Enhancement of Speech Communication Between Human and Machine

M.Sc. Thesis

In

Electrical and Electronics Engineering

University of Gaziantep

Supervisor

Prof. Dr. Arif NACAROĞLU

By

Hasan Feyzi ÖZUSTAOĞLU

August 2006

T. C.

GAZİANTEP UNIVERSITY
GRADUATE SCHOOL OF
NATURAL & APPLIED SCIENCES
ELECTRICAL AND ELECTRONICS ENGINEERING

Name of the thesis: Enhancement of Speech Communication between Human and Machine
Name of the student: Hasan Feyzi Özustaoğlu
Exam date: 18 / 07 /2006

Approval of the Graduate School of Natural and Applied Sciences

Prof. Dr. Sadettin ÖZYAZICI
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science/Doctor of Philosophy.

Assoc. Prof. Dr. Gülay TOHUMOĞLU
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science/Doctor of Philosophy.

(Prof. Dr. Arif NACAROĞLU)
Supervisor

Examining Committee Members	signature
Prof. Dr. Arif Nacaroğlu	_____
Prof. Dr. Rauf Mirzababayev	_____
Prof. Dr. Sadettin Kapucu	_____
Assoc. Prof. Dr. Gülay Tohumoğlu	_____
Asst. Prof. Dr. Tolgay Kara	_____

ABSTRACT

ENHANCEMENT OF SPEECH COMMUNICATION BETWEEN HUMAN AND MACHINE

ÖZUSTAOĞLU, Hasan Feyzi
M.Sc. in Electrical and Electronics Eng.
Supervisor: Prof.Dr. Arif NACAROĞLU
August 2006, 54 pages

Speech is a most natural and efficient way to exchange information for human beings. To create this information exchange between machine and human being and to make a real "intelligent computer," it is important that the machine can "hear", "understand," and "act upon" spoken information, and also "speak" to complete the information exchange. Therefore, "speech recognition" is essential for a computer to reach the goal of natural human-computer communication. Speech Recognition is a fascinating field spanning several areas of computer science and mathematics. Reliable speech recognition is a hard problem, requiring a combination of many techniques; however modern methods have been able to achieve an impressive degree of accuracy but the improved algorithms still have some problems.

This project attempts to examine one of those techniques, and to apply them to build a sample voice recognition system for Turkish. The goal of this thesis is to be able to recognize isolated words, spoken individually. In this work, the speech data is loaded in to the computer using data acquisition card. This data is processed by software prepared in MATLAB. Linear Predictive Coding and Dynamic Time Warping method is used in the program algorithm.

Keywords: signal processing, speech recognition

ÖZET

ÖZUSTAOĞLU, Hasan Feyzi
Yüksek Lisans Tezi, Elektrik Elekronik Müh. Bölümü
Tez Yöneticisi: Prof.Dr. Arif NACAROĞLU
Ağustos 2006, 54 sayfa

Konuşma insanların arasında bilgi alışverişi için en etkili ve doğal bir yoldur. İnsanlar ve makineler arası iletişimi sağlamak için ve akıllı bilgisayarlar elde etmek için ise makinelerin "duyabilir", "anlayabilir" ve buna gore "davranabilir" olması önem kazanmaktadır. Bu durumda bilgisayar-insan iletişiminin sağlanabilmesi için "konuşma tanıma" bir bilgisayara gerekli duruma gelmektedir. . Konuşma tanıma uzun yıllar boyu bilgisayar ve matematik bilimlerinin ilgi alanı olmuştur. Güvenilir konuşma tanıma, bugün modern metodların da etkileyici bir doğruluk payı elde etmelerine rağmen halen bazı problemleri olan ve bir çok tekniği içeren zor bir problemdir.

Bu proje bu tekniklerden birisini ele alacak ve bunu uygulayarak Türkçe için örnek bir ses tanıma sistemi oluşturulmuştur. Tezin hedefi ayrı ayrı söylenmiş kelimeler arasından tanıma yapabilmesidir. Çalışmada konuşma sesleri ses kartı kullanılarak bilgisayara aktarılmış ve bu data MATLAB'da hazırlanmış program yardımı ile işlenmiştir. Program Linear Predictive Coding ve Dynamic Time Warping yöntemleri kullanılarak hazırlanmıştır.

Anahtar kelimeler: sinyal işleme, konuşma tanıma

ACKNOWLEDGMENTS

I would like to give my sincere appreciation to my advisor Prof. Dr. Arif Nacaroğlu, who pointed me to the topic of speech recognition and for serving on my thesis and for providing valuable suggestions and comments.

Also, thanks to my colleagues for helping me by giving speech samples .

My final thanks go to my family for giving me support and sacrifice during my studies on this thesis work.

CONTENTS

LIST OF FIGURES **Page**

LIST OF TABLES **page**

NOMENCLATURE

HMM	Hidden Markov Model
GMM	Gaussian Mixture Model
LPC	Linear Predictive Coding
MFCC	Mel Frequency Cepstral Coefficients
DTW	Dynamic Time Warping
NN	Neural Networks
$H(z)$	Transfer Function of filter
a_c	Coefficient of the pre-emphasis
$x(n)$	Clean speech signal in discrete-time domain
$\hat{E}(n)$	Short-time energy of speech signal
$w(m)$	Weighting Window
ITU	Upper Energy Threshold
ITL	Lower Energy Threshold
$sgn[.]$	Sign of a signal
ZCT	Zero Crossing Threshold
ZCR	Zero Crossing Rate
σ_{ZCR}	Standard deviation of zero crossing rate
G	Gain constant
a	Coefficients of LPC parameter
p	Order of LPC parameter
M	Normalized inner product between vectors
D	Distance between signals

CHAPTER 1

INTODUCTION

1.1 Historical Background of Speech Recognition

The first speech recognition system was built at Bell Labs in the early 1950's. The task of the system was to recognize digits separated by pauses spoken from a single speaker. The system was built using analog electronics and it performed recognition by detecting the resonant frequency peaks (called formants) of the uttered digit.

Research continued into the 1960's and 1970's fueled by the advent of digital computing technology with focus both on the signal processing and the pattern recognition aspects of developing a speech recognizer. The most significant contributions to speech analysis included to the development of the Fast Fourier Transform, cepstral analysis, and linear predictive coding, which replaced the antiquated method of filter banks used in the past. Pattern recognition algorithms such as artificial neural networks, dynamic time warping, and Hidden Markov Models were successfully applied to speech. These methodologies resulted in several functional and useful systems for a variety of applications [1].

In the 1980's and early 1990's, research focused on expanding the capabilities of speech recognition systems to more complex tasks including speaker independent data sets, larger vocabularies, and noise robust recognition. Progress was made by incorporating speech-tailored signal processing methods for feature extraction such as Mel Frequency Cepstral Coefficients. Hidden Markov Models methods were further refined and became the prevailing speech recognition technology. Also, the research community became more organized for facilitation of data and software sharing so that comparisons among researchers could be made.

Standard speech data was compiled and distributed such as the Texas Instruments & Massachusetts Institute of Technology speech corpus and the Re-source Management corpus. Also, speech software toolkits with open source code were made available; the Hidden Markov Modeling Toolkit being one example. The period was punctuated by the release of several commercial products.

1.2 Current Research in Speech Recognition

Current research efforts are focused on several different task dependent categories as well as on the different aspects of speech recognition systems. The task complexities include speaker independent data sets, large vocabulary recognition, spontaneous speech recognition, noise robust recognition, and speaker verification / identification. Additionally, new methodologies have been under investigation, such as rapid speaker adaptation, discriminative training techniques, vocal tract normalization, multi-modal speech recognition, and improved language modeling.

The popularity of speech recognition is increased as a research area. The progress over the last 10-15 years has been largely incremental.

CHAPTER 2

REVIEW OF SPEECH RECOGNITION

2.1 Glance at Speech Recognition

Voice recognition systems have a very strong probability of becoming a necessity in the workplace in the future. Such systems would be able to improve productivity and would be more convenient to use. The idea of a hardware that can recognize any person's voice without the training time involved in currently employed systems is a very promising one, and possibly a marketable one too. Another aspect of this hardware would be in the assistance of hand-disabled people.

The size of vocabulary of a speech recognition system affects the complexity, processing requirements and the accuracy of the system. Some applications only require a few words (e.g. numbers only); others require very large dictionaries (e.g. dictation machines).

A wide variety of techniques are used to perform speech recognition. There are many types of speech recognition. There are many levels of speech recognition / analysis / understanding.

Typically speech recognition starts with the digital sampling of speech. The next stage is acoustic signal processing. Most techniques include spectral analysis; e.g. Linear Predictive Coding Analysis (LPC), Mel Frequency Cepstral Coefficients (MFCC), cochlea modeling and many more.

The next stage is recognition of phonemes, groups of phonemes and words. This stage can be achieved by many processes such as Dynamic Time Warping (DTW), Hidden Markov Modeling (HMM), Neural Networks (NNs), expert systems and combinations of techniques.

Most systems utilize some knowledge of the language to aid the recognition process. Some systems try to "understand" speech. That is, they try to convert the words into a representation of what the speaker intended to mean or achieve by what they said.

2.2 Isolated Word Recognition

An isolated-word system operates on single words at a time - requiring a pause between saying each word. In this recognition form 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. Isolated-word recognition systems, with short pauses between spoken words, are primarily used in small vocabulary command control applications such as name-dialing, Internet navigation, and voice-control of computer menus or accessories in a car. Isolated word recognition systems may use models trained on whole word examples or constructed from concatenation of sub-word models.

2.3 Continuous Word Recognition

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 recognition is the recognition and transcription of naturally spoken speech. Continuous speech recognition systems are usually based on word model networks constructed from compilation of phoneme models using a phonetic transcription dictionary.

2.4 Speaker-dependent and speaker-independent speech recognition

A speaker independent system is developed to operate for any speaker of a particular type (e.g. American, English or Turkish). These systems are the most difficult to develop, most expensive and accuracy is lower than speaker dependent systems. However, they are more flexible.

In a speaker-independent speech database, the effect of variations of different speakers' characteristics results in higher variances and hence broader (as opposed to narrower) probability distributions of speech features. Broader distributions of different acoustic speech sounds result in higher overlaps between those distributions and hence a higher speech recognition error rate.

Unlike speaker-independent systems, speaker-dependent systems do not have to deal with the extra variance, and the consequent additional overlap of the distributions of different speech sounds, due to the variations of speakers' characteristics. Hence, speaker-dependent systems are more accurate than speaker-independent systems. However, the performance of speaker-independent systems can be improved using speaker adaptation methods to modify the parameters of a speaker-independent system towards those of a speaker-dependent system.

2.5 Speech Processing for Recognition

Speech processing is the mathematical analysis and application of electrical signals for information storage and/or retrieval that results from the transduction of acoustic pressure waves gathered from human vocalizations. The teleology of speech processing generally can be subdivided into the broad overlapping categories of speech analysis, coding, enhancement, synthesis and recognition. Speech analysis is the study of the speech production mechanism in order to generate a mathematical model of the physical phenomena. The study of coding endeavors to store speech information for subsequent recovery. Enhancement, in contrast, is the process of improving the intelligibility and quality of noise-corrupted speech signals. The generation of speech from coded instructions is known as speech synthesis. Speech recognition, though, is the process of synthesis in reverse; namely, given a speech

signal, produce the code that generated it. Speech recognition is the task that will be addressed in this work.

For the vast majority of applications, the code that speech recognition systems strive to identify is the units of language, usually phonemes (set of unique sound categories for a language) or words, which can be compiled into text [2]. Speech recognition can then be formulated as a decoding problem, where the goal is to map speech signals to their underlying textual representations. The potential applications of automatic speech recognition systems include computer speech-to-text dictation, automatic call routing, and machine language translation. Speech recognition is a multi-disciplinary area that draws theoretical knowledge from mathematics, physics, engineering, and social science. Specific topics include signal processing, information theory, random processes, machine learning / pattern recognition, psychoacoustics, and linguistics.

The development of speech recognition systems begins with the processing of a given speech signal for the purposes of obtaining the discriminatory acoustic information about that utterance. The discriminatory information is represented by computed numerical quantities called features. Current speech processing techniques are based on modeling speech as a linear stochastic process [3]. The underlying speech production model is what is known as a source-filter model, where the vocal tract is excited by a series of pitch pulses (e.g. vowels) or Gaussian white noise (e.g. fricatives), which ultimately results in certain frequency spectra that humans recognize as sounds.

Features are extracted with the goal of capturing the characteristic frequency spectrum of a particular phoneme. Using the acoustic features, speech recognition algorithms are employed to accomplish the undertaking. The process starts at the phoneme or acoustic level where consecutive, overlapping, short portions (called frames) of speech (~30-50 ms) are assumed to be approximately stationary.

Following feature extraction, machine-learning techniques are employed to use the features for recognition. There are many machine learning algorithms that can be

utilized for the speech recognition task. A common approach is to use statistical pattern recognition methodologies.

A probability density function is usually a continuous, parametric function that is a sum of weighted Gaussian functions, which is called a Gaussian Mixture Model (GMM). In order to track the rapid changes of the vocal tract as a human utters several sounds (phonemes) consecutively, another model must be employed called a Hidden Markov Model (HMM). A HMM is finite state machine where each state has an associated GMM which represents one possible configuration of the vocal tract. HMMs also have parameters called transition probabilities that represent the likelihood of transitioning from one vocal tract configuration to another. The parameters of these statistical models are learned or optimized using training data.

The pattern recognition seen in Figure 2.1 algorithms utilized must be robust to the inherent variability's of speech such as changing speech rates, co-articulation effects and large vocabulary sizes. Other knowledge sources that have discriminatory power, in addition to the acoustic data, can also be incorporated into the recognizer such as language models. The final output of the system can be the single phoneme, a sequence of phonemes, an isolated word, an entire sentence, etc. depending on the specific application and problem domain.

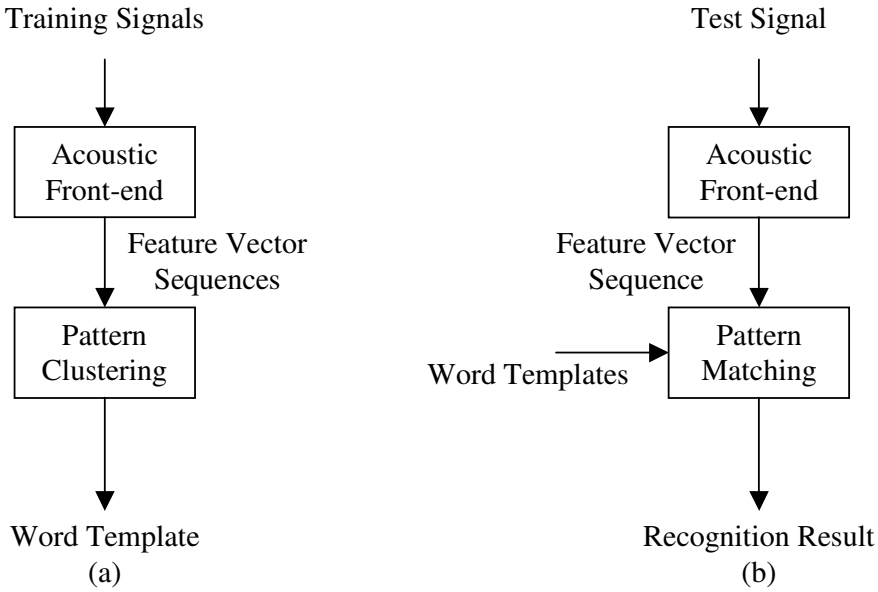

Figure 2.1 Block diagram of simplified speech recognition system

CHAPTER 3

ISOLATED WORD RECOGNITION

3.1 Introduction

After more than 40 years of research, many algorithms have been proposed and implemented for speech recognition. The “pattern-matching” method is one of the well-studied approaches. In this method, the system stores one or more prototypes (templates) for each word in the vocabulary, and compares the incoming speech signal with each of them to find the best-matched one as the recognition result. This approach involves two steps, training the template set, and recognition of the test signal via a pattern matching method.

The training process constructs a template for each word in the vocabulary. Speech signals in the training database are first divided into frames of equal length. Then the acoustic front-end converts each frame into a feature vector that captures the properties of the signal in that frame. These feature vectors will be clustered into groups by the pattern clustering block to form a word model. This process is repeated for every word in the vocabulary. The concept is that if enough versions of a pattern to be recognized are included in the training set, the training procedure should be able to adequately characterize the acoustic properties of the pattern [4].

The recognition procedure first converts the unknown signal into a sequence of feature vectors using the same acoustic front-end as the training process. This feature vector sequence is then compared to each possible word template learned during training by the pattern matching block. A recognition decision is made based on the goodness of the match, which is quantified by a distance function between two sequences of feature vectors, one representing the word model and the other representing the input signal.

Linear Predictive Coding (LPC) is a dominant representation method for the spectral analysis model of a speech signal. It provides a good model to approximate the vocal tract spectral envelope of speaking. The method of LPC is mathematically precise, simple and straightforward to implement in either software or hardware. Also, it works well in recognition applications. Based on these considerations, LPC is used in this project. Dynamic programming is usually applied to combine the distance between each vector pair into the overall distance between two sequences, or the global distance. Its application in speech recognition is known as dynamic time warping (DTW), which stretches or compresses signals nonlinearly to achieve the best matching between them.

3.2 Sampling

The samples extracted from the analog signal must be converted into a digital form. The process of converting the analog waveform representation into a digital code is called analog-to-digital conversion. The inverse process is called digital-to-analog conversion [5].

The analog to digital conversion is generally accomplished by digital signal processing hardware on the computer's sound card (a standard feature on most computers today). The typical sampling rate, 8000 samples per second, is adequate. The spoken voice is considered to be 50 to 4000 Hertz. A sampling rate 8000 gives a Nyquist frequency of 4000 Hertz, which should be adequate for a 4000 Hz voice signal. The sample resolution will be 8 bits per second that sound cards can accomplish.

3.3 Pre-Emphasis

Because speech has an overall spectral tilt of 5 to 12 dB per octave, a pre-emphasis filter of the form

$$H(z) = 1 - a_c z^{-1} \quad 0.9 \le a_c \le 1 \tag{3.1}$$

is normally used [6]. Pre-emphasis causes the speech signal to be spectrally flatten and make it less susceptible to finite precision effect later in the signal processing. There are many other reasons why speech signal is pre-emphasized [7]. The gain of the pre-emphasis circuit is given in Figure 3.1

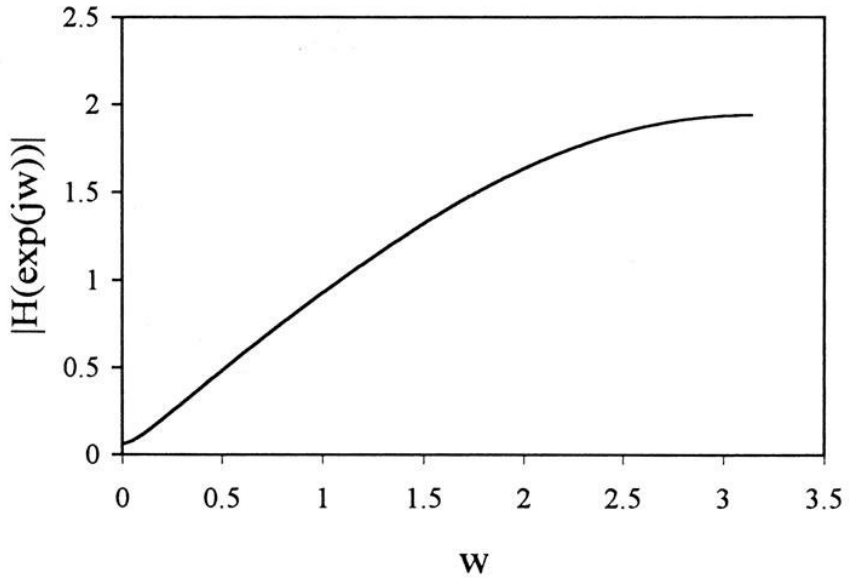

Figure 3.1 Magnitude spectrum of pre-emphasis system for $a_c = 0.95$

The Turkish word “çiçek” is loaded and its waveform, before and after pre-emphasis is given in Figure 3.2 and Figure 3.3 respectively.

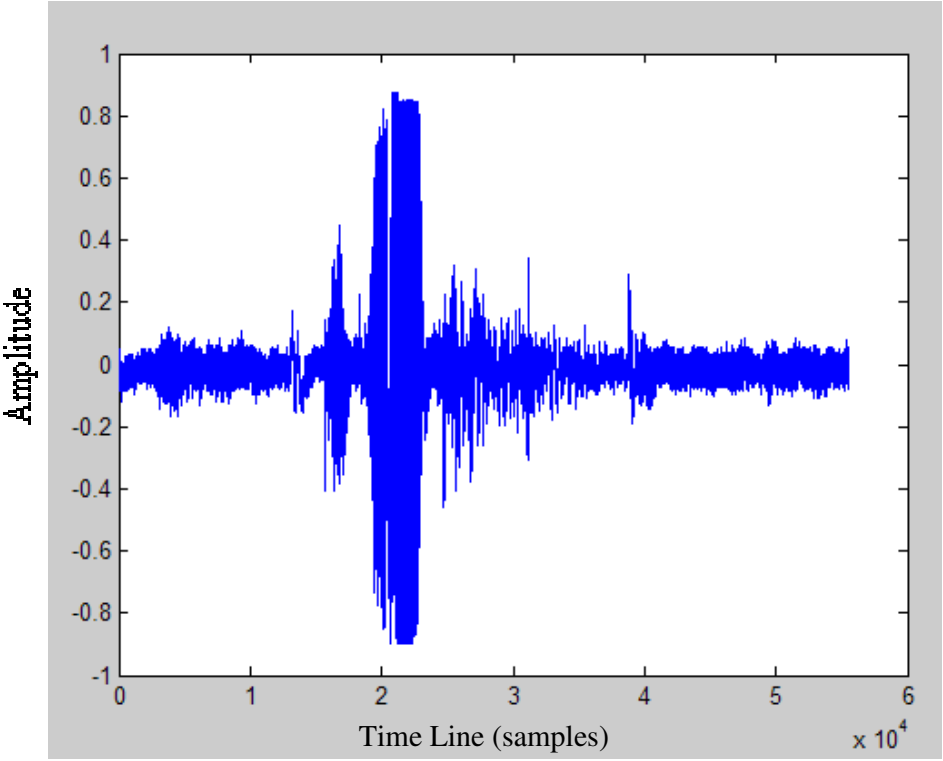

Figure 3.2 Waveform for speech “çiçek” before pre-emphasis

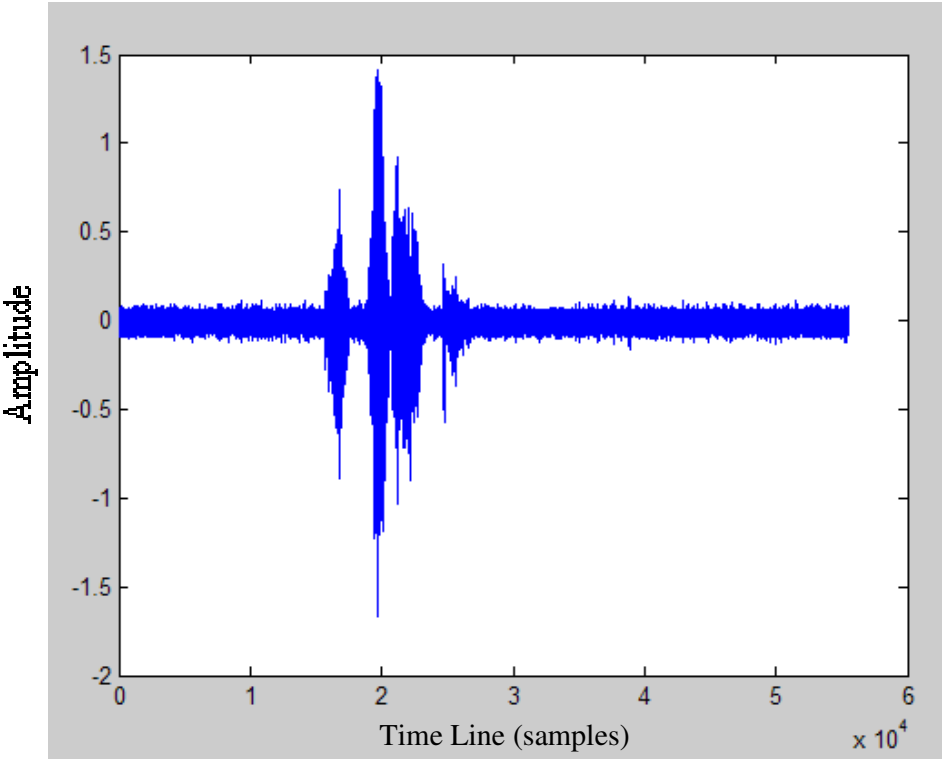

Figure 3.3 Waveform for speech “çiçek” after pre-emphasis

3.4 Finding Speech Beginning and End Point

An important problem in speech recognition is to detect the presence of speech in a background of noise. This problem is often referred to as the start/end point location problem [8].

Our algorithm is based on two measures of speech: short-time energy and zero-crossing rate. The algorithm forms, along with a noise compensation algorithm, a front-end processor of a speech recognizer system. It should be noted here that speech detection is necessary for speech recognition since word boundaries must be approximately known to trigger the recognizer correctly.

Moreover, proper location of regions of speech not only substantially reduces the amount of processing, but also increases the recognition rate. For example, in our experiment two utterances are to be compared by a dynamic time warping (DTW) procedure, each of them including about one third of useful speech information (Figure 3.4).

The regions marked as noise are susceptible to produce errors, as utterances may be considered as “equals” during these regions. This is one of the major difficulties in classifying among words which have same length [9].

A reliable discrimination between voice and silence constitutes a difficult classification problem. It is known that zero-crossing rate generally complements the energy, i.e. during a word, when energy is low, zero-crossing rate is usually high and vice-versa.

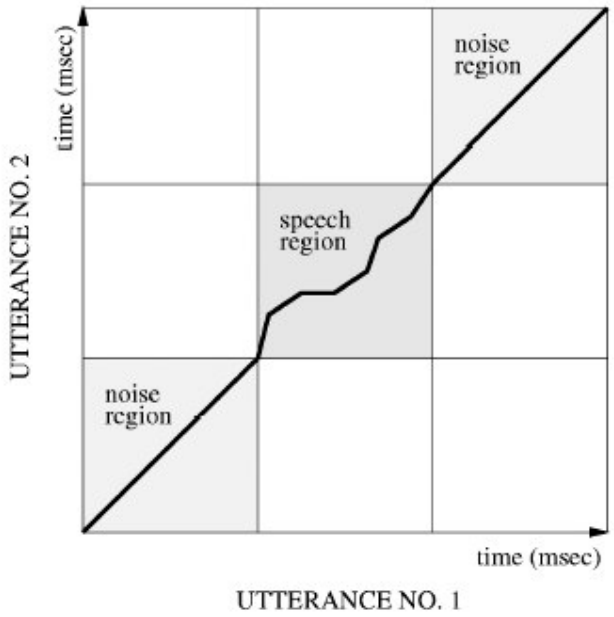

Figure 3.4 DTW alignment for two utterances including about one third of useful speech

Therefore, the algorithm relies on these two measures of speech, one with meaning of magnitude and the other with meaning of frequency:

$$\hat{E}(n) = \sum_{m=0}^{N-1} \left| w(m)x(n-m) \right| \tag{3.2}$$

$$ZCR = \sum_{n=0}^{N-1} \frac{1 - \mathrm{sgn}\left[x(n+1)\right]\mathrm{sgn}\left[x(n)\right]}{2} \tag{3.3}$$

where E is the pseudo-energy of the input signal $x(n)$, $w(m)$ is the weighting window, N is the window size, and ZCR means zero-crossing rate.

The energy is mainly used to discriminate voiced / unvoiced regions of speech. The lower energy threshold ITL is calculated using

$$ITL = \frac{1}{10}\sum_{n=1}^{10} \hat{E}(n) = \frac{1}{100m\,\mathrm{sec}^* f_s} \sum_{n=1}^{100m\,\mathrm{sec}^* f_s} \left| x(n) \right| \tag{3.4}$$

as the average energy in the first 100msec duration of the recording interval in which the speech is not present. It is assumed that upper threshold energy *ITU* is

$$ITU=5*ITL \tag{3.5}$$

In Equation (3.4), f_s is the sampling frequency.

A zero-crossing is also a very useful parameter for speech detection and may be mathematically described by:

$$sgn[x(n+1)] \neq \text{sgn}[x(n)] \tag{3.6}$$

where

$$sgn[x(n)] = \begin{cases} +1 \text{ if } x(n) \geq 0 \\ -1 \text{ if } x(n) < 0 \end{cases} \tag{3.7}$$

Zero-crossing rate is defined as the number of zeros (level) crossing per 10 ms interval. It is assumed that the speech is not present during first frames of the recording interval, so in this interval the statistics of the background silence are measured. These measurements include the average and standard deviation of the zero crossing rate and the average energy. If any of these measurements are excessive the algorithm halts. Otherwise, a zero crossing threshold *ZCT* for unvoiced speech is chosen as the minimum of a fixed threshold which may be formulated as

$$ZCT = \min(25, average(ZCR) + 2\sigma_{ZCR}) \tag{3.8}$$

$$average(ZCR) = \frac{ZCR}{N} \tag{3.9}$$

where *ZCR* is given in Equation (3.3). N is the number of samples.

In Figure 3.5 it can be seen that the beginning and end point found by using this method for speech “pencere”. The Figure 3.6 shows the principles used for start/end point detection.

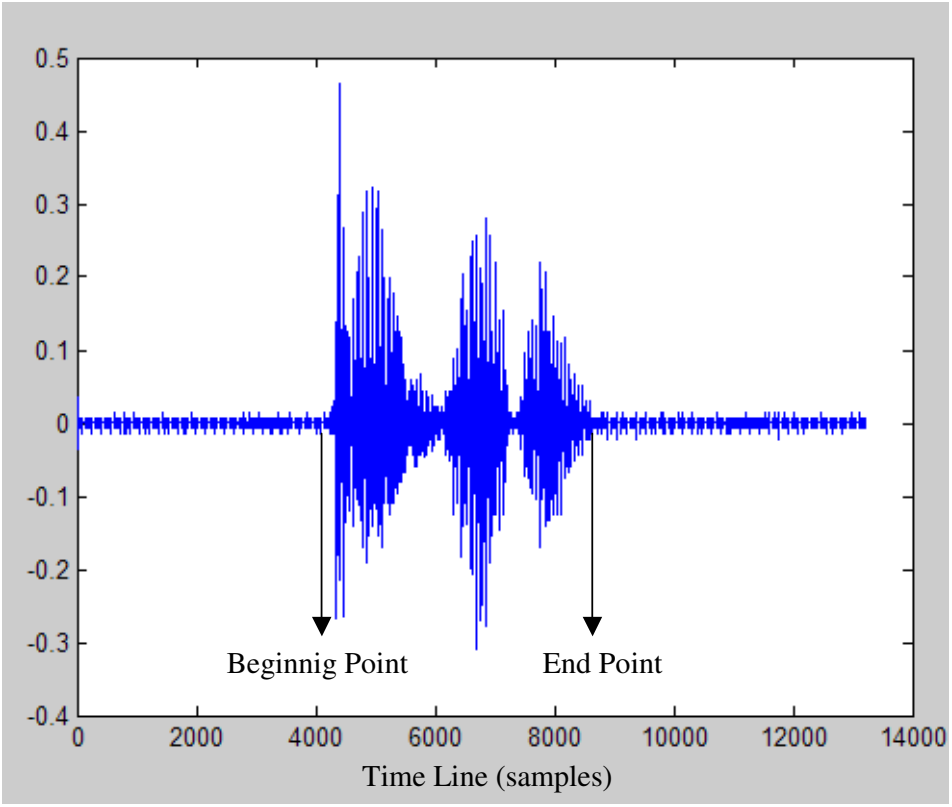

Figure 3.5 Waveform for the speech “pencere” with beginning and end points

ZCR = Zero Crossing
ZCT = Zero Crossing Threshold

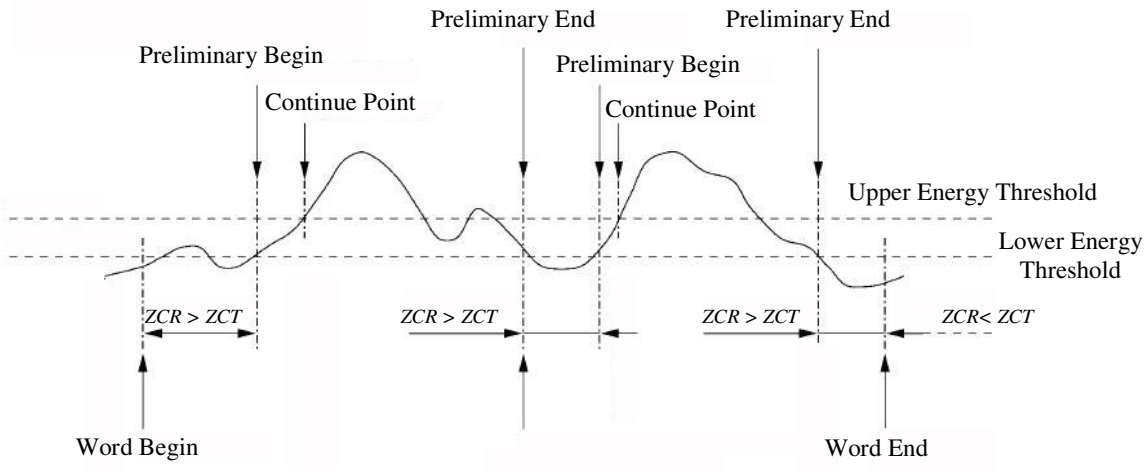

Figure 3.6 The algorithm used for start / end point detection

3.5 Framing of Speech Signal

In most processing tools, it is not appropriate to consider a speech signal as a whole for conducting calculations. A speech signal is often separated into a number of segments called frames. This process of separation is known as framing. Figure 3.7 illustrates how a signal might be divided into frames. Each frame will have the same number of samples although the last frame might have a small variance [10].

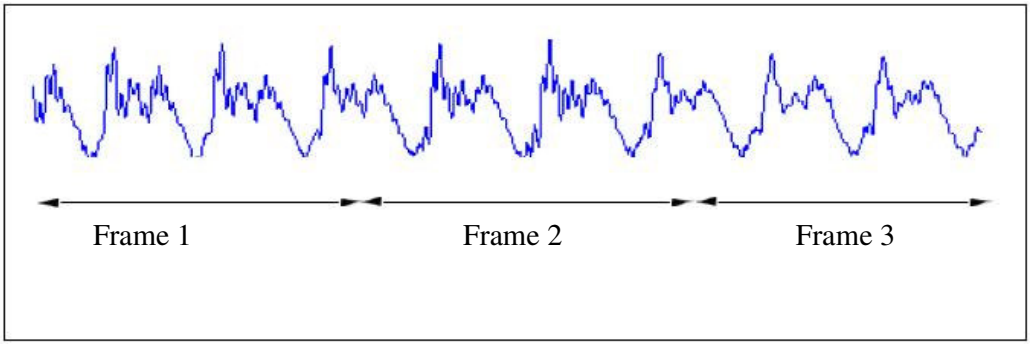

Figure 3.7 A signal is divided into 3 frames

3.6 Windowing of Speech Signal

There are many types of windowing that can be used in digital signal processing. These include Hamming, Hanning, Kaiser, and Chebyshev etc. The purpose of windowing is to make the frame intense in some area while irrelevant in other areas.

A typical window might be the Hamming window. The intensity of the window is much greater around the center than at the edges. When this window is multiplied point to point with a frame, the edges of the frame will become insignificant. Therefore, calculations on the frame will not be affected by the end data.

Hamming windows have the property of low amplitudes at the edges in time domain. Correspondingly, the side lobes become lower at higher frequencies as shown on Figure 3.8.a and 3.8.b.

$$w(m) = 0.54 - 0.46\cos(\frac{2\pi m}{N-1}),\ 0 \le m \le N-1 \qquad (3.10)$$

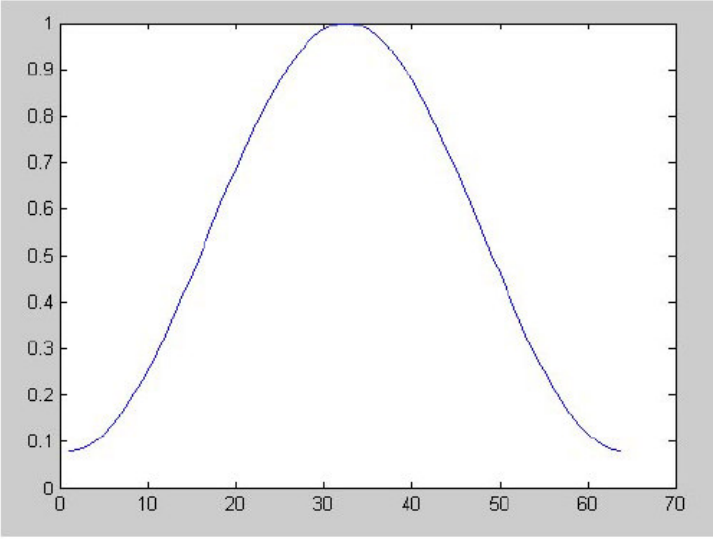

Figure 3.8.a Hamming window in time domain

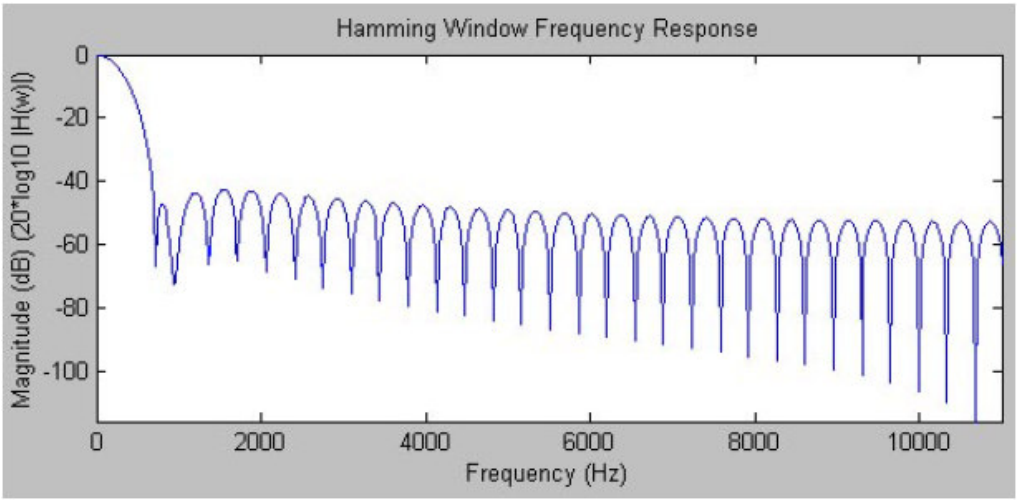

Figure 3.8.b Frequency response of Hamming window

The purpose of these properties is to compensate the effect of spectral leakage when a signal is divided into frames [10]. The side lobes of these windows will break down the unwanted noise contributed in the signal. In the frequency representation, it is desirable to design the properties of the window to have a low noise bandwidth. This can be achieved by reducing the side lobe amplitudes.

However, windowing has negative effects. As the frames are multiplied with the windows, most of the data at the edge of the frame will became insignificant causing

a lost of information. Therefore, we must include a method called overlapping to compensate this loss.

3.7 Overlapping

When a data is framed and windowed, the data at the ends of the frame is much likely to be reduced to zero. This will represent a loss of information. An approach to tackle this problem is to allow overlapping in the sections between frames.

Overlapping will allow adjacent frames to include portions of data in the current frame. This will mean the edges of the current frame will be included as the center data of adjacent frames. Typically, around 60 % of overlapping is sufficient to embrace the lost information. Figure 3.9 shows how a signal is framed, windowed and overlapped.

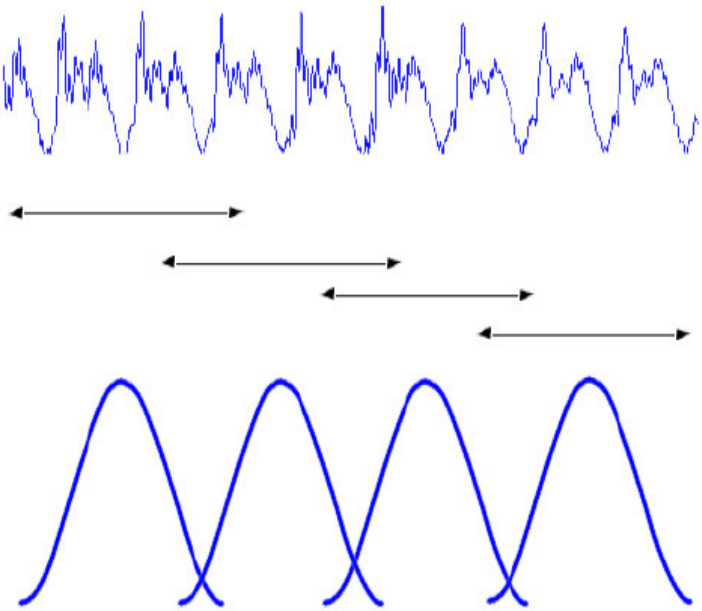

Figure 3.9 Signal is framed. Each frame overlaps the previous frame. And windows are multiplied into each frame

3.8 Linear Predictive Coding of Speech Signal

Linear Predictive Coding is one of the most powerful speech analysis techniques, and one of the most useful methods for encoding good quality speech at a low bit rate. It

provides extremely accurate estimates of speech parameters, and is relatively efficient for computation.

Usually speech signal features are derived from Linear Predictive Coding (LPC), a technique that attempts to derive the coefficients of a filter that (along with a power source) would produce the utterance that is being studied. LPC is useful in speech processing because of its ability to extract and store time varying formant information [11]. Formants are points in a sound's spectrum where the loudness is boosted. What we get from LPC analysis is a set of coefficients that describe a digital filter.

LPC starts with the assumption that a buzzer at the end of a tube produces the speech signal. The glottis (the space between the vocal cords) produces the buzz, which is characterized by its intensity (loudness) and frequency (pitch). The vocal tract (the throat and mouth) forms the tube, which is characterized by its resonances, which are called LPC analyzes the speech signal by estimating the formants, removing their effects from the speech signal, and estimating the intensity and frequency of the remaining buzz. The process of removing the formants is called inverse filtering, and the remaining signal is called the residue. The numbers, which describe the formants and the residue, can be stored or transmitted somewhere else. LPC synthesizes the speech signal by reversing the process: use the residue to create a source signal, use the formants to create a filter (which represents the tube), and run the source through the filter, resulting in speech. Because speech signals vary with time, this process is done on short chunks of the speech signal, which are called frames. Usually 30 to 50 frames per second give intelligible speech with good compression.

LPC method is used to successfully estimate basic speech parameters like pitch, formants and spectra. The principle behind the use of LPC is to minimize the sum of the squared differences between the original speech signal and the estimated speech signal over a finite duration. This could be used to give a unique set of predictor coefficients. These predictor coefficients are normally estimated every frame. The predictor coefficients are represented by a_k. Another important parameter is the gain (G). The transfer function of the time-varying digital filter is given by:

$$H(z) = \frac{G}{1 - \sum a_k z^{-k}} \tag{3.11}$$

The summation is computed starting at $k = 1$ up to p, which will be 10 for the LPC-10 algorithm, and 18 for the improved algorithm. This means that only the first 18 coefficients are transmitted to the LPC synthesizer. The two most commonly used methods to compute the coefficients are the covariance method and the auto-correlation formulation. For our implementation, we used the auto-correlation formula. The reason is that this method is superior to the covariance method in the sense that the roots of the polynomial in the denominator of the above equation is always guaranteed to be inside the unit circle, hence guaranteeing the stability of the system *H(z).* Levinson - Durbin recursion was utilized to compute the required parameters for the auto-correlation method.

Actually LPC determines the coefficients of a forward linear predictor by minimizing the prediction error in the least squares sense. With LPC method we find the coefficients of a *p*th-order linear predictor (FIR filter) that predicts the current value of the real-valued time series x based on past samples.

$$\hat{x}(n) = -a(2)x(n-1) - a(3)x(n-2) - \cdots - a(p+1)x(n-p) \tag{3.12}$$

p is the order of the prediction filter polynomial,

$$a = [1\ \mathrm{a}(2)\ \ldots\ \mathrm{a}(\mathrm{p}+1)] \tag{3.13}$$

LPC uses the autocorrelation method of autoregressive (AR) modeling to find the filter coefficients [12]. The generated filter might not model the process exactly even if the data sequence is truly an AR process of the correct order. This is because the autocorrelation method implicitly windows the data, that is, it assumes that signal samples beyond the length of x are 0. LPC computes the least squares solution to

$$Xa \approx b \tag{3.14}$$

where

$$X = \begin{bmatrix} x(1) & 0 & \dots & 0 \\ x(2) & x(1) & \ddots & \vdots \\ \vdots & x(2) & \ddots & 0 \\ x(m) & \vdots & \ddots & x(1) \\ 0 & x(m) & \ddots & x(2) \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & x(m) \end{bmatrix}, \; a = \begin{bmatrix} 1 \\ a(2) \\ \vdots \\ a(p+1) \end{bmatrix}, \; b = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{3.15}$$

and m is the length of *X*. Solving the least squares problem via the normal equations

$$X^H X a = X^H b \tag{3.16}$$

leads to the Yule-Walker equations [13]

$$\begin{bmatrix} r(1) & r(2)^* & .... & r(p)^* \\ r(2) & r(1) & \ddots & \vdots \\ \vdots & \ddots & \ddots & r(2)^* \\ r(p) & \dots & r(2) & r(1) \end{bmatrix} \begin{bmatrix} a(2) \\ a(3) \\ \vdots \\ a(p+1) \end{bmatrix} = \begin{bmatrix} -r(2) \\ -r(3) \\ \vdots \\ -r(p+1) \end{bmatrix} \tag{3.17}$$

where

$$r = [\text{r(1) r(2) ... r(p+1)}] \tag{3.18}$$

is an autocorrelation estimate for *X*. Yule-Walker equations can be solved by Levinson-Durbin Algorithm which solves the symmetric Toeplitz system of linear equations [14].

3.9 Similarity Matrix

We will use a utility to calculate the full local-match matrix calculating the distance between every pair of frames from the sample and template signals A and B, which have same rows [15].

$$EA = \sqrt{(\sum A^2)} \text{ and } EB = \sqrt{(\sum B^2)} \tag{3.19}$$

Normalized inner product between vectors will be

$$M = \frac{A'*B}{(EA)'*(EB)} \tag{3.20}$$

3.10 Dynamic Time Warping

Many algorithms are currently used for searching a best matching path between two signals. Some of the more successful techniques include hidden Markov models applied at both the phoneme and at the word level. However, dynamic programming (DP) remains the most widely used algorithm for real-time recognition systems. It is considered sufficiently mature to solve sequential decision problems. Silverman and Morgan gave detailed description of the history of the application of DP in speech recognition in [16].

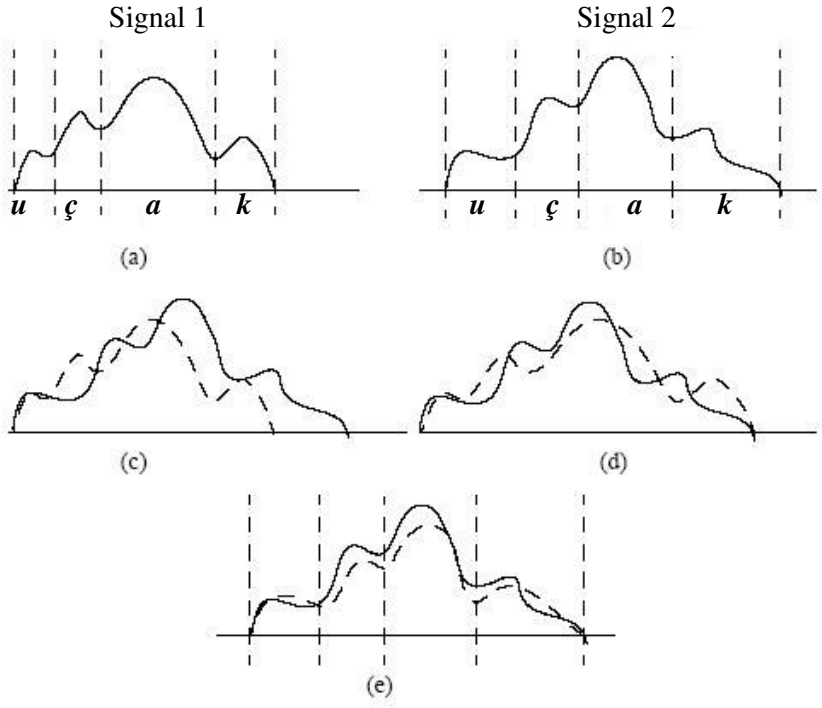

Figure 3.10 Comparison of different time normalizations for the word "*uçak*"

Because of different accents, speaking styles of speakers, etc., waveforms of one word may have a lot of differences. Even spoken by the same speaker, differences still exist because of speaking rate, loudness and stress. Some patterns may have a longer duration and higher amplitude; others may be shorter and weaker. Parts *(a)* and *(b)* of Figure 3.10 show two repetitions of the word "*uçak*". Comparing these two signals, phones *u, ç* and *k* of signal 1 are shorter, and phone *a* is longer, as part *(c)* of Figure 3.10. Here, we use a dotted line to show signal 1, and a solid line to show signal 2 so that they can be distinguished in the same plot. The problem of comparing these two signals as they are is that their lengths are not the same. From previous sections of this chapter, we know that a speech signal can be represented by a sequence of feature vectors. Let's use indicates $\vec{a} = \{a_1, a_2, ...a_I\}$ to indicate signal 1, and $\vec{r} = \{r_1, r_2, ...r_J\}$ to indicate signal 2. The lengths of these two signals may not be the same so that I does not necessarily equal to J

A simple normalization scheme is to extend the shorter signal linearly to the length of the longer one. Let's assume that $J < I$, and use $D(\vec{r}, \vec{a})$ to denote the distance between signal $\vec{a}$ and $\vec{r}$, which can be computed as

$$D(\vec{r}, \vec{a}) = \sum_{i=1}^{I} d_f(r_j a_i) \quad j = \frac{J}{I} * i \tag{3.21}$$

where $d_f(r_j a_i)$ denotes the distance between frame $\vec{a}$ and $\vec{r}$.

The result of this linear time normalization alignment is shown as part *(d)* of Figure 3.10. It assumes that the speaking rate is proportional to the duration of the utterance, and is independent of the sound being spoken. But from the plot of signal $\vec{a}$ and $\vec{r}$, we know that this assumption may not model the real situation of speech utterance. A good normalization scheme should change the duration of one signal according to the characters of the other signal, to achieve a reasonable matching between them, so that the distance $D(\vec{r}, \vec{a})$ can be minimized. Part *(e)* of Figure 3.10 shows a nonlinear time normalization example. We can see that the parts of *u, ç* and *k* of signal 1 are extended, and part is *a* is compressed [17].

Dynamic programming is commonly used to solve sequential decision problems. The nonlinear matching described here is usually performed in a grid plane as shown in Figure 3.11. In the grid plane, signal $\vec{a}$ is aligned along the x-axis, and signal $\vec{r}$ is aligned along the y-axis. Each intersection in this plane is defined as a node, where node *(i, j)* means matching frame *i* of signal $\vec{a}$ with frame *j* of signal $\vec{r}$. The node *(0,0)* is called the original node, which is a dummy node that all paths start from. The cost associated with this matching is defined as the frame distance $d_f(i, j)$ between feature vectors *a(i)* and *r(j)*. The cost at node is defined to be zero, which is $d_f(0,0) = 0$

A path is defined as the catenation of node pairs $(i_{k-1}, j_{k-1}) \longrightarrow (i_k, j_k)$, which means extending from node (i_{k-1}, j_{k-1}) to node (i_k, j_k). Here we use i_k to indicate the index of signal $\vec{a}$ at time *k*, and j_k is the index of signal $\vec{r}$ at time *k*.

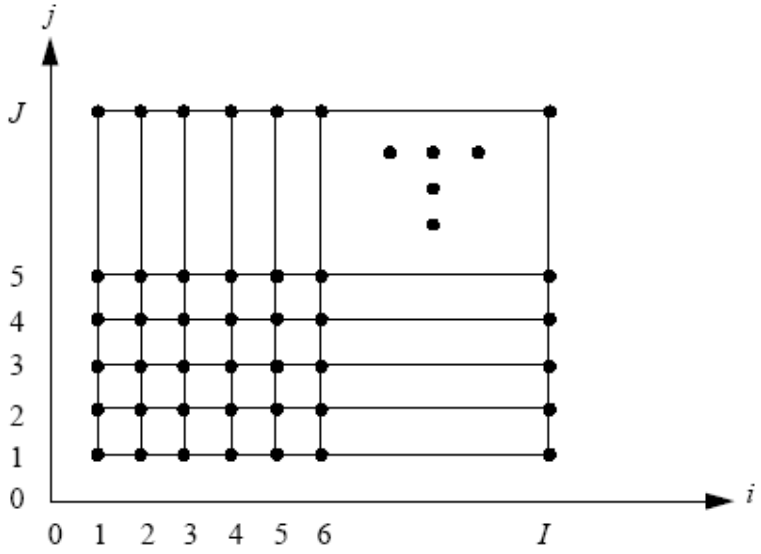

Figure 3.11 The grid plane to illustrate dynamic programming

A path which starts from node *(0,0)* and ends at node (i_k, j_k) has an overall cost, which means the accumulated cost from the starting point of this path until it meets node (i_k, j_k). We use $D(i_k, j_k)$ to denote this type of cost, and it is defined as

$$D(i_k, j_k) = D(i_{k-1}, j_{k-1}) + d_f(i_k, j_k) \tag{3.22}$$

Since node (0,0) is the dummy-starting node for all paths, we define

$$D(0,0)=0 \tag{3.23}$$

Tracing the right part of equation (3.22) back to $k=1$, we get

$$D(i_k\, j_k) = \sum_{m=0}^{k} d_f(i_m, j_m) \tag{3.24}$$

The problem can be written as finding a sequence of nodes (i_k, j_k) which minimizes the accumulated cost for a complete path ending at node *(I, J)*

$$\begin{aligned} D(i_k, j_k) &= \min\left[D(i_{k-1}, j_{k-1})\right] + d_f(i_k, j_k) \\ &= \min\left[\sum_{m=0}^{k} d_f(i_m, j_m)\right] \end{aligned} \tag{3.25}$$

In this equation, $(i_k, j_k) = (I, J)$.

The pattern matching method in speech recognition is used to measure the distance between the test signal and all of the templates, and then pick the template with smallest distance. The word that this template refers to is the recognition result. Therefore, the recognition problem can be simplified as finding the distance between a signal and a template. We know that the challenge of this problem is the diversity of the speaking behavior of humans. The differences imported due to speaking rate or style etc., should not affect the recognition result. We indicated here that dynamic programming can be used to find a nonlinear matching path between the test signal and the template, so that the distance between them is minimized. This application of dynamic programming is called dynamic time warping.

Dynamic time warping uses the same grid plane of Figure 3.8 to define the search space. Here, signal $\vec{a}$ is the test signal, and $\vec{r}$ is the template signal. We will search for the best path along the index of the test signal. As the specific application of dynamic programming in speech recognition, DTW applies particular constraints to the problem according to *a priori* information known about the speech signal, and these constraints define how abruptly the time-scale of one signal can be changed relative to the other signal. The paths of the search space, which are considered unreasonable, are pruned.

The commonly used constraints will be discussed in the following subsections [17].

1. Endpoint Constraints

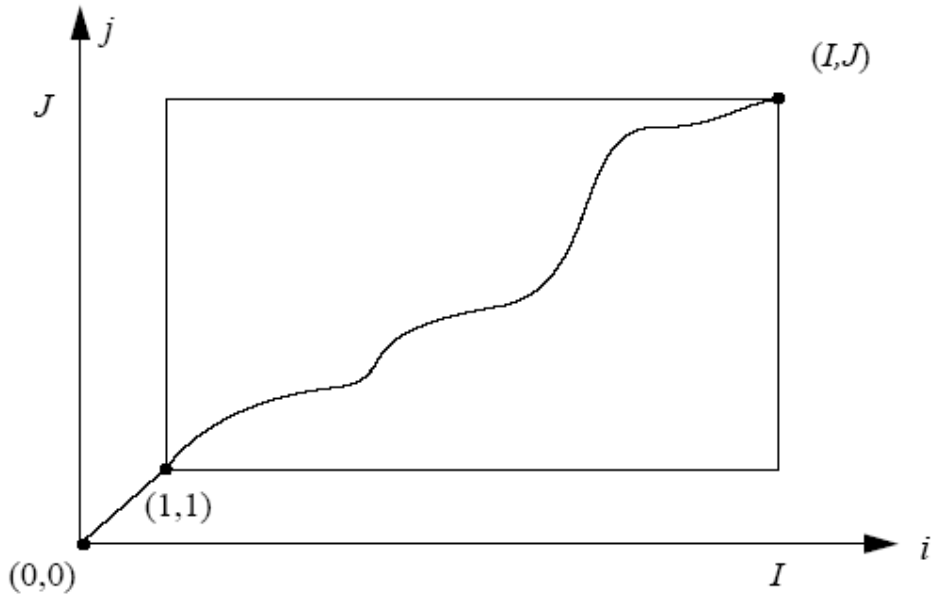

Figure 3.12 The ending-point constraints

Starting point constraint: the first frame of the test signal must be matched with the first frame of the template signal:

$$(i_1, j_1) = (1,1) \tag{3.26}$$

Ending point constraint: the last frame of the test signal must be matched with the last frame of the reference signal:

$$(i_k, j_k) = (I, J) \tag{3.27}$$

Under these constraints, a typical matching path would look like the curve in Figure 3.12.

2. Continuity and monotonicity constraints

The continuity constraint states that there is no "breaking" point in a path, which means

$$i_k - i_{k-1} = 1 \tag{3.28}$$

In other words, the matching must be made along the axis of the test signal one frame after another.

The monotonicity constraint requires vectors of a word to be clustered in monotonically increasing order, which means

$$j_k - j_{k-1} \geq 0 \tag{3.29}$$

Figure 3.13 is an example of non-monotonic matching. The signal of "ağır" is listed along x-axis, and "ağrı" is listed along y-axis. The phoneme sets of both the words "ağır" and "ağrı" are *{a,ğ, r, ı}*. Without monotonicity constraints, these two signals can be matched along the solid line, which gives the incorrect information that these two signals are similar.

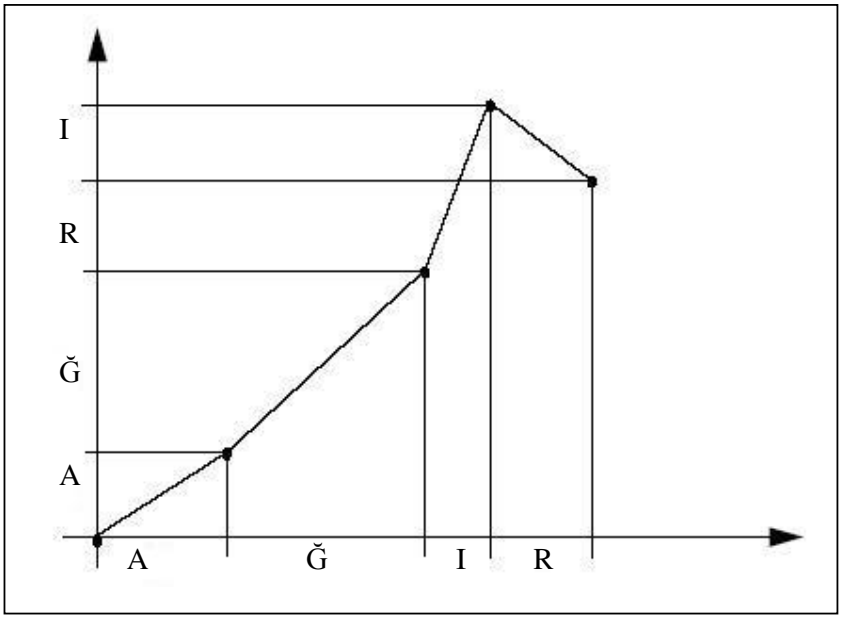

Figure 3.13 Non-monotonic matching between “ağır” and “ağrı”

3. Global Path Constraint

This constraint is used to restrict the extent of compression or expansion of speech signals over long ranges of time. The variation of the speaking rate of human beings is considered to be limited in a reasonable range, which means that we can prune the unreasonable search space, and limit the search to the “legal” region. The commonly used global path constraints are shown in Figure 3.14.

Here the search region is limited by four straight lines whose slopes are s and 1/s, and pass through node *(1,1)* and *(I, J)*. The shadow area is the legal search region

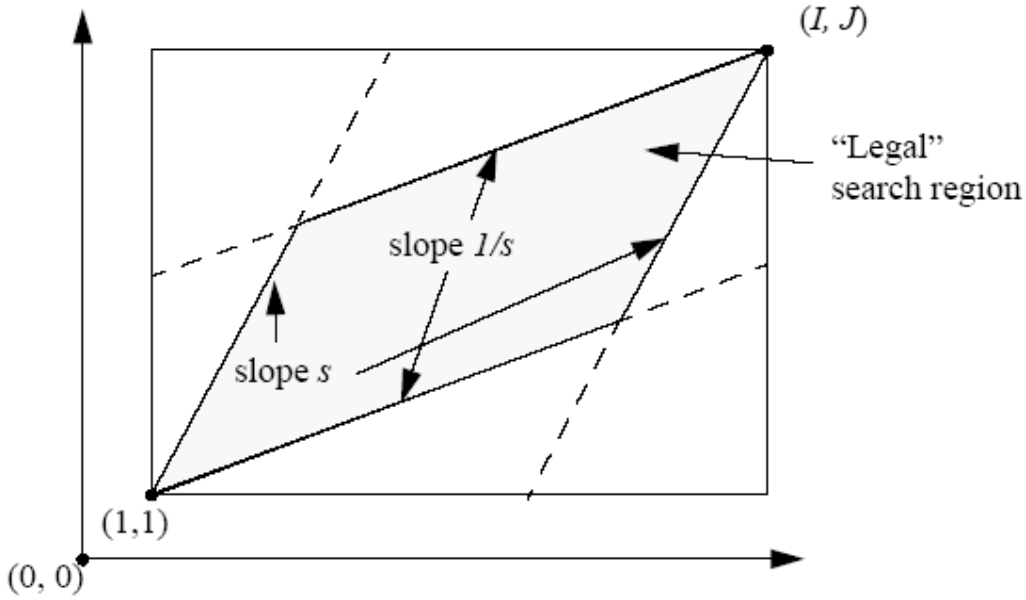

Figure 3.14 The global path constraints

CHAPTER 4

EXPERIMENTAL SET

The apparatus consists of several MATLAB programs running on the Windows operating system on a personal computer with a sound card. The programs can be used interactively.

As discussed in the previous chapters, speech recognition will be implemented by Dynamic Time Warping method and LPC feature extraction method by using MATLAB program. Database of reference patterns will be created which are LPC operation results from recorded isolated word speech after finding beginning and end points of the word. And then test pattern of speech sound will be computed which will be recognized. Later on, this test pattern will be compared with each recorded reference pattern which is added to database one by one. Depending on the Dynamic Time Warping Algorithm the shortest distance will be calculated and this word will be displayed as recognized word. This procedure can be seen in Figure 4.1.

4.1 Algorithm of Program

Step 1 Reference word is sampled by 8 KHz sampling rate with 8bits.

Step 2 Coefficients of pre-emphasis is calculated by using Equation (3.1).

Step 3 The pre-emphasis filter calculated in Step 2 is applied to sampled word.

Step 4 Primary input is framed with 10msec duration.

Step 5 Each frame is windowed with Hamming Window stated in Equation (3.10).

Step 6 The energy of framed signal is calculated by using Equation (3.2).

Step 7 Lower Threshold Energy and Upper Threshold Energy of speech signal are calculated by using Equation (3.4) and (3.5) respectively.

Step 8 Zero Crossing Rate (ZCR) of speech signal and average of ZCR are calculated by using Equation (3.3) and (3.9) respectively.

Step 9 Threshold of zero crossing rate ZCT is calculated by using Equation (3.8).

Step 10 From the first index of the energy of speech signal, the point where the energy of speech signal is greater or equal to lower threshold energy is found by comparing the value of energy of speech signal with the value of lower threshold energy.

Step 11 From the found point at step 10, the another point where the energy of speech signal is greater or equal to upper threshold energy is found and is used to determine beginning point of the speech signal at the next steps.

Step 12 From the last index of the energy of the speech signal, the point where the energy of speech signal is greater or equal to lower threshold energy is found by comparing the value of energy of speech signal with the value of lower threshold energy.

Step 13 From the found point at step 12, another point where the energy of speech signal is greater or equal to upper threshold energy is found and is used to determine end point of the speech signal at the next steps.

Step 14 From the found point at step 11 to back by 25 points, i.e. 250 msec interval preceding the initial beginning point, and the number of intervals at which the zero crossing rate exceeds the threshold zero crossing rate is counted. If the number of times the threshold is exceeded is 3 or more, the starting point is set back to the first point (in time) where the threshold is exceeded. If it does not exceed the threshold zero crossing rate, the beginning point of speech signal, which is found at step 11, is kept.

Step 15 From the point at step 13 to forward by 25 points, the number of intervals at which the zero crossing rate exceeds the threshold zero crossing rate is counted. If the number of times the threshold is exceeded is 3 or more, the end point is set forward to the last point (in time) at which the threshold is exceeded. If it does not exceed the threshold zero crossing rate, the end point of speech signal, which is found at step 13, is kept.

Step 16 The signal is framed by 360 samples with 120 samples of overlapping.

Step 17 Each frame is windowed with Hamming Window stated in Equation (3.10).

Step 18 LPC coefficients are calculated for each frame stated in Equation (3.13).

Step 19 The data collected in Step 17 is recorded in database for each reference word.

Step 20 The same procedure till this step is repeated for test word.

Step 21 As stated in Equation (3.20) normalized inner product between vectors is calculated for each reference word and test word one by one.

Step 22 Minimum distance is calculated for each comparison by using DTW method stated in Equation (3.25).

Step 23 Among those calculated distances the word with smallest distance result is displayed as recognized word.

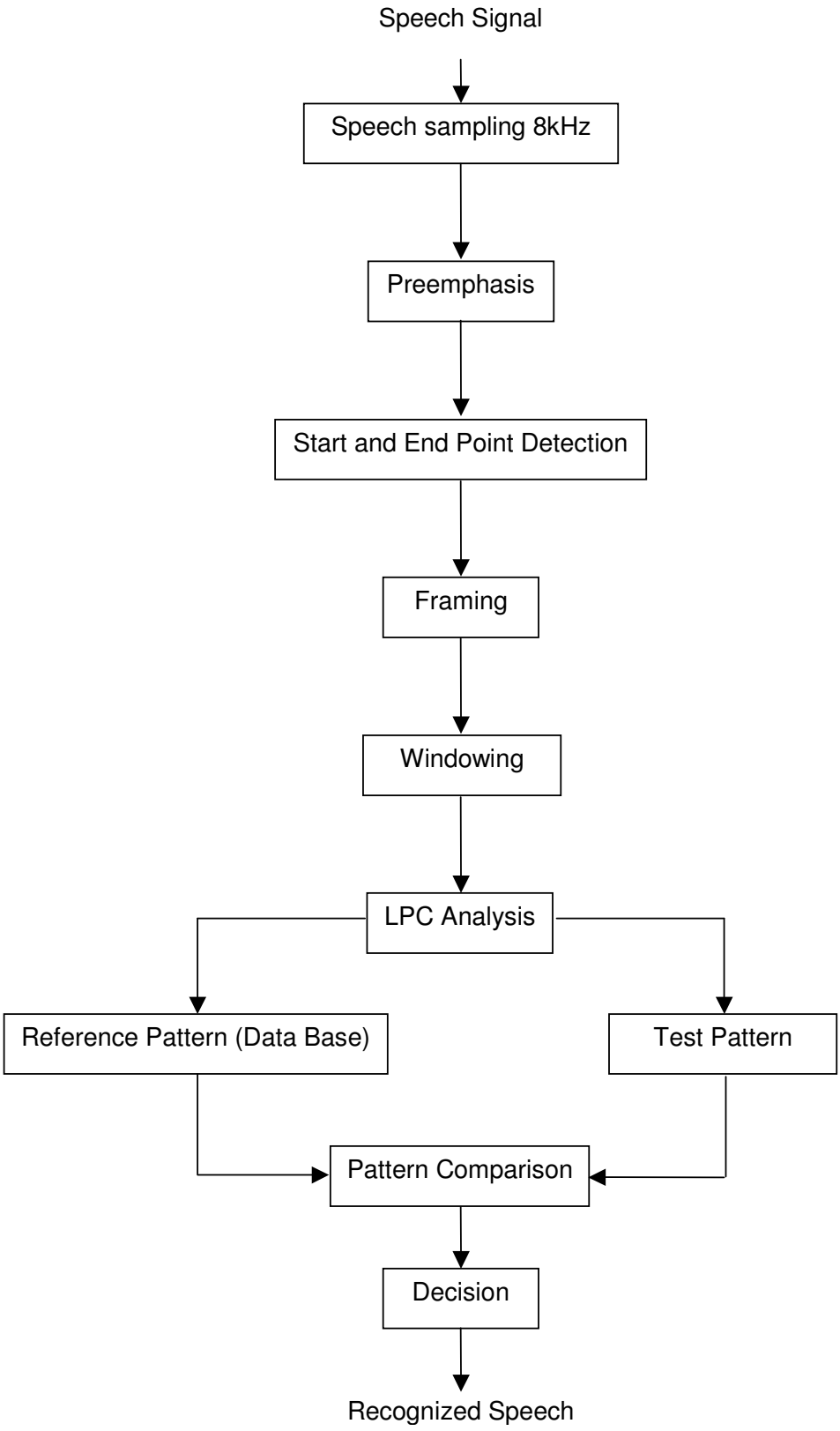

Figure 4.1 Flow Chart of Speech Recognition

4.2 Source Codes of Program

You can find the source code of the experiment below.

4.2.1 Main Program

This part of program is the main graphical user interface of the program which calls other subroutine programs which record sample words, recognize the word, reset the database and quit from the program.

```
M = menu({'ISOLATED WORD'...
,'SPEECH RECOGNITION'...
,' '...
,'What would you like to do?'...
,' '}...
,'Load Reference Sound for Data Base'...
,'Load Test Sound for Recognition'...
,'Reset Data Base'...
,'Data Base Info'...
,'Quit');

switch M
case 1  % Record sample words
reference;
case 2  % Recognize the word
recognition;
case 3 % Reset Database
button = questdlg('Do you want to continue?',...
'Continue Reseting','Yes','No','No');
if strcmp(button,'Yes')
  if exist('C:\MATLAB6p5\work\THESIS\count.dat','file')
     delete('C:\MATLAB6p5\work\THESIS\reference.dat');
     delete('C:\MATLAB6p5\work\THESIS\count.dat');
  else
```

```
      errordlg('Data Base does not exist...','File Error');
  end
elseif strcmp(button,'No')
end
main;
case 4 % Database Info
   clc
   m=0; cnt=0; FN=0; i=0;
   if exist('C:\MATLAB6p5\work\THESIS\count.dat','file')
      [cnt,FN]=textread('C:\MATLAB6p5\work\THESIS\count.dat','%d %s ');
      [m,n]=size(FN);
   if m==1
      disp([sprintf('%d',m),' sound is recorded:'])
      disp('-----')
   else
      disp([sprintf('%d',m),' sounds are recorded:'])
      disp('-----')
   end
   for i =1:m
      disp(['Sound ID:',sprintf('%d',cnt(i,1))])
      disp(['File:',FN{i,1}])
      disp('-----')
   end
   else
     msgbox ('There is no data in Data Base...')
   end
      main;
otherwise % Quit
clear
clc
break;
end
```

4.2.2 Start and End Point Detection

The theoretical explanation of the start and end point detection was given in section 3.4. Here the source code which prepares the required procedure is given.

```
function [N1,N2] = epd(signal)

fs=8000;
winSizeMs=10;   % should be 10ms
winShiftMs=10;  % should be 10ms
silenceEndMs=100;
backoffMs=250;

winSize=winSizeMs*fs/1000;
winShift=winShiftMs*fs/1000;
silenceEnd=silenceEndMs*fs/1000;
mywindow=hamming(winSize);

signedSignal=sign(signal);
j=1;
for i=1:winShift:length(signal)-winSize
  energy(j)=(sum((abs(signal(i:i+winSize-1))).*mywindow));
  zc(j)=sum(abs(signedSignal(i+1:i+winSize)-signedSignal(i:i+winSize-1)));
  j=j+1;
end
silenceRange=1:length(1:winShift:silenceEnd-winSize);

IF=(((25*winSizeMs)/10)/10000)*fs;
IZC=mean(zc(silenceRange));
zcstd=std(zc(silenceRange));
IZCT=min(IF,IZC+2*zcstd);



IMX=max(energy);
```

```
IMN=mean(energy(silenceRange));
I1=0.03*(IMX-IMN)+IMN;
I2=4*IMN;
ITL=min(I1,I2);
ITU=5*ITL;

N1=0;
N2=0;

% get end points

duration=length(energy);
backoffLength=length(1:winShift:((backoffMs*fs)/1000)-winSize);

done=0;

% start estimation

for m=1:duration
  if and(energy(m)>=ITL,~done)
    for i=m:duration
      if energy(i)<ITL
        break
      else
        if energy(i)>=ITU
          if ~done
            N1=i-(i==m);
            done=1;
          end
          break
        end
      end
    end
  end
```

```
end
startID=max(N1-backoffLength,1);
endID=N1;
M1=sum(zc(startID:endID)>=IZCT);
if M1>=3
  for i=startID:endID
    if zc(i)>=IZCT
      N1=i;
      break;
    end
  end
end

done=0;

% end estimation

for m=duration:-1:1
  if and(energy(m)>=ITL,~done)
    for i=m:-1:1
      if energy(i)<ITL
        break;
      else
        if energy(i)>=ITU
          if ~done
            N2=i+(i= =m);
            done =1;
          end
          break
        end
      end
    end
  end
end
```

```
startID=N2;
endID=min(N2+backoffLength,length(zc));
M2=sum(zc(startID:endID)>=IZCT);
if M2>=3
  for i=max([1 startID]):endID   % max added by MPC to prevent negative indices
    if zc(i)>=IZCT
      N2=i;
      break;
    end
  end
end

warpRatio=round(length(signal)/length(energy));
N1=N1*warpRatio;
N2=N2*warpRatio;
```

4.2.3 Feature Extraction

The theoretical explanation of LPC method was given in section 3.8. Here the source code which prepares the required procedure is given. Also framing, windowing and overlapping functions are included as discussed in sections 3.5, 3.6, 3.7.

```
function [feat] = feat(x)

n=floor(length(x)/120);
if 360+((n-1)-1)*120 >length(x)
n=n-1;
else
end
nn=360;
w =.54 -.46*cos(2*pi*(0: nn-1)'/(nn-1));
i=0;
j=0;
```

```
for i=1:n-1
x1=0;
k=0;
for j=1+(i-1)*120:360+(i-1)*120
k=k+1;
x1(k)=x(j);
end
x1=x1'.*w;
a(i,:)=lpc(x1,12);
b1=a(i,:);
feat(i,:)=b1;
end

dif=abs((n-1)*120-length(x));
x1=0;
k=0;
j=0;
for j=(n-1)*120+1:dif+120*(n-1)
k=k+1;
x1(k)=x(j);
end
w =.54-.46*cos(2*pi*(0:dif-1)'/(dif-1));
x1=x1'.*w;
a(n,:)=lpc(x1,12);
b1=a(n,:);
feat(n,:)=b1;
```

4.2.4 Reference Pattern

The calculated feature vectors are stored in database for recognition action. In this section the required procedure for recording the calculated vectors to database is given.

```
N1=0; N2=0; i=0; j=0; z=0; zz=0;

clear
clc

% record sound file

[FileName,PathName] = uigetfile('*.wav','Load the Wav File for Data Base');
if FileName==0
   errordlg('Loading is cancelled by user','File Error');
else

[z]=wavread(sprintf('%s\%s',PathName,FileName));

% prefilter

z=z-mean(z);
pre=[1 -15/16];
z=filtfilt(pre, 1, z);

% find start and end points

[N1,N2]=epd(z);

zz=z(N1:N2);

% Count the number of recorded files and record their names
```

```
cnt=0;
if exist('C:\MATLAB6p5\work\THESIS\count.dat','file')
[cnt,FN]=textread('C:\MATLAB6p5\work\THESIS\count.dat','%d %s\n');
[cnt,name]=size(cnt);
cnt=cnt+1;
else
   cnt=1;
end

fp2=fopen('count.dat','a+');
fprintf(fp2,'%d %s\n',cnt,FileName);
fclose(fp2);

% create reference pattern

[R_feat]=feat(zz);

% add this reference pattern to database

fp3=fopen('reference.dat','a+');

[q,q1]=size(R_feat);
clc
fprintf(fp3,'%d %f %f\n',cnt,q,q1);
for i=1:q
for j=1:q1
fprintf(fp3,'%f ',R_feat(i,j));
end
fprintf(fp3,'\n');
end

fclose(fp3);
end
main;
```

4.2.5 Test Pattern

The same action calculated for reference word till here is done for the test word also. The source code below is the code for this operation.

```
function [kod,dist]=test(x,cnt)

[T_feat]=feat(x);
T_feat=T_feat';

kod(cnt)=0;
dist(cnt)=0;
fp1=fopen('reference.dat','r');
i=0;

for i=1:cnt
q=0;
q1=0;
q2=0;
q=0;
R_feat=0;
q2=fscanf(fp1,'%f',[ 1,1 ]);
q=fscanf(fp1,'%f',[ 1,1 ]);
q1=fscanf(fp1,'%f',[ 1,1]);
R_feat=fscanf(fp1,'%f',[q1,q]);
SM = sm(R_feat,T_feat);
[p,q,C] = dtw(1-SM);
tcost=C(size(C,1),size(C,2));
dist(i)=tcost;
kod(i)=q2;
end

fclose(fp1);
```

4.2.6 Similarity Matrix

The source code written below is a similarity matrix calculation which computes normalized inner products between two vectors.

```
function M = sm(A,B)

% calculate a matrix between feature matrices A and B.
% size(M) = [size(A,2) size(B,2)]; A and B have same # rows.

EA = sqrt(sum(A.^2));
EB = sqrt(sum(B.^2));
M = (A'*B)./(EA'*EB);
```

4.2.7 Dynamic Time Warping

We need to use dynamic programming to find a min-cost path through calculated similarity matrix M as discussed in Section 3.10.The source code of this program which prepares the required procedure is given.

```
function [p,q,D] = dtw(M)

[r,c] = size(M);

% costs
D = zeros(r+1, c+1);
D(1,:) = NaN;
D(:,1) = NaN;
D(1,1) = 0;
D(2:(r+1), 2:(c+1)) = M;


phi = zeros(r,c);
```

```
for i = 1:r;
  for j = 1:c;
    [dmax, tb] = min([D(i, j), D(i, j+1), D(i+1, j)]);
    D(i+1,j+1) = D(i+1,j+1)+dmax;
    phi(i,j) = tb;
  end
end

% Traceback from top left
i = r;
j = c;
p = i;
q = j;
while i > 1 & j > 1
  tb = phi(i,j);
  if (tb = = 1)
    i = i-1;
    j = j-1;
  elseif (tb == 2)
    i = i-1;
  elseif (tb == 3)
    j = j-1;
  else
    error;
  end
  p = [i,p];
  q = [j,q];
end

% Strip off the edges of the D matrix before returning
D = D(2:(r+1),2:(c+1));
```

4.2.8 Recognition

Among those calculated distances in previous sections, the word with smallest distance result is displayed as recognized word. The required source code of this algorithm for finding smallest distance among the recorded words is given.

```
time = cputime;

N1=0; N2=0; cnt=0; dist=0; dist1=0; i=0; iii=0; ikod=0; j=0; kod=0; kod1=0;
mdist=0; name=0;

pre=0;z=0;zz=0;

% Check if there is any file loaded in Database

if exist('C:\MATLAB6p5\work\THESIS\count.dat','file')

[cnt,FN]=textread('C:\MATLAB6p5\work\THESIS\count.dat','%d %s\n');
[cnt,name]=size(cnt);

[FileName,PathName] = uigetfile('*.wav','Select the Wav File for Recognition');

if FileName==0
   errordlg('Loading is cancelled by user','File Error');
else

[z]=wavread(sprintf('%s\%s',PathName,FileName));

% prefilter

z=z-mean(z);
pre=[1 -15/16];
z=filtfilt(pre,1,z);
```

```
% find start and end point

[N1,N2]=epd(z);

zz=z(N1:N2);

% recognition

[kod,dist]=test(zz,cnt);
[mdist,ikod]=min(dist);

if mdist ==0
jjj=0;
for iii=1:cnt
if dist(iii)~0
jjj=jjj+ 1;
dist1(jjj)=dist(iii);
kod1(jjj)=kod(iii);
else
end
end
else
kod1=kod;
dist1=dist;
end

[mdist1,ikod1]=min(dist1);
rec_word=kod1(ikod1);


for i=1:cnt
if rec_word == i
      msgbox({['Recognized word is ',sprintf('%s',FN{i,1})]},'Speech Recognition',
'help', 'replace');
```

```
else
end
end

cputime-time;
end
main;
else
   msgbox('There is no data in Data Base for comparison...')
   main;
end
```

4.3 Experiment and Results

The proposed theory in the previous chapters are applied to assess the performance of the speech recognition system for random Turkish words “pencere”, “kitaplık”, “bilgisayar”, “ilaç”, deneme”, çiçek”, “araba”, “gözlük”, “resim”, “kedi”. These 10 words are recorded from 10 different speakers.

The analog speech signals sampled at 8 kHz and digitized into 8 bits are stored in PC by using sound card. The speech signals in the digital form are pre-emphasized by first order digital filter with zero at z = 15/16.

The recognition system performance depends on the accuracy of the identification of the end points of the speech signal. So, after framing and windowing process, the starting and end points of the signal are found by comparing the energy of the signal with the threshold level. Then, the zero crossing rate is computed to take several more samples from the beginning point and from the end point.

The signals which are pre-emphasized and cleared from silence regions are windowed in the form of 45msec duration frames by using Hamming window. The subsequent frames are applied at each 15msec apart. The durations are chosen to make the signals in these intervals stationary signals.

The procedure explained above is executed for all words and the resultant feature vectors are stored into the computer to build up the reference data base.

And also the same procedure is repeated for 10 speakers for all words and they are used to test the recognition algorithm.

The result of this experiment can be seen below in Table 4.1. In the table, the numbers show recognition results. Matching and non-matching words can be seen.

Table 4.1 Result of Recognition between Reference Pattern and Test Pattern

	REFERENCE PATTERN										
		PENCERE	KİTAPLIK	BİLGİSAYAR	İLAÇ	DENEME	ÇİÇEK	ARABA	GÖZLÜK	RESİM	KEDİ
TEST PATTERN	PENCERE	9					1				1
	KİTAPLIK		7					1			
	BİLGİSAYAR			6							
	İLAÇ			2	8						
	DENEME	1		1	1	9			2		
	ÇİÇEK				1		6			1	
	ARABA		3			1	1	9			
	GÖZLÜK								7		
	RESİM			1					1	7	
	KEDİ						2			2	9

CHAPTER 5

CONCLUSION AND FUTURE WORKS

In this project speech recognition is examined. Recently many studies focused on the speech recognition techniques presented different techniques and different accuracies. In [18], a speech recognition system is designed to control industrial robot manipulator with 20 different word comments. Of course different combinations of signal enhancement, speech coding, and speech recognition techniques may result with different accuracies. And also the type of the word, which will be recognized, is important. The algorithm presented in this work is based on teach and decide method. So the algorithm works not only for Turkish words but also for any other language. One disadvantage is that the words must be recognizable.

Linear Prediction Coding is choosed as a representation for the spectrum of a speech signal, and a template set is generated. So all words are represented with less number of samples which reduces the comparison time. Comparing to other techniques such as FFT and power spectral approach both in literature and in this work, it has been observed that LPC is the best coding method. This does not mean that the new coding methods will not be improved.

Next, Dynamic Time Warping algorithm is used to search for the least cost matching path between a test signal and a template. At this step of algorithm HMM may be used. But since DTW works with the base of matrix operation and since matrix algebra is always faster then the iterative techniques it has been prefered. Unfortunately the optimization of time and accuracy pairs are not satisfied in the same method.

The system was implemented in MATLAB. The program is tested by 10 different speakers. 10 different words are recorded by those 10 different speakers which results 100 data signals generating the template set. And the accuracy is tested to be

77 %. This result is affected by experimental nuances such as the distance the speaker is from the micro-phone during data collection, the noise, microphone and data acquisition card used for sound recording, and the fluency, loudness and sharpness of the word spoken.

During the development of this project, understanding of the theory involved in a DTW speech recognition system is learned. Besides the dynamic time warping algorithm, linear prediction modeling is examined. This project can be a good beginning to the development of tools for speech recognition.

For future work; better results can be taken by trained speakers with noise cancellation programs. Alternative methods may be searched for beginning and end point detection algorithm which affects also the accuracy. The sampling frequency and resolution may be higher and other LPC orders may be used which requires extra hard disk and memory capacity. The other computer languages such as C++, assembly language may decrease the computation time.

REFERENCES

[1] M.H. Savoji, "A Robust Algorithm for Accurate Endpointing of Speech Signals." Speech Communication-89, pp.45-60, 1989

[2] B. Gold and N. Morgan, Speech and Audio Signal Processing. New York: John Wiley & Sons Inc., 2000.

[3] J. R. Deller, J. H. L. Hansen, and J. G. Proakis, Discrete-Time Processing of Speech Signals, vol. IEEE Press, Second ed. New York, 2000.

[4] L. Rabiner and B.H. Juang, Fundamentals of Speech Recognition, Prentice-Hall, Englewood Cliffs, New Jersey, USA, 1993.

[5] Ergün ERÇELEBİ A.Ph. D. Thesis "Computer Aided Design of an Adaptive Filter And Its Applications to Signal Processing, 1999"

[6] Rabiner, L. R., S.E. Levinson, A. E. Rosenberg et al. "Isolated and Connected Word Recognition Theory and Selected Applications" IEEE Transactions on Communications, vol.29, pp.621-659, May 1981.

[7] Hansen, J. H.L , and M.A.Clements, "Stress Compensation and Noise Reduction Algorithms for Robust Speech Recognition", Proceeding of *The IEEE International Conference on Acoustics, Speech, and Signal Processing, Glasgow-Scotland*, vol.1 pp. 266-269, 1989.

[8] Rabiner, L. R., M. R. Sambur, "An Algorithm for Determining the Endpoints of Isolated Utterances", *The Bell System Technical Journal*, Vol. 54, No. 2, pp. 297-315, 1975.

[9] Sima, M., V. Croitoru, D. Burileanu, "Performance Analysis on Speech Recognition Using Neural Networks", Proceedings of the International Conference on Development and Application Systems, Suceava, Romania, pp. 259-266, 1998.

[10] L.R Rabiner and R.W. Schafer, *Digital Processing of Speech Signals*, Prentice-Hall, Englewood Cliffs, N.J., 1978.

[11] F. Itakura, "Minimum Prediction Residual Principle Applied to Speech Recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 23, no. 1, pp. 67-72, February 1975

[12] Jackson, L.B., Digital Filters and Signal Processing, Second Edition, Kluwer Academic Publishers, 1989. pp. 255-257.

[13] Orfanidis, S.J., Optimum Signal Processing. An Introduction. 2nd Edition, Prentice-Hall, Englewood Cliffs, NJ, 1996.

[14] Ljung, L., System Identification: Theory for the User, Prentice-Hall, 1987, pp. 278-280.

[15] F.K. Song, A.E. Rosenberg and B.H. Juang, "A vector quantisation approach to speaker recognition", *AT&T Technical Journal*, Vol. 66-2, pp. 14-26, March 1987.

[16] H.F. Silverman and D.P. Morgan, "The Application of Dynamic Programming to Connected Speech Recognition," IEEE ASSP Magazine, pp. 7-25, July 1990.

[17] J.R. Deller, J.G. Proakis, J.H.L. Hansen, Discrete Time Processing of Signals, MacMillian Publishing Co., New York, New York, USA, 1993.

[18] DOGAN H.Sair, "Vocal Control of An Industrial Robot Manipulator", University of Gaziantep, January 1995