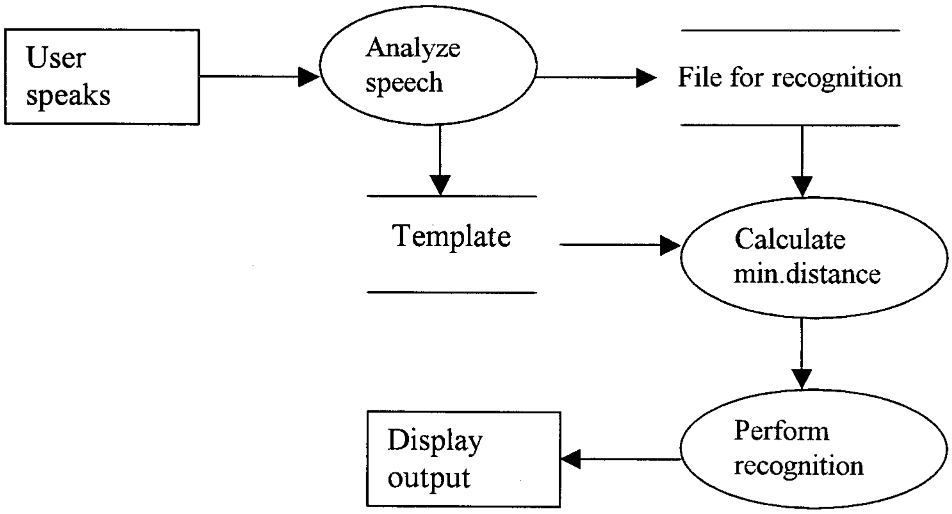
DATA FLOWS

Data Flow Diagram (DFD) is a graphical technique that depicts information flow and the transforms that are applied as data move from input to output. DFD may be used to represent a system or software at any level of abstraction. In fact DFDs may be portioned into levels that represent increasing information flow and functional details.

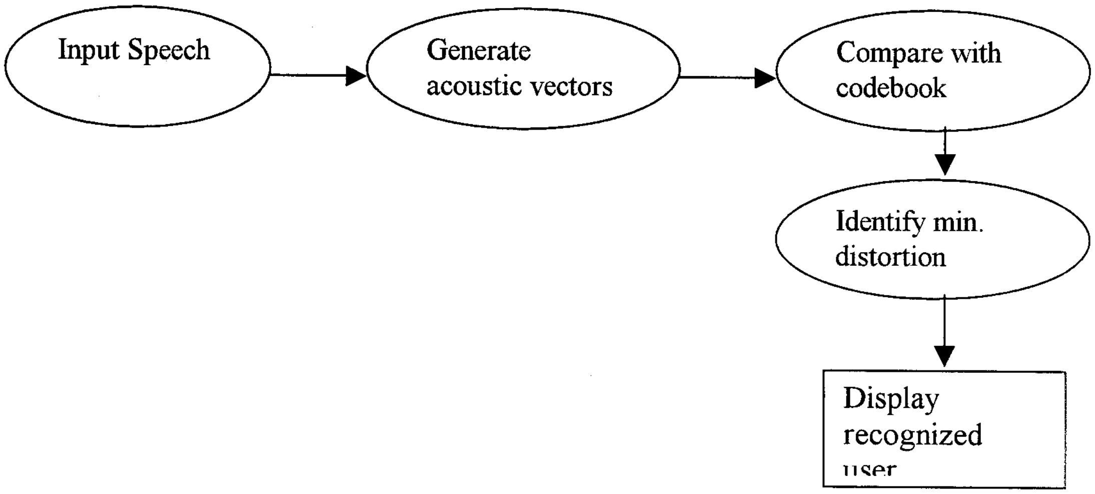
Graphical Notations used in DFD

▭	Inputs/Outputs to the Software
○	Processing the inputs
→	Data Flows
=	Data Stores

Isolated word recognition:



Automatic Voice Recognition:



Fundamentals

Fundamentals

TYPES OF SPEECH RECOGNITION

Speech recognition systems can be separated in several different classes by describing what types of utterances they have the ability to recognize. These classes are based on the fact that one of the difficulties of ASR is the ability to determine when a speaker starts and finishes an utterance. Most packages can fit into more than one class, depending on which mode they're using.

Isolated Words

Isolated word recognizers usually require each utterance to have quiet (lack of an audio signal) on *both* sides of the sample window. It doesn't mean that it accepts single words, but does require a single utterance at a time. Often, these systems have "Listen/Not-Listen" states, where they require the speaker to wait between utterances (usually doing processing during the pauses). Isolated Utterance might be a better name for this class. An isolated-word system operates on single words at a time - requiring a pause between saying each word. This is the simplest form of recognition to perform because the end points are easier to find and the pronunciation of a word tends not affect others. Thus, because the occurrences of words are more consistent they are easier to recognize.

Connected Words

Connect word systems or connected utterances are similar to isolated words, but allow separate utterances to be "run-together" with a minimal pause between them. A continuous speech system operates on speech in which words are connected together, i.e. not separated by pauses.

Continuous Speech

Recognizers with continuous speech capabilities are some of the most difficult to create because they must utilize special methods to determine utterance boundaries. A continuous speech system operates on speech in which words are connected together, i.e. not separated by pauses. Continuous speech is more difficult to handle because of a variety of effects. First, it is difficult to find the start and end points of words. Another problem is "co-articulation". The production of each phoneme is affected by the production of surrounding phonemes, and similarly the start and end of words are affected by the preceding and following words. The recognition of continuous speech is also affected by the rate of speech (fast speech tends to be harder). Continuous speech recognizers allow users to speak almost naturally, while the computer determines the content. Basically, it's computer dictation.

Spontaneous Speech

There appears to be a variety of definitions for what spontaneous speech actually is. At a basic level, it can be thought of as speech that is natural sounding and not rehearsed. An ASR system with spontaneous speech ability should be able to handle a variety of natural speech features such as words being run together, "ums" and "ahs", and even slight stutters.

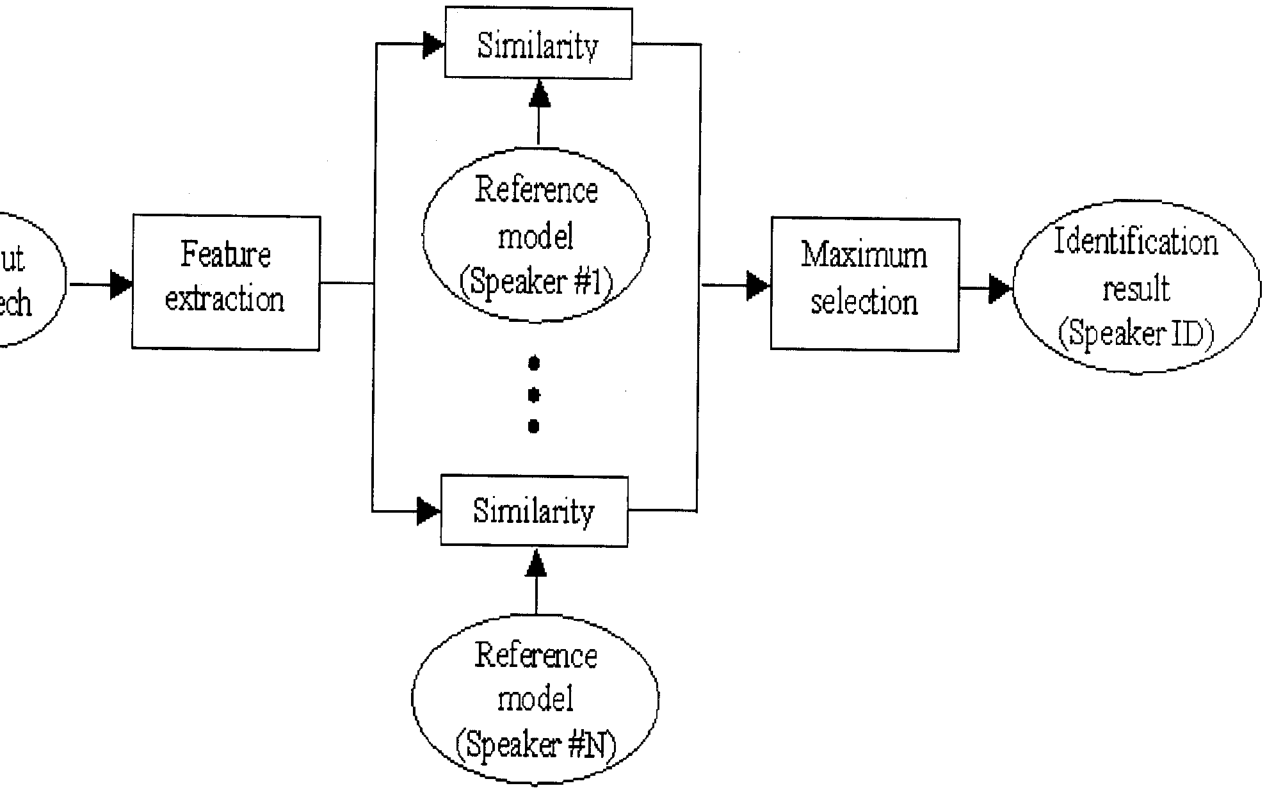
PRINCIPLES OF SPEAKER RECOGNITION

Speaker recognition can be classified into identification and verification. *Speaker identification* is the process of determining which registered speaker provides a given utterance. *Speaker verification*, on the other hand, is the process of accepting or rejecting the identity claim of a speaker. Figure 1 shows the basic structures of speake identification and verification systems.

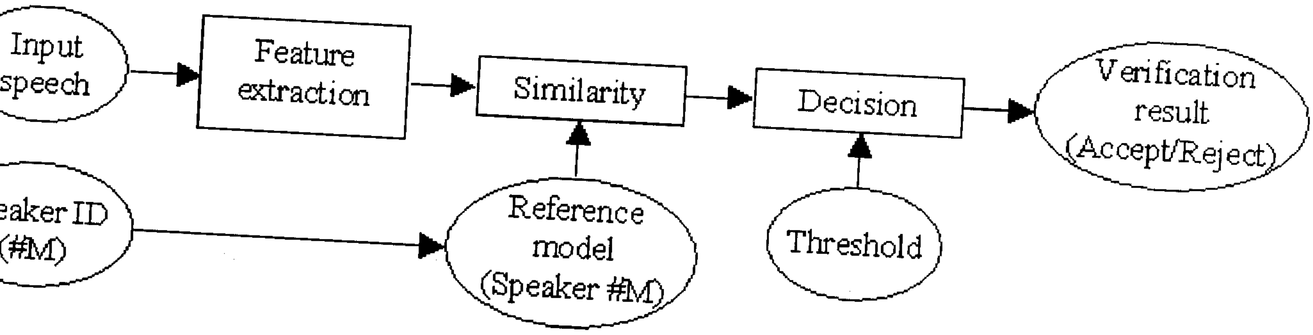
Speaker recognition methods can also be divided into *text-independent* and *t* *dependent* methods. In a text-independent system, speaker models capture characte of somebody's speech which show up irrespective of what one is saying. In a text-dependent system, on the other hand, the recognition of the speaker's identity is based on his or her speaking one or more specific phrases, like passwords, card numbers, PIN codes, etc.

All technologies of speaker recognition, identification and verification, text-independent and text-dependent, each has its own advantages and disadvantages and may requires different treatments and techniques. The choice of which technology to use is application-specific. At the highest level, all speaker recognition systems contain two main modules: *feature extraction* and *feature matching*. Feature extraction is the process that extracts a small amount of data from the voice signal that can later be used to represent each speaker. Feature matching involves the actual procedure to identify the unknown speaker by comparing extracted features from his/her voice input with the ones from a set of known speakers.

(a) Speaker identification



ut
ech
Feature extraction
Similarity
Reference model (Speaker #1)
Similarity
Reference model (Speaker #N)
Maximum selection
Identification result (Speaker ID)



(b) Speaker verification

Figure 1. Basic structures of speaker recognition systems

All speaker recognition systems have to serve two distinguish phases. The first one is referred to the enrollment sessions or training phase while the second one is eferred to as the operation sessions or testing phase. In the *training phase*, each egistered speaker has to provide samples of their speech so that the system can build or rain a reference model for that speaker. In case of speaker verification systems, in ddition, a speaker-specific threshold is also computed from the training samples. uring the testing (operational) phase , the input speech is matched with stored reference odel(s) and recognition decision is made.

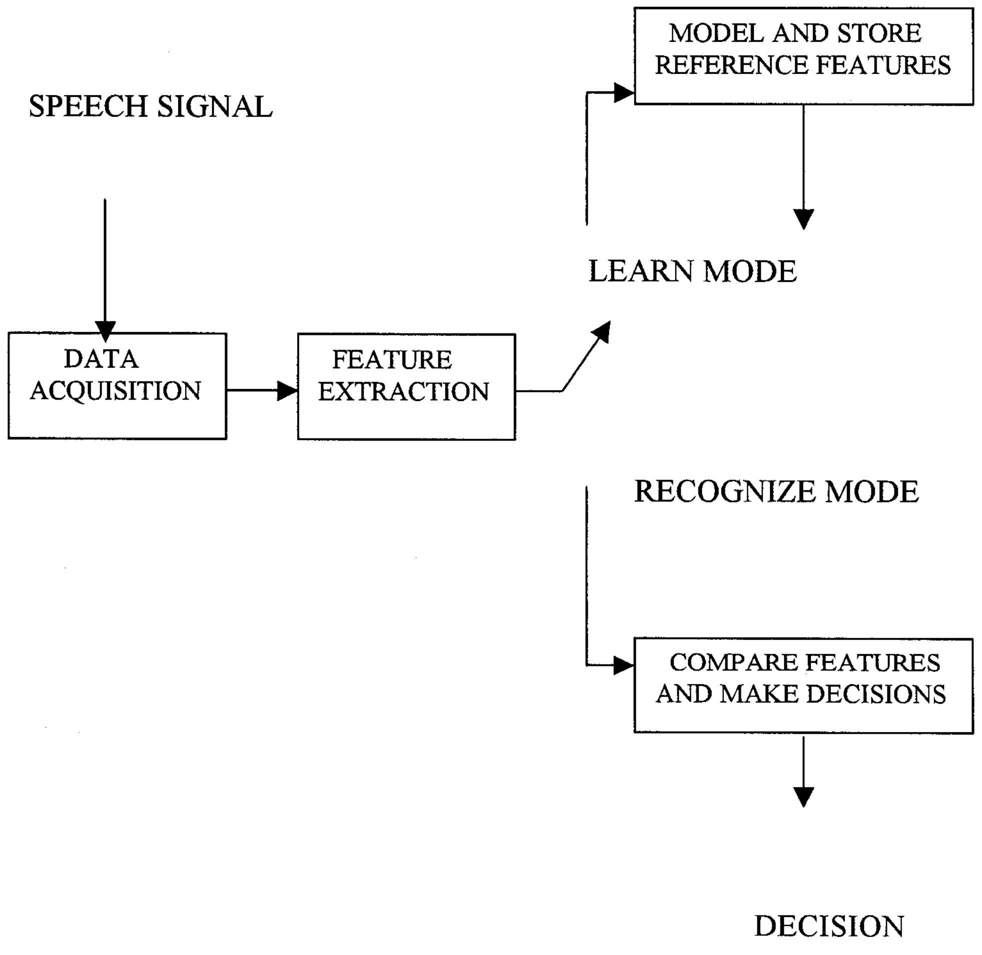
Speaker recognition is a difficult task and it is still an active research area. utomatic speaker recognition works based on the premise that a person's speech hibits characteristics that are unique to the speaker. However this task has been allenged by the highly *variant* of input speech signals. The principle source of iance is the speaker himself. Speech signals in training and testing sessions can be atly different due to many facts such as people voice change with time, health ditions (e.g. the speaker has a cold), speaking rates, etc. There are also other factors, ond speaker variability, that present a challenge to speaker recognition technology. amples of these are acoustical noise and variations in recording environments.

System Implementation

System Implementation

ISOLATED WORD RECOGNITION

The process of speech recognition includes the capture of a speech utterance i.e., data acquisition, analysis of raw speech into a suitable set of parameters i.e., feature extraction, the comparison of these features against some previously stored templates (reference features) and a decision making process. This can be depicted by the following block diagram



Data Acquisition:

Data acquisition deals with the capture of the audio signals as they are uttered. These signals may be constituted by speech and other noises present in the environment. Typically a system must precondition the signal received and only then allow it to be processed. This eliminates or at least reduces the effect of environmental noise.

Sound is inherently analog. The process of converting an analog signal into its digital form is known as sampling. This process establishes the frequency of the waveform.

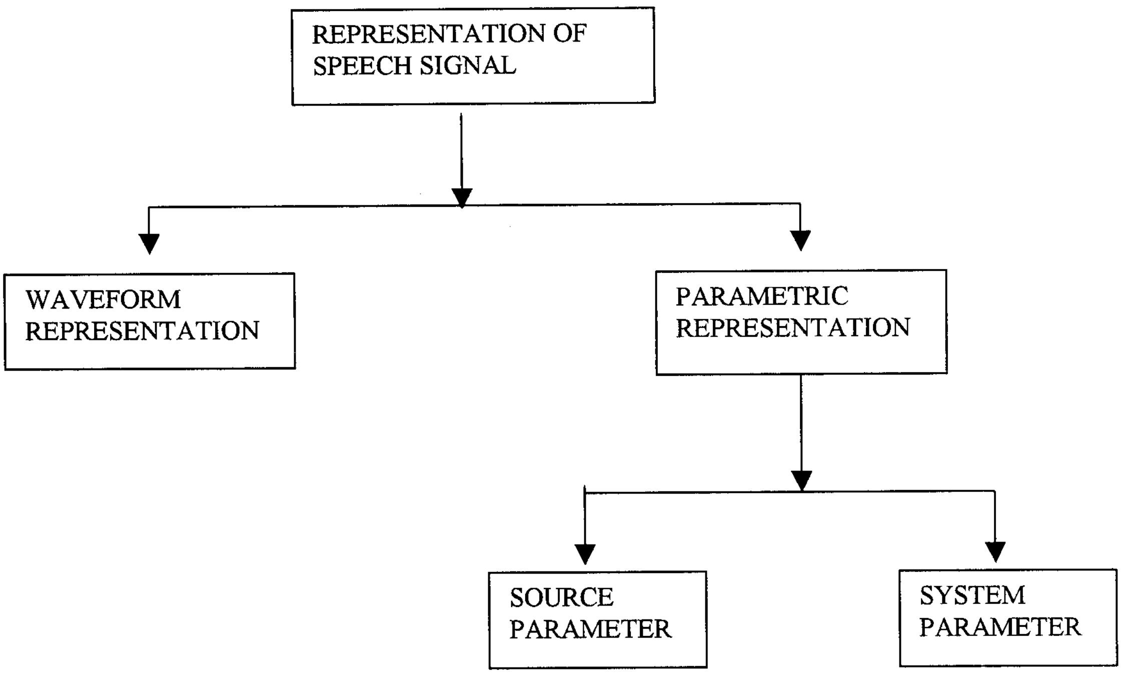
Waveform files store digital audio in files with extension WAV. Wave files are a type of RIFF file. Basic building block of a RIFF file is called a chunk. Each chunk consists of the following fields:

- A four character code used as the identifier
- A value specifying the size of data in the chunk
- The data field itself.

The wave file format is given by
‘RIFF’<file_size>’WAVE’<header chunk><data chunk>

Feature Extraction

Feature extraction involves the representation of the speech signal as a set of parameters, which characterize the signal. Signal processing of the speech signal in terms of its parameters are as below.



Representation Of Speech Signal

The following are the steps to be adopted in obtaining a parametric representation of the signal

End – point location:

This deals with the problem of locating the beginning and end of a speech utterance in a background of noise. In particular, in automatic speech recognition of isolated words, it is essential to locate the regions of the speech signal that correspond to the word. Two simple time domain measurements – energy and zero-crossing rate are used for this purpose.

Windowing:

There are two different types of filters. They are the finite impulse response(FIR) and infinite impulse response(IIR)filters. A finite impulse response filter is being used. Once the filter's impulse response is available, it is easy to find the coefficients of the non-recursive with the impulse response. Windowing is carried out for a filter with finite impulse response in order to obtain the filter coefficients.

The formula for the Hanning window is given by

$W(n) = 0.5 - 0.5 \cos 2\pi n / M, 0<n<M$

$= 0$, otherwise

where n – sample number in a particular frame

M-number of samples in a particular frame

Pre – Emphasis :

Filter coefficients obtained after analysis in each frame is then subjected to pre-emphasis. In order to improve the signal to noise ratio, pre-emphasis is carried out. The formula implemented during pre-emphasis is given by

$P = a_i - \alpha . a_{I-1}$

where

$\alpha = 0.95$,a constant

$a_i =$ sample i's windowed value

$a_{I-1} =$ sample (I-1)'s windowed value

Autocorrelation:

The parameters of the speech signal may be distorted with time. In order to minimize this distortion, autocorrelation is carried out. Some of the auto correlation function are

a) The correlation function of a periodic signal is also periodic.
b) It is an even function.
c) It attains its maximum value for R(0).
d) R(0) is the energy for deterministic signals.

The formula employed in auto-correlation is

$$R(1) = \sum_{i=1}^{99} a_i . a_{i+2}$$

$$R(2) = \sum a_i . a_{i+2}$$

.

.

.

$$R(10) = \sum_{i=1}^{90} a_i . a_{i+1}$$

inear Predictive Coding:

This method is the most predominant method for determining the basic speech
rameters such as pitch, formants etc. The basic principle in predictive analysis is that a
eech sample can be approximated as a linear combination of past speech samples. By
inimizing the sum of the squared differences between the actual speech samples and the
early predicted ones, a unique set of predictor coefficients can be determined. Linear
edictive analysis produces basically an all- pole model with only resonances.

Durbin's recursive solution for the autocorrelation equations is used to determine
e LP parameters which may be stated by the following formulae

$$k_i = \frac{[R(i) - \sum \alpha_j^{(i-1)} . R(i-j)]}{E^{(i-1)}}$$

$$\alpha_i^{(i)} = k_i$$

$$\alpha_j^{(i)} = \alpha_j^{(i-1)} - k_i . \alpha_{(i-j)}^{(i-1)}$$

$$E^{(i)} = (1 - k_i^2) E^{(i-1)}$$

where α- predictor coefficient

E-energy

epstrum Analysis:

Once LPC coefficients have been determined, the cepstrum of the overall LPC is mputed.

$$H(n)=\alpha_n+\Sigma(k/n)h(k).\alpha_{(n-k)} \;,1\leq n$$

Cepstral analysis provides a technique for separating the source and system arameters by computing the spectral envelope which reflects the system characteristics.

eighted cepstrum analysis:

Weights have to be assigned for the cepstral coefficients. This is done as follows:

$$C(n)=n*C(n)$$

where C(n) represents the Nth cepstral coefficient

The digital audio is a stream of amplitudes, sampled at about 16,000 times per second. It's a wavy line that periodically repeats while the user is speaking. While in this form, the data isn't useful to speech recognition because it's too difficult to identify any patterns that correlate to what was actually said.

To make pattern recognition easier, the PCM digital audio is transformed into the "frequency domain." Transformations are done using a windowed fast-Fourier transform. The output is similar to what a spectrograph produces. In frequency domain, we can identify the frequency components of a sound. From the frequency components, it's possible to approximate how the human ear perceives the sound.

The fast Fourier transform analyzes every 1/100th and converts into the frequency omain. Each 1/100th of a second results is a graph of the amplitudes of frequency omponents, and describing the sound heard for that 1/100th of a second. The speech ecognizer has a database of several thousand such graphs, called a codebook, that identify fferent types of sounds the human voice can make. The sound is "identified" by matching to its closest entry in the codebook, producing a number that describes the sound. This mber is called the "feature number." (Actually, there are several feature numbers nerated for every 1/100th of a second but the process is easier to explain assuming only e.)

ecognition Phase:

Dynamic time warping is a non-linear time normalization technique that mpensates for the time scale difference between patterns. The speech signal is analyzed ame wise and each frame is represented by a feature vector .The test utterance having I ames can then be represented as a pattern T of I features=T(1).T(2)......T(I).Similarly the ference utterance is denoted by R=R(1).R(2)...R(J).These patterns are the input to the arping process and the task of the DTW algorithm is to find the warping path j=w(i) that gins at point (1,1) ends at point (I,J) and minimizes the following distance D, between the st and reference patterns over all possible paths.

$$D=\sum d[T(i).R(w(i))]$$

/here d[T(i).R(w(i))] is the local distance between frame i of the test pattern and frame (i) of the reference pattern.

UTOMATIC VOICE RECOGNITION

eech Feature Extraction

The purpose of this module is to convert the speech waveform to some type of rametric representation (at a considerably lower information rate) for further analysis and ocessing. This is often referred as the *signal-processing front end.*

The speech signal is a slowly timed varying signal (it is called *quasi-stationary*). 1 example of speech signal is shown in Figure 2. When examined over a sufficiently short riod of time (between 5 and 100 msec), its characteristics are fairly stationary. However, er long periods of time (on the order of 1/5 seconds or more) the signal characteristic hange to reflect the different speech sounds being spoken. Therefore, *short-time spectral nalysis* is the most common way to characterize the speech signal.

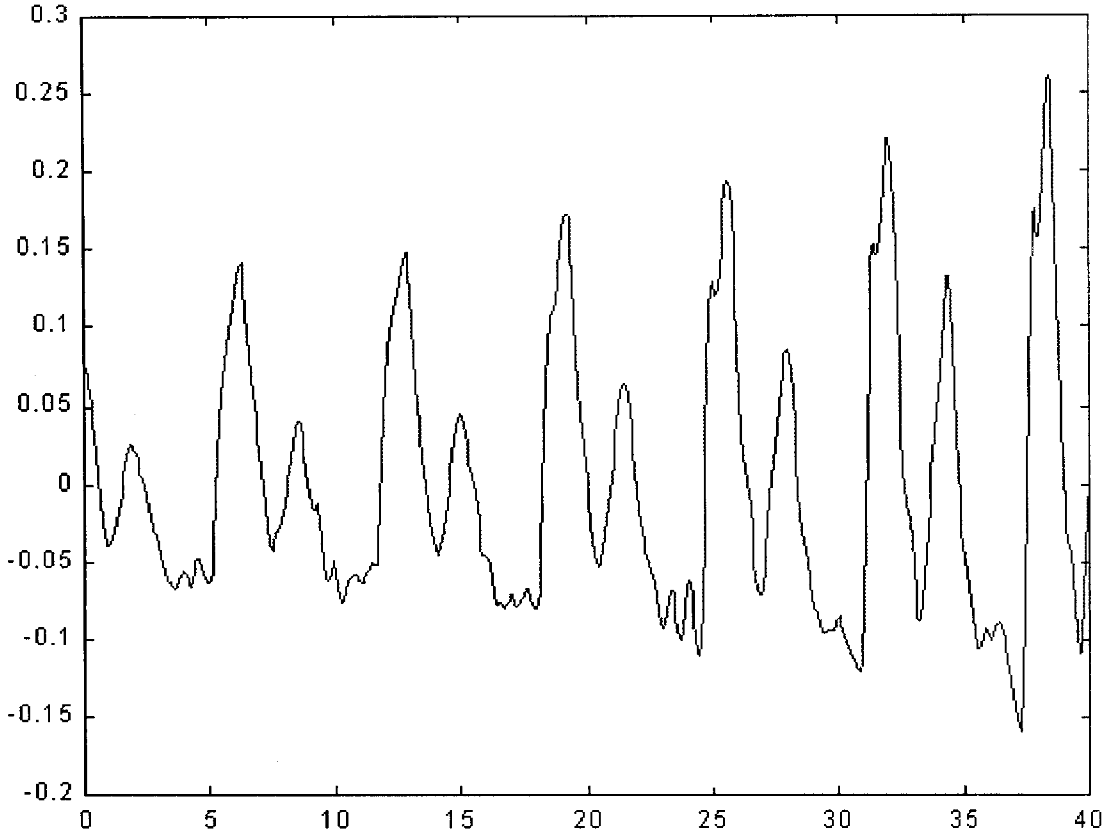


Figure 2. An example of speech signal

A wide range of possibilities exist for parametrically representing the speech signal for the speaker recognition task, such as Linear Prediction Coding (LPC), Mel-Frequency Cepstrum Coefficients (MFCC), and others. MFCC is perhaps the best known and most popular, and these will be used in this project.

MFCCs are based on the known variation of the human ears critical bandwidths with frequency, filters spaced linearly at low frequencies and logarithmically at high frequencies have been used to capture the phonetically important characteristics of speech. This is expressed in the *mel-frequency* scale, which is a linear frequency spacing below 1000 Hz and a logarithmic spacing above 1000 Hz. The process of computing

Mel-frequency cepstrum coefficients processor

A block diagram of the structure of an MFCC processor is given in Figure 3. The speech input is typically recorded at a sampling rate above 10000 Hz. This sampling frequency was chosen to minimize the effects of *aliasing* in the analog-to-digital conversion. These sampled signals can capture all frequencies up to 5 kHz, which cover most energy of sounds that are generated by humans. As been discussed previously, the main purpose of the MFCC processor is to mimic the behavior of the human ears. In addition, rather than the speech waveforms themselves, MFFCs are shown to be less susceptible to mentioned variations.

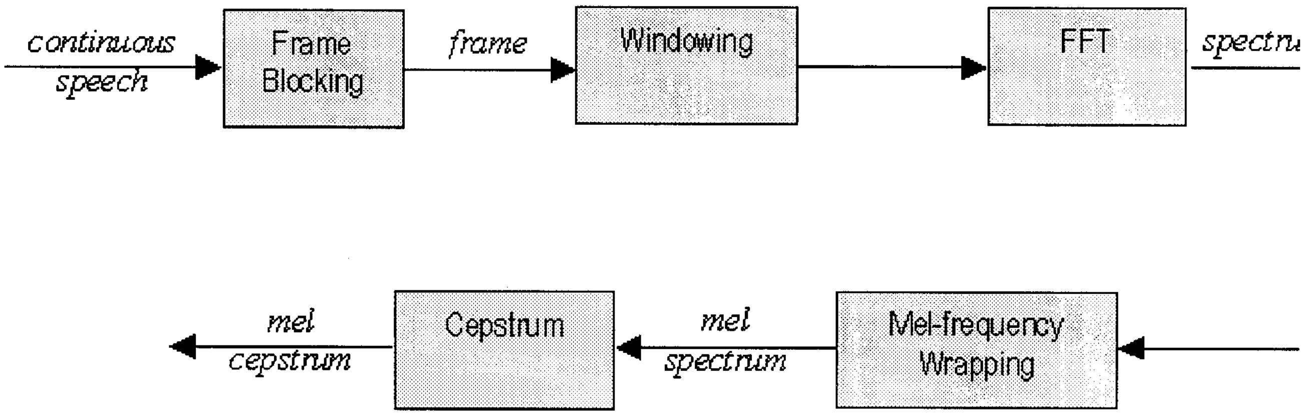


Figure 3. Block diagram of the MFCC processor

Frame Blocking

In this step the continuous speech signal is blocked into frames of N samples, with adjacent frames being separated by M ($M < N$). The first frame consists of the first N samples. The second frame begins M samples after the first frame, and overlaps it by $N - M$ samples. Similarly, the third frame begins $2M$ samples after the first frame (or M samples after the second frame) and overlaps it by $N - 2M$ samples. This process continues until all the speech is accounted for within one or more frames. Typical values for N and M are $N = 256$ (which is equivalent to ~ 30 msec windowing and facilitate the fast radix-2 FFT) and $M = 100$.

Windowing

The next step in the processing is to window each individual frame so as to minimize the signal discontinuities at the beginning and end of each frame. The concept here is to minimize the spectral distortion by using the window to taper the signal to zero at the beginning and end of each frame. If we define the window as $w(n), 0 \le n \le N-1$, where N is the number of samples in each frame, then the result of windowing is the signal

$$y_l(n) = x_l(n)w(n), \quad 0 \le n \le N-1$$

Typically the *Hamming* window is used, which has the form:

$$w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right), \quad 0 \le n \le N-1$$

Fast Fourier Transform (FFT)

The next processing step is the Fast Fourier Transform, which converts each frame of N samples from the time domain into the frequency domain. The FFT is a fast algorithm to implement the Discrete Fourier Transform (DFT) which is defined on the set of N samples $\{x_n\}$, as follow:

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

Note that we use j here to denote the imaginary unit, i.e. $j = \sqrt{-1}$. In general X_n?s are complex numbers. The resulting sequence $\{X_n\}$ is interpreted as follow: the zero frequency corresponds to n = 0, positive frequencies $0 < f < F_s/2$ correspond to values $1 \le n \le N/2-1$, while negative frequencies $-F_s/2 < f < 0$ correspond to $N/2+1 \le n \le N-1$. Here, F_s denotes the sampling frequency.

The result after this step is often referred to as *spectrum* or *periodogram.*

Mel-frequency Wrapping

As mentioned above, psychophysical studies have shown that human perception of the frequency contents of sounds for speech signals does not follow a linear scale. Thus for each tone with an actual frequency, *f*, measured in Hz, a subjective pitch is measured on a scale called the ?mel? scale. The *mel-frequency* scale is a linear frequency spacing below 1000 Hz and a logarithmic spacing above 1000 Hz. As a reference point, the pitch of a 1 kHz tone, 40 dB above the perceptual hearing threshold, is defined as 1000 mels. Therefore we can use the following approximate formula to compute the mels for a given frequency *f* in Hz:

$$mel(f) = 2595 * \log_{10}(1 + f/700)$$

One approach to simulating the subjective spectrum is to use a filter bank, spaced uniformly on the mel scale (see Figure 4). That filter bank has a triangular bandpass frequency response, and the spacing as well as the bandwidth is determined by a constant mel frequency interval. The modified spectrum of $S(\omega)$ thus consists of the output power of these filters when *S(ω)* is the input. The number of mel spectrum coefficients, *K*, is typically chosen as 20.

Note that this filter bank is applied in the frequency domain, therefore it simply amounts to taking those triangle-shape windows in the Figure 4 on the spectrum. A useful way of thinking about this mel-wrapping filter bank is to view each filter as an histogram bin (where bins have overlap) in the frequency domain.

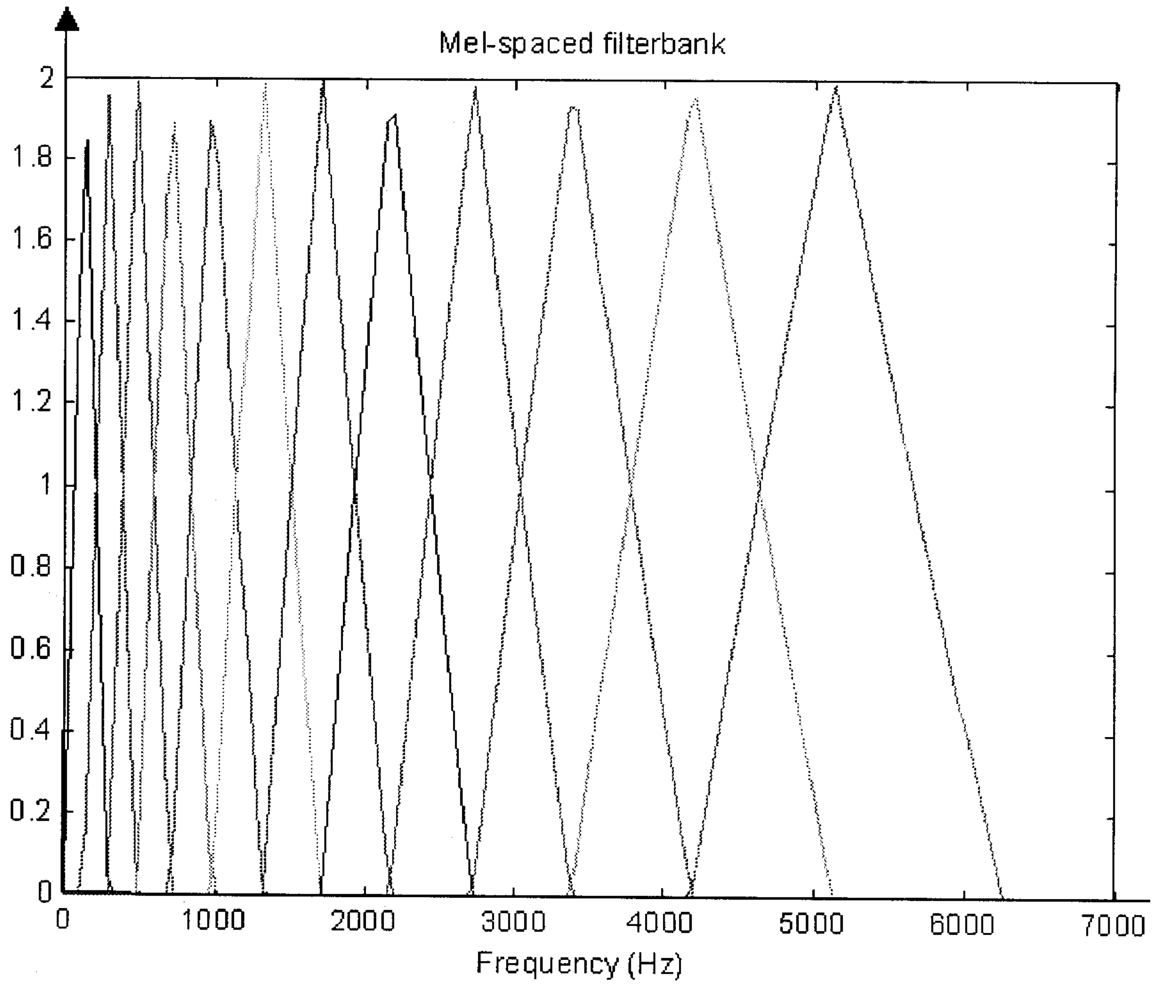


Figure 4. An example of mel-spaced filterbank

Cepstrum

In this final step, we convert the log mel spectrum back to time. The result is called the mel frequency cepstrum coefficients (MFCC). 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 (and so their logarithm) are real numbers, we can convert them to the time domain using the Discrete Cosine Transform (DCT). Therefore if we denote those mel power spectrum coefficients that are the result of the last step are $\tilde{S}_k, k = 1,2,\ldots,K$, we can calculate the MFCC's, $\tilde{c}_n$, as

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

Note that we exclude the first component, $\tilde{c}_0$, from the DCT since it represents the mean value of the input signal which carried little speaker specific information.

Feature Matching

The problem of speaker recognition belongs to a much broader topic in scientific and engineering so called *pattern recognition*. The goal of pattern recognition is to classify objects of interest into one of a number of categories or classes. The objects of interest are generically called *patterns* and in our case are sequences of acoustic vectors that are extracted from an input speech using the techniques described in the previous section. The classes here refer to individual speakers. Since the classification procedure in our case is applied on extracted features, it can be also referred to as *feature matching*.

Furthermore, if there exists some set of patterns that the individual classe of which are already known, then one has a problem in *supervised patte recognition*. This is exactly our case since during the training session, we label ea input speech with the ID of the speaker (S1 to S8). These patterns comprise *training set* and are used to derive a classification algorithm. The remaining patt are then used to test the classification algorithm; these patterns are collectively referred to as the *test set*. If the correct classes of the individual patterns in the test set are also known, then one can evaluate the performance of the algorithm.

The state-of-the-art in feature matching techniques used in speaker recognition include Dynamic Time Warping (DTW), Hidden Markov Modeling (HMM), and Vector Quantization (VQ). In this project, the VQ approach will be used, due to ease of implementation and high accuracy. VQ is a process of mapping vectors from a large vector space to a finite number of regions in that space. Each region is called a *cluster* and can be represented by its center called a *codeword*. The collection of all codewords is called a *codebook*.

Figure 5 shows a conceptual diagram to illustrate this recognition process. In the figure, only two speakers and two dimensions of the acoustic space are shown. The circles refer to the acoustic vectors from the speaker 1 while the triangles are from the speaker 2. In the training phase, a speaker-specific VQ codebook is generated for each known speaker by clustering his/her training acoustic vectors. The result codewords (centroids) are shown in Figure 5 by black circles and black triangles for speaker 1 and 2, respectively. The distance from a vector to the closest codeword of a codebook is called a VQ-distortion. In the recognition phase, an input utterance of an unknown voice is "vector-quantized" using each trained codebook and the *total VQ distortion* is computed. The speaker corresponding to the VQ codebook with smallest total distortion is identified.

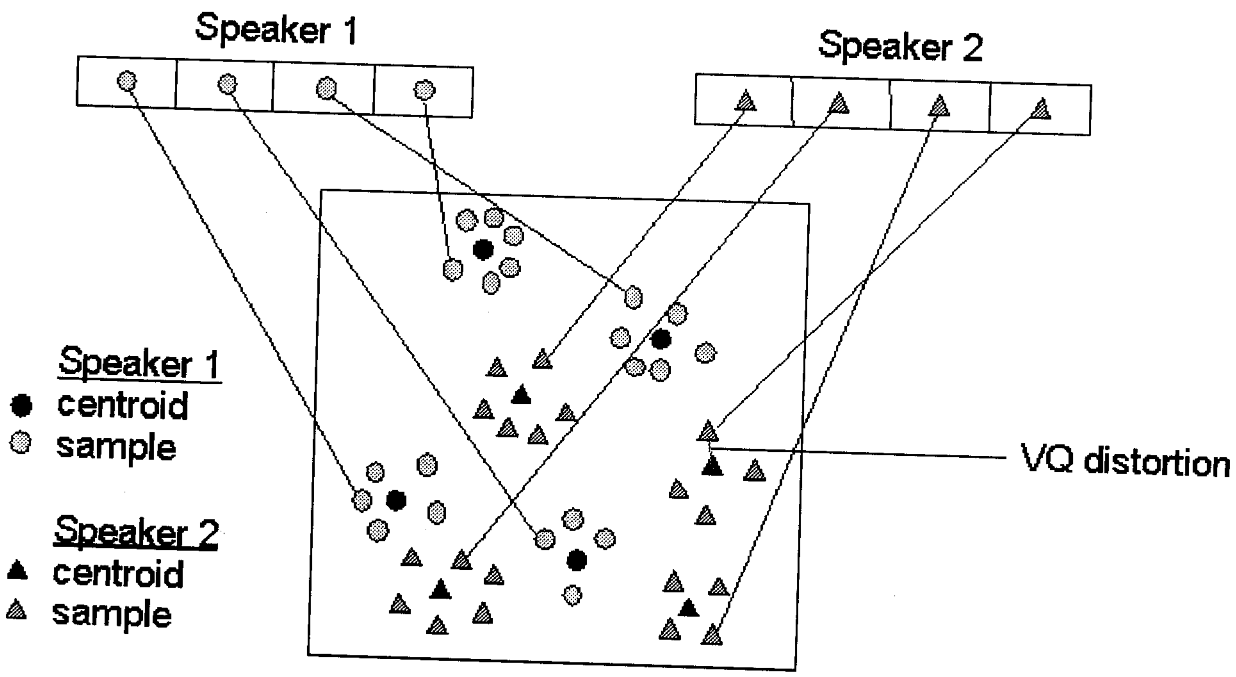


Figure 5. Conceptual diagram illustrating vector quantization codebook formation.

One speaker can be discriminated from another based of the location of centroids.

Clustering the Training Vectors

After the enrolment session, the acoustic vectors extracted from input speech of a speaker provide a set of training vectors. As described above, the next important step is to build a speaker-specific VQ codebook for this speaker using those training vectors. There is a well-know algorithm, namely LBG algorithm [Linde, Buzo and Gray, 1980], for clustering a set of L training vectors into a set of M codebook vectors. The algorithm is formally implemented by the following recursive procedure:

- Design a 1-vector codebook; this is the centroid of the entire set of training vectors (hence, no iteration is required here).

- Double the size of the codebook by splitting each current codebook $\mathbf{y}_n$ according to the rule

$$\mathbf{y}_n^+ = \mathbf{y}_n(1+\varepsilon)$$

$$\mathbf{y}_n^- = \mathbf{y}_n(1-\varepsilon)$$

where n varies from 1 to the current size of the codebook, and ε is a splitting parameter (we choose ε=0.01).

- Nearest-Neighbor Search: for each training vector, find the codeword in the current codebook that is closest (in terms of similarity measurement), and assign that vector to the corresponding cell (associated with the closest codeword).

- Centroid Update: update the codeword in each cell using the centroid of the training vectors assigned to that cell.

Iteration 1: repeat steps 3 and 4 until the average distance falls below a preset threshold

Iteration 2: repeat steps 2, 3 and 4 until a codebook size of M is designed.

Intuitively, the LBG algorithm designs an M-vector codebook in stages. It starts first by designing a 1-vector codebook, then uses a splitting technique on the codewords to initialize the search for a 2-vector codebook, and continues the splitting process until the desired M-vector codebook is obtained.

Figure 6 shows, in a flow diagram, the detailed steps of the LBG algorithm. "*Cluster vectors*" is the nearest-neighbor search procedure which assigns each training vector to a cluster associated with the closest codeword. "*Find centroids*" is the centroid update procedure. "*Compute D (distortion)*" sums

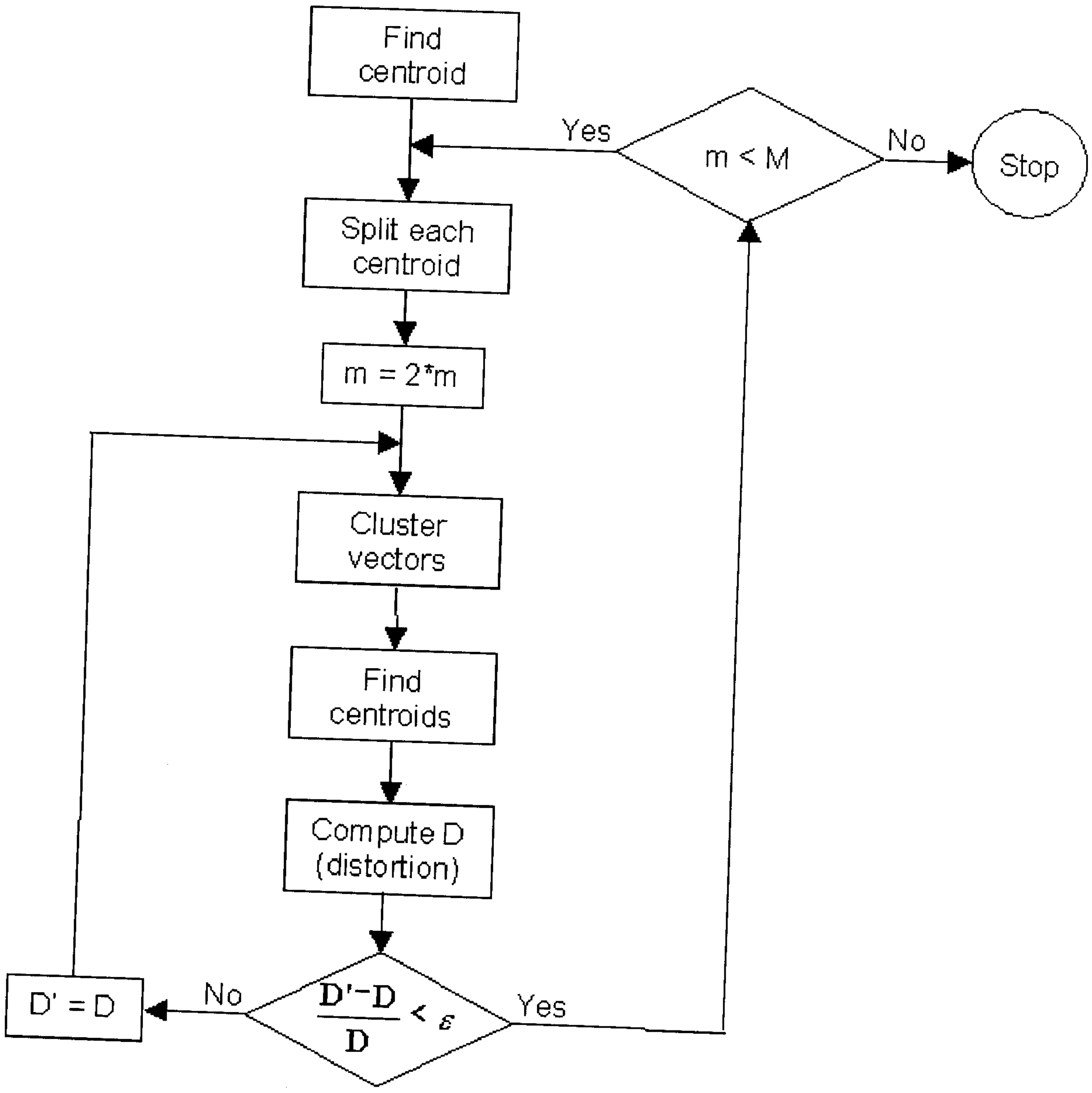


Figure 6. Flow diagram of the LBG algorithm

the distances of all training vectors in the nearest-neighbor search so as to determine whether the procedure has converged.

System Testing

System Testing

Adding speech recognition adds a few more things that could go wrong with an application.

Sound Cards

Different sound cards and driver versions - Make sure to test on different sound cards and driver versions. The variation in the sound cards' mixers and quality might cause problems with the application.

Test for sound card - Identify if the user doesn't have a sound card, or the user's sound card can't handle IWR/AVR, and inform the user.

More than one sound card - The user might have more than one sound card. Make sure they can choose the right one.

Usability

Can the user get their microphone working? - Do extensive testing to make sure the user can plug in their microphone and use the microphone correctly.

Does the user understand how to talk to the PC? - Make sure that users can figure out how to talk to their PCs.

Miscellaneous

Proper machine speed - Some engines require Pentium-speed or better to function. Inform the user if their PC is too slow for speech recognition and/or text-to-speech. Some older Pentiums with faulty math coprocessors will have their math coprocessor disabled by Windows NT. If the speech engine relies on floating point, it may be too slow on these machines.

Uses & Applications

Uses and Applications

Although any task that involves interfacing with a computer can potentially use ASR, the following applications are the most common right now.

DICTATION

Dictation is the most common use for IWR systems today. This includes medical transcriptions, legal and business dictation, as well as general word processing. In some cases special vocabularies are used to increase the accuracy of the system.

COMMAND AND CONTROL

AVR systems that are designed to perform functions and actions on the system are defined as Command and Control systems. Utterances like "Open Netscape" and "Start a new xterm" will do just that.

TELEPHONY

Some PBX/Voice Mail systems allow callers to speak commands instead of pressing buttons to send specific tones.

MEDICAL/DISABILITIES

Many people have difficulty typing due to physical limitations such as repetitive strain injuries (RSI), muscular dystrophy, and many others. For example, people with difficulty hearing could use a system connected to their telephone to convert the caller's speech to text.

EMBEDDED APPLICATIONS

Some newer cellular phones include C&C speech recognition that allows utterances such as "Call Home". This could be a major factor in the future of ASR.

For many applications, voice-based identity verification is attractive either as an adjunct or alternative to signatures, photos, passwords, keys and cards, et cetera, and an alternative to more complex methods based on imaging and analyzing a person's face, hand shape, fingerprint, retina, iris, et cetera.

A person's voice is a personal identifier that cannot be misplaced, forgotten, forged, or stolen --: "*Let me in, Hal.*"

> **AND: Impostors will be deterred by knowing that a copy of their voice is being saved for security analysis and possible use as evidence against them.**

The hardware for voice identity verification is inexpensive. It is included at no tangible cost in most PC's now on the market. We have demonstrated that compact, lower-cost hardware can also be developed for special applications.

Some areas where voice verification can be used as an added layer of security or as an alternative to other approaches include:

- Site or machine access security/control
- Computer or network/data access security
- Employee attendance monitoring/verification
- Access to gun lockers, drug lockers, et cetera
- Telephone and ATM financial transactions
- Other Point-of-Sale/Transaction verification
- "Smart Guns" for designated users

Limitations

Limitations

LIMITATIONS OF IWR

It becomes difficult to locate the beginning and end of an utterance if there are:

1.Weak fricatives (/f/,/th/,/h/) at the beginning or end.

2.Weak plosive bursts (/p/,/t/,/k/) at the beginning or end.

3.Nasals at the end.

4.Voiced fricatives which become devoiced at the end of words.

5.Trailing off of vowel sound at the end of the utterance.

LIMITATIONS OF AVR

It is important to understand the limits of current SR technology and how these limits affect system performance. The limitations of current AVR technology are:

Recognition accuracy can be affected by regional dialects, quality of the microphone, and the ambient noise level during a speech session. Much like the problem with pronunciation, dialect variations can hamper AVR performance. If our software is implemented in a location where the common speech contains local slang or other region-specific words, the voice may be misinterpreted or not recognized at all.

Poor microphones or noisy office spaces also affect accuracy. A system that works fine in a quiet, well-equipped office may be unusable in a noisy facility. In a noisy environment, the AVR is more likely to confuse similar-sounding words such as *out* and *pout,* or *in* and *when*. For this reason it is important to emphasize the value of a good microphone and a quiet environment when performing AVR activities.

Future Work

Future work

There are many factors involved in speech recognition. Although speech ecognition technology seems relatively new, computer scientists have been continually leveloping it for the past 40 years. They've made great strides in improving the systems ınd processes, but the futuristic idea of the computer hearing and understanding we are still a long way off. However, there are numerous on-going projects that deal with topics such as the following:

• Visual cues to help computers decipher speech sounds that are obscured by nvironmental noise
• Speech-to-speech translation project for spontaneous speech
• Building synthetic voices

Due to time constraints, we were only able to add eight people to our database. Ve recorded each person saying 'zero' twice and other words to attempt penetrating the ystem. We had a total of 30 words recorded to test the system.22 of the words succeeded n its task, either acknowledging the correct person or rejecting the wrong person .The ones that failed identified the wrong person.

The main cause of failure in the system is our voice identification portion. In our ystem, we calculate the Euclidean distance between the spoken word and the codebooks; hen the lowest value of the distances is identified as the correct person. Ideally we would ıave liked to set a threshold and accept the person whose distance falls below the hreshold. After a series of tests we found that was not possible. There could have been ufficient amount of noise in the background to interfere with the spoken word.

Our future work will be to expand the database of users and eliminate mimicked oices.

Epilogue

Epilogue

To summarize, our project primarily deals with providing speech recognition utilities. The human voice is considered the most common form of communication and is an ideal form of personal identification, because a person's voice can never be lost, stolen or shared without your knowledge. This motivated us to work on this project.

This project is a high level overview of how speech recognition works. It's not nearly enough detail to actually write a speech recognizer, but it exposes the basic concepts. Most speech recognition systems work in a similar manner, although not all of them work this way. Recognition accuracy can be affected by regional dialects , quality of the microphone and the ambient noise level during a speech session.

We have attempted to provide speech services in a user-friendly manner. With each new release of Windows, speech services are bound to become more powerful and user-friendlier. As hardware continues to become more powerful and cheaper, speech recognition should continue to become more accurate and useful. Future work could be done on designing web browsers to be speech-enabled. However, although speech recognition will work automatically, a web designer can use a little bit of care and get speech recognition to work even better.

Bibliography

Bibliography

WEBSITES

http://www-speech.sri.com/

http://www.smartcomputing.com/editorial/article.asp?article=articles/archive

http://joda.cis.temple.edu/~pwang/

http://tcts.fpms.ac.be/synthesis/introtts.html

www.voicerecognition.com

http://www.otolith.com/pub/u/howitt/lpc.tutorial.html

http://www.cs.uwa.edu.au/~eunjung/LPCproj/nlpchlac.html

http://dmoz.org/Computers/Algorithms/Speech_Recognition/

http://htk.eng.cam.ac.uk/

LITERATURE

VC++ Multimedia-Jarol Scott

Programming Microsoft Visual C++,1998,fifth edition

Win32 API programming-Charles Petzold

Programming in Turbo C++ - Robert Lafore

Discrete time signal processing-Alan.V.Oppenheim & Ronald Schafer

MATLAB Manual

Annexure

Annexure

SOURCE CODE

ISOLATED WORD RECOGNITION

```
/*Recognition of isolated words*/

#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include<string.h>
#include<dos.h>
#include<math.h>
#include<graphics.h>
#define pi 3.14
#define alpha 0.95
int nfr;        /*no. of frames*/
double tfr1;    /*no.of samples in a frame*/
void endpt();
void wow();
void pre(float a[40][100]);
void corr(float pe[40][100]);
void linear(float r[40][10]);
void cepst(float a1[40][10]);
void wtdcp(float h[40][10]);
void graphplot();
void recog();
void say1(int);
void say2(int);
int ch;
void main()
{
FILE *ifp,*ofp;
int p=44;
char in_file[14],s;
clrscr();
printf("\nMENU");
printf("\n---------");
printf("\nPlease choose between the set of words you would like to recognise");
printf("\n1.Words: Two - Five");
printf("\n2.Words: Six - Nine");
printf("\nEnter your choice");
scanf("%d",&ch);

printf("\nEnter file for recognition:");
scanf("%s",in_file);
ifp = fopen(in_file,"r");
if (ifp == NULL)
{
printf("\n File cannot be opened");
exit(0);
}
printf("\n please wait");
fflush(stdin);
ofp = fopen("res","w");
fseek(ifp,p,SEEK_SET);
while ((s=fgetc(ifp))!= EOF)
{
fprintf(ofp,"%d\n",s);
}
fclose(ifp);
fclose(ofp);
endpt();
wow();
graphplot();
recog();
}
/* end point location*/

void endpt()
{
FILE *ofp,*ofp1;
int temp,temp1,flag=1;
ofp = fopen("res","r");
if(ofp == NULL)
{
printf("\n --File cannot be opened");
exit(0);
}
putc('.',stdout);
ofp1 = fopen("res1","w");
```

```
rewind(ofp);
while(flag)
{
fscanf(ofp,"%d",&temp);
if ((temp == 127) || (temp == -128))
  continue;
else
{
 flag=0;
 do
 {
  if ((temp == 127) || (temp == -128))
   continue;
   else
   fprintf(ofp1,"\n%d",temp);
 }
 while(fscanf(ofp,"%d",&temp) !=EOF);
 }
 }
 fclose(ofp);
 fclose(ofp1);
 }
 /* windowing */
 /* calculates filter coefficients */
 void wow()
 {
 FILE *ofp1,*windptr;
 float w[40][100],a[40][100],tt;
 int i,j,nsamp,inp;
 ofp1 = fopen("res1","r");
 if (ofp1 == NULL)
 {
 printf("\n ++file cannot be opened ");
 exit(0);
 }
 windptr = fopen("wind","w");
 rewind(ofp1);
 nsamp = 3000;
 nfr = 30;
 tfr1 = nsamp/nfr;
 for(i=1;i<=nfr;i++)
 for(j=1;j<=tfr1;j++)
 {
 a[i][j] = 0.0;
```

```
w[i][j] = 0.0;
fscanf(ofp1,"%d",&inp);
tt = ((2*pi*j) / tfr1) / 180 *pi;
w[i][j] = 0.5 - 0.5*cos(tt);
a[i][j] = inp * w[i][j];
fprintf(windptr,"%f\n\t\t",a[i][j]);
}
fclose(windptr);
putc('.',stdout);
pre(a);
}
/*pre-emphasis */
/* improves signal to noise ratio*/
void pre(float a[40][100])
{
FILE *preptr;
float pe[40][100];
int i,j;
preptr = fopen("preemp","w");
for(i=1;i<=nfr;i++)
for(j=2;j<=tfr1;j++)
{
pe[i][j-1] = 0.0;
pe[i][j-1] = a[i][j] - alpha*a[i][j-1];
fprintf(preptr,"%f\n\t\t",pe[i][j-1]);
}
fclose(preptr);
putc('.',stdout);
corr(pe);

}

//auto corr
void corr(float pe[40][100])
{
 printf("\n calling corr");
 FILE *autoptr;
 float r[40][10];
 int i,j,ind,sa,x;
 autoptr=fopen("autocor","w");
 for(i=1;i<=nfr;i++)
  {
   sa=99;x=1;ind=1;
   do
 {
  r[i][ind]=0;
  for(j=2;j<=sa;j++)
   {
      r[i][ind]=r[i][ind]+pe[i][j-1]*pe[i][j-1+x];
   }
   fprintf(autoptr,"%f\n",r[i][ind]);
   x++;
   sa--;ind++;
  }
 while(sa>=90);
 }
fclose(autoptr);
printf("\nauto over");
linear(r);
}


/* linear predictive coding */
void linear(float r[40][10])
 {
 FILE *lpcptr;
 float k[40][10],a1[40][10];
 int i,j;
 lpcptr = fopen("lpca","w");
 for(i=1;i<=nfr;i++)
 {
 for(j=2;j<=10;j++)
 {
 if (j == 2)
 k[i][j-1] = r[i][j] / r[i][j-1];
 else
 k[i][j-1] = (r[i][j]*r[i][j-2] - r[i][j-1]*r[i][j-1]) / (r[i][j-2]*r[i][j-2] - r[i][j-1]*r[i][j-1]);
 }
 }
 for(i=1;i<=nfr;i++)
 {
 for(j=2;j<=10;j++)
 {
 if(j == 2)
 a1[i][j-1] = r[i][j] / r[i][j-1];
 else
 a1[i][j-1] = (r[i][j]*r[i][j-2] - r[i][j-1]*r[i][j-1]) / (r[i][j-2]*r[i][j-2] - r[i][j-1]*r[i][j-1]);
 fprintf(lpcptr,"%f\n\t\t",a1[i][j-1]);
}
}
fclose(lpcptr);
putc('.',stdout);
cepst(a1);
 }

 //cepstrum
 void cepst(float a1[40][10])
 {
 FILE *cepptr;
   float h[40][10],sum;
   int i,j,l;
   cepptr=fopen("cep","w");
   for(i=1;i<=nfr;i++)
    {
     h[i][1]=a1[i][1];
     fprintf(cepptr,"%f\n",h[i][1]);
     for(j=2;j<=9;j++)
       {
           for(l=1;l<=j-1;l++)
             sum=sum+(l/j)*h[i][l]*a1[i][j-l];
           h[i][j]=a1[i][j]+sum;
           fprintf(cepptr,"%f\n",h[i][j]);

            }
      }
    fclose(cepptr);
    putc('.',stdout);
    printf("\ncepstrum over");
    wtdcp(h);

    }


    //weighted cep

    void wtdcp(float h[40][10])
    {
     FILE *wtdptr;
     float htd[40][10];
     int i,j;
     wtdptr=fopen("wcptt","w");
      for(i=1;i<=nfr;i++)
 {
  for(j=1;j<=9;j++)
    {
        htd[i][j]=j*h[i][j];
        fprintf(wtdptr,"%f\n",htd[i][j]);
        }
  }
 fclose(wtdptr);
 putc('.',stdout);
}

void recog()
{
 FILE *inppat;
  float tty1=0.0,tty2=0.0,tty=0.0;
 FILE *tf,*nam;
 int i2,i=1,count,j,k=1,flag=0;
 double sum[5],min;
 char name[10][10];
 inppat=fopen("trn_fil","r");
 if (inppat == NULL)
 {
        printf("\n//file cannot be opened");
         exit(0);
 }
 i=1;
 while(fscanf(inppat,"%s",name[i])!=EOF)
 {
    i++;

 }
  fclose(inppat);
  tf = fopen("wcptt","r");
 if (tf == NULL)
 {
 printf("\n ==file cannot be opened");
 exit(0);
 }
 rewind(tf);
 switch(ch)
 {
  case 1:
         i=1;
         flag=1;
```

```
      break;
case 2:
      i=5;
      break;
}

 for(j=i;j<=i+3;j++)
       {
       nam=fopen(name[j],"r");
       printf("%s",name[j]);
         if (nam == NULL)
        {
        printf("\nfile cannot be opened");
        exit(0);
        }
        i2=1;
        while (i2 <= 220)
        {
                 fscanf(tf,"%f",&tty1);
                 fscanf(nam,"%f",&tty);
                 tty2 = fabs(tty1-tty);
                 sum[k]+=tty2;
                 i2++;
         }
         printf("\n sum %f",sum[k]);
         fclose(nam);
         k++;
         rewind(tf);

    }

   min=sum[1];
   for (i=1;i<=4;i++)
         {
          if(sum[i] <= min)
          {
           min = sum[i];
           count = i;
           printf(" i value %d",i);
          }
    }
    printf("\nmin %f",min);
    printf("\ncount %d",count);
    if(flag==1)
```

```
     say1(count);
    else
     say2(count);
   }

void graphplot()
{
FILE *windptr,*preptr,*cepptr,*autoptr,*wtdptr;;
float wintemp,pretemp,ceptemp,autotemp,wtdtemp;
int xx=1,prevx=1,prevy=100;
float tmpnum;
int d=DETECT,m;
initgraph(&d,&m,"c:\tc\bgi");
cleardevice();
windptr = fopen("wind","rb");
rewind(windptr);
setcolor(RED);
line(1,100,getmaxx()-5,100);
setcolor(CYAN);
line(30,getmaxy()-50,80,getmaxy()-50);
settextstyle(TRIPLEX_FONT,HORIZ_DIR,2);
outtextxy(82,getmaxy()-60,"t");
line(30,getmaxy()-50,30,getmaxy()-100);
outtextxy(10,getmaxy()-130," w(t)");
while ((fscanf(windptr,"%f",&tmpnum) != EOF))
{
wintemp = tmpnum * 100;
putpixel(xx,wintemp+100,GREEN);
setcolor(GREEN);
line(prevx,prevy,xx,wintemp+100);
prevx = xx;
prevy = wintemp+100;
xx++;
}
fclose(windptr);
setcolor(MAGENTA);
settextstyle(TRIPLEX_FONT,HORIZ_DIR,3);
outtextxy(500,400,"WINDOWING");
setcolor(YELLOW);
settextstyle(TRIPLEX_FONT,HORIZ_DIR,1);
outtextxy(500,getmaxy()-30,"press any key");
getch();
```

```
cleardevice();
xx = 1;
prevx = 1;
prevy = 100;
preptr = fopen("preemp","rb");
rewind(preptr);
setcolor(RED);
line(1,100,getmaxx()-5,100);
setcolor(CYAN);
line(30,getmaxy()-50,80,getmaxy()-50);
settextstyle(TRIPLEX_FONT,HORIZ_DIR,2);
outtextxy(82,getmaxy()-60,"t");
line(30,getmaxy()-50,30,getmaxy()-100);
outtextxy(10,getmaxy()-130," p(t)");
while ((fscanf(preptr,"%f",&tmpnum) != EOF))
{
pretemp = tmpnum * 100;
putpixel(xx,pretemp+100,GREEN);
setcolor(GREEN);
line(prevx,prevy,xx,pretemp+100);
prevx = xx;
prevy = pretemp+100;
xx++;
}
fclose(preptr);
setcolor(MAGENTA);
settextstyle(TRIPLEX_FONT,HORIZ_DIR,3);
outtextxy(500,400,"PRE-EMPHASIS");
setcolor(YELLOW);
settextstyle(TRIPLEX_FONT,HORIZ_DIR,1);
outtextxy(500,getmaxy()-30,"press any key");
getch();


 cleardevice();
 xx = 1;
 prevx = 1;
 prevy = 100;
 autoptr = fopen("autocor","rb");
 rewind(autoptr);
 setcolor(RED);
 line(1,100,getmaxx()-5,100);
 setcolor(CYAN);
```

```
line(30,getmaxy()-50,80,getmaxy()-50);
settextstyle(TRIPLEX_FONT,HORIZ_DIR,2);
outtextxy(82,getmaxy()-60,"t");
line(30,getmaxy()-50,30,getmaxy()-100);
outtextxy(10,getmaxy()-130," c(t)");
while ((fscanf(autoptr,"%f",&tmpnum) != EOF))
{
autotemp = tmpnum * 100;
putpixel(xx,autotemp+100,GREEN);
setcolor(GREEN);
line(prevx,prevy,xx,autotemp+100);
prevx = xx;
prevy = autotemp+100;
xx++;
}
fclose(autoptr);
setcolor(MAGENTA);
settextstyle(TRIPLEX_FONT,HORIZ_DIR,3);
outtextxy(330,400,"auto corr");
setcolor(YELLOW);
settextstyle(TRIPLEX_FONT,HORIZ_DIR,1);
outtextxy(500,getmaxy()-30,"press any key");
getch();

 cleardevice();
 xx = 1;
 prevx = 1;
 prevy = 100;
 cepptr = fopen("lpca","rb");
 rewind(cepptr);
 setcolor(RED);
 line(1,100,getmaxx()-5,100);
 setcolor(CYAN);
 line(30,getmaxy()-50,80,getmaxy()-50);
 settextstyle(TRIPLEX_FONT,HORIZ_DIR,2);
 outtextxy(82,getmaxy()-60,"t");
 line(30,getmaxy()-50,30,getmaxy()-100);
 outtextxy(10,getmaxy()-130," c(t)");
 while ((fscanf(cepptr,"%f",&tmpnum) != EOF))
 {
 ceptemp = tmpnum * 100;
 putpixel(xx,ceptemp+100,GREEN);
 setcolor(GREEN);
 line(prevx,prevy,xx,ceptemp+100);
prevx = xx;
prevy = ceptemp+100;
xx++;
}
fclose(cepptr);
setcolor(MAGENTA);
settextstyle(TRIPLEX_FONT,HORIZ_DIR,3);
outtextxy(330,400,"LINEAR PREDICTIVE CODING");
setcolor(YELLOW);
settextstyle(TRIPLEX_FONT,HORIZ_DIR,1);
outtextxy(500,getmaxy()-30,"press any key");
getch();


 cleardevice();
 xx = 1;
 prevx = 1;
 prevy = 100;
 wtdptr = fopen("wcptt","rb");
 rewind(wtdptr);
 setcolor(RED);
 line(1,100,getmaxx()-5,100);
 setcolor(CYAN);
 line(30,getmaxy()-50,80,getmaxy()-50);
 settextstyle(TRIPLEX_FONT,HORIZ_DIR,2);
outtextxy(82,getmaxy()-60,"t");
line(30,getmaxy()-50,30,getmaxy()-100);
outtextxy(10,getmaxy()-130," c(t)");
while ((fscanf(wtdptr,"%f",&tmpnum) != EOF))
{
wtdtemp = tmpnum * 100;
putpixel(xx,wtdtemp+100,GREEN);
setcolor(GREEN);
line(prevx,prevy,xx,wtdtemp+100);
prevx = xx;
prevy = wtdtemp+100;
xx++;
}
fclose(wtdptr);
setcolor(MAGENTA);
settextstyle(TRIPLEX_FONT,HORIZ_DIR,3);
outtextxy(330,400,"weigted vectors");
setcolor(YELLOW);
settextstyle(TRIPLEX_FONT,HORIZ_DIR,1);
```

```
outtextxy(500,getmaxy()-30,"press any key");
getch();
}
void say1(int count)
{
int d=DETECT,m;
initgraph(&d,&m," ");
 setcolor(CYAN);
 settextstyle(TRIPLEX_FONT,HORIZ_DIR,6);

 switch(count)
 {
  case 1:
    outtextxy(getmaxx()/2-5,getmaxy()/2,"TWO");
    break;
   case 2:
    outtextxy(getmaxx()/2-5,getmaxy()/2,"THREE");
    break;
   case 3:
    outtextxy(getmaxx()/2-5,getmaxy()/2,"FOUR");
    break;
   case 4:
    outtextxy(getmaxx()/2-5,getmaxy()/2,"FIVE");
    break;

    }
    printf("\n%d",count);
    delay(1000);
   closegraph();
   getch();
   }
   void say2(int count)
    {
   int d=DETECT,m;
   initgraph(&d,&m," ");
   setcolor(CYAN);
   settextstyle(TRIPLEX_FONT,HORIZ_DIR,6);
   switch(count)
    {
     case 1:
      outtextxy(getmaxx()/2-5,getmaxy()/2,"SIX");
      break;
      case 2:
      outtextxy(getmaxx()/2-5,getmaxy()/2,"SEVEN");
 break;
case 3:
  outtextxy(getmaxx()/2-5,getmaxy()/2,"EIGHT");
  break;
  case 4:
  outtextxy(getmaxx()/2-5,getmaxy()/2,"NINE");
  break;
 }
 delay(1000);
closegraph();
getch();

}
```

AUTOMATIC VOICE RECOGNITION

% Initialize.m

```
no_users = 0;
codebook = cell(25,3);
save no_users no_users;
save codebook codebook;
```

%avr.m

```
clear all;
loop = 1;

while loop == 1,
    clc;
    disp('           1. Add New User');
    disp('           2. Improve User DataBase');
    disp('           3. Identify User Utterance');
    disp('           4. Exit');
    option = input('Enter your choice: ');
    disp('Activity in Progress...');
    switch option
    case 1,
       AddNew;
    case 2,
       Improve;
    case 3,
       Identify;
    otherwise,
       loop = 0;
       break;
    end
    disp('Task completed successfully...');
    pause;
```

```
end
```

%Addnew.m

```
function AddNew

load codebook;
load no_users;

disp(' ');
disp(' ');
name = input('Enter your full name: ');
file = input('Enter the file name : ');

[s, fs] = wavread(sprintf('data\\train\\%s',file));
ps = PreProcess(s);
m = mfcc(ps,fs);

no_users = no_users + 1;
codebook{no_users,1} = name;
codebook{no_users,2} = adapt(m, codebook{no_users,2});
codebook{no_users,3} = Distortion(m,codebook{no_users,2});

save codebook codebook;
save no_users no_users;
```

%Identify.m

```
function Identify()
load no_users;
load codebook;

file = input('Enter the file name : ');
[s, fs] = wavread(sprintf('data\\test\\%s.wav',file));
ps = PreProcess(s);
if(length(ps) > 1024)
   ps(1025:end) = [];
end
m = mfcc(ps,fs);
user = 0;
distortion = inf;
for i = 1:no_users,
   dist = Distortion(m,codebook{i,2});
   if dist < distortion
      distortion = dist;
      user = i;
   end
end
disp(sprintf('The utterance was given by Miss. %s',codebook{user,1}));
codebook{user,2} = adapt(m, codebook{user, 2});
save codebook codebook;
```

%Adapt.m

```
function newcode = adapt(sample,oldcode)
if(length(oldcode) == 0)
   newcode = vqLBG(sample);
else
   temcode = vqLBG(sample);
   temcode = [temcode; oldcode];
   newcode = vqLBG(temcode);
end
```

%Improve.m

```
function Improve

load codebook;
load no_users;
disp(' ');
disp(' ');
name = input('Enter your full name: ');
file = input('Enter the file name : ');

[s, fs] = wavread(sprintf('data\\train\\%s',file));
ps = PreProcess(s);
m = mfcc(ps,fs);

for i = 1:no_users,
   if(deblank(codebook{i,1}) == deblank(name))
      user = i;
   end
end

codebook{user,2} = adapt(m, codebook{user, 2});
save codebook codebook;
```

%Distortion.m

```
function distortion = Distortion(sample,codebook)
[m,n] = size(sample);
[k,l] = size(codebook);
dtot = 0;
for i = 1:m,
   dmin = inf;
   for j = 1:k,
      d = sum(abs(sample(i,:)-codebook(j,:)).^2);
      if d<dmin
         dmin = d;
      end
   end
   dtot = dtot + dmin;
end
distortion = dtot/m;
```

%melbank.m

```
function m = melbank(L,N,fs)
N2 = N/2;
fk = (1:N2+1)/N*fs;
m = zeros(L,N2+1);
melmul = 2595;
meldiv = 700;
mL = melmul*log(1+fs/2/meldiv);
dm = mL/L;
mn = (0:L+1)*dm;
f = meldiv*(exp(mn/melmul) - 1);

fnm1 = f(1:L);
fn   = f(2:L+1);
fnp1 = f(3:L+2);
hmax = 2./(fnp1 - fnp1+1);
for i = 1:L,
   m(i,:) = ((fk > fnm1(i)) & (fk <= fn(i))).*hmax(i).*((fk-
fnm1(i))/(fn(i)-fnm1(i)))...
      + ((fk > fn(i)) & (fk <= fnp1(i))).*hmax(i).*((fnp1(i)-
fk)/(fnp1(i)-fn(i)));
end
```

%Mfcc.m

```
function c = mfcc(x,fs)
%Frame length.
N = 256;
%Offsets.
M = 64;
%no. of filters.
L = 16;

lenx=length(x);
%Calculation of total no. of frames in the run length sequence.
nf = fix((lenx-N+M)/M);
%Matrix initialization.
f=zeros(nf,N);
%window coefs.
w = hamming(N);

for i = 1:nf,
   %frame extraction.
   a = x((i-1)*M+1:(i-1)*M+N);
   %windowing.
   f(i,:) = (a .* w)';
   %spectrum.
   f1(i,:) = fft(f(i,:));
   %positive freq part extraction.
   f2(i,1:N/2+1) = f1(i,1:N/2+1);
```

```
end

%power spectrum.
f3 = abs(f2).^2;
%Mel Filter coefs in freq domain.
m = melbank(L,N,fs);
%spectrum after mel filtering.
for i = 1:nf,
   for j = 1:L,
      s(i,j) = sum(f3(i,:) .* m(j,:));
   end
end

%cepstrum with mel filtering.
sl = log10(s);
c = zeros(nf,L);
for i = 1:nf,
   c(i,:) = dct(sl(i,:));
end
%for i = 1:nf,
%   for n = 1:L,
%      for l = 1:L,
%         c(i,n) = c(i,n) + sl(i,l)*cos(n*pi*(l-0.5)/L);
%      end
%   end
%end

%c(:,8:end) = [];
%plot(c(:,1),c(:,2),'b.');
%hold on;
%C = vqLBG(c,16);
%plot(C(:,1),C(:,2),'r.');
```

%Preprocess.m

```
function t = PreProcess(s)
%[s, fs] = wavread('data\\train\\s2.wav');
%plot(s);
%pause;
%De Silencing.
t = 0;
len = 128;
th = .05;
m = length(s);
nf = m/len;
for i = 1:nf,
   if(sum(abs(s((i-1)*len+1:i*len)).^2) > th)
      t = [t;s((i-1)*len+1:i*len)];
   end
end
%plot(s1);

%PreEmphasize.
%m = length(s1);
```

```
%s2(1) = s1(1);
%for i = 2:m,
 %  s2(i) = s1(i) - 0.98*s1(i-1);
%end

%Normalisation.
%s3 = max(s1);
%t = s1/s3;
```

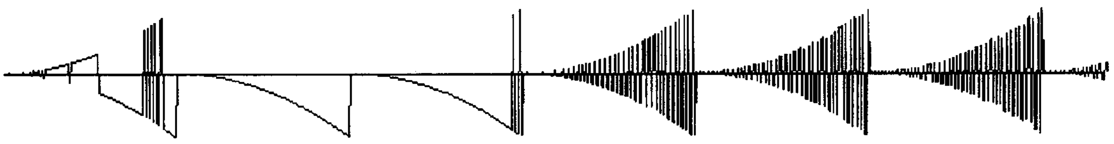
%Vqlbg.m

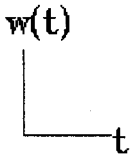
```
function C = vqLBG(x)
N = 32;
[n,L] = size(x);        %n - no of vectors , L - dimension of each vector.
C = zeros(N,L);         %CodeBook.
B = zeros(N,L);         %Bins or Partitions.
vecs = zeros(N,1);      %No. of Vectors in a Bin.
dist = zeros(N,1);      %Total distortion in a bin.
d = Inf;                %Dist of each centroid from the selected vector.
q = 1;                  %No. of bins.
Dmm1 = Inf;             %Distortion of the previous level.
Dm = 0;                 %Avg Distortion of the current level.
tolerance = 0.001;      %Tolerance value for the distortion.
split = 0.2;            %Split factor for the centroids.
M = 50;                 %Maximum optimisation iterations.
m = 0;
%Initial Centroid.
for i = 1:n,
   B(1,:) = B(1,:) + x(i,:);
end
C(1,:) = B(1,:) / n;

%Initial Split.
%C(2,:) = C(1,:) * (1-split);
%C(1,:) = C(1,:) * (1+split);
%q = 2;

while q<=N/2,
   getout = 0;
   %Split the CodeBook;
   for i = 1:q,
      C(i+q,:) = C(i,:) * (1-split);
      C(i,:)   = C(i,:) * (1+split);
   end
   q = 2*q;
   while getout == 0 & m <= M,
      %Optimisation.
      m = m+1;
      B = zeros(N,L);         %Bins or Partitions.
      vecs = zeros(N,1);      %No. of Vectors in a Bin.
      dist = zeros(N,1);      %Total distortion in a bin.
      for i = 1:n,
         dmin = Inf;
         imin = 0;
```

```
        for j = 1:q,
           d = (sum((x(i,:) - C(j,:)).^2));
           if dmin > d
              dmin = d;
              imin = j;
           end
        end
        %           [dmin,imin] = min(d);
        dist(imin)   = dist(imin) + dmin;
        vecs(imin)   = vecs(imin) + 1;
        B(imin,:)    = B(imin,:) + x(i,:);
     end
     Dm = sum(dist)/(n*L) + 0.0000001;
     if(abs((Dmm1 - Dm)/Dm) > tolerance)
        Dmm1 = Dm;
        %Find new CodeBook.
        empty = 0;
        for i = 1:q,
           if vecs(i) > 0
              C(i,:) = B(i,:)/vecs(i);
           else
              empty = 1;
           end
        end
        %Take proper action for empty bins.
        if empty ==  1
           for i = 1:q,
              if(vecs(i) == 0)
                 %Find densely populated cell.
                 %Place the empty centroid nearer to the most densely
populated centroid.
                 vmax = 0;
                 imax = 0;
                 for j = 1:q,
                    if vecs(j) > vmax
                       vmax = vecs(j);
                       imax = j;
                    end
                 end
                 C(i,:) = C(imax,:) * (1+split);
                 vecs(imax) = 1;
              end
           end
        end
     else
        getout = 1;
     end
  end
end
%plot(C(:,1),C(:,2),'r.');
```

OUTPUT SCREEN

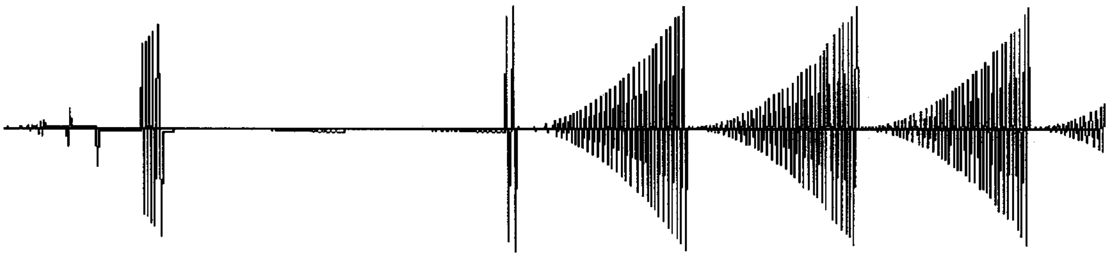
```
ISOLATED WORD RECOGNITION

Enter number of files for training:1

Enter file for analysis:seven.wav

Please wait......

Autocorrelation over.....

Cepstrum  analysis over...

Enter new file name :pat1


Weighted Cepstrum analysis over...

LPC over...

Windowing over...


MENU
---------
Please choose between the set of words you would like to recognise

1.Words: Two - Five

2.Words: Six - Nine

Enter your choice 2


Enter file for recognition:seven.wav
```

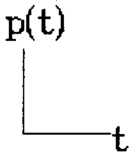




WINDOWING

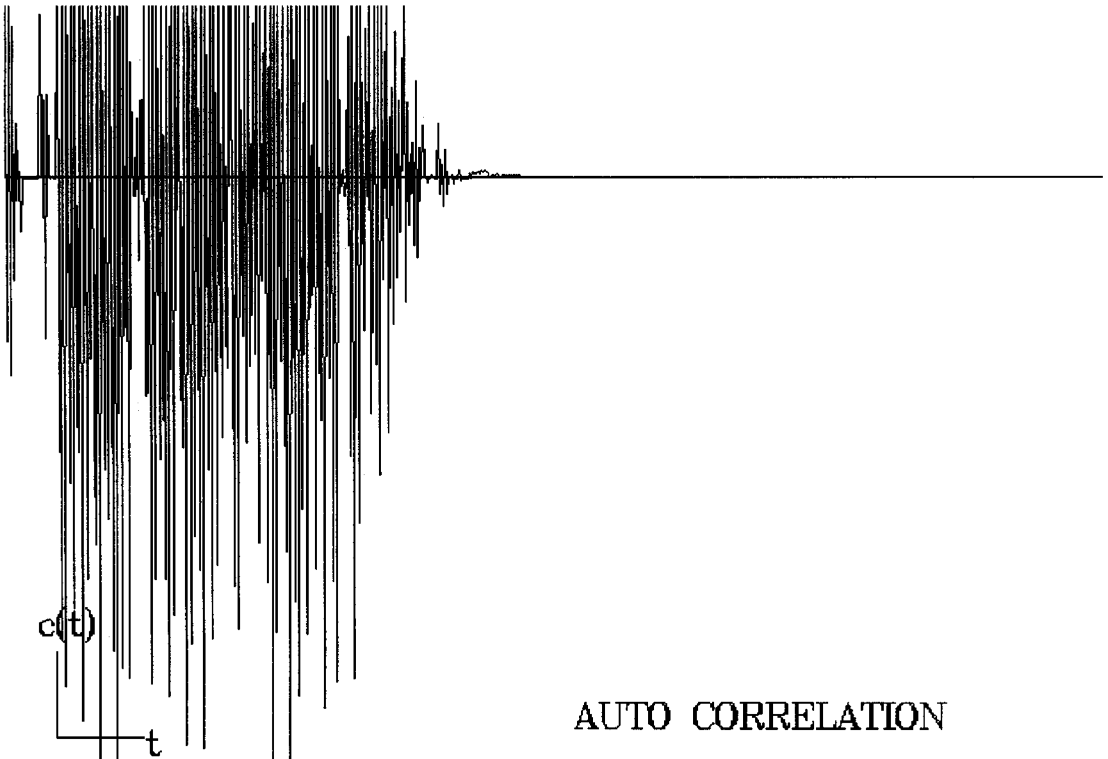
press any key





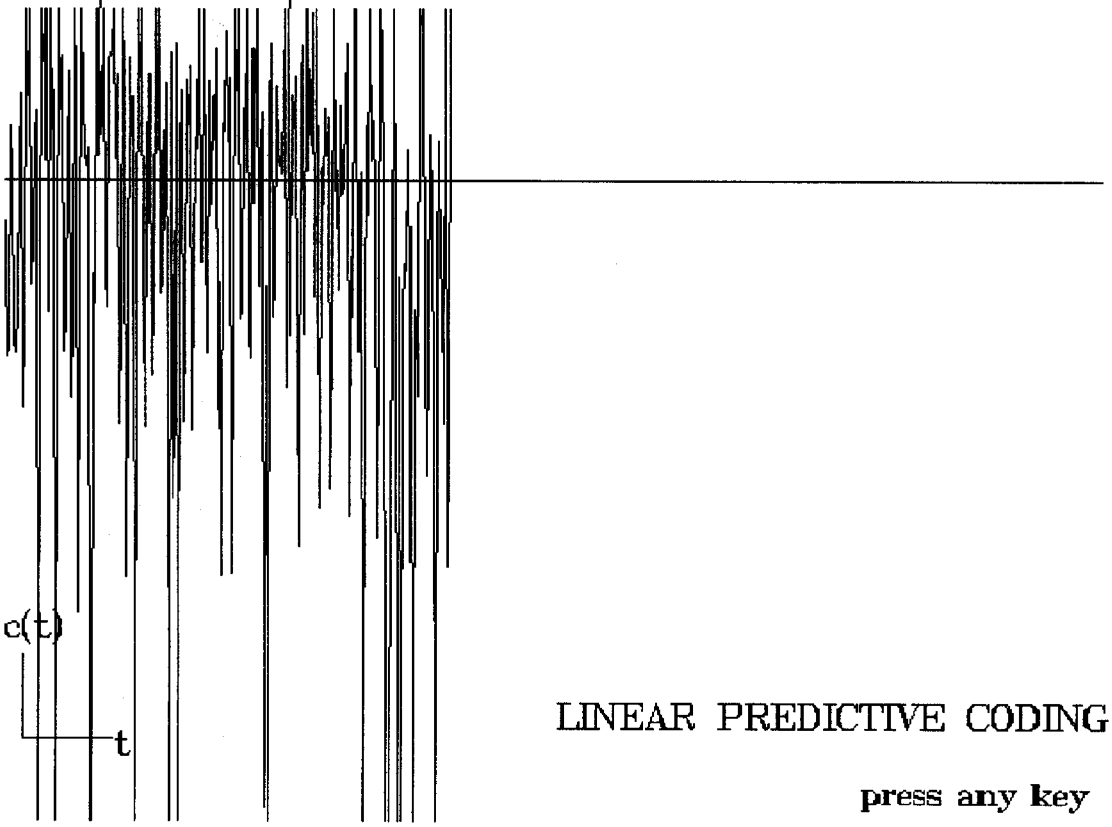
PRE–EMPHAS

press any key



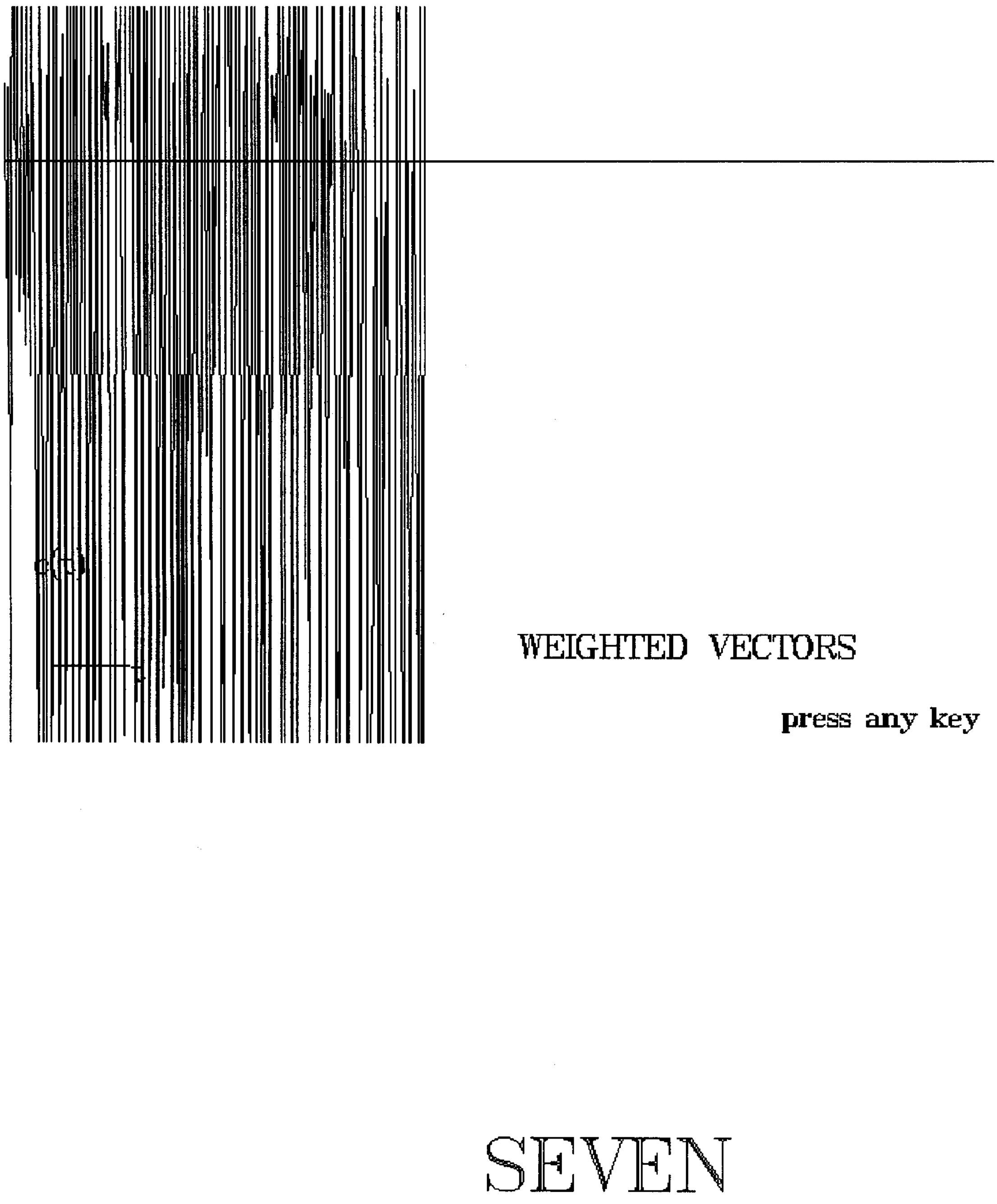
AUTO CORRELATION

press any key



LINEAR PREDICTIVE CODING

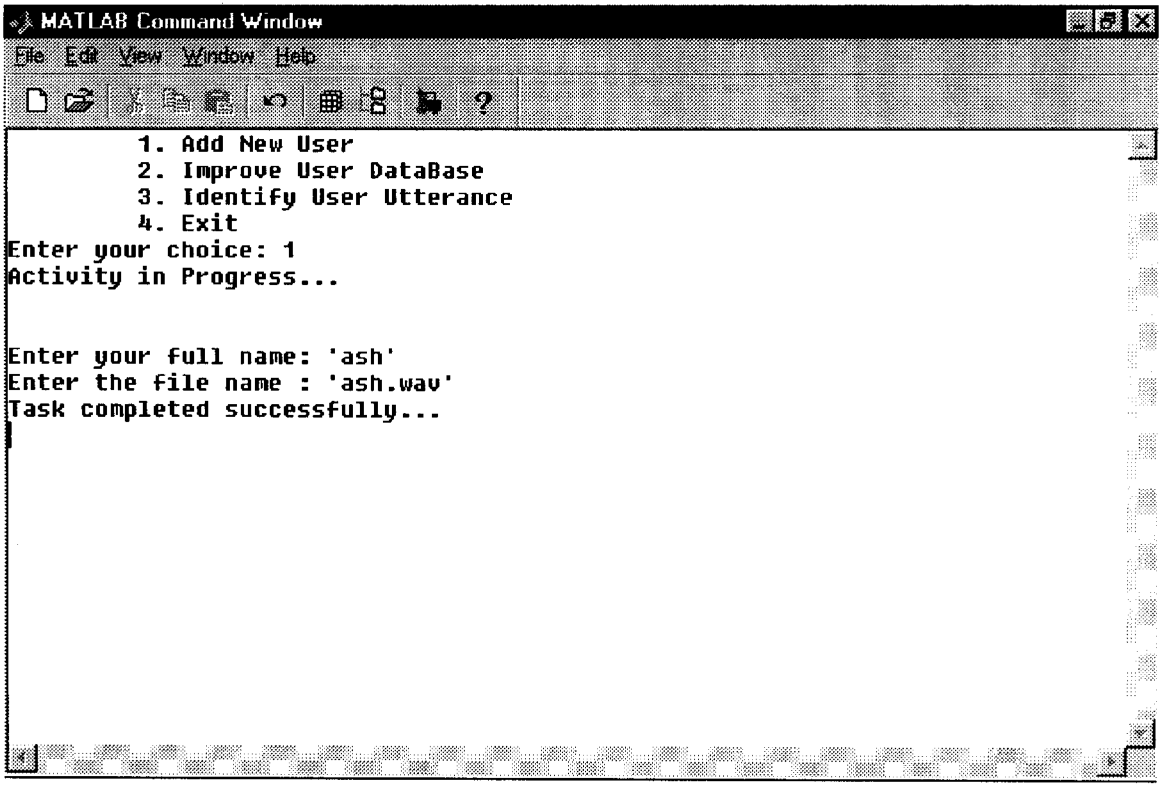
press any key



WEIGHTED VECTORS

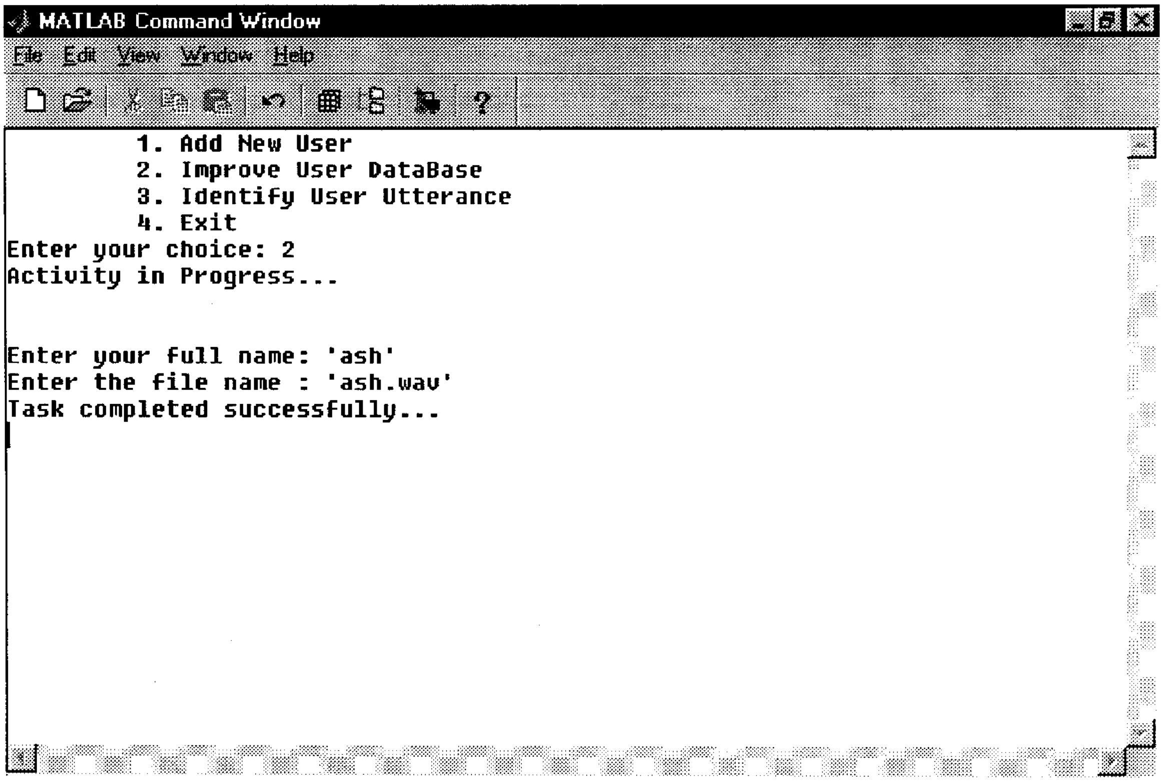
press any key

SEVEN



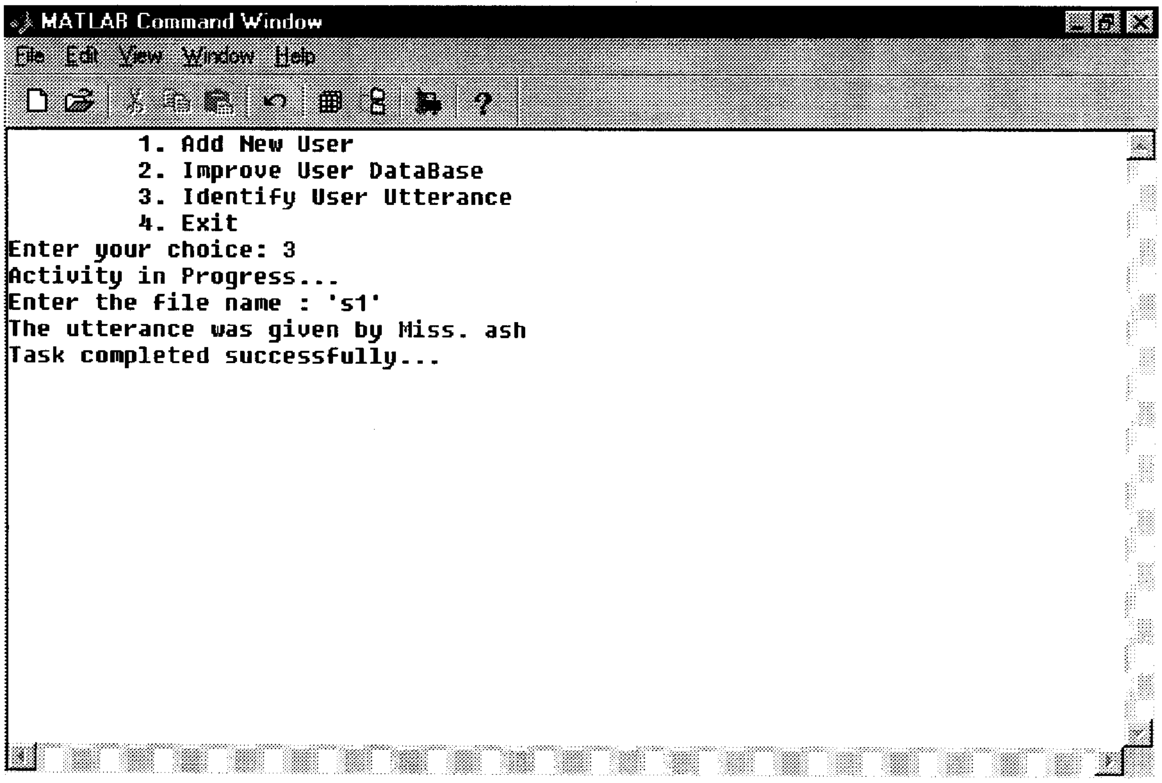
```
         1. Add New User
         2. Improve User DataBase
         3. Identify User Utterance
         4. Exit
Enter your choice: 1
Activity in Progress...


Enter your full name: 'ash'
Enter the file name : 'ash.wav'
Task completed successfully...
```

MATLAB Output For Adding new user



```
          1. Add New User
          2. Improve User DataBase
          3. Identify User Utterance
          4. Exit
Enter your choice: 2
Activity in Progress...


Enter your full name: 'ash'
Enter the file name : 'ash.wav'
Task completed successfully...
```

MATLAB Output For Improving User Database



```
         1. Add New User
         2. Improve User DataBase
         3. Identify User Utterance
         4. Exit
Enter your choice: 3
Activity in Progress...
Enter the file name : 's1'
The utterance was given by Miss. ash
Task completed successfully...
```

MATLAB Output For Identifying user utterance