

# Isolated Word Recognition for a Speaker Dependent System

Project Report

Submitted by

Kavitha Natarajan

Rajan .K

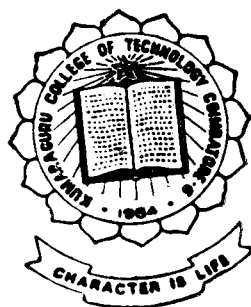
Karpagam .U

Venkatesan .A

Under the Guidance of

Prof. N. Shanmugam, M.Sc. (Engg)., M.S. (Hawaii)., Sr. M.I.E.E.E.,

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS  
FOR THE DEGREE OF  
BACHELOR OF ENGINEERING IN  
COMPUTER SCIENCE AND ENGINEERING  
OF THE BHARATHIAR UNIVERSITY, COIMBATORE



1995-96

Department of Computer Science and Engineering  
Kumaraguru College of Technology  
Coimbatore-641 006

# Bharathiar College of Technology

Coimbatore - 641 006.

Department of Computer Science and Engineering

## Certificate

This is to Certify that the Report entitled

"ISOLATED WORD RECOGNITION FOR  
A SPEAKER DEPENDENT SYSTEM"

has been submitted by

Kavitha Natarajan

Mr./Ms. Rajan. K. Karpagam. V, Venkatesan. N

in partial fulfilment of the requirements for the award of Degree of Bachelor of Engineering in the Computer Science and Engineering Branch of the Bharathiar University, Coimbatore - 641 006 during the academic year 1995-'96

\_\_\_\_\_  
(Guide)

\_\_\_\_\_  
(Head of Department)

Certified that the Candidate was Examined by us in the Project Work Viva-Voce Examination held on \_\_\_\_\_ and the University Register

Number is \_\_\_\_\_

\_\_\_\_\_  
(Internal Examiner)

\_\_\_\_\_  
(External Examiner)

*Dedicated to*

*our*

*Beloved Parents*

## ACKNOWLEDGEMENT

The authors place on record their heartfelt gratitude to the Principal **Dr. S. Subramanian**, for providing the infrastructure to conduct the study.

With great veneration and sincere gratitude, they record their profound thanks to their guide, **Prof. P. Shanmugam** M.Sc(Engg)., M.S(Hawaii)., M.I.E.E.E., M.I.C.S.I., M.I.S.T.E., Head, Department of Computer Science and Engineering for his systematic and able guidance, sustained support and constructive criticism.

They remain indebted to **Dr. R. Sundar**, Assistant Professor, Department of Computer Science and Engineering, I.I.T, Madras for sparing his precious time and his enormous efforts to enlighten us on the obscure areas of the project work. His critical accumen and clear judgement have been of unfailing help.

They are grateful to **Mrs. S. Devaki** B.E., M.S., for the timely encouragement and immense help offered. Sincere thanks are due to the Faculty members and the non-teaching staff.

Forever they remain obliged to their parents and the Imani family for being a source of inspiration and great help.

# CONTENTS

<u>TOPIC</u>	<u>PAGE NO.</u>
SYNOPSIS	1
INTRODUCTION	3
1.1 Purpose of Speech Recognition	3
1.2 Problem and its Objective	4
1.3 Solution Details	4
SPEECH PRODUCTION	6
2.1 Introduction	6
2.2 Speech Production Mechanism	6
2.3 Articulatory Phonetics	9
2.4 Acoustic - Phonetics	11
2.5 Coarticulation	11
2.6 Prosodic Features	12
2.7 Model for Speech Production	12
SPEECH RECOGNITION	14
3.1 Introduction	14
3.2 Principle Elements	15
3.2.1 Data Acquisition	16
3.2.2 Feature Extraction	19
3.2.3 Recognition Phase	29

IMPLEMENTATION	31
4.1 Minimum Requirements	31
4.2 Language Used	31
4.3 Algorithm Used	31
4.4 Data Structures	32
4.5 Program Implementation	33
4.6 Flow Chart	35
RESULTS AND CONCLUSION	40
FUTURE SCOPE	41
APPLICATIONS	42
REFERENCES	44
SAMPLE OUTPUTS	45
APPENDIX	48



## SYNOPSIS

Notwithstanding the spectacular advances in modern technology, speech still remains the most effective and efficient means of communication between humans. The effectiveness and efficiency of speech for communication is due to the fact that speech, language and human intelligence have evolved together. It is for this reason that research on man - machine communication using speech is attractive.

The lungs and the associated respiratory muscles constitute the source of power for the speech production system. This power is used to generate the quasi - periodic acoustic signal by means of the vibrating vocal cords for voiced sounds such as vowels. For fricative sounds, it is converted into an aperiodic signal due to the high velocity frictional flow of air through a narrow constriction formed in the mouth. For plosive sounds, it is converted into short bursts of noise by the sudden release of pressure which is built up by completely closing the vocal tract for short durations. Thus, all of the above mechanisms convert the more or less steady pressure of the lungs into an acoustic signal which is used for exciting the vocal tract system to generate audible speech sounds.

The shape of the vocal tract uniquely determines the sound that is produced. Speech recognition may be visualized as determining or surmising from the information contained in the speech signal,



the causative movements of the articulators and from thence the spoken message. This is done by extracting certain parameters from the speech signal. Hence the acoustic signal has to be analyzed in either time domain or frequency domain. Time domain analysis has been carried out for isolated words. The steps in analysis are End - point location which is carried out to determine the beginning and end of a speech signal, Windowing to obtain the filter coefficients, pre-emphasis, auto-correlation, linear predictive coding, cepstrum and weighted cepstrum analysis to estimate the short time smoothed power spectrum. The parameters obtained are stored as templates. Recognition is done using the dynamic time warping algorithm which is a non - linear time normalization technique which compensates for the time scale differences between patterns.



## INTRODUCTION

### 1.1 PURPOSE OF SPEECH RECOGNITION

The purpose of speech is communication. Speech communication is the transfer of information from one person to another person via speech. There are two ways of characterizing the communication potential of speech. One is information theory in which speech can be represented in terms of its message content or information. Another way is to characterize speech in terms of the signal carrying the message information, i.e., the acoustic waveform. The chain of events from the conception of a message in the speaker's brain to the perception of the message in the listener's brain is called the speech chain. This chain consists of a speech production process, transmission through a medium and a speech perception process.

The information communicated through speech is of a discrete nature and it can be represented as a concatenation of elements from a finite set of symbols. These symbols may be classified as phonemes. English language can be represented as a set of around 42 phonemes.

There are two major factors to be considered in any system;

- 1) Preservation of message content in the speech signal.
- 2) Representation of message content in a form that is convenient for transmission or storage.<sup>[SA1]</sup>



## **1.2 PROBLEM AND ITS OBJECTIVE**

The problem is mainly concerned with surmising the numerals zero through nine as they are uttered. Isolated word recognition is an endeavour towards a world where machines learn how to communicate with humans rather than vice - versa.

The problem of speech recognition has engaged the attention of speech researchers for several decades, not merely for its practical applications but also due to the intrinsic challenge in replicating an activity that only the human brain was capable of performing.

## **1.3 SOLUTION DETAILS**

Speech recognition consists in assigning to the time domain acoustic signal a sequence of labels from the label bank or a vocabulary consisting of a finite number of distinct labels. The labels here are words. Recognition here is done based on some degree of correspondence between the utterance of the word and its acoustic manifestation. This correspondence becomes very complicated due to speaking rate fluctuations, noise, articulatory laxity and context dependencies in the acoustic signal. Consequently, speech recognition remains as yet an open problem. There on, the time domain analysis of the acoustic signal of speech is carried out to represent the word as a



## ISOLATED WORD RECOGNITION

set of vectors forming a template. The whole process has been implemented using 'C' language with Visual Basic as the front end tool.

## **SPEECH PRODUCTION**

### **2.1 INTRODUCTION**

Speech signals are composed of a sequence of sounds. These sounds and the transitions between them serve as a symbolic representation of information. The arrangement of these sounds (symbols) is governed by the rules of language. The study of these rules and their implications in the human communication is the domain of *linguistics* and the study and classification of the sounds of speech is called *phonetics*.

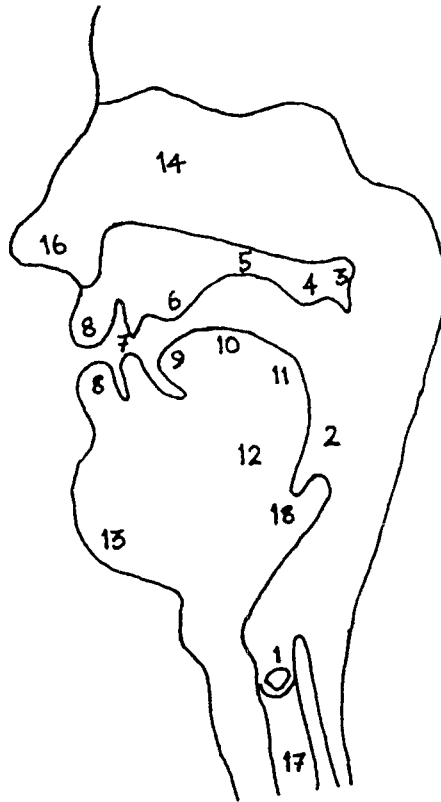
### **2.2 SPEECH PRODUCTION MECHANISM**

The organs that are involved in producing speech can be divided into three main groups: lungs, larynx and parts of the mouth and nose. The lungs are the source of an airflow that passes through the larynx and oral tract before exiting the mouth as pressure variations constituting the speech signal. The source of most speech occurs in the larynx where vocal folds can partially or completely obstruct airflow from the lungs. During normal breathing, the vocal folds remain sufficiently parted to allow free air passage without creating sound. If they are closed sufficiently, airflow may be hindered enough to create a turbulent noise at the glottis, the variable opening between the vocal folds. Speech arising from such a noise is called whisper. Noise can be generated by the same mechanism farther up in the vocal tract at a narrow constriction involving the tongue and roof of the mouth or the lips and



## ISOLATED WORD RECOGNITION

teeth. A noise source primarily excites the portion of the vocal tract in front of its source constriction and the shorter cavities excited lead to higher frequencies than for whisper. All sounds involving noise are aperiodic.



1. VOCAL FOLDS
2. PHARYNX
3. VELUM
4. SOFT PALATE
5. HARD PALATE
6. ALVEOLAR RIDGE
7. TEETH
8. LIPS
9. TONGUE TIP
10. BLADE
11. DORSUM
12. ROOT
13. MANDIBLE
14. NASAL CAVITY
15. ORAL CAVITY
16. NOSTRIL
17. TRACHEA
18. EPIGLOTTIS

CROSS-SECTIONAL VIEW OF THE VOCAL TRACT

If the vocal tract or glottis is closed completely, air flow ceases and no sound emerges. A class of sounds called stops or plosives utilize such air flow interruptions of 20 - 150 ms with different acoustic aspects depending on whether the closure occurs at the vocal tract or at the lips. Fricative sounds employ a narrow fixed vocal tract constriction, while plosive sounds close and release a full occlusion. Another class of sounds called the sonorant sounds, excite the vocal tract through periodic vocal fold vibrations. The rate of vibration is called the fundamental frequency ( $F_0$ ). The period between successive vocal fold closures,  $T_0 = 1/F_0$ , is called the pitch period. Phonation (vibration) occurs when

- a) Vocal folds are elastic and close together
- b) There is a sufficient difference between subglottal pressure and supraglottal pressure.

The vocal tract is the most important component in the speech production process. It has two different speech functions

- 1) It modifies the spectral distribution of energy in the glottal pulse and
- 2) It can contribute to the generation of stop and fricative sounds.

Different sounds are primarily distinguished by their periodicity, spectral shape and duration. The vocal tract can be modelled as an acoustic tube with resonances called formants and antiresonances. The



periodicity of excitation of the vocal tract is called the pitch.

### 2.3 ARTICULATORY PHONETICS

Speech sounds can be analysed, described and classified in relation to the chain of events that take place while the speaker communicates the message to the listener. Articulatory, acoustic and perceptual aspects of speech are normally considered for such an analysis.

Speech sounds may be associated with phonetic features or articulatory configuration. The sound [s], for example, has features voiced, fricative and alveolar, and producing an [s] involves an open glottis, a raised velum and a single narrow constriction in the alveolar region. Articulatory phonetics relates features of each sound to the position and gestures of vocal tract articulators involved in speech production.

In many speech applications, the signal is divided into segments whose boundaries coincide with points of major changes in the vocal tract shape. The gestures associated with successive phonemes overlap to such an extent that each phonemes' features often affect several preceding and ensuing phonemes. Words are usually divided into parts called syllables. Each syllable has one vowel obligatory and may contain optionally consonants before and after the vowel.

Consonants are classified in terms of manner and place of articulation. Manner of articulation concerns how the vocal tract restricts the air

flow. It determines the path air flow takes and the degree to which it is impeded by vocal tract constrictions. Completely stopping air flow by an occlusion creates a stop consonant. Vocal tract constrictions of varying degree occur in fricatives, glides, liquids and vowels. The state of velum can neither be opened(lowered) as in nasal sounds or closed as in oral sounds or positioned somewhere inbetween(nasalized words).

Place of articulation refers to the location or the point of constriction made along the vocal tract by the articulators. Vowels are classified based on the basis of tongue and lip positions. Place of articulation is associated with consonants, rather than vowels because consonants use a relatively narrow constriction. The following points are traditionally associated with consonant constrictions

- 1) Labial : Both lips constrict (/p/, /b/, /m/, /w/).
- 2) Labiodental : Lower lip contacts upper teeth (/f/, /v/).
- 3) Dental : Tongue tip touches the back of the upper incisor teeth (/t/, /d/).
- 4) Interdental : Tongue tip protrudes between upper and lower teeth.
- 5) Alveolar : Tongue tip approaches alveolar ridge (/t/, /d/, /s/, /r/, /l/).
- 6) Postalveolar : Tongue tip curls up (/n/, /s/, /l/).
- 7) Palatal : Tongue blade or dorsum constricts with the hard palate (/ts/, /dz/, /j/).
- 8) Velar : Back of tongue approaches soft palate or velum (/k/, /g/, /kh/, /gh/).
- 9) Pharyngeal : Constriction at the pharynx.
- 10) Glottal : Complete obstruction of air flow at the glottis.



## 2.4 ACOUSTIC - PHONETICS

The relationship between phonemes and their acoustic realizations forms the basis for many speech applications such as coding and recognition. Acoustic phonetics describes these relationships. Acoustic phonetics considers the differentiations of sounds on an acoustic basis.

## 2.5 COARTICULATION

The phenomenon of coarticulation involves changes in the articulation and consequently the acoustics of a phoneme due to its phonetic or phonological structural context, immediate or across the syllables. Coarticulation occurs in varying degrees depending on context. Some coarticulatory features are universal and some others are language specific. It is essential to examine the effects of coarticulation on individual phonemes and develop a knowledge base for the changes in their spectra. This knowledge can be used to improve the performance of the speech recognition systems and also to bring naturalness in the synthetic speech. There are two types of coarticulations, the forward coarticulation and the carryover coarticulation. The forward coarticulation is called anticipatory or right to left because the target of the phoneme on the right induces motion on the articulator during a prior phoneme. Carryover coarticulation or left to right coarticulation is the one in which some of the features of the previous phoneme persist into the



following one. Coarticulation affects not only the spectral characteristics but the durations of the phonemes also.

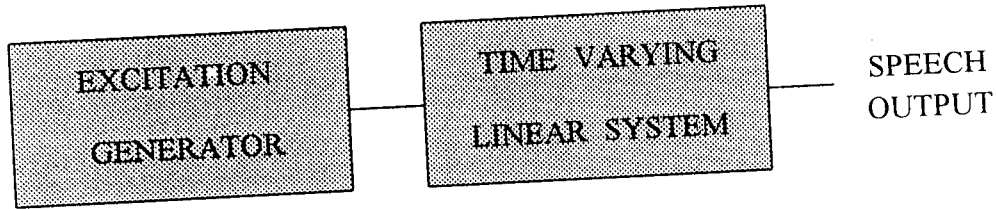
## **2.6 PROSODIC FEATURES**

These features are also known as suprasegmentals. The suprasegmental knowledge refers to aspects like rhythm, melody and stress. The prosodic features are quantity (duration), stress (intensity) and intonation (pitch). These factors although give cues about segmental properties, but relate more to the quality, naturalness and speaker identity in a speech signal.

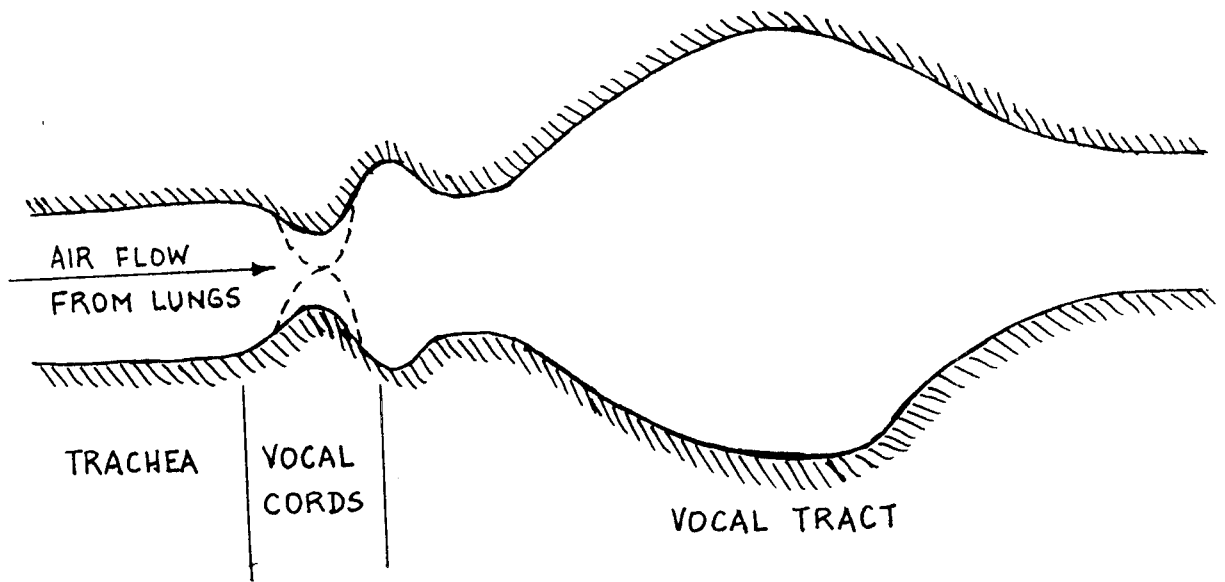
Application of prosodic rules are useful in syntactical boundary detection, stressed syllable location, speech rate by measuring inter stress intervals. Prosodic features are important aids to speech recognition and synthesis.

## **2.7 MODEL FOR SPEECH PRODUCTION**

A simple model for speech production consists of an excitation generator and a time varying linear system. This is a source - system model for speech production.



SOURCE - SYSTEM MODEL FOR SPEECH PRODUCTION



SCHEMATIC REPRESENTATION OF THE VOCAL SYSTEM

The glottis and vocal folds represent the excitation generator and the vocal tract is represented by the time varying linear system. The parameters of the source and the system can be appropriately varied for the production of different sounds.

**SPEECH RECOGNITION****3.1 INTRODUCTION**

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. It is difficult to recognize continuous speech whereas some commercial success has been achieved by IWR systems. Further CSR and IWR systems may be classified as speaker dependent recognition and speaker independent recognition systems. In the former, the system is capable of recognizing speech in a particular dialect uttered by a single speaker. Independent recognition is very difficult to achieve as it requires a large amount of training. The voice of different people differ based on their sex, context, pitch and duration.

Automatic speech recognition systems have a variety of options to be considered. They are

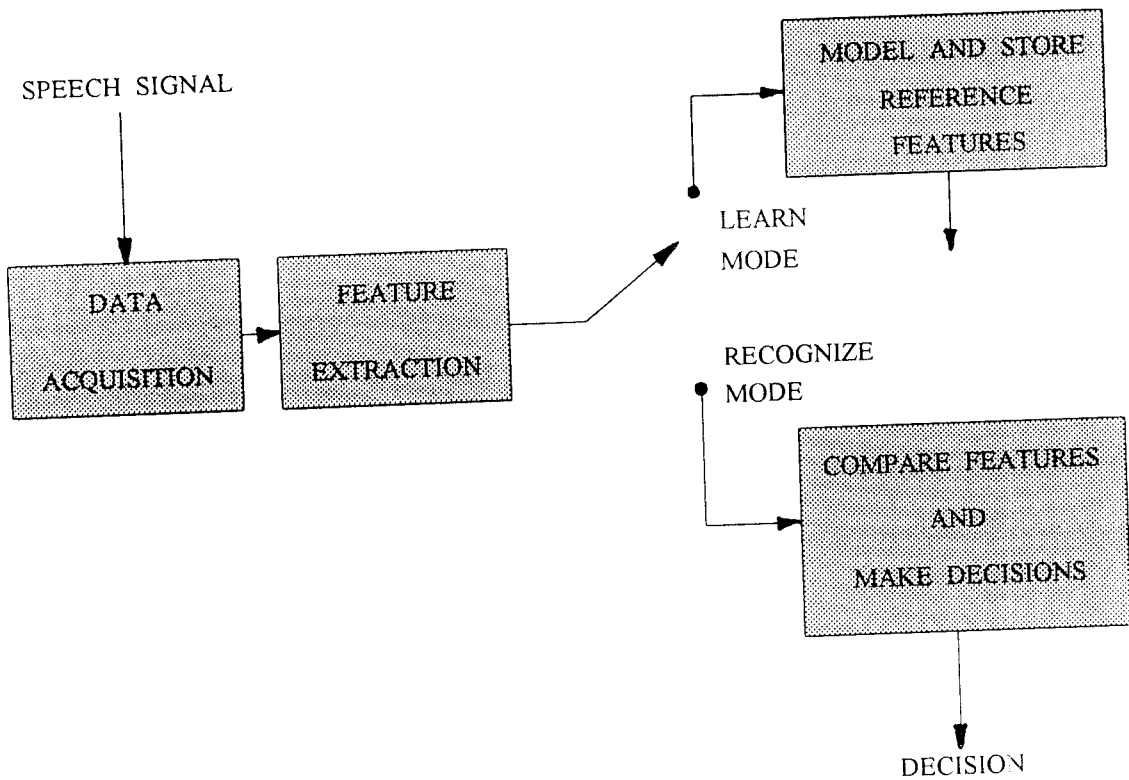
- 1) Type of speech - isolated words, continuous speech.
- 2) Number of speakers - single speaker system, multiple designated speakers, unlimited population.

- 3) Type of speakers - cooperative, casual, male, female, child.
- 4) Speaking environment - soundproof booth, computer room, public place.
- 5) Transmission system - high quality microphone, close talking microphone, telephone.
- 6) Type and amount of system training - no training, continuous training, fixed training.
- 7) Vocabulary size - small vocabulary(1 - 20 words), medium vocabulary(20 - 100 words), large vocabulary(greater than 100 words).

This project concerned with isolated word recognition for a speaker dependent system having a computer room speaking environment. It is a single speaker system with the female as the speaker. Transmission system used is a close talking microphone.

### **3.2 PRINCIPLE ELEMENTS**

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



### ELEMENTS OF SPEECH RECOGNITION

#### 3.2.1 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 atleast reduces the effect of environmental noise.

Sound is inherently analog. Therefore, sound recorded should be converted into its digital form. Converting an analog waveform into a digital form works by taking small samples of the waveform at fixed intervals as sound is captured. The process of converting an analog signal into its digital form is known as *sampling*. This process establishes the frequency of the waveform. At the same time, the values of the waveform amplitude are captured, defining the amount of information stored per sample.

Three characteristics determine the quality and size of the digital waveform file. They are

1) *Frequency* : The higher the frequency of sampled sound, the more is the disk space required for storage. The three standard sampling frequencies are 11.025 KHz, 22.05 KHz and 44.1 KHz.

2) *Amount of information* : This measures the precision with which the sample is measured. Sampling at 8 or 16 bit resolution is possible. An 8 bit sample divides each sample into 256 equal units whereas a 16 bit sample divides each sample into 65,536 equal units. The higher the sampling resolution, the more accurately the sample resembles the original analog waveform.

3) *Number of channels* : This specifies whether a recording produces a one-channel waveform (known as monoaural or just mono) or produces two channels of waveforms (known as stereo).

There is usually a trade-off between sound quality and storage space required. A sampling rate of 11.025 KHz does not give a good sound quality but utilizes less storage space. Sound has been captured at a rate of 11.025 KHz with a sampling resolution of 8 bits per sample and mono recording. This is the first step where discretization or digitization of the analog speech signal is done.

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

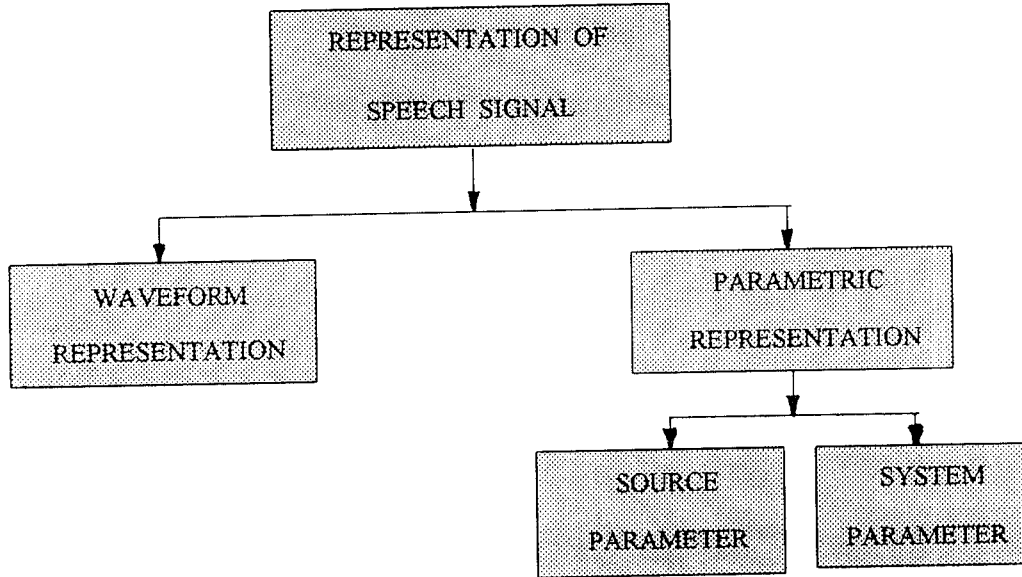
'RIFF'<file\_size>'WAVE'<header chunk><data chunk>

The first four bytes are the word RIFF. These identify the file as a multimedia resource interchange file. The first chunk is of type WAVE. This identifies the file as a waveform digital audio file. The next chunk is the fmt chunk. This specifies the format of the data contained in the rest of the file.



### 3.2.2 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 is carried out to represent the speech signal in terms of its parameters .



REPRESENTATION OF SPEECH SIGNAL

The representation of speech signal can be classified into 2 groups namely, waveform representation and parametric representation. The former is mainly concerned with a sampling and quantization process which preserves the signal. In parametric representation, the signal is further processed to obtain the parameters of the model for speech production.



Processing may be carried out in either the time domain or the frequency domain. Time domain methods involve processing the waveform of the speech signal directly. Frequency domain methods involve either explicitly or implicitly some form of spectrum representation.

The assumption in most speech processing schemes is that the properties of the speech signal change relatively slowly with time. This assumption leads to a variety of short-time processing methods in which the short segments of speech signal are isolated and processed as if they were short segments from a sustained sound with fixed properties. This may be repeated as often as desired. These short segments, known as analysis frames, often overlap each other. The result of processing on each frame may result in a single number or a set of numbers. Such processing produces a new time dependent sequence which can serve as a representation of the speech signal.

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

1) End - point location : This deals with the problem of locating the beginning and end of a speech utterance in a background of noise i.e., it becomes necessary to separate the voiced region from the unvoiced region. 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.

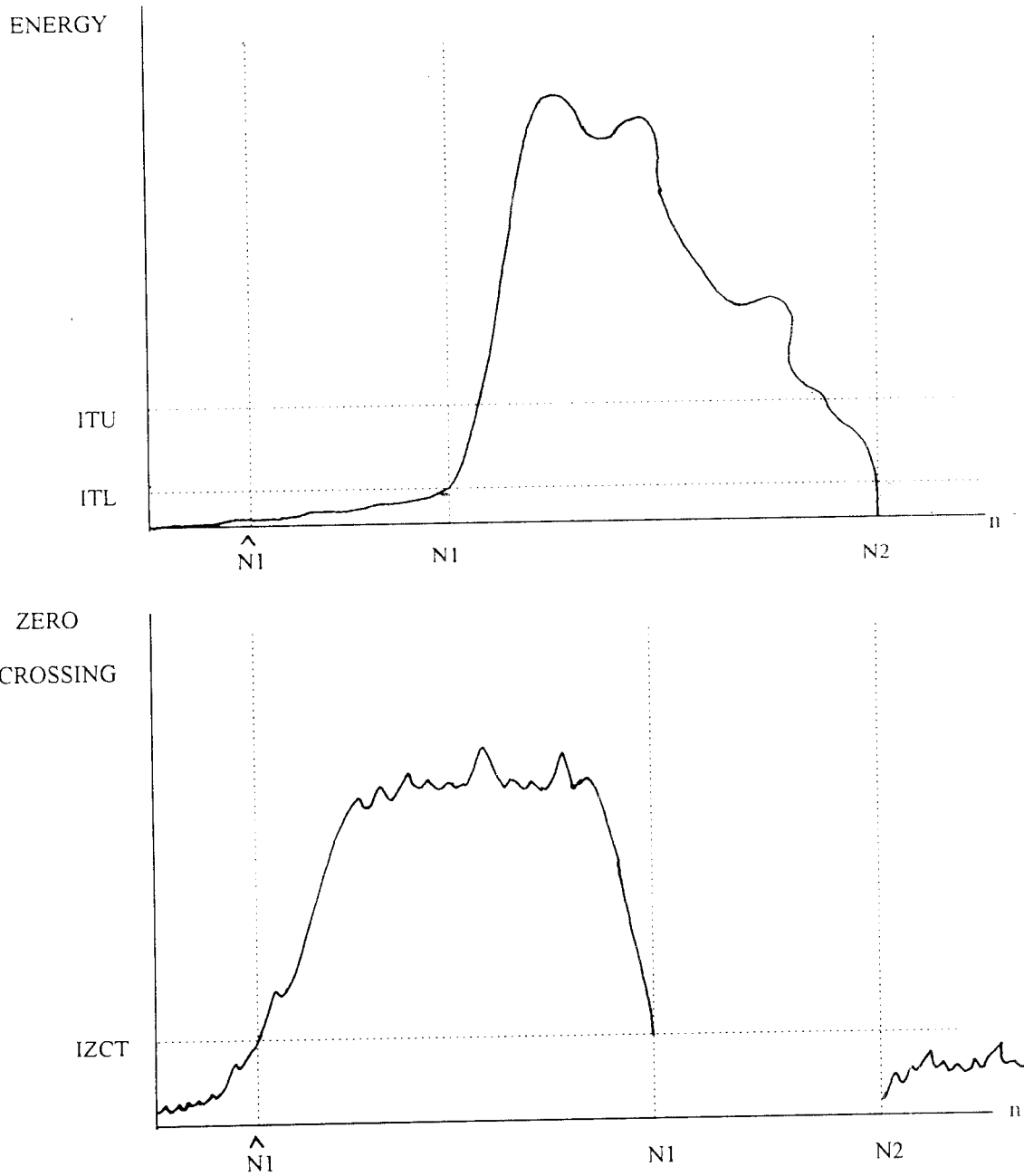
It becomes difficult to locate the beginning and end of an utterance if there are

- 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 sounds at the end of the utterance.

In spite of the difficulties posed by the above situations, energy and zero-crossing rate representations can be combined to serve as the basis of locating the beginning and end of a speech signal.

The algorithm for end - point location uses the following representations ( i ) the number of zero - crossings per 10 msec frame and ( ii ) the average magnitude computed with a 10 msec window. Both functions are computed for the entire record interval at the rate of 100 times / sec. Assuming that there is no speech in the first 100 msec interval, the mean and standard deviation of the average magnitude and zero - crossing rate is determined which gives a statistical characterization of the background noise. Using this statistical characterization, average magnitude in the interval, zero-crossing rate and energy threshold are computed. The average

magnitude profile is searched to find the interval in which it exceeds the threshold (ITU). Assuming that the beginning and the ending lie outside this interval, we work backwards from the point where magnitude ( $M_n$ ) first exceeds ITU, the point where magnitude falls below a lower threshold (ITL) is tentatively selected as the beginning point ( $N_1$ ). A similar procedure is carried out to determine the end point ( $N_2$ ). This double threshold procedure ensures that dips in the average magnitude function do not falsely signal the end point. Assuming again that the beginning and end points do not lie in the interval  $N_1$  to  $N_2$ , it is worked backwards from  $N_1$ , comparing the zero-crossing rate with a threshold IZCT which is determined from the statistics of zero-crossing rate for the background noise. If the zero-crossing rate exceeds the threshold 3 or more times, the beginning point is moved to the first point at which the zero-crossing threshold was exceeded. Otherwise  $N_1$  is chosen as the beginning point. A similar procedure is carried out to determine the end point.



Average magnitude and zero - crossing rate measurements for a word with a strong fricative in the beginning.



2) 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.

There are different types of windows such as rectangular, Hamming, Hanning window etc. Here Hanning window which is the most efficient and simple has been implemented. The formula is given by

$$w(n) = 0.5 - 0.5 \cos 2\pi n / M, 0 < n \leq M \\ = 0, \text{ otherwise}$$

where  $n$  - sample number in a particular frame

$M$  - number of samples in a particular frame

The sample values of the speech signal are multiplied with the window value. The main purpose of windowing is filtering. An efficient filter produces adequate smoothing. It is an advantage to have the impulse response (window) to be positive since this causes the average magnitude to be positive. FIR filters (Hanning impulse responses) have the advantage that the output can easily be computed at a lower sampling rate than the input by moving the window more than one sample between computations.



3) Pre - Emphasis : Filter coefficients obtained after analysis in each frame is then subjected to pre-emphasis. Frequency of a speech signal varies from 300 Hz - 3 KHz. In the beginning, the speech signal generally has a high noise level. Hence the signal to noise ratio is generally low. In order to improve the signal to noise ratio(i.e., to reduce the noise level in the speech signal), pre-emphasis is carried out. This improves the quality of the signal.

The formula implemented during pre - emphasis is given by

$$p_i = a_i - \alpha \cdot a_{i-1}$$

where  $\alpha = 0.95$ , a constant

$a_i$  = sample  $i$ 's windowed value

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

Hence if 100 samples are considered to be in one frame, windowing produces 100 values and pre - emphasis produces 99 values in one frame.

4) Autocorrelation : The parameters of the speech signal may be distorted with time. In order to minimize this distortion, autocorrelation is carried out. The autocorrelation function representation of the signal is a convenient way of displaying certain properties of the signal. Some properties of the autocorrelation 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.

At certain periodic intervals, the autocorrelation function attains a maximum. Regardless of the time origin of the signal, the period can be estimated by finding the location of the first maximum in the autocorrelation function. The autocorrelation function contains much more information about the detailed structure of the signal. Hence it is important to consider how the definition of the autocorrelation function can be adapted to obtain a short - time autocorrelation function representation of speech.

The formula employed in auto - correlation depends on the order of the polynomial. For a 10th order polynomial the correlation coefficients are as follows

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

$$R(2) = \sum_{i=1}^{98} a_i \cdot a_{i+2}$$

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

[SA2]





5) Linear predictive coding : This method is the most predominant technique for determining the basic speech parameters such as pitch, formants etc. The importance of this method lies both in its ability to provide extremely correct estimates of the speech parameters and its relative speed of computation.

The basic principle in linear predictive analysis is that a speech sample can be approximated as a linear combination of past speech samples. By minimizing the sum of the squared differences between the actual speech samples and the linearly predicted ones, a unique set of predictor coefficients can be determined. Linear predictive analysis produces basically an all - pole model with only resonances.

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

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

$$\alpha_i^{(i)} = k_i \quad (2)$$

$$\alpha_j^{(i)} = \alpha_j^{(i-1)} - k_i \cdot \alpha_{(i-j)}^{(i-1)} \quad (3)$$

$$E^{(i)} = (1 - k_i^2) E^{(i-1)} \quad (4)$$

where  $\alpha$  - predictor coefficient

E - energy

Equations (1)-(4) are solved recursively for  $i = 1, 2, \dots, p$

While solving for the predictor coefficients for a predictor of order  $p$ , the solutions for the predictor coefficients of all orders less than  $p$  is also obtained.

6) Cepstrum analysis : Once LPC coefficients have been determined, the cepstrum of the overall LPC is computed. If the overall LPC has an impulse response  $\hat{h}(n)$  and complex cepstrum  $h(n)$  then  $\hat{h}(n)$  can be obtained from the following recursive relation.

$$\hat{h}(n) = \alpha_n + \sum (k/n) h(k) \cdot \alpha_{n-k} \quad , 1 \leq n.$$

Cepstral analysis provides a technique for separating the source and system parameters by computing the spectral envelope which reflects the system characteristics. The cepstral analysis thus provides a procedure for computing the spectral envelope. This can be used to extract the formant frequencies by the peak picking method.

7) Weighted 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.



Now the weighted cepstrum coefficients are the parameters and these coefficients are represented in the form of vectors. Many utterances of the isolated words are made and the vectors are classified into pattern classes and stored as templates. A vector of size 270 is obtained for a 3000 sample pattern.

### 3.2.3 Recognition Phase:

Dynamic time warping is a non - linear time normalisation technique which compensates for the time scale difference between patterns. In an isolated word recognition system, the test utterance of a word is recognized by comparing it with the reference utterance of the same word. As the two utterances are likely to be of different length due to speaking rate variations, they need to be time normalised before comparison. A linear time normalization which uniformly compresses or stretches the test utterance to the length of the reference pattern is not adequate. This is because the effects of speaking rate change across the individual speech events would, in general, be non-linear. To time align similar speech events between the two utterances, non-linear time normalization is necessary.


The speech signal is analyzed frame wise and each frame is represented by a feature vector. The test utterance having I frames can then be represented as a pattern T of I features :  $T = T(1) . T(2) . . .$

$T(I)$ . Similarly the reference utterance is denoted by  $R = R(1) . R(2) . . . R(J)$ . These patterns are the input to the warping process and the task of the DTW algorithm is to find the warping path  $j = w(i)$  that begins at point  $(1,1)$ , ends at point  $(I,J)$  and minimizes the following distance  $D$ , between the test and reference patterns over all possible paths.

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

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

The DTW is a very powerful technique which has contributed to the success of speech recognition in a big way.

 ISOLATED WORD RECOGNITION  
**IMPLEMENTATION**

### **4.1 MINIMUM REQUIREMENTS**

The minimum requirements are an 80386 machine with 4Mb RAM with a multimedia kit for speech capturing.

### **4.2 LANGUAGE USED**

The code has been written in 'C' language which is known for its versatility, simplicity and portability. Visual Basic has been used as the front end tool for the design of menu.

### **4.3 ALGORITHM USED**

#### **4.3.1 Analysis and storage of templates**

- (i) Capturing of words using multimedia kit and the digitization of the analog signal.
- (ii) Locating the end-point using energy and zero-crossing rate.
- (iii) Dividing the samples of the speech signal into analysis frames.
- (iv) Performing windowing using the Hanning window formula in each frame.
- (v) Performing pre-emphasis using the formula mentioned in the previous section.
- (vi) Performing autocorrelation and determining the correlation coefficients for a tenth order polynomial.



- (vii) Performing linear predictive coding and determination of predictor coefficients.
- (viii) Performing cepstrum analysis and determination of cepstrum coefficients from LP coefficients.
- (ix) Assigning weights to the cepstrum coefficients.
- (x) Storage of weighted values in files which are the templates.

#### **4.3.2 Recognition**

- (i) Get test pattern and perform speech analysis and store the vector.
- (ii) Consider I to be the number of frames in the test pattern and J be the number of frames in a particular reference pattern.
- (iii) Start at frame (1,1)
- (iv) Calculate minimum distance between the test and reference pattern given by
$$D = \sum d[T(i).R(w(i))]$$
where  $w(i)$  is the warping path.
- (v) Continue step(iv) upto the frame (I,J)
- (vi) Display the recognized word

#### **4.4 DATA STRUCTURES**

Structure of wave file is given by

```
typedef struct _WaveFmt
{
    WORD wFormatTag ; // Indicates PCM set to 1 //
    WORD nChannels ; // Number of channels 1 - mono //
    WORD nSamplesPerSec ; // Sampling rate //
    WORD nAvgBytesPerSec ; // Average number of bytes per sec //
    WORD nBlockAlign ; // Block Alignment //
} ffmt ;
```

Temporary files have been used to store temporary values obtained after each module. The values obtained are stored in two - dimensional arrays. Static storage allocation is used. The output vectors are stored as templates in files.

#### **4.5 PROGRAM IMPLEMENTATION**

There are two main modules used. They are

- \* Speech analysis and storage of reference patterns
- \* Recognition of test pattern

##### **Speech analysis :**

In the analysis part, different patterns of isolated words are uttered. Ten different patterns of each word have been considered based on

the context, pitch etc. After analysis the resultant weight vector is stored in the file "wcptt". For each pattern the file is renamed to some other file. Hence for the word 'one', ten different files are used to store the templates corresponding to each utterance.

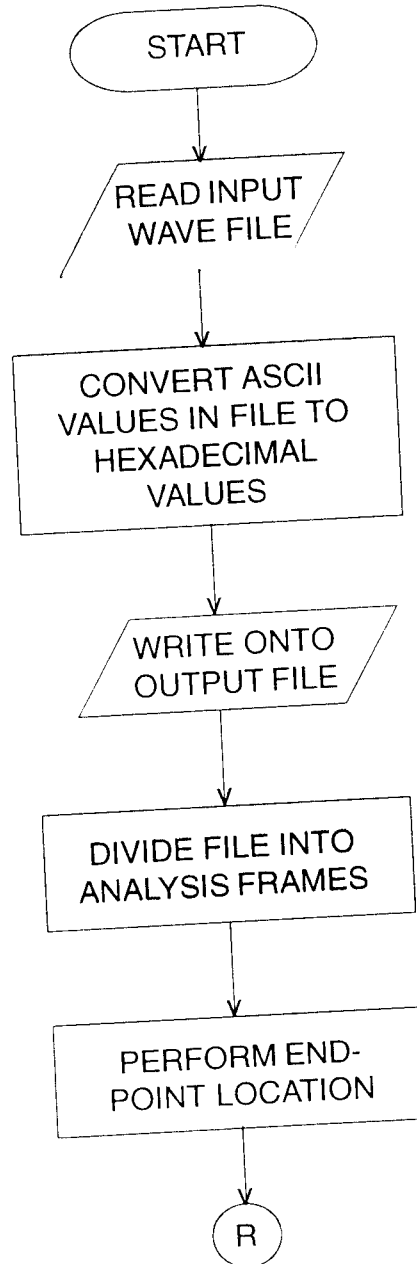
### **Recognition :**

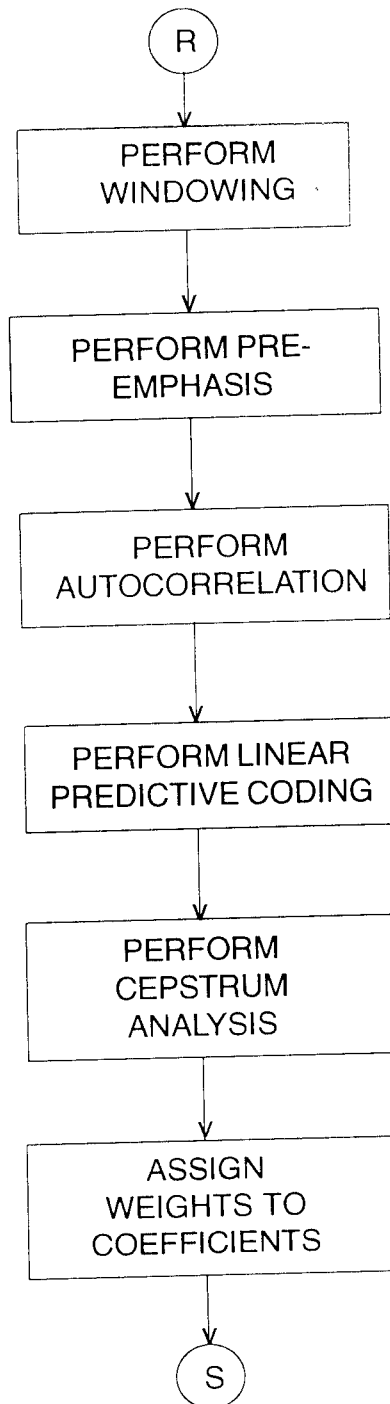
A new file `trn_fil` is created during this stage and it contains the names of all the files which have the templates corresponding to each word uttered. The weight vectors of the word to be recognized are in the file 'wcptt'.

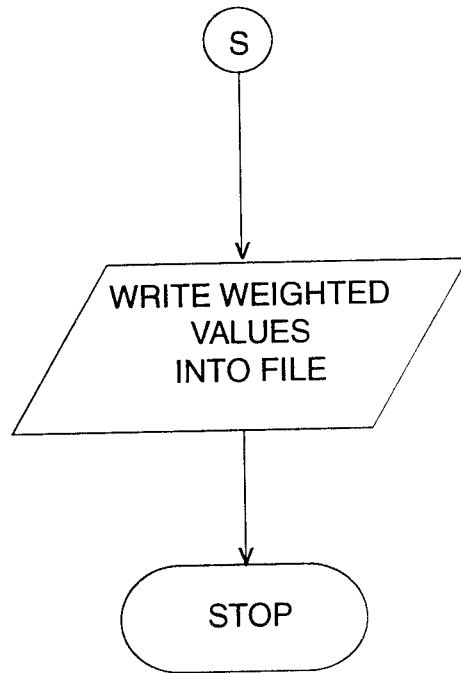


4.6 FLOWCHART

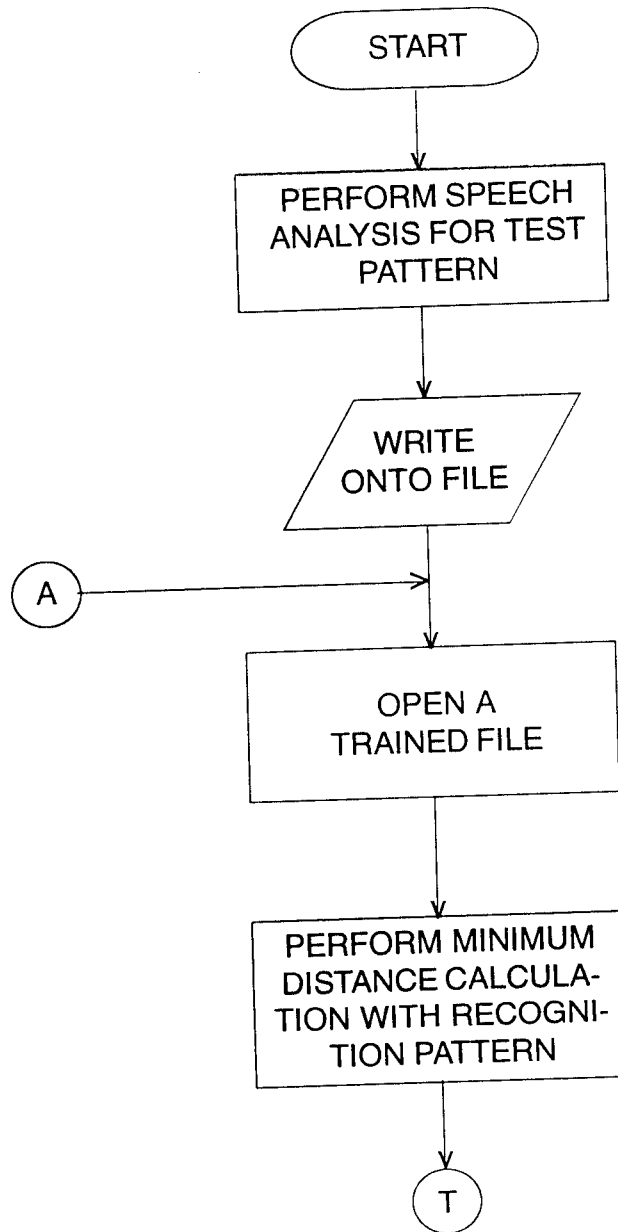
SPEECH ANALYSIS

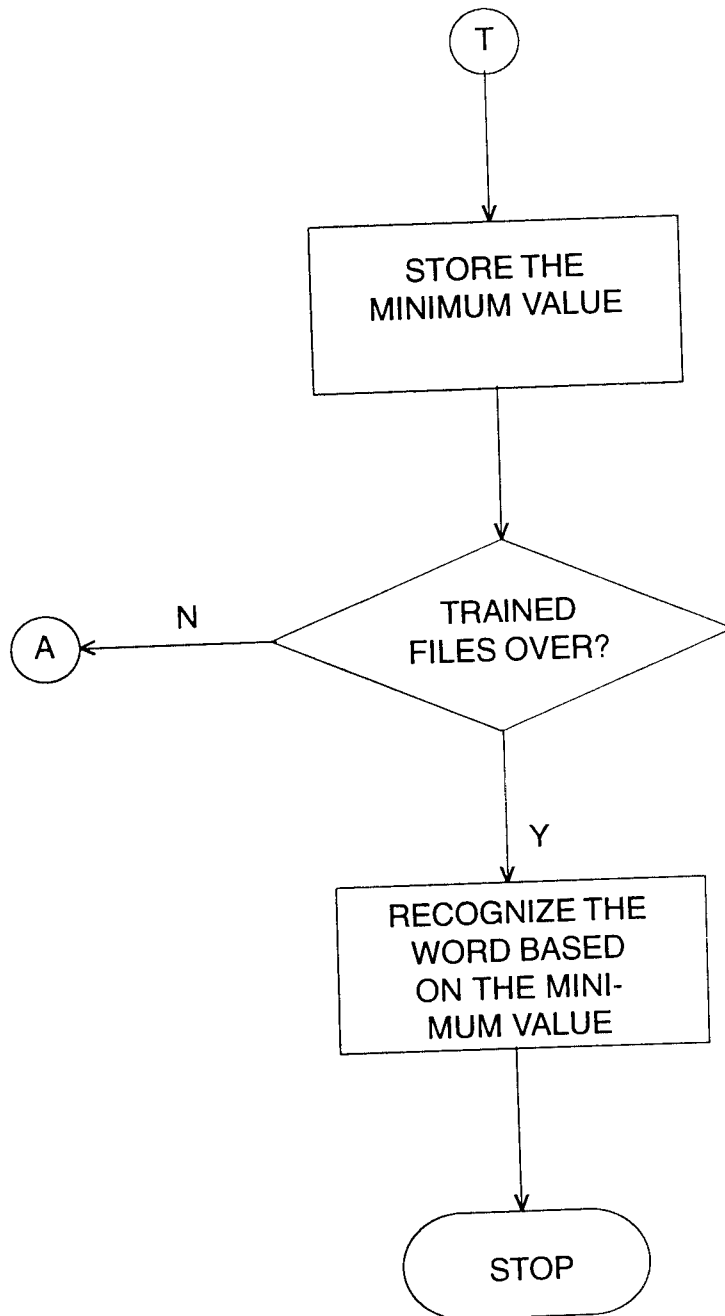






## SPEECH RECOGNITION







## RESULTS AND CONCLUSION

An isolated word recognition scheme has been implemented and its performance evaluated. The criteria for evaluation should include the algorithm efficiency and percentage of correct recognition obtained.

The speech analysis part takes around 40 seconds to be executed on a 386 machine and this time depends on the number of samples of the speech signal taken at a time. The more the number of samples, the more the time taken. The recognition part also takes around 40 seconds. Efficient recognition is obtained if there are a number of reference patterns(atleast 20) for each word uttered.

The algorithm designed has been tested for the word 'two'. Isolated word recognition systems have achieved some commercial success. Speaker dependent, small vocabulary, isolated word recognition systems operate reasonably well for well delimited task environments.



## FUTURE SCOPE

In speech analysis, a 10 pole LP analysis has been carried out when a 2 pole LP analysis is sufficient for numerals. Frequency domain analysis can also be implemented. The recognition part for isolated words has been done using template matching which has been implemented using the dynamic time warping algorithm. The recognition obtained is not very good. Also there is no training actually involved. Training here is just the storage of templates after analysis. On - line training facility may be provided. Hidden Markov Models (HMM) and Artificial Neural Networks (ANN) are two other methods available which involve a training phase and then the recognition phase. An efficient amount of recognition is obtained using the above two procedures.

## APPLICATIONS

Speech recognition finds a number of potential applications. Some of them are voice operated typewriter, voice communication with computers etc. Speech recognition can be used in any place where the "Hands Busy, Eyes Busy" condition exists.

The potential applications of speech recognition are as follows

- \* Domestic appliance system
- \* Office applications
- \* Industrial applications
- \* Transport applications
- \* Medical applications
- \* Application in public service
- \* Military application
- \* Aids for the handicapped

Domestic appliance system : This system controls the operation of domestic appliances through voice. When the speaker uses the name of any particular device, the device is activated immediately. These devices can be controlled as desired by giving commands.

Office applications : In the office, voice - controlled, telephone operating machines are widely used. The incoming calls can be directed to the appropriate extension by using either number or name. It would also





be possible to reprogram the exchange by voice so that the calls are temporarily diverted during absences.

Industrial applications : Speech recognition is used in computer controller machine loads which are programmed by means of a keyboard. It also finds use in inspection and quality control.

Transport applications : Speech recognition is used in flight management systems, training of air-traffic controllers, for booking and making time table enquiries and to have a direct access to the reservation system.

Applications in public service : It may find use in the postal system. It can be used to provide the police immediate access to database by voice over radio links.

Military applications : In warships, speech recognition can be used to, control weapon systems.

Aids for the handicapped : Speech recognition proves to be an excellent aid for the handicapped. The deaf people can be made aware of the movements of the tongue, to give them an idea of what exactly is being uttered.

**REFERENCES:**

1. Whither speech recognition - the next 25 years by David. B. Rose and Jay. G. Wilpon, IEEE communications magazine, November 1993.
2. Speaker - Independent, Speaker - Dependent and Speaker - Adaptive recognition by Xuedong Huang and Kai - Fu Lee, IEEE transactions on speech and audio processing, Vol I, April 1993.
3. Speech technology, IIT Madras.
4. Digital processing of speech signals by L. R. Rabiner and R. W. Schafer.
5. Digital signal processing by Alan. V. Oppenheimer and R. W. Schafer.
6. Speech communication - Human and machine by Douglas O' Shaughnessey.

```

/* Speech analysis and storage of templates */
# include <stdio.h>
# include <stdlib.h>
# include <conio.h>
# include <math.h>
# include <string.h>
# include <dos.h>
# include <io.h>
# include <graphics.h>

# define pi 3.14
# define alpha 0.95

int nfr;      /* Number of frames */
double tfrl; /* Number of samples in a frame */
char re_file[7];

void endpt();
void wow();
void pre(float a[][]);
void corr(float pe[][]);
void linear(float r[][]);
void cepst(float al[][]);
void wtdcp(float h[][]);
void graplot();

void main()
{
    FILE *ifp,*ofp;
    int p=44;
    char train[14],s;
    int count=0,num=0;

    clrscr();

    printf("\n Enter the number of files for training : ");
    scanf("%d",&num);
    while(count < num)
    {
        fflush(stdin);
        printf("\n Enter file name for training : ");
        gets(train);

        while ((ifp = fopen(train,"rb")) == NULL)
        {
            printf("\n\t File cannot be opened...");
            printf("\n\t Re-Enter file name...");
            gets(train);
        }
        ofp = fopen("res","wb");

        fseek(ifp,p,SEEK_SET);
        while ((s=fgetc(ifp))!= EOF)
        {
            fprintf(ofp,"%d\n",s);
        }
        fclose(ifp);
        fclose(ofp);
    }
}

```

```

    endpt();
    wow();
    clrscr();
    graplot();
    count++;
}
}

/* End point location */
/* Function determines the start and end point in a speech signal */

void endpt()
{
    FILE *ofp,*ofpl;
    int temp,temp1,flag=1;

    ofp = fopen("res","rb");
    if (ofp == NULL)
    {
        printf("\n File cannot be opened ");
        exit(0);
    }
    ofpl = fopen("resi","wb");
    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
                    *printf(ofpl,"\n%d",temp);
            }
            while(fscanf(ofp,"%d",&temp)!=EOF);
        }
    }

    fclose(ofp);
    fclose(ofpl);
    return;
}

/* windowing */
/* Function to calculate filter coefficients */

void wow()
{
    FILE *windptr,*ofpl;
    float w[40][120],a[40][120],tt;
    int i,j,nsamp,inp;

    ofpl = fopen("resi","rb");

```

```

if (ofpl == NULL)
{
    printf("\n File cannot be opened ");
    exit(0);
}
windptr = fopen("wind","wb");
rewind(ofpl);

nsamp = 3000;
nfr = 30;
tfrl = (nsamp / nfr);
/*printf("\n No: of samples in each frame = %f",tfrl);*/

for (i=1; i<=nfr; i++)
    for (j=1; j<=tfrl; j++)
    {
        a[i][j] = 0.0;
        w[i][j] = 0.0;
        fscanf(ofpl,"%d",&inp);
        /*fprintf(windptr,"\n\t%d",inp);*/
        tt = ((2*pi*j) / tfrl) / 180 * pi;
        w[i][j] = 0.5 - 0.5*cos(tt);
        a[i][j] = inp * w[i][j];
        fprintf(windptr,"%f\n",a[i][j]);
    }

fclose(windptr);
printf("\n windowing over... ");
pre(a);
return;
}

/* pre-emphasis */
/* Function improves signal to noise ratio */

void pre(float a[40][100])
{
    FILE *preptr;
    float pe[40][100];
    int i,j;

    preptr = fopen("preemp","wb");
    /*fprintf(preptr,"\t Values after pre-emphasis\n",);*/

    for (i=1; i<=nfr; i++)
        for (j=2; j<=tfrl; j++)
        {
            pe[i][j-1] = 0.0;
            pe[i][j-1] = a[i][j] - alpha*a[i][j-1];
            fprintf(preptr,"%f\n",pe[i][j-1]);
        }

    fclose(preptr);
    printf("\n preemphasis over...");
    corr(pe);
    return;
}

/* auto - correlation */
/* Function calculates the correlation coefficients */

```

```

void corr(float pe[40][100])
{
    FILE *autoptr;
    float r[40][10];
    int i,j,ind,sa,x;

    autoptr = fopen("autocor","wb");
    /*fprintf(autoptr,"\t Values after auto-correlation :*/

    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("\n auto-correlation over...");
    linear(r);
    return;
}

/* linear predictive coding */
/* Function calculates the linear predictive coefficients */

void linear(float r[40][10])
{
    FILE *lptr;
    float k[40][10],a1[40][10];
    int i,j;

    lptr = fopen("lpca","wb");
    /*fprintf(lpcptr,"Linear predictive coefficients :*/
    /*egyptr = fopen("energy","wb");
    fprintf(egyptr,"Energy value\n");*/

    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)
            e[i][j-1] = (1-k[i][j-1])*k[i][j-1];
        else
            e[i][j-1] = (1-k[i][j-1])*k[i][j-1]*e[i][j-2];
        fprintf(egyptr,"%f\n",e[i][j-1]);
    }
}*/

for (i=1; i<=nfr; i++)
{
    for (j=2; j<=10; j++)
    {
        if (j == 2)
            a[i][j-1] = r[i][j] / r[i][j-1];
        else
            a[i][j-1] = (r[i][j]*r[i][j-2] - r[i][j-1]*r[i][j-3]) /
                (r[i][j-2]*r[i][j-3] - r[i][j-3]*r[i][j-4]);
        fprintf(lptra,"%f\n",a[i][j-1]);
    }
}

fclose(lptra);
/*fclose(egyptr);*/
printf("\n linear prediction over...\n");
cepst(a);
return;
}

/* cepstrum analysis*/
/* Function calculates the cepstrum coefficients */

void cepst(float a[40][10])
{
    FILE *cepptr;
    float b[100][100],sum;
    int i,j,1;

    cepptr = fopen("cep","wb");
    /*fprintf(cepptr,"Cepstrum coefficients \n");*/

    for (i=1; i<=nfr; i++)
    {
        b[i][1] = a[i][1];
        fprintf(cepptr,"%f\n",b[i][1]);
        for (j=2; j<=9; j++)
        {
            for (l=1; l<= j-1; l++)
                sum = sum + (1/j) * b [i][l] * a[i][j-l];
            b[i][j] = a[i][j] + sum;
            fprintf(cepptr,"%f\n",b[i][j]);
        }
    }
}

```

```

fclose(cepptr);
printf(" cepstrum analysis over...");
wtddcp(h);
return;
}

/* Weighted cepstrum analysis */
/* Function assigns weights to the cepstrum coefficients X.

void wtddcp(float h[40][10])
{
    FILE *tf,*wtdptr;
    union REGS inregs,outregs;

    float htd[40][10];
    char old[] = "wcptt";
    int i,j;

    wtdptr = fopen("wcptt","wb");
    /*fprintf(wtdptr,"\n weighted cepstrum coefficients ");*/

    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);
    printf("\n weights assigned...");
    printf("\n\n Enter filename for renaming : ");
    scanf("%s",re_file);

    /* Renaming file */
    inregs.x.dx = (int) &old[0];
    inregs.x.di = (int) &re_file;
    inregs.h.ah = 0x54;
    intdos(&inregs,&outregs);
    if(outregs.x.cflag == 0)
        printf("\n wcptt renamed to %s\n", re_file);
    else
        printf("\n File not renamed...\n");

    tf = fopen("trn_fil","a+");
    fprintf(tf,"%s\n",re_file);
    fclose(tf);
    return;
}

void graphplot()
{
    FILE *windptr,*preptr,*autoptr,*cepptr,*wtdptr;
    float wintemp,pretemp,autotemp,ceptemp,wtdtemp;
    int xx=1,prevx=1,prevy=100;
    float tmpnum,tmpnum1,tmpnum2,tmpnum3,tmpnum4;
    int d=DETECT,m;

    initgraph(&d,&m,"c:\tc\bgi");

```



```

cleardevice();

windptr = fopen("wind","rb");
rewind(windptr);
while ((fscanf(windptr,"%f",&tmpnum) != EOF))
{
    wintemp = tmpnum * 100;
    putpixel(xx,wintemp+100,2);
    line(prevx,prevy,xx,wintemp+100);
    prevx = xx;
    prevy = wintemp+100;
    xx++;
}
fclose(windptr);
getch();
printf("\n Windowing over...");

cleardevice();
xx = 1;
prevx = 1;
prevy = 100;
preptr = fopen("preemp","rb");
rewind(preptr);
while ((fscanf(preptr,"%f",&tmpnum1) != EOF))
{
    pretemp = tmpnum1 * 100;
    putpixel(xx,pretemp+100,2);
    line(prevx,prevy,xx,pretemp+100);
    prevx = xx;
    prevy = pretemp+100;
    xx++;
}
fclose(preptr);
getch();
printf("\n Pre-emphasis over...");

cleardevice();
xx = 1;
prevx = 1;
prevy = 100;
autoptr = fopen("autocor","rb");
rewind(autoptr);
while ((fscanf(autoptr,"%f",&tmpnum2) != EOF))
{
    autotemp = tmpnum2 * 1000;
    putpixel(xx,autotemp+100,2);
    line(prevx,prevy,xx,autotemp+100);
    prevx = xx;
    prevy = autotemp+100;
    xx++;
}
fclose(autoptr);
getch();
printf("\n Autocorrelation over...");

cleardevice();
xx = 1;
prevx = 1;

```

```

prevy = 100;
cepptr = fopen("lpca","rb");
rewind(cepptr);
while((fscanf(cepptr,"%f",&tmpnum3) != EOF))
{
    ceptemp = tmpnum3 * 10;
    putpixel(xx,ceptemp+100,2);
    line(prevx,prevy,xx,ceptemp+100);
    prevx = xx;
    prevy = ceptemp+100;
    xx++;
}
fclose(cepptr);
printf("\n Linear prediction over...");
getch();

cleardevice();
xx = 1;
prevx = 1;
prevy = 100;
wtdptr = fopen(re_file,"rb");
rewind(wtdptr);
while((fscanf(wtdptr,"%f",&tmpnum4) != EOF))
{
    wtdtemp = tmpnum4 * 10;
    putpixel(xx,wtdtemp+100,2);
    line(prevx,prevy,xx,wtdtemp+100);
    prevx = xx;
    prevy = wtdtemp+100;
    xx++;
}
fclose(cepptr);
getch();
printf("\n vector representation over...");
}

```

```

    /* Recognition of isolated words */

#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <math.h>
#include <string.h>
#include <graphics.h>

#define pi 3.14
#define alpha 0.95

int nfr;          /* Number of frames */
double tfrl;     /* Number of samples in a frame */

void endpt();
void wow();
void pre(float a[][]);
void corr(float pe[][]);
void linear(float r[][]);
void cepst(float al[][]);
void wtdcp(float h[][]);
void graplot();
void recog();
void say1();
void say2();

void main()
{
    FILE *ifp,*ofp;
    int p=44;
    char in_file[14],s;

    clrscr();
    printf("\n Enter 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();
    graplot();
    recog();
}

```

```

/* end point location */
/* Determines start and end of a signal */

void endpt()
{
    FILE *ofp,*ofpl;
    int temp,temp1,flag=1;

    ofp = fopen("res","r");
    if (ofp == NULL)
    {
        printf("\n File cannot be opened ");
        exit(0);
    }
    putc('.',stdout);
    ofpl = 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(ofpl,"\n%d",temp);
            }
            while(fscanf(ofp,"%d",&temp)!=EOF);
        }
    }

    fclose(ofp);
    fclose(ofpl);
}

/* windowing */
/* Calculates filter coefficients */

void wow()
{
    FILE *ofpl,*windptr;
    float w[40][100],a[40][100],tt;
    int i,j,nsamp,inp;

    ofpl = fopen("res1","r");
    if (ofpl == NULL)
    {
        printf("\n File cannot be opened ");
        exit(0);
    }
    windptr = fopen("wind","w");
    rewind(ofpl);

    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 - correlation */
/* Calculates correlation coefficients */

void corr(float pe[40][100])
{
  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(autopttr,"%f\n\t\t",r[i][ind]);
    x++;
    sa--;
    ind++;
}
while (sa >= 90);
}

fclose(autopttr);
putc('.',stdout);
linear(r);
}

/* linear predictive coding */
/* Calculates linear predictive coefficients */

void linear(float r[40][10])
{
    FILE *lpcptr;
    float k[40][10],a1[40][10];
    int i,j;

    lpcptr = fopen("lpc","w");
    /*egypttr = fopen("energy","w");
    fprintf(egypttr,"Energy value\n");*/

    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)
                e[i][j-1] = (1-k[i][j-1]*k[i][j-1]);
            else
                e[i][j-1] = (1-k[i][j-1]*k[i][j-1])*e[i][j-2];
            fprintf(egypttr,"%f\n",e[i][j-1]);
        }
    }*/

    for (i=1; i<=nfr; i++)
    {

```

```

    for (j=2; j<=10; j++)
    {
        if (j == 2)
            al[i][j-1] = r[i][j] / r[i][j-1];
        else
            al[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",al[i][j-1]);
    }
}

fclose(lpcptr);
/*fclose(egypt);*/
putc('.',stdout);
cepst(al);
}

/* cepstrum analysis*/
/* Calculates cepstrum coefficients */

void cepst(float al[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] = al[i][1];
        for (j=2; j<=9; j++)
        {
            for (l=1; l<= j-1; l++)
                sum = sum + (1/j) * h [i][l] * al[i][j-l];
            h[i][j] = al[i][j] + sum;
            fprintf(cepptr,"%f\n\t\t",h[i][j]);
        }
    }

    fclose(cepptr);
    putc('.',stdout);
    wtdcp(h);
}

/* Weighted cepstrum analysis */
/* Assigns weights for cepstrum coefficients */

void wtdcp(float h[40][10])
{
    FILE *wtdptr;
    float htd[40][10];
    int i,j;

    wtdptr = fopen("wcpt","w");

    for (i=1; i<=nfr; i++)
    {
        for (j=1; j<=9; j++)
        {
            htd[i][j] = j * h[i][j];

```

```

        fprintf(wtdptr,"%f\n\t\t",htd[i][j]);
    }
}

fclose(wtdptr);
putc('.',stdout);
}

/* recognition phase */
void recog()
{
    FILE *tf,*inppat,*r1;
    char req_fil[8];
    float tty1=0.0,tty2=0.0,tty=0.0,min;
    int i2,count,i;
    double sum[30];

    tf = fopen("trn_fil","r");
    if (tf == NULL)
    {
        printf("\n File cannot be opened ");
        exit(0);
    }
    rewind(tf);

    inppat = fopen("wcpt","r");
    if (inppat == NULL)
    {
        printf("\n File cannot be opened ");
        exit(0);
    }

    fflush(stdin);
    for (i=1; i<=6; i++)
        sum[i] = 0;

    i = 1;
    while(i <= 6)
    {
        fgets(req_fil,9,tf);
        if ((r1 = fopen(req_fil,"r")) == NULL)
        {
            printf("\n File cannot be opened ");
            exit(0);
        }
        rewind(r1);
        rewind(inppat);

        i2=1;
        while (i2 <= 270)
        {
            fscanf(r1,"%f",&tty1);
            fscanf(inppat,"%f",&tty);
            tty2 = fabs(tty1-tty);
            sum[i]+=tty2;
            i2++;
        }
        fclose(r1);
        fgets(req_fil,9,tf);
        i++;
    }
}

```



```

fclose(tf);
fclose(inppat);
/*for (i=1; i<=7; i++)
    printf("\n sum%d = %f",i,sum[i]);*/

/* sorting for recognition of word file */
min = sum[1];
for (i=1; i<=6; i++)
{
    if (sum[i] < min)
    {
        min = sum[i];
        count = i;
    }
}

/* Determining the pattern class */
if ((count >= 1) && (count <= 3))
    say1();
if ((count >= 4) && (count <= 6))
    say2();
}

void graplot()
{
    FILE *windptr,*preptr,*autoptr,*cepptr,*wdptr;
    float wintemp,pretemp,autotemp,ceptemp,wdtemp;
    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);
    setttextstyle(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(450,400,"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," r(t)");

```

```

while ((fscanf(autopttr, "%f", &tmpnum) != EOF))
{
    autotemp = tmpnum * 500;
    putpixel(xx, autotemp+100, GREEN);
    setcolor(GREEN);
    line(prevx, prevy, xx, autotemp+100);
    prevx = xx;
    prevy = autotemp+100;
    xx++;
}
fclose(autopttr);

setcolor(MAGENTA);
setttextstyle(TRIPLEX_FONT, HORIZ_DIR, 3);
outtextxy(430, 400, "AUTOCORRELATION");
setcolor(YELLOW);
setttextstyle(TRIPLEX_FONT, HORIZ_DIR, 1);
outtextxy(500, getmaxy()-30, "Press Any Key");
getch();

cleardevice();
xx = 1;
prevx = 1;
prevy = 100;
ceppttr = fopen("lpc", "rb");
rewind(ceppttr);

setcolor(RED);
line(1, 100, getmaxx()-5, 100);

setcolor(CYAN);
line(30, getmaxy()-50, 80, getmaxy()-50);
setttextstyle(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(ceppttr, "%f", &tmpnum) != EOF))
{
    ceptemp = tmpnum * 5;
    putpixel(xx, ceptemp+100, GREEN);
    setcolor(GREEN);
    line(prevx, prevy, xx, ceptemp+100);
    prevx = xx;
    prevy = ceptemp+100;
    xx++;
}
fclose(ceppttr);

setcolor(MAGENTA);
setttextstyle(TRIPLEX_FONT, HORIZ_DIR, 3);
outtextxy(330, 400, "LINEAR PREDICTIVE CODING");
setcolor(YELLOW);
setttextstyle(TRIPLEX_FONT, HORIZ_DIR, 1);
outtextxy(500, getmaxy()-30, "Press Any Key");
getch();

cleardevice();
xx = 1;
prevx = 1;

```

```

prevy = 100;
wtdptr = fopen("wcpt","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," wt(t)");

while((fscanf(wtdptr,"%f",&tmpnum) != EOF))
{
    wtdtemp = tmpnum * 0.5;
    putpixel(xx,wtdtemp+100,GREEN);
    setcolor(GREEN);
    line(prevx,prevy,xx,wtdtemp+100);
    prevx = xx;
    prevy = wtdtemp+100;
    xx++;
}
fclose(cepptr);

setcolor(MAGENTA);
settextstyle(TRIPLEX_FONT,HORIZ_DIR,3);
outtextxy(400,400,"WEIGHTED VECTORS");
setcolor(YELLOW);
settextstyle(TRIPLEX_FONT,HORIZ_DIR,1);
outtextxy(500,getmaxy()-30,"Press Any Key");
getch();
}

void say1()
{
    int d=DETECT,m;

    initgraph(&d,&m," ");
    setcolor(CYAN);
    settextstyle(TRIPLEX_FONT,HORIZ_DIR,6);
    outtextxy(getmaxx()/2-5,getmaxy()/2,"ONE");
    delay(5000);
    closegraph();
}

void say2()
{
    int d=DETECT,m;

    initgraph(&d,&m," ");
    setcolor(CYAN);
    settextstyle(TRIPLEX_FONT,HORIZ_DIR,6);
    outtextxy(getmaxx()/2-5,getmaxy()/2,"TWO");
    delay(5000);
    closegraph();
}

```

*Windowing*

0.000038  
0.000149  
0.000340  
0.000605  
0.000945  
0.001361  
-0.001794  
-0.002285  
0.002989  
-0.003630  
-0.004611  
-0.005487  
0.006287  
0.007351  
-0.008573  
0.009678  
0.010752  
0.012248  
0.013322  
0.014881  
0.016406  
0.018151  
-0.019045  
-0.020737  
-0.023251  
-0.024540  
-0.026464  
-0.029636  
-0.032043  
-0.034021  
0.035750  
0.038708  
-0.041165  
0.042657  
0.045937  
-0.047433  
0.051337  
-0.053716  
-0.057492  
0.059998  
0.063034  
0.065088  
0.069333  
0.071433  
0.072287  
0.079343  
0.082829  
0.082935  
0.088586  
0.091488  
0.097524  
0.099763  
0.102793  
0.109331  
0.110695  
0.115696  
0.119864  
0.124105  
0.129464  
0.132808

0.140618  
0.141806  
-0.146416  
-0.152327  
-0.160924  
-0.160686  
-0.168282  
-0.176114  
-0.181330  
0.185152  
0.188965  
0.192768  
0.196559  
0.205262  
0.210844  
0.211305  
0.218677  
0.226214  
0.235791  
0.241794  
0.245906  
0.254027  
0.258191  
0.262330  
0.266442  
0.279396  
0.283657  
0.285567  
0.292089  
0.303541  
0.312800  
0.314632  
0.321503  
0.325800  
0.335467  
0.345323  
-0.358189  
-0.362726  
-0.367222  
-0.377668  
-0.000038  
-0.000150  
-0.000340  
-0.000595  
-0.000945  
0.001361  
-0.001852  
-0.002439  
0.003062  
0.003780  
0.004538  
0.005444  
-0.006440  
-0.007410  
-0.008438  
-0.009755  
-0.010838  
-0.012054  
-0.013755  
-0.014761  
-0.016142

-0.018297  
-0.019680  
-0.020910  
-0.022689  
-0.024743  
-0.026901  
-0.028460  
-0.030529  
-0.032401  
-0.034309  
-0.037172  
-0.040511  
-0.042657  
-0.044467  
-0.047044  
-0.050926  
-0.053716  
-0.057036  
-0.059998  
-0.063034  
-0.066675  
0.069888  
0.072595  
0.075324  
0.077439  
0.080841  
0.084317  
0.087866  
0.090738  
0.093623  
0.098140  
0.101950  
0.104958  
0.107973  
0.112874  
0.115966  
0.120069  
0.122155  
0.128489  
0.133922  
0.137195  
0.141654  
0.147413  
0.153321  
0.159379  
0.164243  
0.171954  
0.177046  
0.183682  
-0.191988  
-0.195877  
-0.201353  
-0.206904  
-0.209158  
-0.211305  
-0.215121  
-0.224390  
-0.228306  
-0.236037  
-0.240004  
-0.245963



-0.254060  
-0.262330  
-0.272941  
-0.279396  
-0.288195  
0.292532  
0.294463  
0.301112  
0.307835  
0.312095  
0.318911  
0.323151  
0.330057  
0.337035  
0.346907  
0.359848  
0.361346  
0.371673  
0.000037  
0.000150  
0.000340  
0.000600  
0.000938  
0.001350  
0.001838  
-0.002439  
-0.003062  
-0.003750  
-0.004538  
-0.005401  
-0.006237  
-0.007174  
-0.008236  
-0.009294  
-0.010578  
-0.011762  
-0.013105  
-0.014521  
-0.016142  
-0.017861  
-0.019521  
-0.021429  
-0.023439  
-0.025554  
-0.027776  
0.029636  
0.031539  
0.033751  
0.035750  
0.037479  
0.039531  
0.041616  
0.043732  
0.046267  
0.047640  
0.049817  
0.052017  
0.054718  
0.057487  
0.059796  
0.062677

*Pre-emphasis*

0.000113  
0.000199  
0.000282  
0.000370  
0.000463  
-0.003087  
-0.000581  
0.005160  
-0.006470  
-0.001162  
-0.001107  
0.011500  
0.001378  
-0.015557  
0.017822  
0.001558  
0.002034  
0.001686  
0.002225  
0.002269  
0.002565  
-0.036289  
-0.002644  
-0.003551  
-0.002451  
-0.003151  
-0.004496  
-0.003889  
-0.003580  
0.068070  
0.004745  
-0.077937  
0.081763  
0.005414  
-0.091074  
0.096398  
-0.102485  
-0.006462  
0.114615  
0.006037  
0.005205  
0.007499  
0.005567  
0.004425  
0.010671  
0.007454  
0.004247  
0.009798  
0.007331  
0.010610  
0.007115  
0.008018  
0.011678  
0.006830  
0.010536  
0.009952  
0.010234  
0.011564  
0.009818  
0.014451  
0.008219

-0.281131  
-0.013232  
-0.016213  
-0.007808  
-0.015631  
-0.016246  
-0.014021  
0.357415  
0.013071  
0.013251  
0.013429  
0.018531  
0.015845  
0.011003  
0.017937  
0.018471  
0.020887  
0.017793  
0.016201  
0.020417  
0.016865  
0.017048  
0.017229  
0.026276  
0.018231  
0.016093  
0.020800  
0.026056  
0.024437  
0.017472  
0.022603  
0.020371  
0.025958  
0.026629  
-0.686246  
-0.022447  
-0.022632  
-0.028807  
-0.000114  
-0.000198  
-0.000272  
-0.000380  
0.002259  
-0.003145  
-0.000679  
0.005379  
0.000871  
0.000947  
0.001133  
-0.011611  
-0.001292  
-0.001399  
-0.001738  
-0.001572  
-0.001757  
-0.002304  
-0.001694  
-0.002119  
-0.002962  
-0.002298  
-0.002214

-0.002824  
-0.003189  
-0.003396  
-0.002904  
-0.003492  
-0.003398  
-0.003528  
-0.004579  
-0.005198  
-0.004171  
-0.003944  
-0.004800  
-0.006234  
-0.005336  
-0.006006  
-0.005814  
-0.006037  
-0.006793  
0.133229  
0.006201  
0.006359  
0.005881  
0.007275  
0.007518  
0.007765  
0.007265  
0.007422  
0.009199  
0.008717  
0.008105  
0.008263  
0.010300  
0.008735  
0.009902  
0.008090  
0.012442  
0.011858  
0.009968  
0.011319  
0.012842  
0.013278  
0.013724  
0.012833  
0.015923  
0.013690  
0.015488  
-0.366486  
-0.013488  
-0.015270  
-0.015619  
-0.012598  
-0.012605  
-0.014381  
-0.020025  
-0.015135  
-0.019147  
-0.015769  
-0.017959  
-0.020395  
-0.020973  
-0.023727

-0.020102  
-0.022769  
0.566317  
0.016558  
0.021372  
0.021779  
0.019651  
0.022420  
0.020186  
0.023063  
0.023482  
0.026724  
0.030286  
0.019491  
0.028394  
0.000115  
0.000198  
0.000277  
0.000368  
0.000459  
0.000555  
-0.004185  
-0.000745  
-0.000841  
-0.000975  
-0.001089  
-0.001106  
-0.001249  
-0.001420  
-0.001470  
-0.001749  
-0.001713  
-0.001931  
-0.002071  
-0.002347  
-0.002526  
-0.002554  
-0.002883  
-0.003082  
-0.003287  
-0.003500  
0.056024  
0.003384  
0.003789  
0.003687  
0.003517  
0.003926  
0.004062  
0.004197  
0.004721  
0.003687  
0.004558  
0.004691  
0.005302  
0.005506  
0.005183  
0.005871  
0.006663  
0.006961  
0.008536  
0.008361

# *Autocorrelation*

-0.013950  
-0.007360  
0.044953  
-0.029429  
-0.002225  
0.005336  
-0.126651  
-0.008757  
0.006868  
-0.002906  
0.009243  
0.013466  
0.008967  
0.007646  
0.007828  
0.007039  
0.010729  
0.006251  
0.010974  
0.006232  
-0.003526  
-0.005841  
-0.001547  
-0.002103  
-0.007776  
-0.001768  
0.002972  
0.003174  
0.007533  
0.008331  
-0.002763  
-0.007747  
-0.002216  
0.001046  
-0.002733  
0.002549  
0.000553  
0.000984  
0.009969  
0.007295  
-0.010741  
-0.012287  
-0.000890  
-0.005112  
-0.003092  
-0.003263  
-0.007954  
0.006344  
0.017377  
0.021780  
-0.004928  
-0.003530  
-0.006940  
-0.003102  
0.000184  
0.003350  
0.006414  
0.003044  
0.003740  
0.003464  
-0.020100  
-0.018332



-0.018380  
-0.012399  
-0.006287  
-0.000398  
0.006422  
-0.009410  
0.002604  
0.002666  
-0.012304  
-0.002503  
-0.006577  
0.001419  
0.005015  
0.004122  
0.005195  
0.002168  
0.005302  
0.005808  
-0.017486  
-0.014046  
-0.018679  
-0.008616  
-0.013107  
-0.006324  
-0.005522  
-0.005740  
0.000669  
-0.000588  
-0.008399  
-0.005304  
-0.007709  
-0.004262  
-0.021962  
-0.018521  
-0.013339  
-0.013287  
-0.003914  
-0.000465  
0.004264  
0.003962  
0.003551  
0.003898  
0.004121  
0.002864  
0.004235  
0.003998  
0.003278  
0.004367  
-0.002525  
-0.001287  
-0.002695  
-0.001916  
0.000322  
0.001320  
0.005861  
0.007533  
0.010627  
0.014313  
0.001219  
0.000439  
0.001368  
0.001378

12.145488  
-3.508173  
0.203465  
1.028859  
3.796390  
-3.394325  
4.171316  
6.662565  
12.640018  
8.200175  
5.967045  
0.803271  
2.384579  
4.509150  
2.484132  
6.012354  
1.474087  
4.277343  
119.502983  
0.811277  
0.631480  
1.726866  
3.529497  
14.649733  
4.345350  
-2.156994  
2.890083  
-722.226013  
-0.510277  
0.929194  
-0.448100  
2.753269  
0.870165  
16.299341  
6.322186  
4.663747  
-8.624229  
11.534057  
0.509535  
2.181996  
2.566990  
-5.050730  
-3.691007  
-0.523284  
5.239170  
-1.980947  
0.820443  
0.360187  
2.279888  
2.262592  
-315.663208  
2.769323  
-9.439011  
4.633284  
36.473736  
-14.941942  
0.914191  
-4.199041  
1.086622  
-54.125961  
2.773955

0.003058  
0.003791  
0.007280  
0.007236  
0.007592  
0.009175  
-0.014160  
-0.012945  
-0.006951  
-0.007069  
-0.010418  
-0.010758  
-0.001832  
-0.004932  
-0.002644  
-0.000105  
-0.007005  
-0.019876  
-0.022946  
-0.087782  
-0.476487  
-0.056516  
0.013816  
0.016747  
0.017390  
0.076212  
0.003577  
0.002766  
-0.258013  
-0.003983  
-0.007940  
-0.201238  
0.010686  
0.010866  
0.247823  
0.009886  
-0.000435  
-0.000907  
0.000302  
-0.077468  
0.006216  
0.007485  
-0.102664  
0.012964  
0.014890  
0.014497  
0.003206  
-0.011552  
-0.032984  
-0.008384  
-0.005501  
-0.005156  
-0.042374  
-0.003551  
-0.004345  
0.027256  
0.004114  
0.005603  
0.004941  
0.004609  
0.003562  
0.004794

0.004785  
0.007251  
0.007203  
0.006866  
0.002757  
-0.000392  
0.000893  
-0.000631  
0.002818  
0.005075  
0.006774  
0.006015  
0.009628  
-0.007045  
0.008545  
0.009875  
0.006800  
0.006659  
-0.011022  
-0.011295  
-0.009602  
-0.008405  
-0.009203  
-0.008134  
0.011724  
0.010646  
0.010422  
0.010542  
0.009525  
0.009678  
0.008927  
0.008458  
0.008196  
0.007466  
0.012734  
0.013296  
0.012213  
0.012277  
0.012004  
0.010949  
0.011236  
0.011250  
0.010902  
0.010934  
0.014686  
0.013312  
0.010014  
0.011427  
0.010886  
0.010426  
0.009257  
0.006359  
0.008872  
0.008475  
-0.006618  
-0.006770  
-0.005754  
-0.006832  
-0.007012  
-0.008978  
-0.008825  
-0.007317

*LP analysis*

0.527624  
-4.851464  
0.917405  
-0.836627  
-0.188100  
-10.767628  
1.004696  
-0.059292  
-0.735923  
1.456878  
1.026581  
0.223432  
0.534467  
2.650654  
2.937110  
1.084547  
1.034565  
1.001484  
1.656595  
1.321810  
0.311775  
-3.746032  
1.012573  
-0.457527  
2.530858  
-9.892773  
0.649275  
2.804239  
1.028845  
-0.236180  
1.300340  
0.753268  
-8.257204  
0.355689  
-6.857834  
0.936910  
1.143872  
3.973377  
0.412987  
0.922634  
0.429482  
-12.882918  
1.595798  
-7.756952  
0.625820  
0.716172  
1.837885  
1.042322  
-0.282763  
-1.087420  
0.897361  
1.034269  
0.461936  
0.729312  
0.912031  
0.491145  
62.927853  
-0.207516  
-0.302849  
-1.029625

0.912790  
1.518186  
-0.389797  
0.203465  
0.514430  
1.265463  
-0.848581  
0.834263  
1.110427  
1.805717  
1.025022  
0.663005  
0.803271  
1.192289  
1.503050  
0.621033  
1.202471  
0.245681  
0.611049  
14.937873  
0.090142  
0.631480  
0.863433  
1.176499  
3.662433  
0.869070  
-0.359499  
0.412869  
-90.278252  
-0.056697  
0.929194  
-0.224050  
0.917756  
0.217541  
3.259868  
1.053698  
0.666250  
-1.078029  
1.281562  
0.509535  
1.090998  
0.855663  
-1.262683  
-0.738201  
-0.087214  
0.748453  
-0.247618  
0.091160  
0.360187  
1.139944  
0.754197  
-78.915802  
0.553865  
-1.573168  
0.661898  
4.559217  
-1.660216  
0.914191  
-2.099521  
0.362207  
-13.531490

0.554791  
13.451713  
0.442277  
0.929005  
-0.373528  
2.837471  
0.677249  
-9.265680  
-0.449585  
1.012514  
-0.043679  
-0.378727  
0.448889  
-44.412971  
0.773159  
-180.811859  
1.000280  
0.030543  
-15.648933  
1.003658  
-0.056974  
-654.069458  
1.000174  
2.081893  
1.507708  
95.973526  
0.999703  
-0.103732  
39.931728  
0.996088  
-0.163585  
0.629338  
-3.602835  
1.942140  
1.038340  
0.109235  
0.323848  
56.106480  
1.004678  
0.096200  
18.437063  
1.362038  
0.764830  
0.201700  
-1.145827  
1.099988  
0.576819  
138.394775  
0.610181  
-3.015561  
-0.142084  
0.309845  
0.854445  
5.309163  
1.477051  
0.374206  
0.763138  
2.990187  
2.389796  
1.155678  
1.608620



0.380748  
-63.138885  
2.550141  
3.568948  
0.077427  
0.822452  
1.161871  
0.908019  
0.367176  
0.766014  
4.714019  
0.553491  
2.942581  
0.154858  
0.199685  
-0.923311  
1.044117  
1.453870  
0.509459  
2.633645  
-1.463061  
0.619151  
0.482488  
12.957636  
0.538580  
0.906423  
-0.783273  
0.673841  
0.711721  
0.051874  
-0.807600  
-0.842421  
0.921417  
0.648467  
1.022903  
3.820142  
1.033132  
0.466748  
-4.897727  
0.595592  
-4.478067  
-0.045435  
1.936083  
0.112773  
0.126344  
0.702191  
0.340493  
2.611622  
0.023076  
-176.234573  
0.999933  
0.030511  
0.867940  
-0.353928  
0.769084  
0.770142  
0.000881  
-2.104566  
0.842364  
0.763050

# *Weighted Vectors*

0.527624  
-9.702929  
2.752216  
-3.346510  
-0.940500  
-64.605766  
7.032873  
-0.474332  
-6.623304  
1.456878  
2.053161  
0.670295  
2.137870  
13.253271  
17.622660  
7.591832  
8.276519  
9.013354  
1.656595  
2.643620  
0.935324  
-14.984127  
5.062866  
-2.745162  
17.716005  
-79.142181  
5.843473  
2.804239  
2.057690  
-0.708540  
5.201361  
3.766339  
-49.543224  
2.489824  
-54.862671  
8.432193  
1.143872  
7.946755  
1.238960  
3.690535  
2.147411  
-77.297508  
11.170588  
-62.055614  
5.632378  
0.716172  
3.675771  
3.126965  
-1.131052  
-5.437101  
5.384168  
7.239886  
3.695486  
6.563806  
0.912031  
0.982290  
188.783554  
-0.830062  
-1.514244  
-6.177752  
6.389530

80.710274  
3.095937  
7.432037  
-3.361753  
2.837471  
1.354498  
-27.797041  
-1.798339  
5.062568  
-0.262072  
-2.651090  
3.591113  
-399.716736  
0.773159  
-361.623718  
3.000841  
0.122173  
-78.244667  
6.021946  
-0.398818  
-5232.555664  
9.001562  
2.081893  
3.015416  
287.920593  
3.998811  
-0.518661  
239.590363  
6.972619  
-1.308679  
5.664039  
-3.602835  
3.884279  
3.115020  
0.436938  
1.619239  
336.638885  
7.032745  
0.769602  
165.933563  
1.362038  
1.529660  
0.605099  
-4.583307  
5.499938  
3.460914  
968.763428  
4.881448  
-27.140049  
-0.142084  
0.619690  
2.563336  
21.236650  
7.385256  
2.245234  
5.341963  
23.921494  
21.508160  
1.155678  
3.217240

1.142245  
-252.555542  
12.750707  
21.413689  
0.541992  
6.579615  
10.456839  
0.908019  
0.734351  
2.298043  
18.856077  
2.767453  
17.655483  
1.084009  
1.597481  
-8.309797  
1.044117  
2.907740  
1.528376  
10.534581  
-7.315304  
3.714903  
3.377416  
103.661087  
4.847224  
0.906423  
-1.566545  
2.021523  
2.846885  
0.259369  
-4.845602  
-5.896946  
7.371340  
5.836207  
1.022903  
7.640284  
3.099395  
1.866991  
-24.488632  
3.573555  
-31.346468  
-0.363480  
17.424751  
0.112773  
0.252689  
2.106574  
1.361970  
13.058108  
0.138453  
-1233.641968  
7.999462  
0.274595  
0.867940  
-0.707856  
2.307252  
3.080569  
0.004405  
-12.627396  
5.896550  
6.104403