SPEECH RECOGNITION SYSTEM

by

MILAN G. MEHTA, B.S.E.

A THESIS

IN

ELECTRICAL ENGINEERING

Submitted to the Graduate Faculty
of Texas Tech University in
Partial Fulfillment of
the Requirements for
the Degree of

MASTER OF SCIENCE

IN

ELECTRICAL ENGINEERING

Approved

August, 1996

ACKNOWLEDGEMENTS

I wish to express my sincere gratitude to Dr. Micheal Parten, my graduate advisor, for his guidance, valuable suggestions, encouragement, and moral support throughout the period of my study and research at the Texas Tech University. I am grateful to Dr. Thomas Krile and Dr. Mary Baker for serving as members of my thesis committee. I would like to thank the Department of Electrical Engineering for providing financial assistance throughout my graduate studies. I appreciate the support and encouragement offered by my friends. I am most grateful to my parents and my family, for their support during my graduate studies.

TABLE OF CONTENTS

ABSTRACT

Automatic Speech Recognition (ARS) has progressed considerably over the past several decades, but still has not achieved the potential imagined at its very beginning. Almost all of the existing applications of ASR systems are PC based. This thesis is an attempt to develop a speech recognition system that is independent of any PC support and is small enough in size to be used in a daily use consumer appliance. This system would recognize isolated utterances from a limited vocabulary, provide speaker independence, require less memory and be cost-efficient compared to present ASR systems. In this system, speech recognition is performed with the help of algorithms such as Vector Quantization and Zero Crossing. Several features of a Digital Signal Processor (DSP) have been utilized to generate and execute the algorithms for recognition. The final system has been implemented on Texas Instruments TMS320C30 DSP. The system, when implemented using the Vector Quantizer approach, achieved an accuracy of 94% for a vocabulary of 6 words and a recognition time of 6 seconds. The zero crossing approach resulted in an accuracy of 89% for the same vocabulary while the recognition time was 0.8 seconds.

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Introduction

Technology has increased (and will continue to increase) the rate at which data is processed by machines in our lives. These machines include computers, televisions, microwave ovens and many others. As the data processing rate increases, the machines begin to wait for human data input, rather than humans waiting for the machines to respond.

There are many input technologies available today that include key pad units (keyboards, remote controls, or membrane switch matrices) and pointing devices (mice, joysticks, or digitizing tablets). However, most of these devices have slow data input rates. Consequently, as machines gain features and data processing power, they spend less time working and more time waiting for human input.

One technology that has been in existence for some time, but still has limited application is Automatic Speech Recognition (ASR).

1.2 Speech Recognition

In the last two decades, attempts have been made to automate the recognition of human speech. The term "Speech Recognition" is one that covers many different approaches to the problem of recognizing human speech. It ranges from isolated word

recognition to continuous speech recognition, from speaker-dependent recognition to speaker-independent recognition, and from a small vocabulary to a large vocabulary. The simplest scenario is speaker-dependent, isolated word recognition on a small vocabulary and the most complex is a speaker-independent, continuous speech recognition on a large vocabulary. In any case, the speech recognition problem, as developed over the years, is a highly computation intensive problem; it requires fast processors, and a large amount of memory. Many attempts have, therefore, been made to speed up the process by using various techniques. Thus, the implementation of speech recognition systems using digital signal processing integrated circuits (DSP's) is becoming increasingly attractive. The advantages of using a DSP are, increase in speed of the algorithm development, long term reliability, noise immunity and the ability to perform complex computations unthinkable in the analog domain.

1.3 Statement of the problem

1.3.1 Description of the project

This project examines known speech recognition techniques (specifically template matching and zero-crossing methods) and applies new processing power and data storage to the execution of the techniques. An attempt has been made in this project to develop a Speech Recognition card which is small in size, cost efficient, and can perform fast recognition so that it can be used in real time applications. The system has to be small enough to fit into a small appliance like a car radio or a microwave. This system is based

on the template based technique for recognition. The two techniques that are used for recognition in this project are Vector Quantization (VQ) and Zero Crossing. For speech recognition using VQ, Linear Predictive Coding (LPC) analysis has to be performed on the set of input utterances. Once the LPC coefficients are determined, the VQ algorithm generates a codebook that is a representation of the word. This codebook is used as a lookup table during speech recognition. In speech recognition using the zero-crossing technique, the number of zero crossings of the input speech within specific intervals of time, are calculated and statistical parameters, such as the standard deviation and the mean are calculated for the zero crossings. The result of this statistical approach leads to speech recognition.

In both the above cases, the algorithm was divided into two steps: a training step and a recognition step. During the training phase, speech signals were recorded from different people, and a reference utterance (template) library was built from this recorded data. During the recognition step, the recognized utterance was compared with each utterance in the reference library, and a decision was made regarding the identity of the test utterance.

Thus, the aim of this thesis is to develop a speaker independent, isolated word, limited vocabulary speech recognition system that is small enough to fit in a small household appliance and can be operated in real time. Having indicated the requirements for the speech recognition system, the major objectives of this project can be defined.

1.3.2 Objectives of the project

The main objectives of this thesis are as follows:

- To develop and test a speaker independent Speech Recognition system via two approaches, the first one being the Template Matching approach using the VQ and LPC algorithms while the second approach is the Zero Crossing technique. The system is to be built using a TMS320C30 Digital Signal Processor application development board.
- To evaluate the performance of the system in terms of computing time, disk, storage and word accuracy.

The next chapter will provide insight into the various Speech Recognition techniques and the choice of the appropriate technique for the present SR system. Chapter 3 gives an overview of the entire SR system and describes each stage of the system in particular. Chapter 4 explains the two techniques used for the purpose of Speech Recognition in the present system. After explaining the techniques for SR in Chapter 4, Chapter 5 describes the implementation of the system and evaluates the systems performance based on the results from the two recognition techniques. The final chapter provides the results and conclusions for the entire system and suggests further improvements in the system.

CHAPTER 2

SPEECH RECOGNITION TECHNIQUES

The first step in solving the speech recognition problem is to understand its complexity. There are four basic components to be considered in understanding the operation of a speech recognition system. First, the recognition system must have some form of encoding and representing a set of utterances that it will recognize. Second, during recognition, there must be some type of pattern-matching algorithm that compares a representation of a particular input utterance with the representations in the known vocabulary. Third, given the comparison of some input with the vocabulary, there must be an algorithm that decides which utterance in the vocabulary was produced. Finally there must be some kind of user interface to the functions and operation of the recognition system. The goal of this thesis is to develop a card that is small in size, low in cost and, if used in some of the daily use appliances, such as a car radio or a stereo system, should perform the function of real time speech recognition with high accuracy. To develop such an SR system, an insight into the speech analysis process is needed. In addition, different features of an SR system that can affect the overall performance of the system must be examined.

2.1 Characteristics of Speech Recognition Systems

There are a number of ways to classify speech recognition systems. A very important classification is whether the system is speaker independent or speaker dependent [26]. Speaker-dependent systems recognize a particular person's utterances only if that person has previously stored examples of his or her speech in the system. Speaker-independent systems recognize speech without prior experience with a particular person.

Another difference among systems is whether they accept discrete, connected, or continuous speech. Isolated word recognizers require that the speaker utters the word with pauses between each word. This is adequate for many applications but is far from being a natural way of communicating. Systems that recognize connected speech accept a sequence of concatenated citation-form words. Some words are separated by pauses while some are connected. Thus the task of spotting the beginning and ending of words becomes important for the recognizer. Finally, in continuous speech recognition, the speaker talks naturally and the system recognizes strings of words. This system is more complex than the previous system, because, the allowable strings of words would have to be defined either by an artificial rule set designed for the application [38], or by an attempt to model the grammar of a natural language [10]. These rules or models usually render the system slower than its discrete counterpart [10]. Finally, systems can differ depending upon the size of the vocabulary. Some system can recognize only a small set of words [24] (as few as ten digits), while other can recognize larger sets of words, even tens of thousands [26].

2.2 Applications of Speech Recognition Systems

One of the major applications of ASR is in telecommunications. One use is in cellular phones whereby a person can dial a number by just speaking the name of the person he wishes to call [38]. Other telephone applications include automatic operator-assisted calls (without a human operator) and replacement of touch-tone input with speech input(for banking, airlines, etc.). These applications are most often vocabulary limited, but offer speaker independence [29]. One of the fastest growing areas of applications is in system [29] control. As the electronic content of devices in our work and our lives increases, ASR seems a natural solution. Many applications such as automatic data retrieval are designed for use in personal computers. One such system is the IBM Personal Dictation System (IDPS). It is a speaker dependent, discrete speech recognition system. The system is available with dictionaries for subjects like medicine, radiology, and journalism having vocabulary sizes ranging from 16,000 to 30,000 words [6]. The system requires a 486 PC along with an adapter card with a Digital Signal Processor (DSP).

The development of any SR system, including the ones discussed above, requires software, along with hardware. The software for an SR system includes algorithms to perform the preprocessing of the speech waveform and those to carry out analysis on the speech data and perform the classification. The hardware of the system might be a PC, a mini-computer, a mainframe, or a DSP. The features of several systems with different software and hardware specifications are discussed in the next section.

2.3 Algorithms in Speech Recognition

There are a lot of approaches to speech recognition. Algorithms like zero crossing, energy measurement and feature extraction are based on the acoustic-phonetic approach. Algorithms such as template matching come under the pattern recognition approach, while algorithms that depend on knowledge sources, stochasticity of speech signals and neural networks are based on the artificial intelligence approach. However, an important approach to speech recognition is stochastic modeling, in particular stochastic modeling using Hidden Markov Models [22] in conjunction with the Viterbi algorithm. Among these the most popular and accurate algorithm is the template based Dynamic Time Warping [27].

2.3.1 Zero-Crossing and Energy-based Speech recognition

As compared to other approaches, zero-crossing and energy-based recognition systems require far fewer computations and fewer sets of parameters [12]. For the purpose of recognition, sets of parameters are obtained from several frames of speech data. Parameters such as average zero-crossing rate, density of zero-crossings within frames, excess threshold duration, standard deviation of the zero-crossing within frames, mean zero-crossing within frames, and energy estimates for each frame have all been used. These parameters are then compared with fixed thresholds to determine the spoken word. For example, the zero-crossing rate at the beginning of words starting with strong fricatives are higher than for words starting with weak consonants. This classifies a set of

words into two groups and makes the process of recognition easier. A system developed by Chok-ki Chan [30] is based on the parameters such as zero crossing and energy. It is a speaker dependent, isolated Cantonese digit and words, limited vocabulary SR system, developed and implemented on a PC-386, with a recognition accuracy of 97.2%. This system worked reasonably well for isolated digits but when tested for isolated words, the accuracy dropped to 76% [30]. One more example of such a system is the one developed by Chan Y. T. [29] on an MC68000 microprocessor based system. It was a speaker-independent, isolated-word, limited vocabulary SR system. The system when operated in speaker-dependent mode had an accuracy of 87%, but when operated in speaker independent mode, the accuracy dropped down to 78%. The system when developed for a 10 word vocabulary, took around 6 minutes [29] to recognize a word but with inclusion of every 3 words in the vocabulary the recognition speed dropped by 40%.

2.3.2 Feature-Dependent Speech Recognition

Feature-dependent speech systems are based on principles of human speech perception. These systems try to mimic human performance in recognizing speech. Some of the features that are used in such systems are: frequency location of the first few formants, the maximum and minimum frequency of the first few formants, duration of aperiodic energy, formant transitions and the ratio of high frequency energy to low frequency energy. To obtain the above sets of features, parameters such as the spectrum, pitch, zero crossings, total energy, energy in low, mid, and high frequency bands are

produced using signal processing routines. The decision structure is arrived at by first selecting a set of measures that are likely to provide a clean grouping of letters into a set of classes. A recursive clustering algorithm is used to determine the optimal grouping of letters into clusters until the errors are minimized. An example of such a system is, FEATURE, a speaker-independent isolated letter recognition system [31]. This system performs recognition on an 80 letter vocabulary generated by 20 different speakers. The recognition accuracy of 85% was obtained across 20 speakers for a speaker-independent mode while an accuracy of 91% was observed when operated in dynamic adaptation mode.[1] The different algorithms for the system were implemented on a Fast Processor Array, Motorola 68000 microprocessor, and VAX11/750.

2.3.3 Template-Based Speech Recognition

Template-based speech recognition systems have a database of prototype speech patterns (templates) that define the vocabulary. The generation of this database is performed during the training mode. During recognition, the incoming speech is compared to the templates in the database, and the template that represents the best match is selected. Since the rate of human speech production varies considerably, it is necessary to stress or compress the time axes between the incoming speech and the reference template. This can be done efficiently using Dynamic Time Warping (DTW). In a few algorithms, like Vector Quantization (VQ), it is not necessary to vary the time axis for each word,

[1] In dynamic adaptation mode the user provides feedback when an error is made, and the system changes the statistical parameters that are used during the classification.

even if any two words have different utterance length. This is performed by splitting the utterance into several different sections and coding each of the sections separately to generate a template for the word. Each word has its own template, and therefore this method becomes impractical as the vocabulary size is increased (> 500 words). Itakura of NTT [2] developed an SR system with the concept of TW for non-linear alignment of speech. The system performed a speaker-dependent isolated-word recognition task with a 97% accuracy on a 200-word vocabulary. Performance of this system degraded to 83%, when the vocabulary size was increased to 1000 words. A template-based system for connected speech was implemented by H. Sakoe [25]. It was a speaker-dependent connected-digit large vocabulary speech recognition system that had an accuracy of 99.6%. It was implemented on a NEAC-3100 computer, with a memory requirement of 10 Kbytes per word, and had a vocabulary of 600-2500 words depending on the training set [25]. This system performed well for speaker-dependent speech recognition, but one of the shortcomings of the system was that it took more than 150 minutes [25] to recognize a word with a vocabulary size of 600. For the same set of parameters, a VQ based recognizer would require around 250 Kbytes to 1.2 Mbytes of memory space. This represents a 130:1 reduction for the VQ recognizer over the DTW model.

In some cases of template matching, no time alignment is needed to perform the recognition, as in the case of the system developed by J. E. Shore et.al. [31]. It was a speaker independent, isolated word, limited vocabulary SR system, implemented using a VQ approach. It had an accuracy of 98% for speaker dependent recognition while for

speaker independent recognition the accuracy reduced to 85%. The system was implemented on a DEC VAX11/750 with floating point accelerator. Template generation required 20-22 minutes, while classification of a single utterance with these templates took about 1-1.5 minutes.

2.3.4 Knowledge-Based Speech Recognition

Knowledge-based speech recognition systems incorporate expert knowledge that is, for example, derived from spectrograms, linguistics, or phonetics. The goal of a knowledge-based system is to include the knowledge using rules or procedures. The drawback of these systems is the difficulty of quantifying expert knowledge and integrating the multitude of knowledge sources [24]. This becomes increasingly difficult if the speech is continuous, and the vocabulary size increases [24]. A good example of a knowledge-based system is HEARSAY, developed on a PDP-11 microcomputer, at Carnegie-Mellon University. It was a speaker-dependent continuous-speech recognition system with a vocabulary of 1011 words. Using a very restrictive syntax (perplexity[2] 4.5), it achieved a recognition accuracy of 87% [23]. It took 13 hours to compile the algorithm developed for HEARSAY. Fifty knowledge sources were used to develop this system and the amount of storage required for each one was about 80 Kbytes. Comparing this storage requirement with that needed for the Vector Quantization algorithm shows a reduction

[2] Perplexity Q is an information theoretic measure of a task's difficulty. It is average number of possible words following a word.

[34] of the order of 30 or more. This would mean far less memory storage would be required for codebooks generated by VQ approach than by the knowledge sources in the above approach.

2.3.5 Stochastic Speech Recognition Systems

Speech recognition based on stochastic modeling is another approach for SR systems. Probabilistic models of speech are used in this approach to deal with incomplete information or uncertainty. The most widely used model is the Hidden Markov model [1] (HMM). The HMM uses states that model generic speech sounds and transitions between states with associated transition probabilities to model the temporal behavior of speech. This model assumes that speech was produced by a hidden markov process. To derive the transition probabilities, an efficient estimate-maximize algorithm, the forward-backward algorithm, is often used [22]. Though the HMM approach can give substantially accurate results, if the time factor is taken into consideration, then algorithms based on template matching using Vector Quantization or DTW prove to be much faster than the ones based on HMM [1]. An example of a system based on the HMM approach is the one developed by Rabiner et al.[4] at Bell Labs. The system used the techniques of Vector Quantization in conjunction with HMM in a speaker-independent, isolated-word recognition system. It achieved an accuracy of 96.3% word accuracy for a 10-digit vocabulary. The system was implemented on a SUN-workstation. Despite the fact that the system could recognize words from any speaker, the system had a drawback, which was

the large amount of time it took to train the system to a given set of words. For the given case, 10 digits took more than 15 hours to train the system [4]. Another example of a system based on HMM and statistical methods [20] is SRI's DECYPHER system, which has a 1,000 word vocabulary, is speaker independent, accepts connected speech input, and has a recognition accuracy of 95.6%. Since DECYPHER is a connected speech recognition system, the recognition algorithm involves more computation than a discrete recognition system (by a factor of 3 [35]), and real time performance is more difficult to achieve. In this system, a statistical bigram grammar is used that reduces perplexity to 60 (without grammar, any word can follow a given word). Using this grammar, the recognition accuracy for DECYPHER improves from 75.5% (no grammar) to 95.6%. This system was implemented on a microcomputer and required more than 16 Mbytes of workspace to perform calculations [20].

2.3.6 Connectionist Speech Recognition Systems

Connectionist speech recognition is based on artificial neural networks that use learning strategies to organize and optimize a network of processing elements (neurons). These networks are used as classifiers or mapping functions to recognize the incoming speech. Thus the speech knowledge or constraints used for speech recognition are distributed among many, but simple processing elements [17]. The concepts and ideas of applying neural networks to speech recognition are relatively new, and researchers are investigating a number of approaches. An example of a connectionist system developed

for keyword spotting [38] is the one implemented using a time-delay neural network [36]. This speech recognition system, implemented on a PC-486, was compared to an HMM recognizer in the task to recognize the phones "B", "D", "G" out of a database of 5,240 Japanese words. For different speakers, the HMM had a recognition accuracy of 90.9% to 97.2%, while the neural net achieved an accuracy of 97.5% - 99.1% [36]. One of the advantages of neural network-based speech recognition systems is that they can achieve the capability of handling a really large vocabulary [36]. One of the disadvantages of the system, which is the same as that of an HMM-based system, is that a large time may be required for training the system.

2.4 Speech Recognition System

As discussed earlier, most of the commercially available systems have been implemented on either a mini-computer or a mainframe. Some of the SR systems were even implemented on PC's. Most of the above mentioned SR systems required a relatively large amount of memory for storing representations of each word. This means that all these systems depend on the enormous computing power and the large mass storage devices of the computer to process the required information. For some of these systems, large computing power and storage capabilities would mean relatively large capital for development of the system. Moreover, most of these systems were implemented in [2,3,8,10] non-real time. None of the above SR systems are small enough to fit in a daily use consumer product such as office equipment, stereo systems, home appliances, or even

car radio for that matter. Nor are they fast enough for real time applications. The use of a Microcontroller (DSP) could prove helpful in developing a Speech Recognition Card that would be cost-efficient and fast enough to operate in real time. This stand-alone SR system would be small in size, since a DSP does not require several peripherals and numeric processors to generate control signals and perform calculations. These peripherals are a necessity in PC-based systems. Second, the enormous computing capabilities of the DSP would help in making real time speech recognition possible. These features of a DSP could lead to development of a small card that could fit into an appliance like a stereo, microwave oven, or the car radio and perform the function of a Speech Recognition system at a relatively low cost.

An attempt has been made in this thesis to develop a DSP-based speech recognition system which later could be converted into a stand-alone system. One of the goals of this project is to evaluate the performance of this system, which would perform the function of isolated-word, speaker-independent speech recognition on a limited vocabulary. The system is based on the TMS320C30 Digital Signal Processor. One of the advantages of implementing it on a DSP, other than the system being cost-efficient and small in size, would be a tremendous increase in processing speed, which would help make the system real-time. As discussed earlier, this approach of implementing the system on a DSP could represent an increase in efficiency over the PC-based speech recognition systems. Considering the features and limitations of different speech recognition algorithms discussed earlier, an attempt has been made in this thesis to develop and

simulate this ASR system, using two SR algorithms, the *template matching algorithm* in conjunction with *Vector Quantization*, and the *Zero Crossing* technique.

The system has been classified into three functional stages: the *Preprocessing* stage, the *Vector Quantization* stage and finally, the *implementation* stage. The different steps involved in the *preprocessing* stage are as follows: *Data Acquisition*, where, the speech input is captured using a microphone, preconditioned and digitized for further processing; *Endpoint Detection*, where the beginning and endpoints of the digitized signal are detected; *Windowing*, where speech frames are "Windowed", to prevent aliasing and finally; *Feature extraction*, where the spectral parameters of the speech signal are extracted. The next stage implements the *Vector Quantization* algorithm to generate a template. The basic concept of Vector Quantization is that a vector of input (e.g., a segment of waveform) can be encoded into an integer that is associated with an entry of a collection (codebook) of reproduction vectors. The reproduction vector chosen is the one that is closest to the input vector in a specified distortion sense. Finally, the last stage has to do with the *implementation* of the entire system. This implementation is actually an explanation of how the system was developed on a PC, the results of the system, and its implementation on a DSP board. This includes the steps as to how the system was implemented, both on the DSP board and the PC.

The second algorithm used for SR purposes in this thesis is the Zero-Crossing algorithm. In the Zero-Crossing algorithm, the test utterance is divided into sections and an analysis is carried out on the number of zero crossings in each section. This analysis

includes calculation of parameters such as the density and standard deviation of zero crossings in each section The results are then compared to those of the reference utterance to perform classification. A comparison of the results of the two SR algorithms is also highlighted.

The next chapter discusses the preprocessing stage of the system, while the subsequent chapter explains the Vector Quantization algorithm along with the Zero-Crossing technique for SR and its incorporation in this system. The final chapter gives a detailed explanation of the DSP based implementation of the SR system. This chapter also tabulates the results of the SR system and performance benefits of the stand-alone system over the DSP-based system and compares the two SR algorithms used in this project.

CHAPTER 3

IMPLEMENTATION OF A SPEECH RECOGNITION SYSTEM

This chapter describes the development of a speech recognition system based on the template matching technique. The three stages for this system are Endpoint detection, LPC preprocessing, and Vector Quantization. This chapter gives a detailed description of Endpoint detection and LPC preprocessing and gives a brief introduction to Vector Quantization, the major portion of which will be covered in the next chapter.

3.1 Typical Template-Based Word Recognition System

Figures 3.1 and 3.2 show the block diagram and the flow chart, respectively, of a typical template-based word recognition system. The incoming speech is preprocessed to detect the start and end points and the amplitude is normalized. If the recognizer is in the training mode, a suitable representation of the word is extracted and stored as a template for the selected word. If the recognizer is operating in the recognition mode, the input word will be compared to each of the stored templates using a suitable distance metric to determine the best match. If the best match exceeds the decision threshold, the match is considered to have been found, but if the match is less than the decision threshold, the input word is not considered to be any of the stored templates and consequently not recognized.

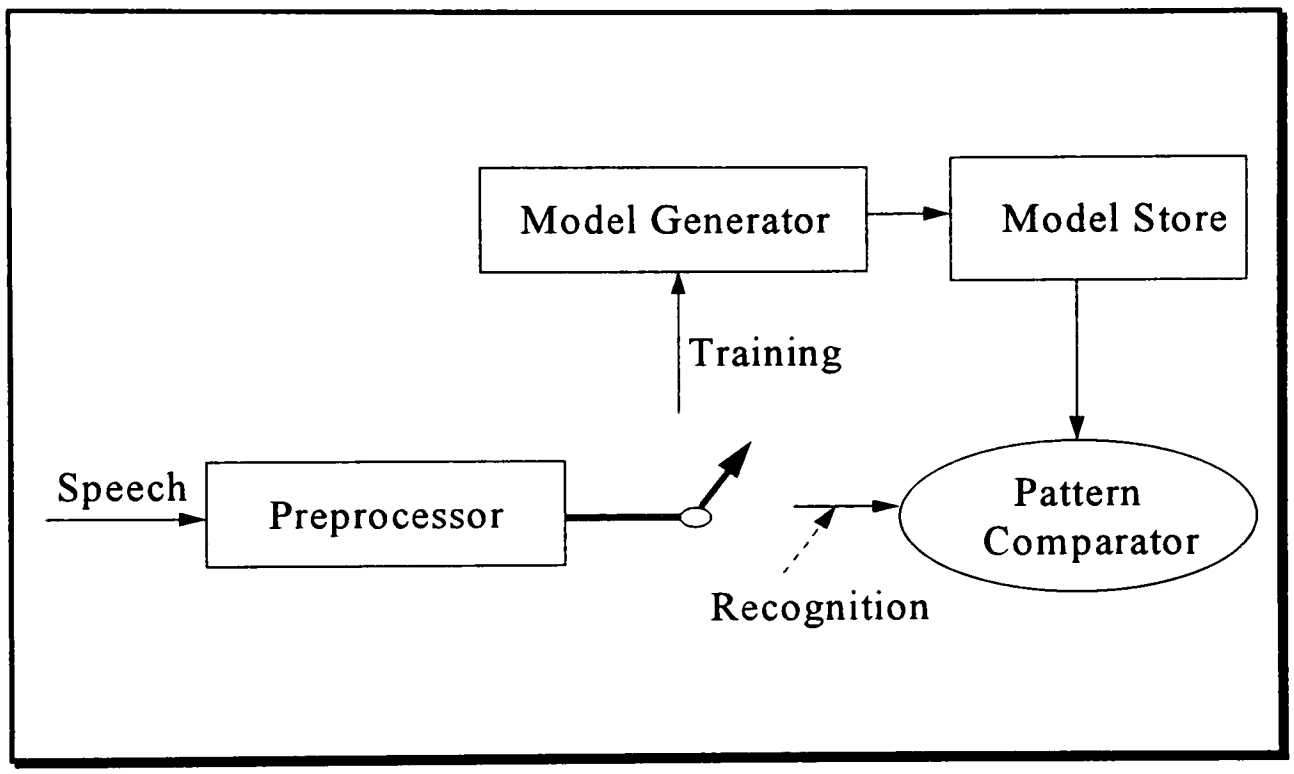


Fig. 3.1 Block diagram of a typical word recognition system

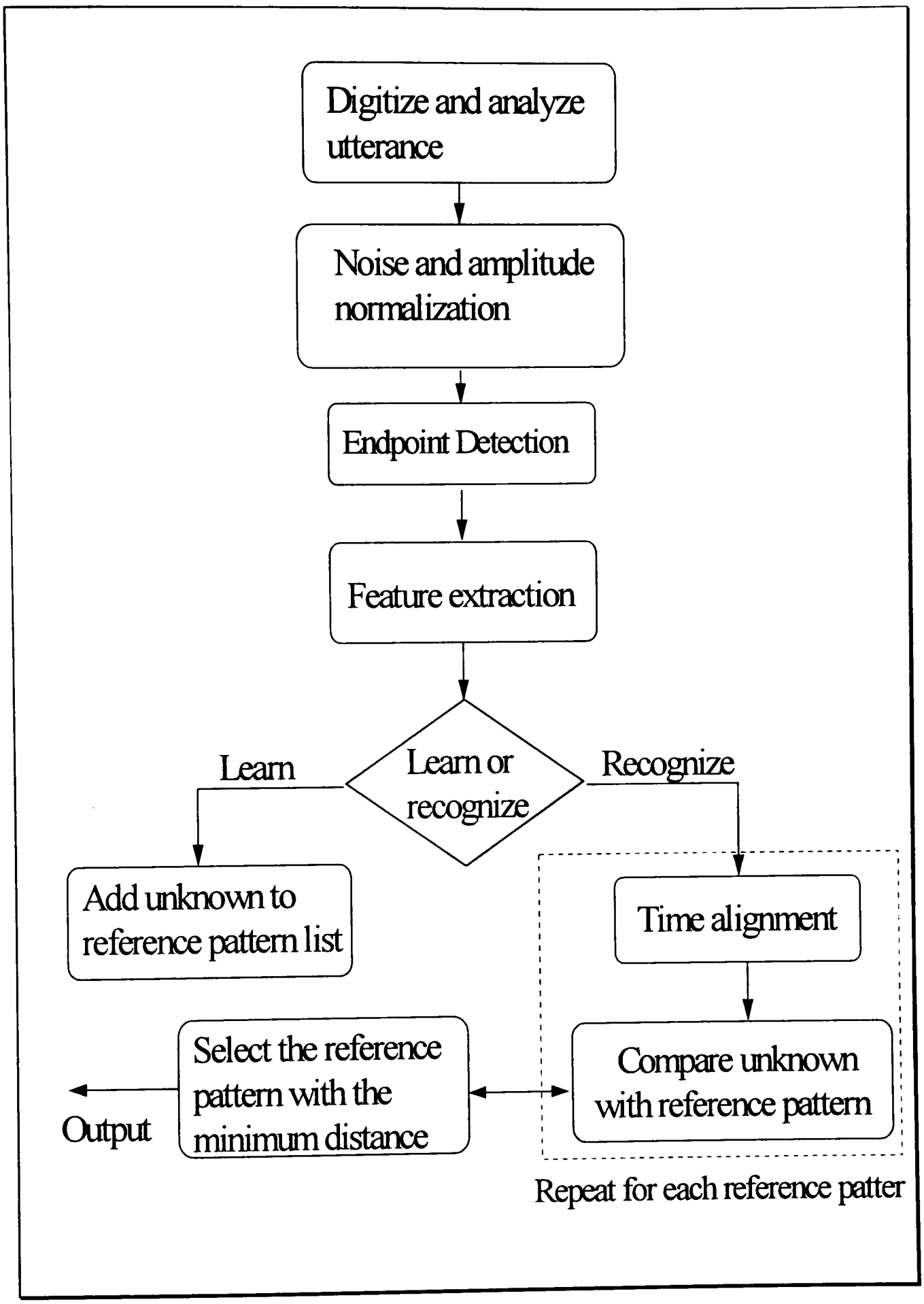


Fig. 3.2 Flow chart for a typical word recognition system

3.2 Endpoint Detection Algorithm

The first task in any speech recognition system is to accurately determine where the word starts and where it ends. Approximate start and endpoints are relatively easily located when the word starts or ends with a strong or plosive consonant like /T/, /K/ or /G/ (e.g., 'tick', 'gate'). A simple thresholding technique using the short term amplitude of the speech signal usually suffices. This means that if the amplitude of the speech signal exceeds a threshold for more than a short period of time, a word start is assumed to have been detected, but if the duration of the sound is short, it is assumed to be noise. Difficulties are encountered when the first or last phoneme of the word has a weak sound such as /F/, /S/, or /H/. When this happens, simple level thresholding is often inadequate, especially when background noise is present. Considering the above facts, the algorithm [17] that is implemented in this system is based on two simple measurements, energy and zero-crossings. Both these parameters are fast and easy to calculate and can give A fairly accurate indication of the presence or absence of speech.

After digitizing and prefiltering the speech data with a bandpass filter, the short-term amplitude sum, A, and zero-crossing, Z, are obtained using an N sample window width. The endpoint detection algorithm appears in Fig. 3.3

$$A = \frac{1}{N}\sum_{i=1}^{N}|x_i| \tag{3.1}$$

$$Z = \sum_{i=1}^{N}\left[1 - \left(\frac{x_i x_{i-1}}{|x_i x_{i-1}|}\right)\right] \tag{3.2}$$

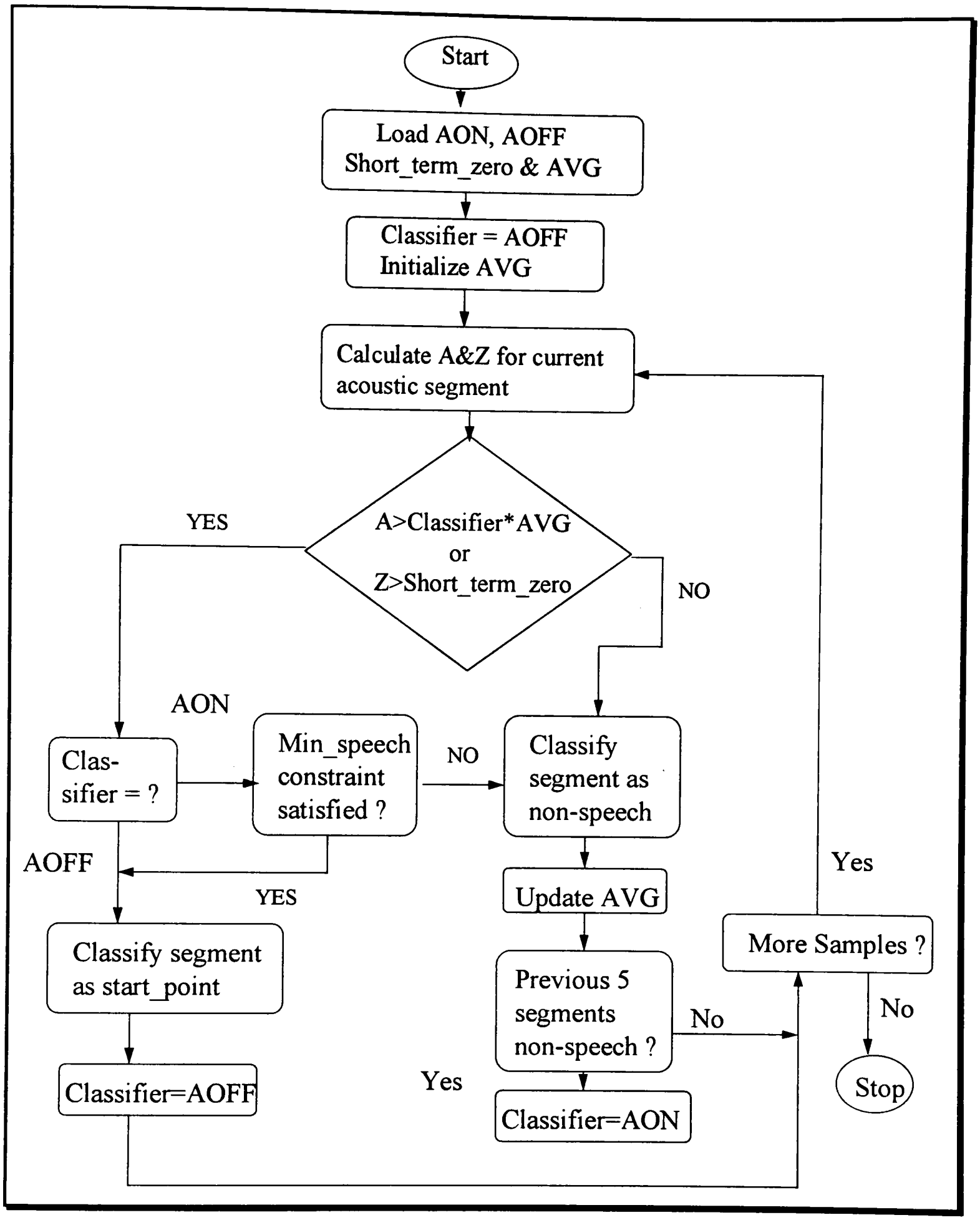


Fig. 3.3 End-point detection algorithm

The parameter, short_time_zero is the zero-crossing rate boundary between speech and non-speech. Min_speech is the minimum number of continuous speech segments needed for any of these segments to be classified as speech. The variable AON is the amplitude threshold multiplier for detecting silence-to-speech transitions. AOFF is the amplitude threshold multiplier for detecting speech-to-silence transitions. To adapt to the background noise variations, the actual amplitude threshold is obtained by multiplying CLASSIFIER (= AON or AOFF) by AVG which is the local average of the 10 most recent non-speech period A values. The first segment with A > Adapt_threshold or Z > Short_time_zero, is classified as speech *only* if the next "Min_speech" segments are also speech. This approach deletes short acoustic burst that might be classified as speech. Following classification of a segment as the start_point, CLASSIFIER = AOFF is initiated. Setting CLASSIFIER = AOFF immediately following a silence prior to a weak stop consonant can result in stop's classification as the end_point since AON > AOFF. Premature activation of AON is thus avoided by requiring 6 continuous silence segments.

3.3 LPC Processor for Speech Recognition

A vast majority of isolated word recognizers use LPC analysis as the front end processing for recognition [8]. The purpose of front end processing is to extract certain spectral features of the speech signals for comparison and further processing. Another way of performing a spectral analysis is the Filter Bank technique. The filter bank technique requires a lot of computation, which is minimized in the LPC technique. The LPC [7] model produces parameters that provide a reasonably good representation of the vocal

tract, as compared to the filter analysis technique [27]. Figure 3.4 shows a block diagram of an LPC preprocessor stage in a Speech Recognition system . The basic steps in preprocessing are indicated below.

1. *Preemphasis--* The digitized speech signal, s(n), is put through a low-order digital system (actually a first-order FIR filter), to spectrally flatten the signal and to make it less susceptible to finite precision effects later in signal processing. The digital system used in the preemphasizer is the fixed first-order system:

$$H(z) = 1 - \tilde{a} z^{-1}, \qquad 0.9 \le a \le 1.0 . \qquad (3.3)$$

In this case. the output of the preemphasis network, s(n), is related to the input to the network, s(n), by the difference equation,

$$\tilde{s}(n) = s(n) - \tilde{a}s(n-1). \qquad (3.4)$$

where, the value chosen for a is 0.97 .

2. *Frame Blocking--* In this step the preemphasized signal, $\tilde{s}(n)$, is blocked into N samples, with adjacent frames being separated by M samples. Fig. 3.5 illustrates the blocking into frames where M was chosen as $M = (1/3)N$. The first illustrated frame consists of N speech samples. The second frame begins M samples after the first frame, and overlaps by $N - M$ samples. This process continuous until all the speech is accounted for within one or more frames. The values for M and N have to be selected such that, $N \ge M$, to make sure that adjacent frames overlap. This would result in LPC spectral estimates to be

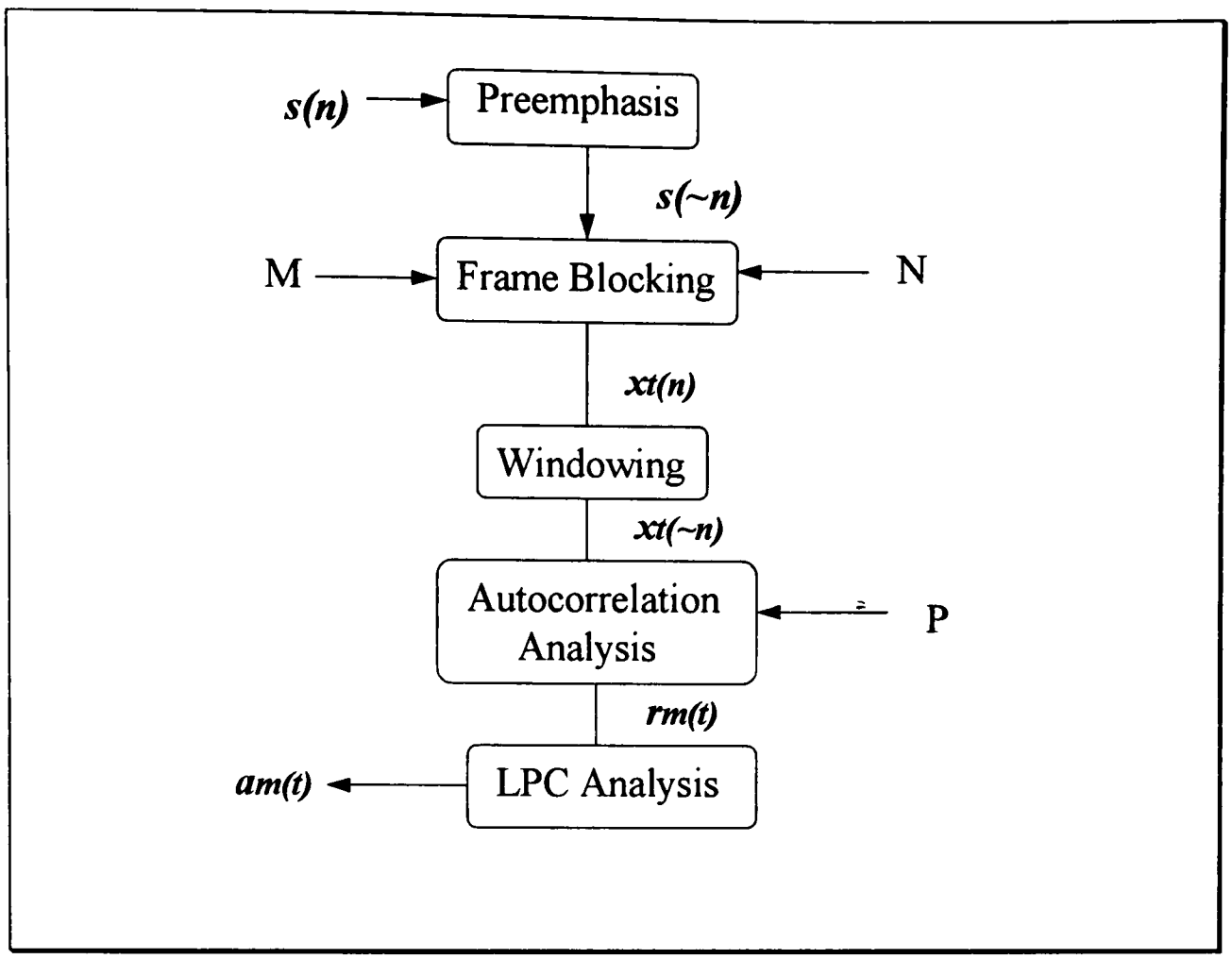


Fig. 3.4 LPC Preprocessor for Speech Recognition

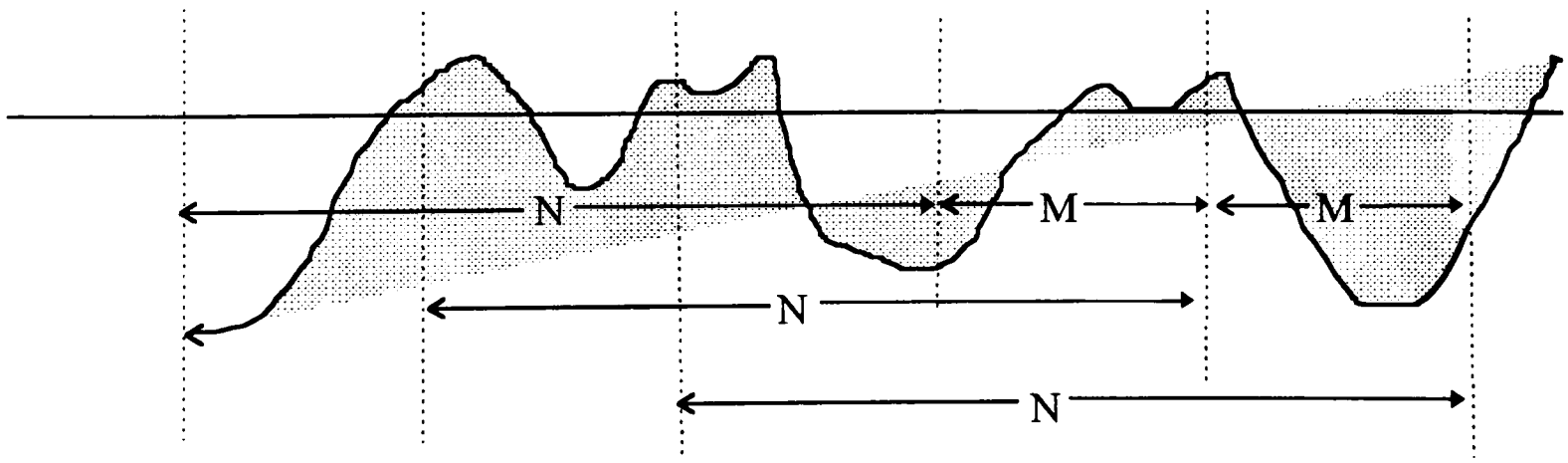


Fig. 3.5 Blocking of speech into overlapping frames

correlated from frame to frame. If M is chosen to be greater than N, some of the speech signal would be lost and the LPC estimates may include a noisy component. Usually for speech recognition applications, N is chosen to be 300.

3. *Windowing--* The next step in preprocessing is to perform windowing on individual frames so as to minimize the signal discontinuities at the beginning and the end of each frame. The concept here is identical to the frequency domain interpretation of the short-time spectrum [27]. This approach uses the window to taper the signal to zero at the beginning and end of each frame. Fig. 3.6 shows results of applying such a function. Consider the window as $w(n), 0 \le n \le N-1$ and the result of the windowing is the signal

$$\tilde{x}_l(n) = x_l(n)w(n), \qquad 0 \le n \le N-1 \tag{3.5}$$

A "typical" window used for the autocorrelation method of LPC is the Hamming window, which has the form

$$w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right), \qquad 0 \le n \le N-1 \tag{3.6}$$

4. *Autocorrelation Analysis--* Each frame of the windowed signal is next autocorrelated to give

$$r_l(m) = \sum_{n=0}^{N-1-m} \tilde{x}_l(n)\tilde{x}_l(n+m), \qquad m = 0,1,2,\ldots\ldots,p \tag{3.7}$$

where the highest value, p, is the order of the LPC analysis. Typically, values of p from 8 to 16 are used. For our case p=8 has been chosen. A side benefit in the autocorrelation analysis is that the zeroth autocorrelation, $R_l(0)$, is the energy of the l^{th} frame.

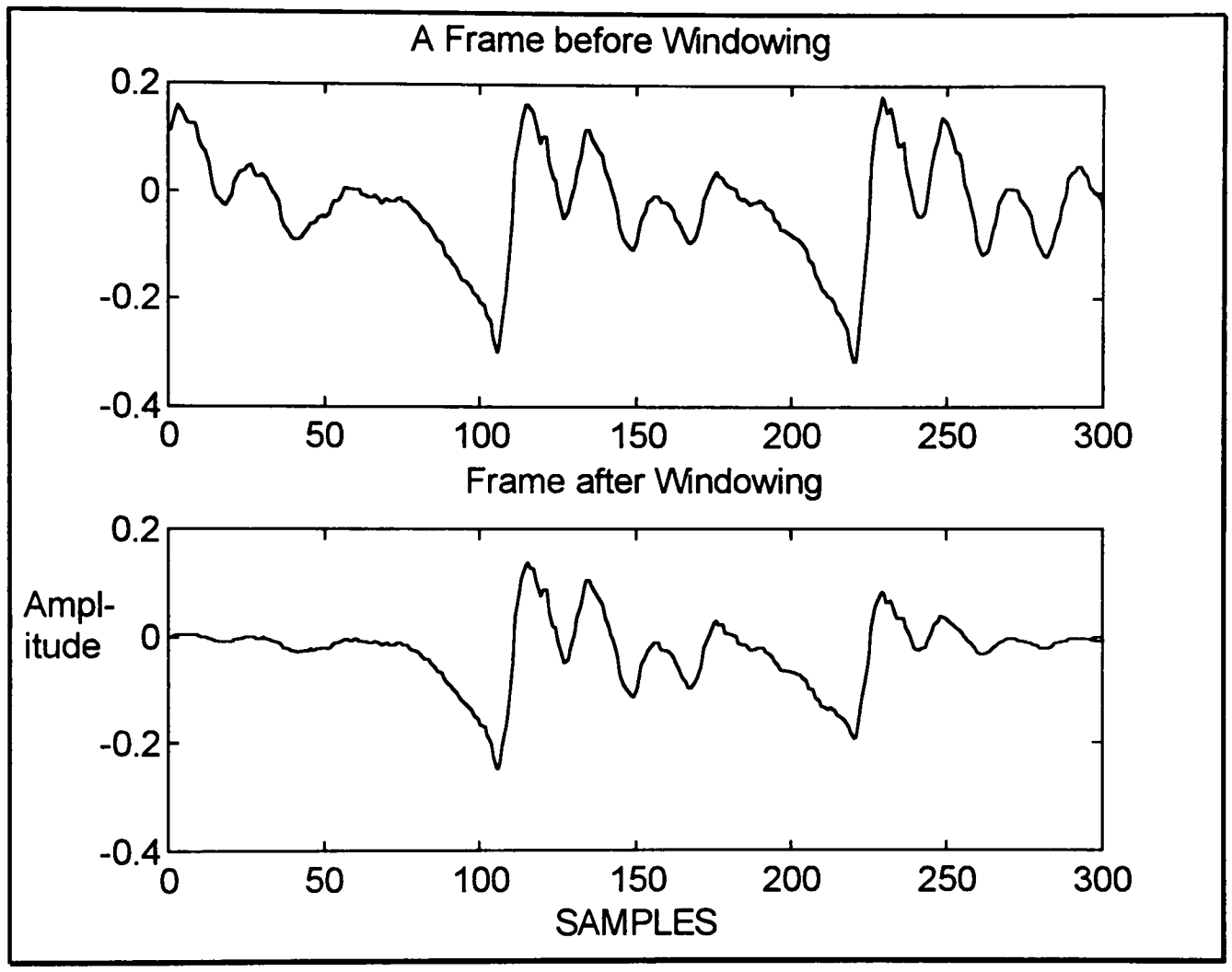


Fig. 3.6 Effects of applying Hamming Window to a frame of speech signal

5. *LPC Analysis--* The final step of the preprocessing stage is the LPC Analysis. The idea behind LPC [32] is to model the vocal tract using time-varying linear filters. As shown in Fig 3.7, the filters are be excited by periodic pulses for voiced speech and random noise for unvoiced speech. The LPC model [10] allows a speech signal, *s(n),* at time *n*, to be represented as a linear combination of the past *p* speech samples, such that

$$s(n) \approx a_1 s(n-1) + a_2 s(n-2) + \ldots + a_p s(n-p) \qquad (3.8)$$

where the coefficients $a_1, a_2, \ldots, a_p$ are assumed constant over the speech analysis frame. One advantage of using LPC is that it is done in the time domain unlike [27] other techniques which use the frequency domain. Nevertheless, frequency domain parameters can be easily calculated from the LPC coefficients.

The predictor coefficients are determined by minimizing the *residual error* (square difference) between the actual speech samples and the linearly predicted ones. The predictor polynomial for the LPC analysis block is as follows,

$$P(z) = 1 - \sum_{k=1}^{p} a_k z^{-k} \qquad (3.9)$$

The residual error mentioned above is given as

$$e(n) = s(n) - \tilde{s}(n) = s(n) - \sum_{k=1}^{p} a_k s(n-k) \qquad (3.10)$$

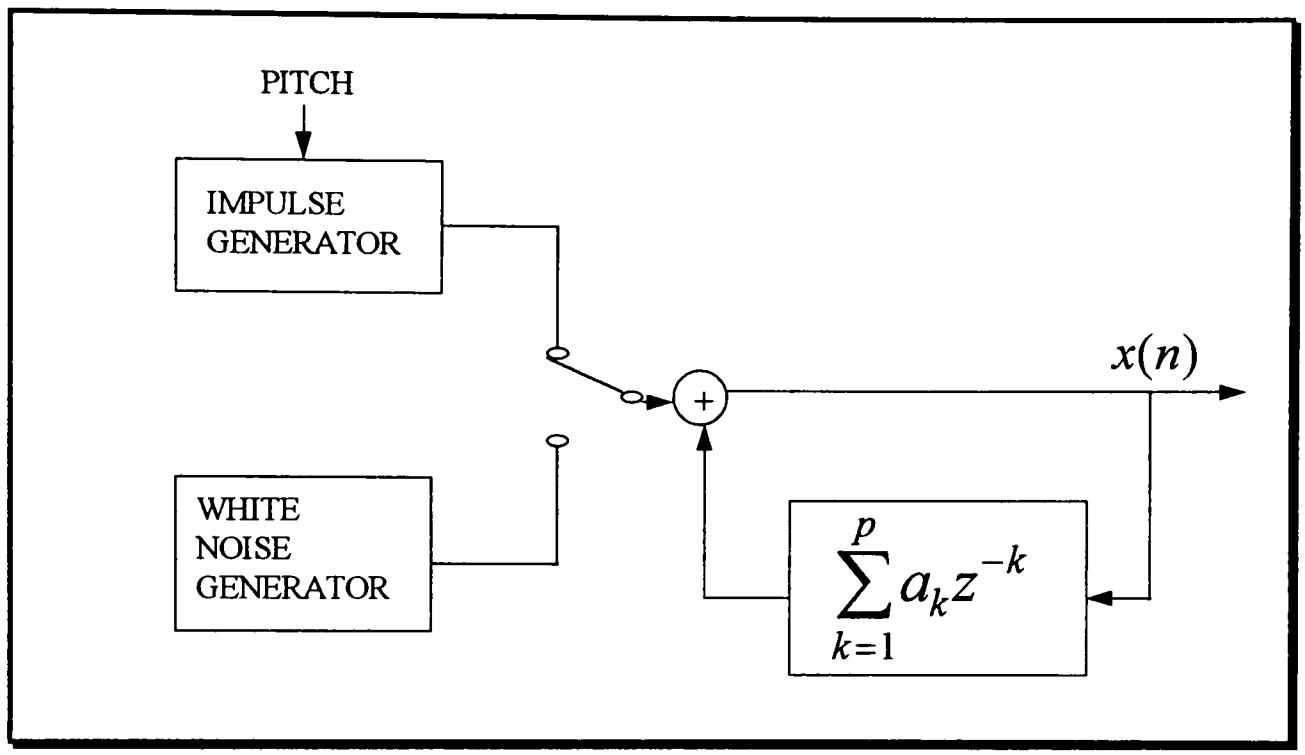


Fig. 3.7 Digital model for speech production

3.4 Vector Quantization

Vector Quantization is the key to this speech recognition system. Usually Vector Quantization is used only for source coding, to reduce the information storage rate, but in the present speech recognition system, it is also used as a template matching (Speech Recognition) algorithm. A few of the advantages of using VQ are reduced storage for spectral analysis vectors, reduced computation for determining similarity of spectral analysis vectors, and discrete representation of speech sounds. In the present case, VQ has been used to reduce the number of spectral vectors representing a speech signal. The basic elements of the VQ algorithm are shown in Fig 3.8.

The training set of spectral vectors $V_1, V_2, \ldots, V_L$, in Fig. 3.8, is derived from an LPC analysis of the speech signal. This training set is used to create the "optimal" set of codebook vectors for representing the spectral variability observed in the training set. The size of the codebook is given by $M = 2^B$, where M has to be such that $L >> M$, so as to be able to find the best set of M codebook vectors in a robust manner.

A measure of similarity, or distance, between a pair of spectral analysis vectors is denoted by d (. , .). This distance helps in clustering the training set vectors as well as associating arbitrary spectral vectors into unique codebook entries. Spectral distance between two vectors v_i and v_j is denoted as $d(v_i, v_j)$. Once the L training vectors have been classified into M clusters, M codebook vectors are chosen as the centroid of each of the M clusters. Finally, a classification procedure is implemented that chooses the codebook vector that is the best match for the input utterance.

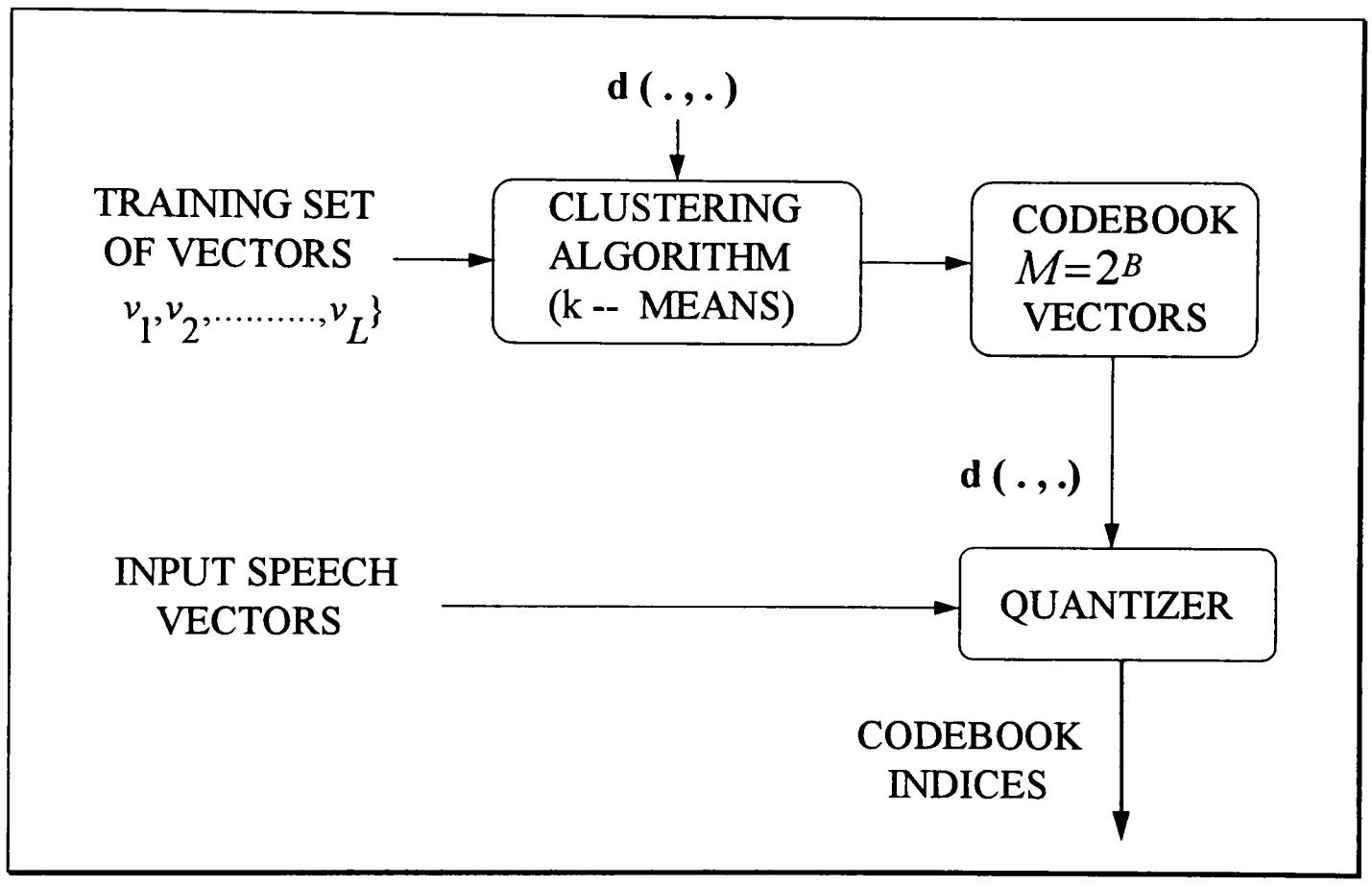


Fig. 3.8 Block diagram of the VQ training and classification structure

CHAPTER 4

VECTOR QUANTIZATION

4.1 Introduction

As discussed in the previous chapter, the result of LPC analysis is a series of vectors, characteristic of the time-varying spectral features of the speech signal. For convenience, the spectral vectors are denoted as V_l, l = 1,2,...,L, where typically each vector is a p-dimensional vector; p being the order of LPC analysis. Vector Quantization is a procedure that generates a small set of reproduction vectors (*codebook*) that best represent the information contained in the LPC spectral representation, V_l, for each word. The generation of a "*codebook*" is a recursive procedure that converges when spectral distortion(discussed later) between the input vector and the reproduction vector reaches a minimum. Vector Quantization, which is usually referred to as a source coding technique can also lead to a good classifier design. In the speech recognition system under consideration, the VQ technique has been implemented both as a source coding technique and a word classifier. The algorithms used for both of the above implementations are discussed in the following sections.

4.2 Fundamentals of Vector Quantization

If the characteristic sets of spectral parameters of a time-varying signal (e.g., speech) are quantized individually, the process is called *scalar quantization*. When these sets of parameters are quantized simultaneously as a vector object, the process is called

vector [34] *quantization.* A detailed explanation of the principles of vector quantization as described by J. Makhoul and L.Linde [34] is given below.

Consider a vector, $\mathrm{x} = [\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_n]^T \in \mathrm{R}^N$, where each $x_i \in R$ and represents a continuous-amplitude random variable. When this vector is considered in *N*-dimensional space, instead of occupying a unique position in the space, it is mapped onto another vector, say **y** by the operation; **y** = q(*x*) where q is the quantization operator and **y** is called the *reconstruction* or the *exemplar* of the category or the output vector for *x*. The vector **y** comprises real-valued elements and is itself a part of a set of vectors, denoted by **Y**. In set notation, this is:

$$y \in Y = \{y_i | 1 \le i \le L\} \quad \text{where} \quad y_i \in R^N . \tag{4.1}$$

The set **Y** is called the codebook, L is the size of the codebook and y_i represents the individual codebook vectors. L is also referred to as the "number of levels" in the codebook. To construct an L-level codebook, *N*-dimensional space is partitioned into L cells, such that:

$$R^N = \bigcup_{i=1}^{L} C_i \quad . \tag{4.2}$$

This partition implies that any point in space is mapped onto (attracted towards) a certain y_i, such that, for every cell C_i there is a vector y_i which acts as a representative for the cell based on certain rules. This vector is then used by the quantization function $q(x) = y_i$ if $x \in C_i$. The codebook design process is called *populating* the codebook. It

may be useful to think of the output (representative vectors) of the quantization algorithm as being not merely a set of classes, but as points in a class-space. The action of the quantization draws the output to a point that has the least distortion or distance with respect to class-space vectors. This point is referred to as the *attractor point*. The set **Y** can thus be thought of as a set of *attractors* in the *n*-dimensional space. Any object introduced into the space is attracted to a certain y_i based on the metric definition in the space.

When an input vector x is quantized by the function q, there is an inherent quantization error called the *distortion measure* which is measured as a distance metric $d(x, y_i)$, also called a dissimilarity measure. The overall average distortion as the vectors $y(n)$ are encoded at different times is given by the expression:

$$D = \lim_{M \to \infty} \frac{\sum_{n=1}^{M} d[x(n), y(n)]}{M}, \qquad (4.3)$$

i.e., at time n, a vector is quantized as a representative vector, y. The distortion or dissimilarity at time $t = n$ in the process will be $d[x(n), y(n)]$. Adding all the distortions and dividing by the number of samples taken, gives the average distortion.

4.3 Codebook Design

The quantization function defined for the purpose of measuring the proximity of x to the codebook vectors, assigns codebook vector y_i to x if and only if $x \in C_i$. For

quantization to be of minimum distortion, the average distortion should be minimized over all the L-level quantizers. This condition depends on two necessary factors:

1. The optimal quantizer is realized by using the minimum distortion or nearest neighbor selection rule. This is given by : $y_i = q(x)$ iff $d(x, y_i) \leq d(x, y_j)$ $\forall 1 \leq j \leq L$ and i not equal to j. In other words, y_i has the minimum distortion, for x compared to all other representative vectors.
2. Each codebook vector is chosen to minimize the average distortion in the cell C_i. Such a codebook vector is then referred to as the centroid of C_i, or y_i = cent(C_i).

Computing the centroid of C_i depends on the definition of distortion. The cells thus defined are called *Voronoi cells* or *Dirichlet regions*. It is desirable to have y_i = cent(C_i) at each level of the codebook design, hence the codebook is re-evaluated at each stage so as to give an optimal representation of the input vectors. This is the fundamental dynamic that characterizes the VQ algorithm.

Two types of codebook generation algorithms have been implemented in this project. These two variations of VQ differ only in the number of codebooks generated for each word or utterance class in the database. The reason for these variations is explained later in the chapter.

4.4 Single Codebook Generation

The single codebook algorithm generates a codebook for each utterance class(e.g. words) using a relatively short training set of vectors containing repetitions of words in the recognition vocabulary. For example, the codebook for the word STOP is designed by running the VQ design algorithm on a training sequence of several repetitions of the word STOP. The algorithm divides the set of training vectors, {x(n)} into L clusters such that, if m is the iterative index, $C_i(m)$ is the i^{th} cluster at the iteration m and $y_i(m)$ is its centroid. The algorithm is called the binary split-generalized Lloyd algorithm or the K-means clustering algorithm [27].

Initializing : Choose an initial set of codebook vectors from the input training set. In this case the initialization generates a vector which is the centroid of the entire training set.

Splitting : Double the size of the codebook by splitting each current codebook y_n according to the rule

$$y_n^+ = y_n(1+\epsilon)$$
$$y_n- = y_n(1-\epsilon)$$

where n varies from 1 to the current size of the codebook, and

$\epsilon = 0.004$ is a splitting parameter.

Classification : Classify the set of input vectors into clusters C_i , using the nearest neighborhood rule.

$$x \in C_i(m) \text{ iff } d[x,y_i(m)] \le d[x,y_i(m)] \text{ for all j}$$

This means that for each training vector, the codeword in the current codebook that is closest (in terms of spectral distance) is found, and that vector is assigned to the corresponding cell (associated with the closest codeword).

Centroid Update : Compute the new codebook vector for each cell by determining the centroid.

Iteration : Repeat steps *Classification* and *Centroid Update* until the average distortion falls below a preset threshold.

Termination: Repeat steps *Splitting* through *Iteration* until a codebook of size M is designed.

Fig. 4.1 shows in a flow diagram, the detailed steps of the binary split VQ codebook generation technique. The box labeled "Classify Vectors" is the nearest-neighbor search procedure, and the box labeled "Find Centroids" is the centroid update procedure of the K-means algorithm. The box labeled "Compute D(Distortion)" sums the distance of all training vectors in the nearest neighbor search so as to determine whether the procedure has converged (i.e., $D = D'$ of the previous iteration).

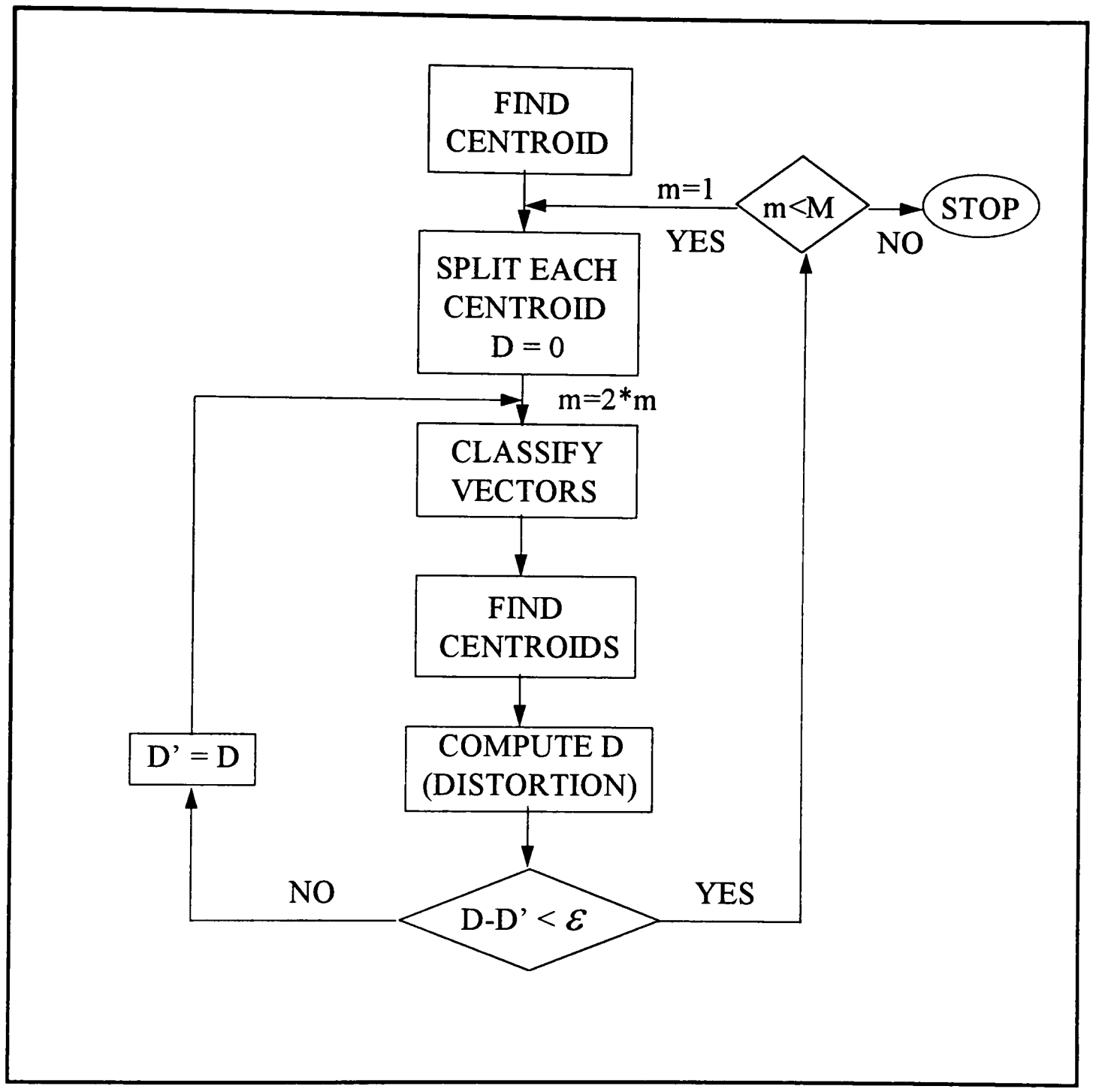


Fig. 4.1 Flow diagram of binary split codebook generation algorithm

4.4.1 Distortion

A key component of most pattern recognition algorithms is a prescribed [27] measurement of dissimilarity between two feature vectors. For speech processing, an important consideration in defining (or choosing) a measure of distance is its subjective meaningfulness. A mathematical measure of distance, to be useful in speech processing, [27] has to have a high correlation between its numerical value and the subjective distance judgment, as evaluated on real speech signals. The distortion caused by reproducing an input vector x by a reproduction vector $\hat{x}$ is assumed to be given by a nonnegative distortion measure $d(x,\hat{x})$. Several [37] distortion measures, like the squared error, holder norm, Minkovski norm and difference distortion, have been proposed for the purpose of distance calculation. All the above distortion measures depend on the vectors x and $\hat{x}$ only through the error vector $(x-\hat{x})$. The distortion measure put to use in this thesis is the one proposed by Itakura, Saito and Chaffee [32] and has the form

$$d(x,\hat{x}) = (x-\hat{x})R(x)(x-\hat{x})^t \tag{4.4}$$

where for each x, $R(x)$, the autocorrelation matrix of the input sequence x_N is as follows,

$$R(x) = \begin{bmatrix} r(0)..r(1)..r(2)...r(K) \\ r(1)...r(0)..r(1)...r(K-1) \\ \vdots \\ \\ r(K)..r(K-1)...r(0) \end{bmatrix}. \tag{4.5}$$

The $(i - j)^{th}$ element of $R(x)$ is given by $r(|i - j|)$, where r_N can be calculated as follows

$$r_N(n) = \sum_{i=0}^{N-|n|-1} x(i)x(i+|n|) \qquad \text{for } n = 0,1,\ldots,N\text{-}1 . \qquad (4.6)$$

There were a few more distortion measures that could have been considered, such as the Itakura-Saito distortion [32] measure and the Likelihood distortion [37] measure. The calculations involved in distance evaluation using the above two measures [37] are relatively complicated compared to the Mahalanobis measure described above. Thus all the distance calculations implemented in this thesis are based on the Mahalanobis distance measure.

4.4.2 Centroid

The problem of centroid calculation is highly dependent on the distortion measure [34] selected. Since the distortion measure in the present case is the Weighted Euclidean distance measure(Mahalanobis distance measure), the centroid calculation is the mean of the vector set,

$\ddot{y} = \frac{1}{L}\sum_{i=1}^{L} x_i$, where $\ddot{y} = (y_1, y_2, \ldots, y_k)$ is the centroid of the input vector

$x_i = (x_1, x_2, \ldots x_L)$. Fig. 4.2 shows the classification procedure implemented on a set of input vectors. The four circles are the codewords of a two-dimensional codebook. The

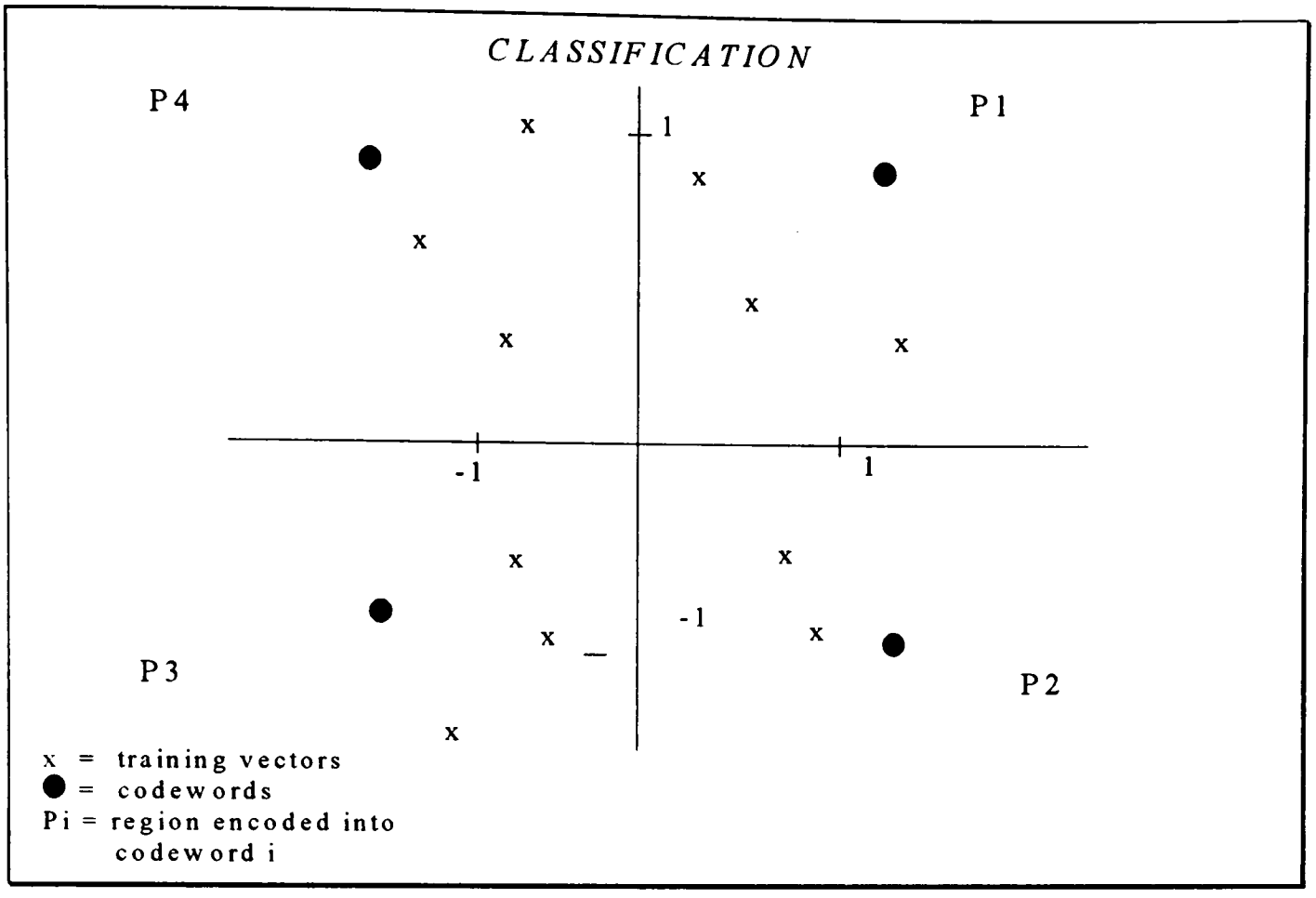


Fig. 4.2 Classification of training vectors before Centroid Update

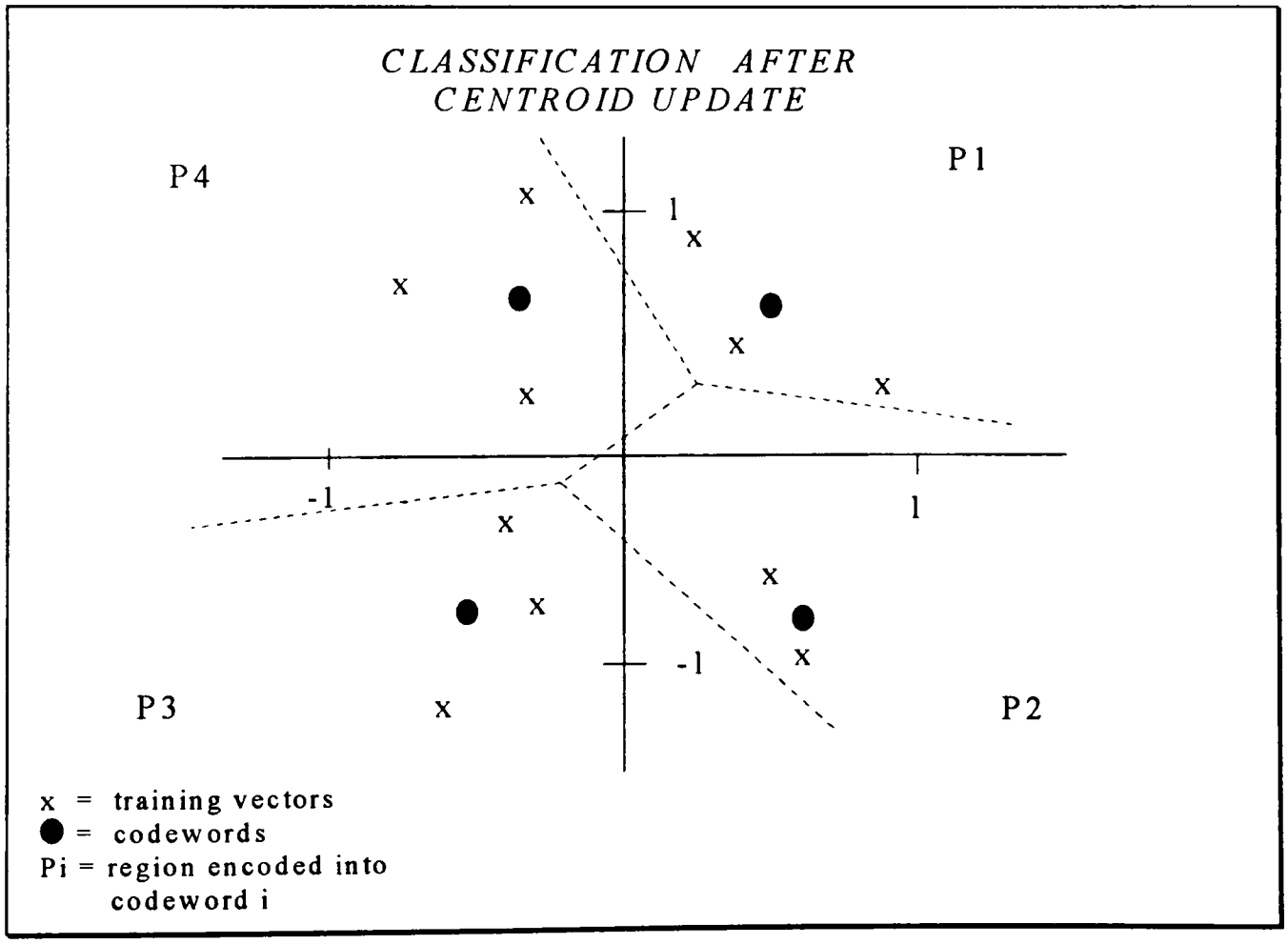


Fig. 4.3 Classification of training vectors after Centroid Update

Voronoi regions are the quadrants containing the circles. The x's were produced with a training sequence of 12 two-dimensional vectors. Each input vector is mapped into the nearest neighbor codeword, that is the circle in the same quadrant. Fig. 4.2 and Fig. 4.3 show that the centroid computation has moved the codewords to better represent the input vectors which yielded those codewords. When the training vectors are used to redefine the positions of codewords, a reduction in the distortion takes place. The broken lines delineates the new Voronoi regions for these codewords.

4.5 Segmental Codebook Generation

The standard vector-quantization approach that uses a single vector quantizer for the entire duration of the utterance for each class sometimes does not preserve the sequential characteristics of the utterance class [26]. This lack of explicit characterization of the sequential behavior is remedied in the segmental VQ approach. In this approach each utterance class is treated as a concatenation of N_S information subsources, each of which is represented by a VQ codebook. This results in a huge matrix of codebook vectors for each word. This encoder is also called a *Matrix Quantizer*. The segmental [26] approach to VQ is a straightforward extension of the memoryless vector quantizer. In this algorithm, the set of training vectors, $\{x(n)\}$, is decomposed into N_S subsources by equally dividing each utterance vector into N_S segments. Then each segment is treated as if it were a separate information source, and all the corresponding segments of each utterance are grouped together to form a training set. This simple segmentation scheme is illustrated in Fig. 4.4. The VQ algorithm described in the previous section is then applied

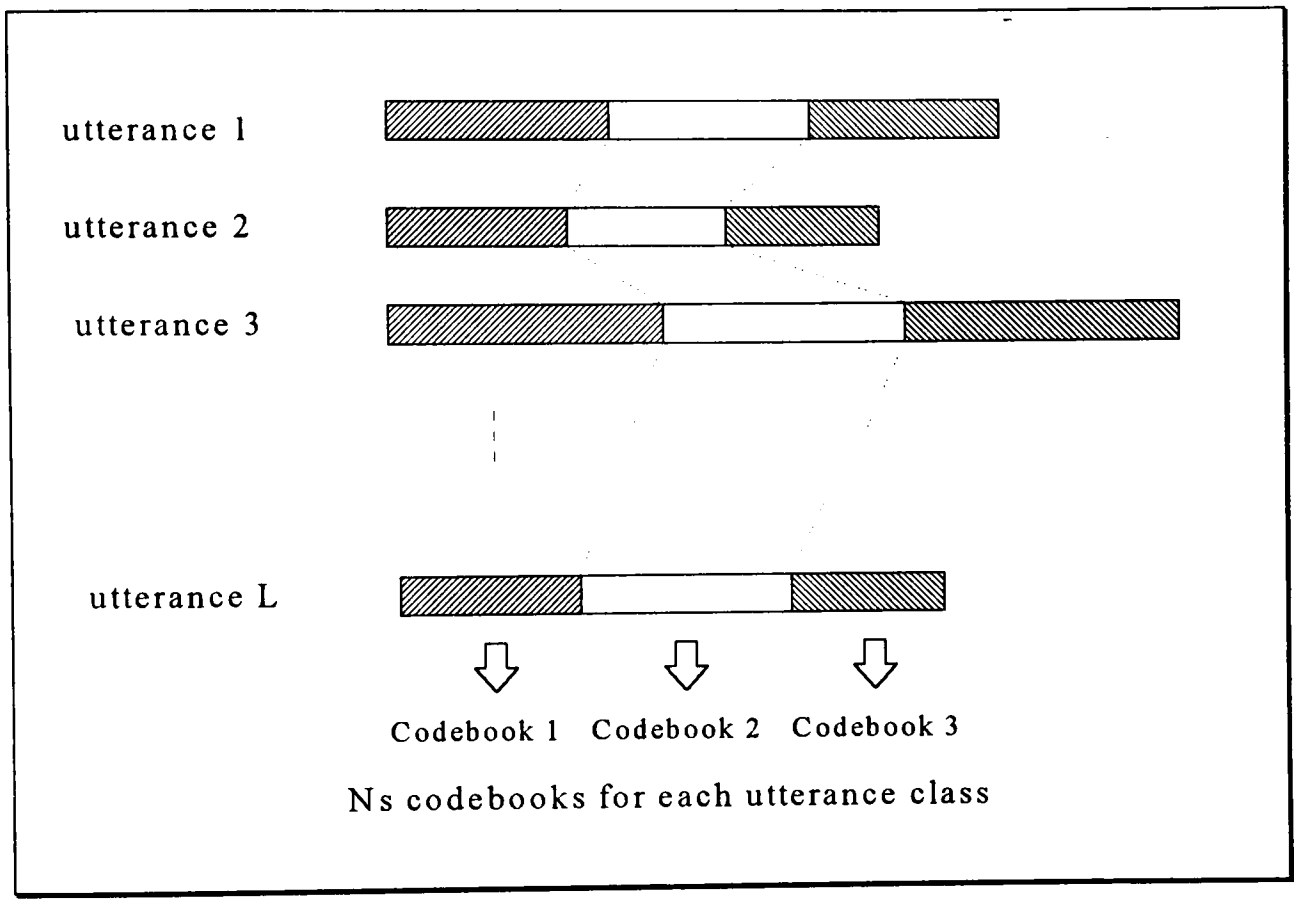


Fig. 4.4 Codebook training for segmental vector quantization

to these sets of training data. Now for each class there are N_S codebooks that have an implicit temporal order because they correspond to different portions of the utterances. Segmental vector quantization requires virtually the same computational complexity as the previous utterance-based VQ as long as the codebook size remains the same. The only complexity increase is in the size of the codebook storage; segmental VQ has N_S codebooks while utterance-based VQ has only one for each utterance class. As indicated in the results in the following chapters, the preserved sequential relationship in the form of codebook concatenation, however, proves to be relatively more efficient than the utterance-based approach.

4.6 Recognition Algorithms

This section discusses the two algorithms used to perform Speech Recognition in this thesis.

4.6.1 Vector Quantization

There are M utterance *classes* (e.g., words) to be recognized. Each utterance *class* is considered as an information source. M sets of training data $\{x_t^{(i)}\}$ where i = 1,2,...M is the class index, are collected. Each training set, contains several utterances of the same class. M codebooks $\{C^{(i)}\}_{i=1}^{M}$ are then designed using the minimum average distortion objective for the M information sources, respectively . Each codebook represents the characterization of the information source (*class*). During the

recognition operation, the M codebooks are used to implement M distinct vector quantizers as shown in Fig 4.5. An unknown utterance $\{x_t\}_{t=1}^{T_u}$ is vector quantized by all M quantizers, resulting in M average distortion scores $D(C^{(i)}), i = 1,2,\ldots,M$ where,

$$D(C^{(i)}) = \frac{1}{T_U}\sum_{t=1}^{T_u} d(x_t, \hat{x}_t^{(i)}) \qquad \text{with } \hat{x}_t^{(i)} \in C^{(i)} \text{ satisfying} \qquad (4.7)$$

$$\hat{x}_t^{(i)} = \arg.. \min_{y_j^{(i)} \in C^{(i)}} d(x_t, y_j^{(i)}) \qquad (4.8)$$

The utterance is recognized as class k if

$$D(C^{(k)}) = \min_i D(C^{(i)}) \qquad (4.9)$$

Thus, the index of the codebook that was classified as the test utterance is used as a representation of the utterance. In some communication applications, where the entire word has to be transmitted over a channel, using this technique only a 1 digit index has to be sent. This represents a large saving in the channel bandwidth for communication applications. A variation of Vector Quantization, the segmental VQ algorithm, explained in the next chapter, recognizes a test utterance in the same way as described above.

4.6.2 Zero-Crossing Technique

As discussed in earlier chapters, zero-crossing algorithms can be used for the purpose of speech recognition. A zero-crossing occurs when the polarity of a time varying signal (e.g. speech) changes. The features used to perform classification of speech signals

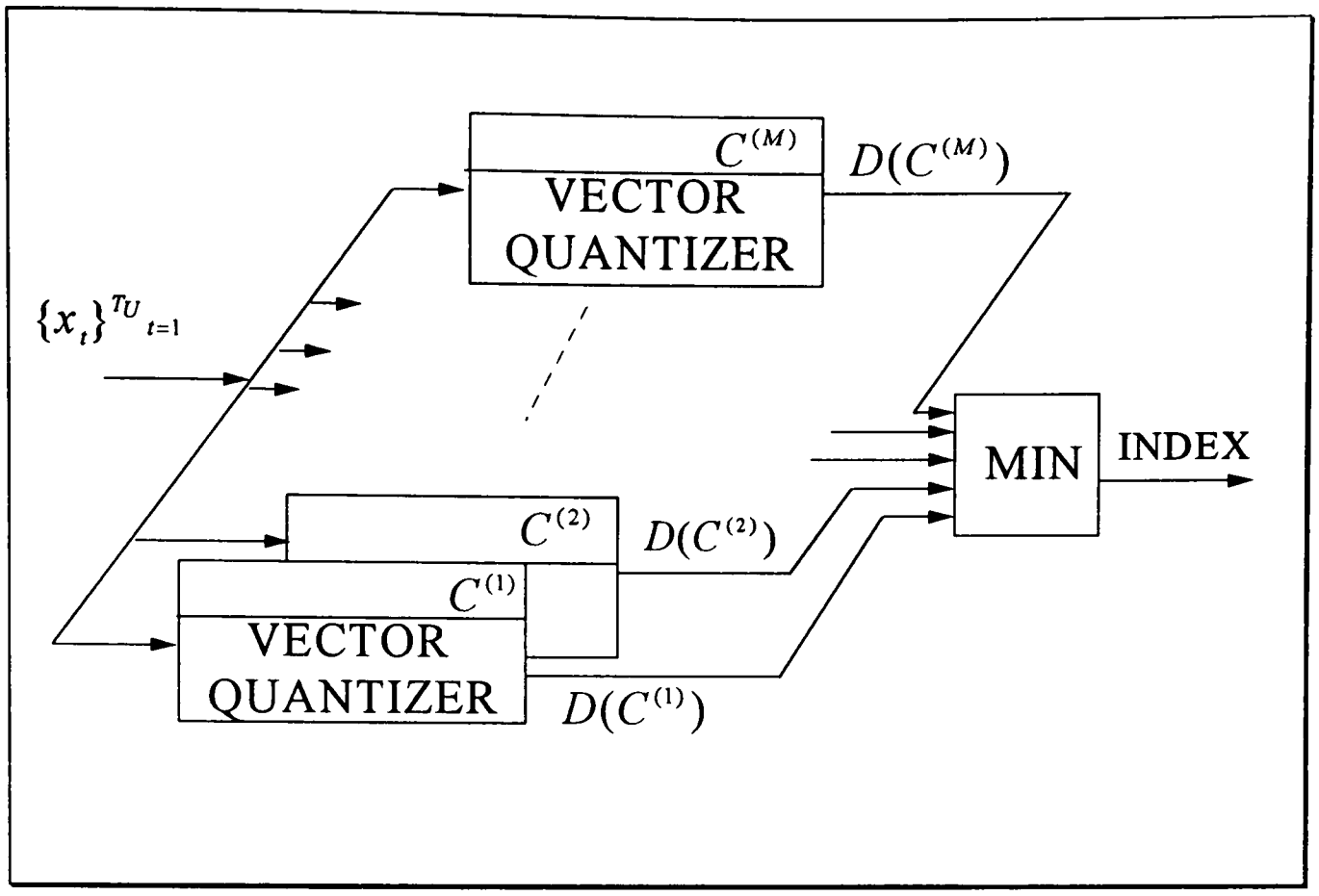


Fig. 4.5 A vector quantizer based speech recognition system

are average zero-crossing rate and excess-threshold duration. The speech signal is divided into N different sections with the help of N time observation windows. $\tau_1, \tau_2, \ldots \tau_N$ are the lengths of a succession of time observation windows then given by

$$Z_k = \frac{M_k}{\tau_k} \qquad k=1,2,\ldots.N. \qquad (4.10)$$

A parameter that needs to be calculated to determine the excess threshold measurement, is, $\boldsymbol{P_k}$, the Zero-Crossing density. It is the time interval between successive zero-crossings within an observation window. L_k is a vector indicating the location of zero crossings in

the *k*th observation window. The zero crossing density is thus defined as

$$P_k(i) = L_k(i) - L_k(i-1) \qquad \begin{matrix} \forall k = 0,1,2,\ldots N \\ \forall i = 1,2,\ldots,\tau_k \end{matrix} \qquad (4.11)$$

The excess threshold measurement consists of the calculation of an excess-threshold function $f(D)$ which is the ratio of the sum of time intervals that exceed a threshold value D (i.e. $i > D$) to the duration of the observation window τ_k. σ_k is the mean of P_k and ℓ_k is the length of P_k. The threshold D is calculated as

$$D = \frac{\sum_{k=1}^{N} \sigma_k * \ell_k}{\sum_{k=1}^{N} M_k} \qquad (4.12)$$

The excess threshold function $f(D)$ is calculated as

$$f(D_k) \approx \frac{1}{\tau_k} \sum_{i=1}^{i=U} P_\tau(i) \qquad \begin{matrix} \forall P_\tau(i) > D \\ k = 1,2,\ldots N \end{matrix} \qquad (4.13)$$

4.6.3 Speech Recognition using Zero-Crossing

To generate the database for the SR task, every speaker has to speak all the words from the vocabulary several times. A class is a collection of repeated utterances of a unique word spoken by different speakers. A feature set, comprised of the excess threshold function and the average zero-crossing rate is evaluated for each group of speech classes. The feature sets for each utterance class in the vocabulary together form the *reference feature set.* Once the *reference feature set* has been calculated, the recognition phase of the algorithm is implemented. This phase generates, a feature set for

the test utterance, called the *test feature set* . The next step is to perform the comparison between the reference feature set and the test feature set on the basis of a spatial distance calculation. This comparison is done on the basis of a Euclidean distance measurement [27]. Euclidean distance is one of the simplest metrics and is the square root of the sum of the squares of the difference between feature sets of the reference signal and test signal. As a simple example consider the two patterns $x = a, b, c, d$ and $y = p, q, r, s$ in Fig. 4.6. The recognition phase, calculates the distance between the *test feature set* and the feature set of each word in the vocabulary. The word in the *reference set* that accumulates the least distance with respect to all the other words in the vocabulary is classified as the spoken word.

The previous section described the two different approaches that were used in the thesis to perform speech recognition. The previous chapters dealt with the different types of algorithms for Speech Recognition applications. The previous chapters also provided insight into the preprocessing stages that a speech signal passes through, before being used in an SR application. The next chapter explains the results of the above mentioned algorithms and evaluates the performance of the entire SR system.

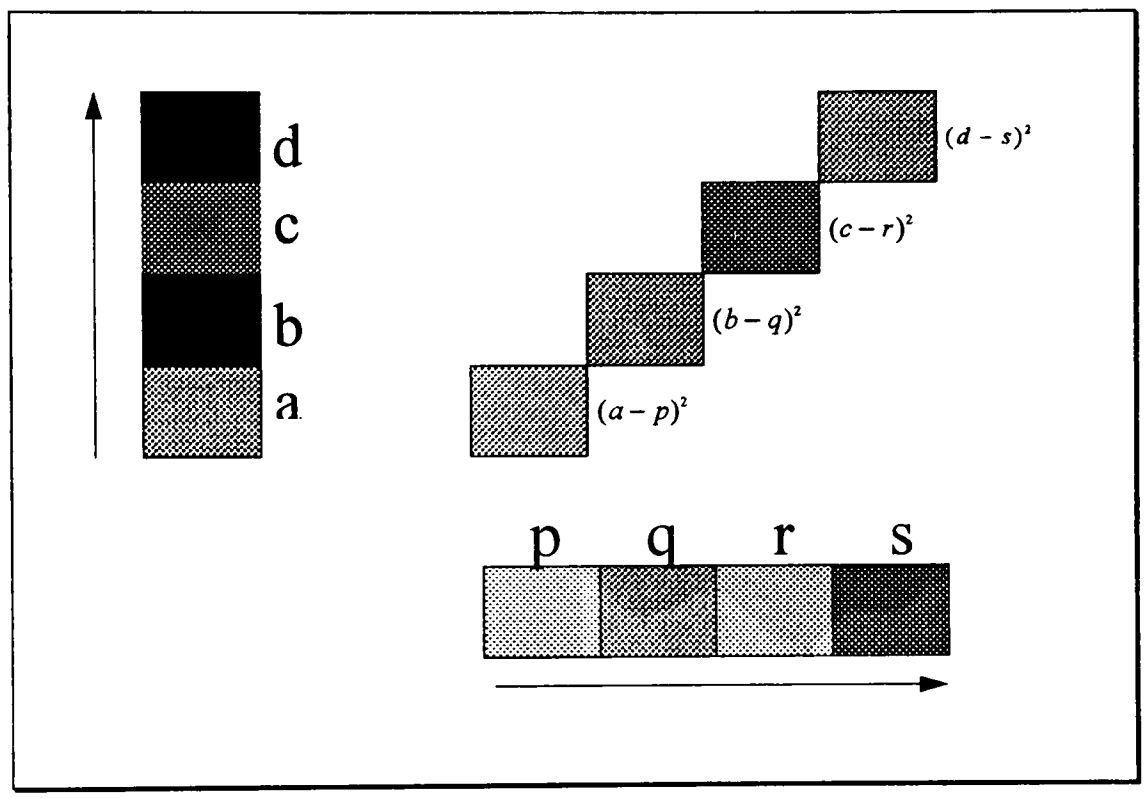


Fig. 4.6 Illustration of the Euclidean distance metric

CHAPTER 5

IMPLEMENTATION OF THE SYSTEM

In previous chapters, the software for implementing the SR system was described. This included the speech coding algorithm and the algorithms to perform Speech Recognition. Once these algorithms are developed, the next step is to implement the entire system on a platform and evaluate the results of the overall system. The goal of this thesis was to develop a small Speech Recognition card based on a Digital Signal Processor. This would make the Digital Signal Processor the implementation platform for the SR system.

Digital Signal Processors are basically high-speed, single-chip microcomputers [39]. Their internal architectures are different and, in most cases, instruction execution times are considerably shorter than in conventional microprocessors. DSP chips such as the TMS320 series [39] from Texas Instruments have a Harvard-type internal architecture. These processors have separate internal buses for instructions and data, which allow them to fetch both simultaneously instead of sequentially and, thereby, reduce instruction execution times. Perhaps the most distinctive characteristic of DSP's is that they have a single-instruction multiply-and-accumulate (MAC) operation. This is particularly useful for digital signal processing, since most algorithms involve the sum-of-products arithmetic operation. Since the DSP used in the current project is the TMS320C30, the next section deals the features of the TMS320C30 only.

5.1 TMS320C30 DSP Microprocessor Introduction

The TMS320C30 [39] is a fast processor (16.7 million instructions per second for an instruction cycle time of 60 nanoseconds) with a large memory space (16 million 32-bit words) and 40-bit floating point arithmetic capabilities. This last feature, a major trend in new DSP devices, was developed to answer the need for quicker, more accurate solutions to numerical problems. DSP algorithms, being very numerically intensive, cause a designer to worry about overflows and the accuracy of results. The introduction of floating-point capabilities reduces these difficulties. DSP microprocessors use a Harvard type architecture [39] in the sense that internally and externally they have have multiple buses to access program instructions, data, or perform DMA transfers. However, they also have a von Neumann style architecture [39] since the memory space is unified, and there is no separation of program and data spaces. As a result, the user can choose to locate programs and data at any desired location. In other words, the unified memory space offers flexibility in placing program and data. It also permits optimal use of the memory space as a trade-off between program and data. The TMS320C30 provides a total memory space of 16 million 32-bit words, a memory space that is several orders of magnitude [39] larger than that of fixed-point devices. Furthermore, it has significantly increased on-chip memory (compared to its predecessors), i.e. , six thousand 32-bit words of *RAM* and *ROM.* The internal memory (both *ROM* and *RAM*) supports two accesses for reads and/or writes in one cycle. This key feature permits high throughput and ease of programming,

since it makes possible three-operand instructions with two operands residing in the memory [39].

The pipelined architecture [27] of the TMS320C30 permits higher throughput to be achieved by the device. With the pipelined architecture one can assume that the result of any instruction will be available for the next instruction. The design philosophy of the TMS320C30 can be described as one having an *interlocked* or *hidden-pipeline* approach.

The CPU consists of an ALU (arithmetic logic unit), a hardware multiplier, and a register file. The register file [39] consists of eight 40-bit, extended-precision registers R0 through R7, eight 32-bit auxiliary registers AR0 through AR7, and twelve 32-bit control registers. The single-cycle hardware multiplier [39] is an integral part of the DSP because any real-time application relies on the fast execution of multiplication operations [39]. The multiplier performs both floating-point and integer multiplications. The architecture of the 320C30 DSP chip has been roughly described and both the internal organization of the device and the external interfaces and the pipeline structure have been discussed. For more information about the addressing modes and other software-related issues and constructs, the reader is directed to the following references [39].

5.1.1 Development Tools

Several algorithms, implemented for the purpose of Digital Signal Processing [27] applications, are sophisticated, and constructing and debugging such algorithms at

assembly-language level becomes a tedious task. The user-friendly development tools provided by Texas Instruments help in minimizing the time required to develop a DSP subroutine. The tools include an optimizing C compiler, a linker, and a DSP operating system. These tools facilitate the development of algorithms on large computers and help in converting those algorithms to 320C30 assembly language. All the assembly language subroutines developed for this project were initially developed in C code and then translated into assembly language using the C compiler as described in Fig. 4.5. The assembled file is then linked with object libraries to generate an executable object module. The assembled code is a COFF (Common Object File Format) object file. A COFF file format promotes modular programming by supporting *sections* (i.e., executable code, initialized data, reserved space for uninitialized data). One of the advantages of the optimized linker provided by Texas Instruments, as a part of development tools, is that any *section* of the COFF code can be relocated either in ROM, internal RAM, or external RAM (on the SPIRIT-30 board). In certain cases, performance critical code is in a ROM-based system, but would run faster if it executed from RAM. Linker directives can be used to load the code in ROM, meanwhile executing it from RAM, thus making a faster, more efficient code. Fig. 5.1 shows how the C code is finally executed on the DSP. The shaded portion of the figure highlights the most common path of software development. The other portions are optional.

5.1.2 The SPIRIT-30 TMS320C30 System Board

The SPIRIT-30 AT is an accelerator card that can be plugged into an IBM compatible computer. The SPIRIT-30 card is built around Texas Instruments' TMS320C30 DSP which offers up to 33 Mflops (Million floating point operations per

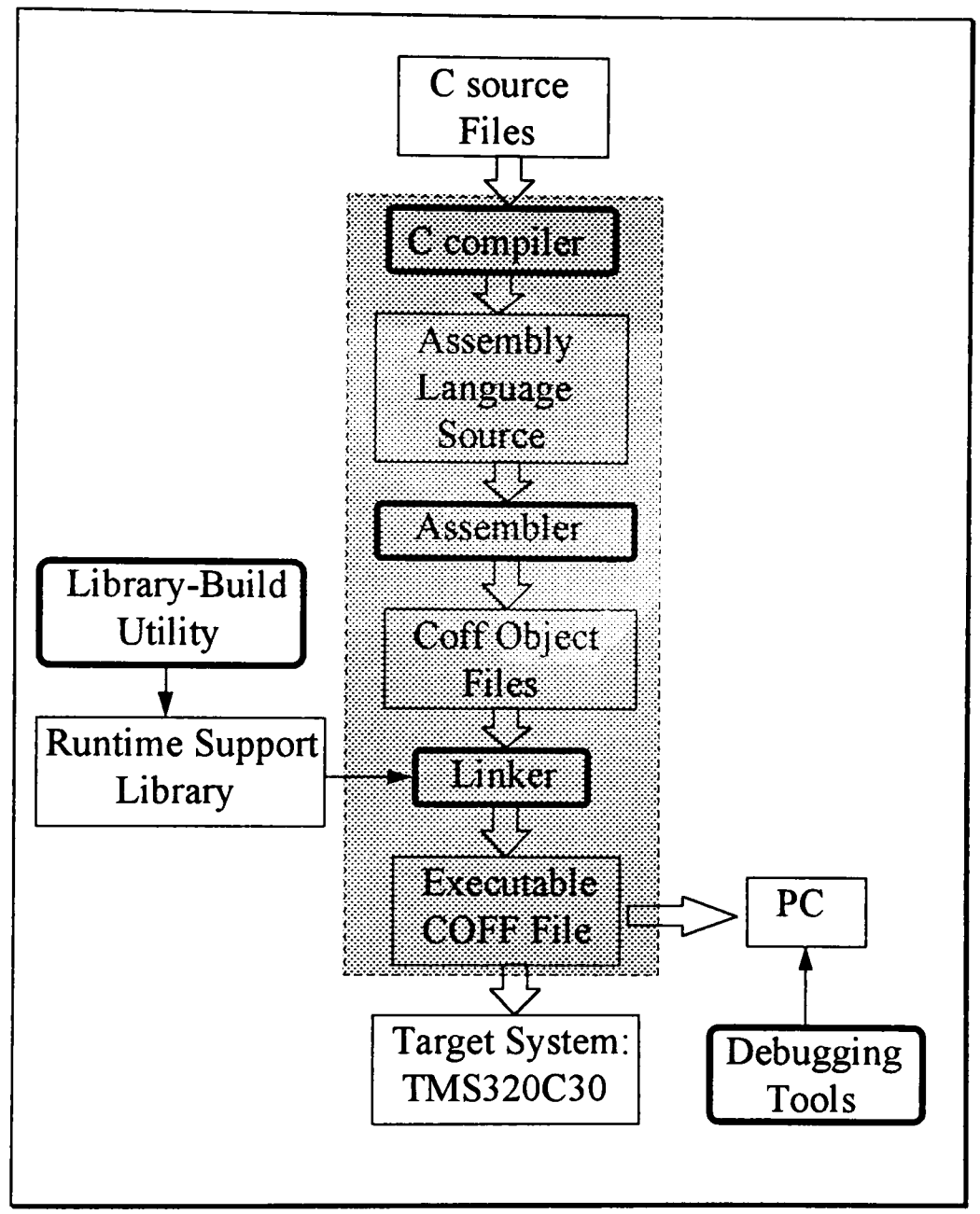


Fig. 5.1 TMS320C30 Software Development Flow

second) capabilities. The features of the card include DMA transfer capabilities between PC and card, run time library, debugger, and PC-to-card communication utilities. One of the features available on this card is the 256Kbytes of memory space. This large amount of memory can prove very useful in several digital signal processing applications that require huge arrays as inputs and generate very large matrices as outputs. The card acts as a target system (as shown in Fig. 5.1) for execution of the executable COFF file. Using the runtime library provided with the card, a windows based master C program can be generated that downloads the COFF file onto the DSP, executes the code, and uploads the results from the DSP to the PC. It is also possible to display the results on the screen in any format (i.e., as a matrix, as an array, or generate a plot). A separate sound card developed by Creative Technologies was used to record all the utterances.

5.2 Implementing the System

The system was implemented on two different systems, the first one was the PC and the second one was the SPIRIT-30 DSP application development system. For the PC based realization, the high level language, MATLAB, was used for developing algorithms and evaluating the results. For the DSP based realization, the algorithms were coded in assembly language and executed directly on the Digital Signal Processor. The system algorithm was implemented in several stages and the details and results of each stage are described in the next section. Fig.5.2 shows the different phases that the system goes through before performing speech recognition.

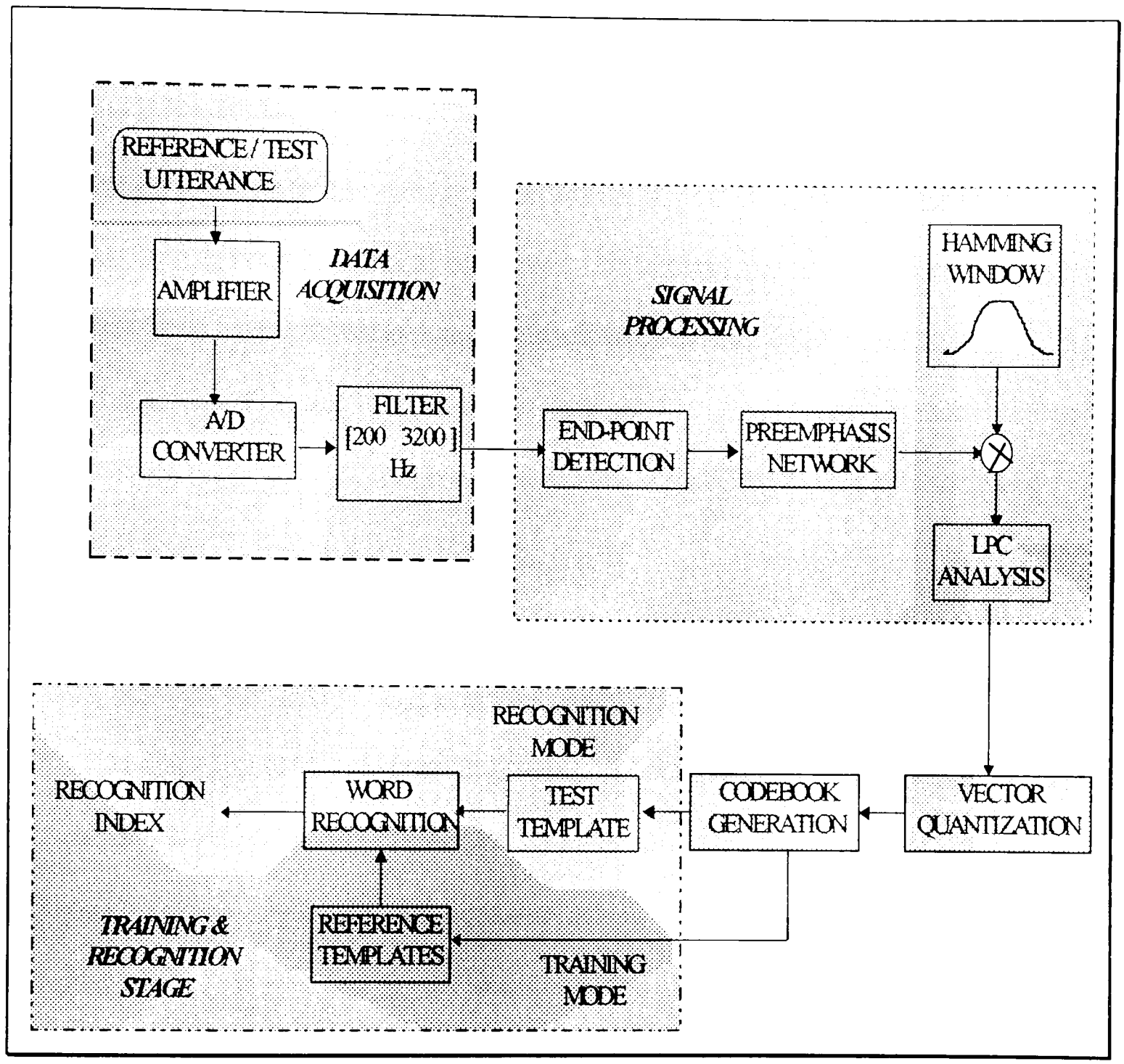


Fig. 5.2 The block diagram of Speech Recognition System

5.3 Data Acquisition and Conversion

Fig. 5.2 shows that the first block in the SR system is Data Acquisition. Here, the speech is recorded using a microphone, amplified and passed on to the A/D converter (16-bit). The converter then samples the speech signal at the rate of 10 kHz and stores the speech in μ-law format for further digital processing. Most of the important characteristics of a speech waveform are in the lower frequency band (200-3200 Hz) [27]. To extract the relevant features of the speech, the digitized data is then passed through a band-pass filter, having lower and upper cut off frequencies of 200 Hz and 3200Hz, respectively. This filter also eliminates the unwanted high frequency noise that could be present in the original signal. The speech data is now ready for the Signal Processing stage.

5.4 Signal Preprocessing

At this point of the recognition process, the available data, as shown in Fig.5.2, is the speech waveform, having silence periods before and after the utterance. The first step in Signal Preprocessing is endpoint detection, where the beginning and ending of the unknown word is determined. Errors in endpoint location increase the probability of impossible. The algorithm used in the system of Fig. 5.2 has been developed to accurately determine the endpoints of a word to within 90% accuracy. The algorithm, as explained in section 3.3, is based on zero crossing and energy values of successive frames of the speech signal. Fig. 5.3 and Fig. 5.4 show two examples of endpoints determined by the system

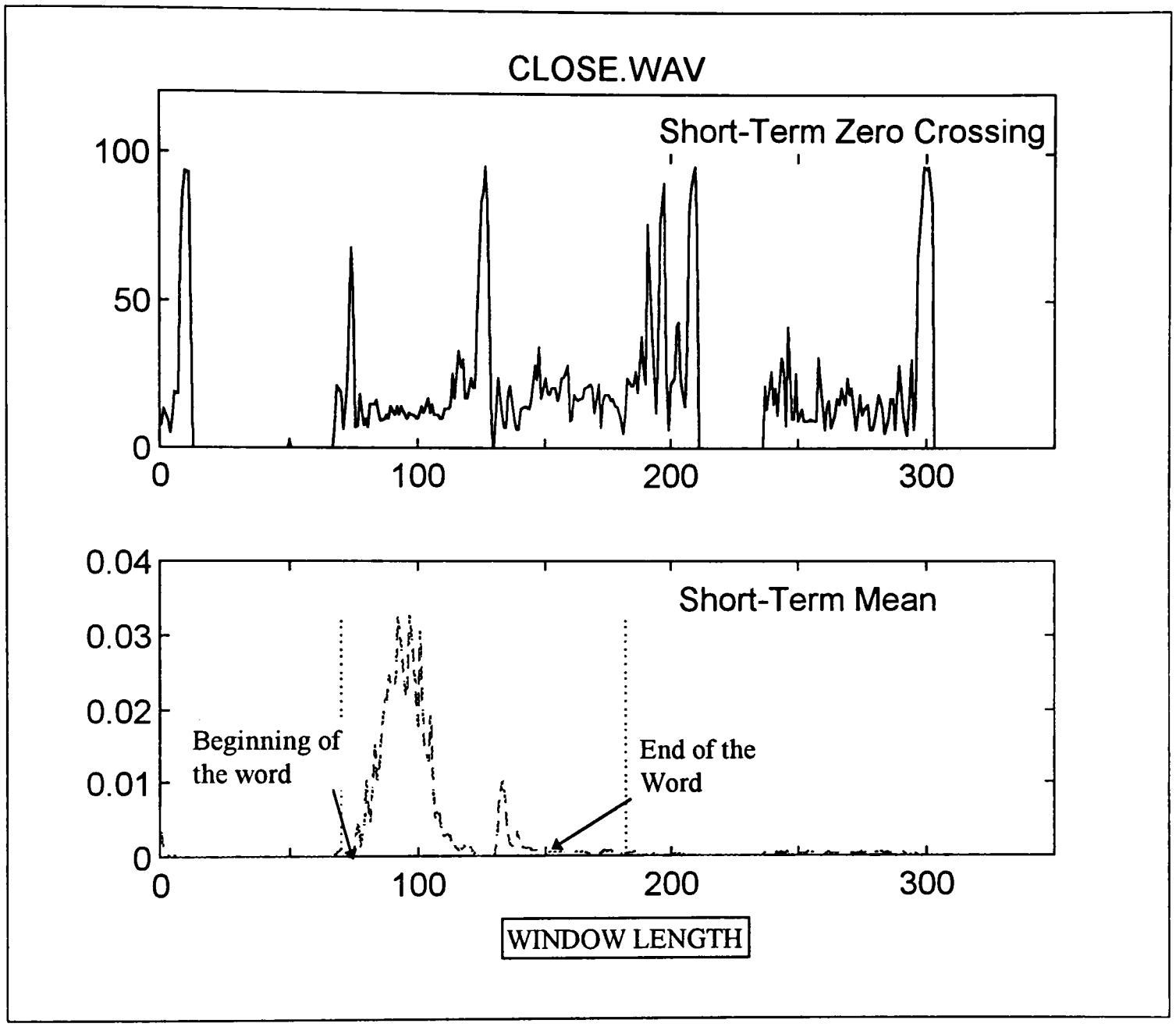


Fig. 5.3 Detection of the beginning and ending of the word CLOSE

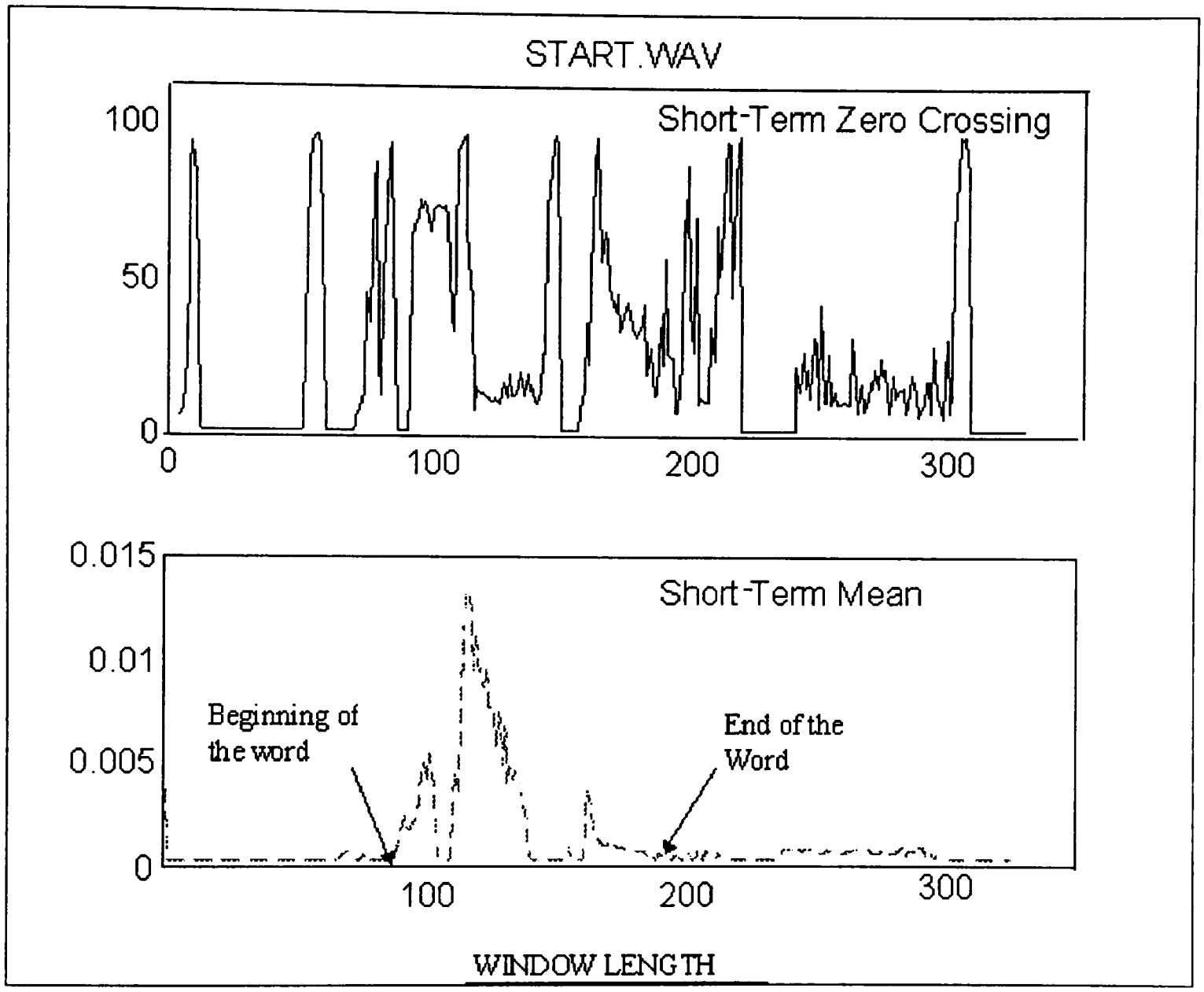


Fig. 5.4 Detection of the beginning and ending of the word START

for isolated spoken words. As seen from the Short-Term Mean in Fig.5.4, the system would have located the endpoint of the word at the base of second peak rather than the third one. The adaptive threshold technique, explained earlier, helps in correctly recognizing the difference between intraword silence and actual silence, and performs accurate endpoint detection.

The speech waveform, after eliminating the silence periods prior to and after the beginning and endpoints, is ready for LPC-based feature analysis. To perform feature analysis, the speech signal is first preemphasized (to spectrally flatten the speech signal and to reduce computational instabilities associated with finite precision arithmetic [27]). The network for performing preemphasis has been described in section 3.3. To perform the LPC-feature analysis, the signal is blocked into N sample sections (frames), the frame size chosen for the system of Fig. 5.2 is 45 ms, i.e., N=300 for a 10 kHz sampling rate. Consecutive frames are spaced M samples apart, where M has been chosen to be 15ms. Since $M < N$, there is an overlap between adjacent frames, which provides smoothing between vectors of feature coefficients. A windowing function is then applied to each of the N frames of data to smooth the abrupt variations at both frame edges. The windowing function used, shown in Fig. 5.5, is the Hamming Window of size 300. LPC analysis generates a set of LPC coefficients for each windowed frame of the signal. The order of LPC analysis, p, indicates the number of coefficients it generates. LPC coefficients are basically a representation of the envelope of the signal spectrum. The higher the order of analysis, the better the coefficients represent the signal spectrum. This is evident from

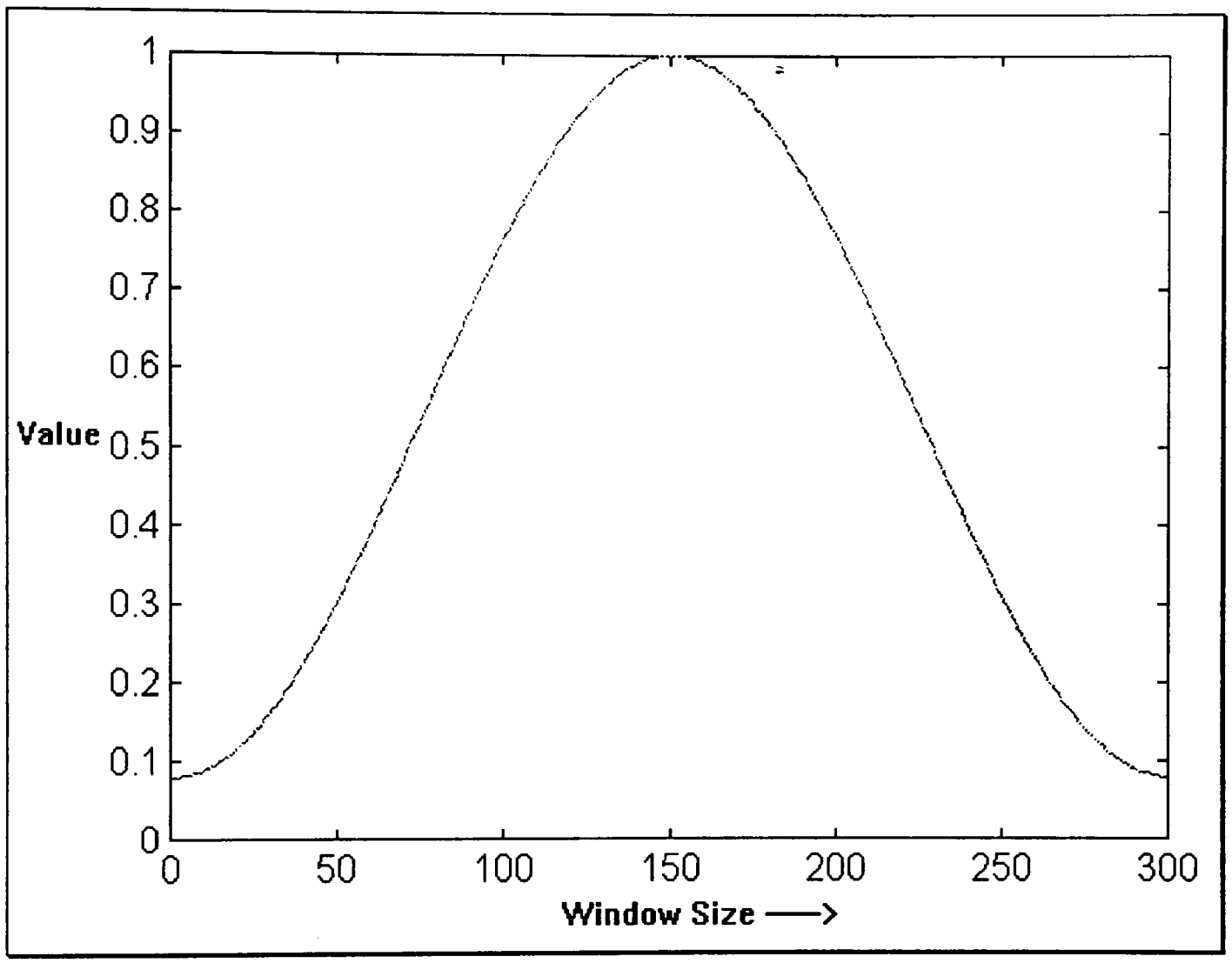
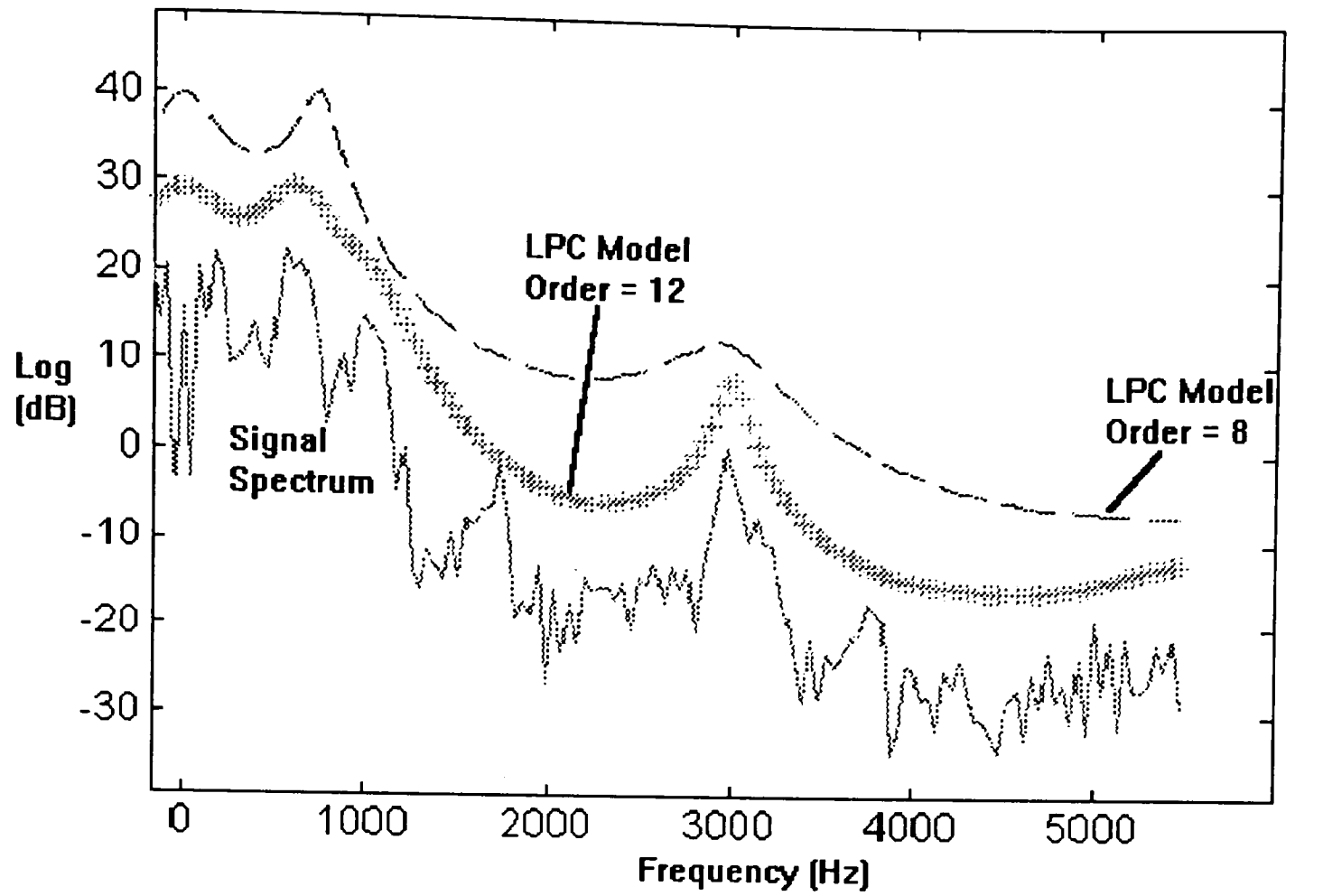


Fig. 5.5 Hamming Window

Figures 5.6 and 5.7, where the log spectrum (as measured via an FFT) of the coefficients is imposed over the signal spectrum of the frame that generated the coefficients. It is seen that the spectrum due to the LPC analysis of order 12, satisfactorily follows the spectral envelope for the frame. The spectrum for the order 14 does not provide a substantial increase in signal information over that for order 12. On the other hand, the LPC model for order 8 does not provide adequate information about the signal spectrum. Thus, keeping the LPC order limited at $p=12$ helps in conveying relevant information about the signal and at the same time provides better computational efficiency. At the end of LPC analysis, there is a set of feature vectors for each word that represent each frame of the input speech.

5.5 Training and recognition of speech data using Vector Quantization

The final stage in the SR system shown in Fig. 5.2 is the training and recognition stage. In the training phase, the system is actually trained with a specific set of reference words from the vocabulary. This includes generation of a reference database using codebooks, which will be used as a lookup during the recognition phase. To perform recognition, given the codebook of test utterances, the SR system has to determine which of the codebooks from the reference database is the best choice for the test word.



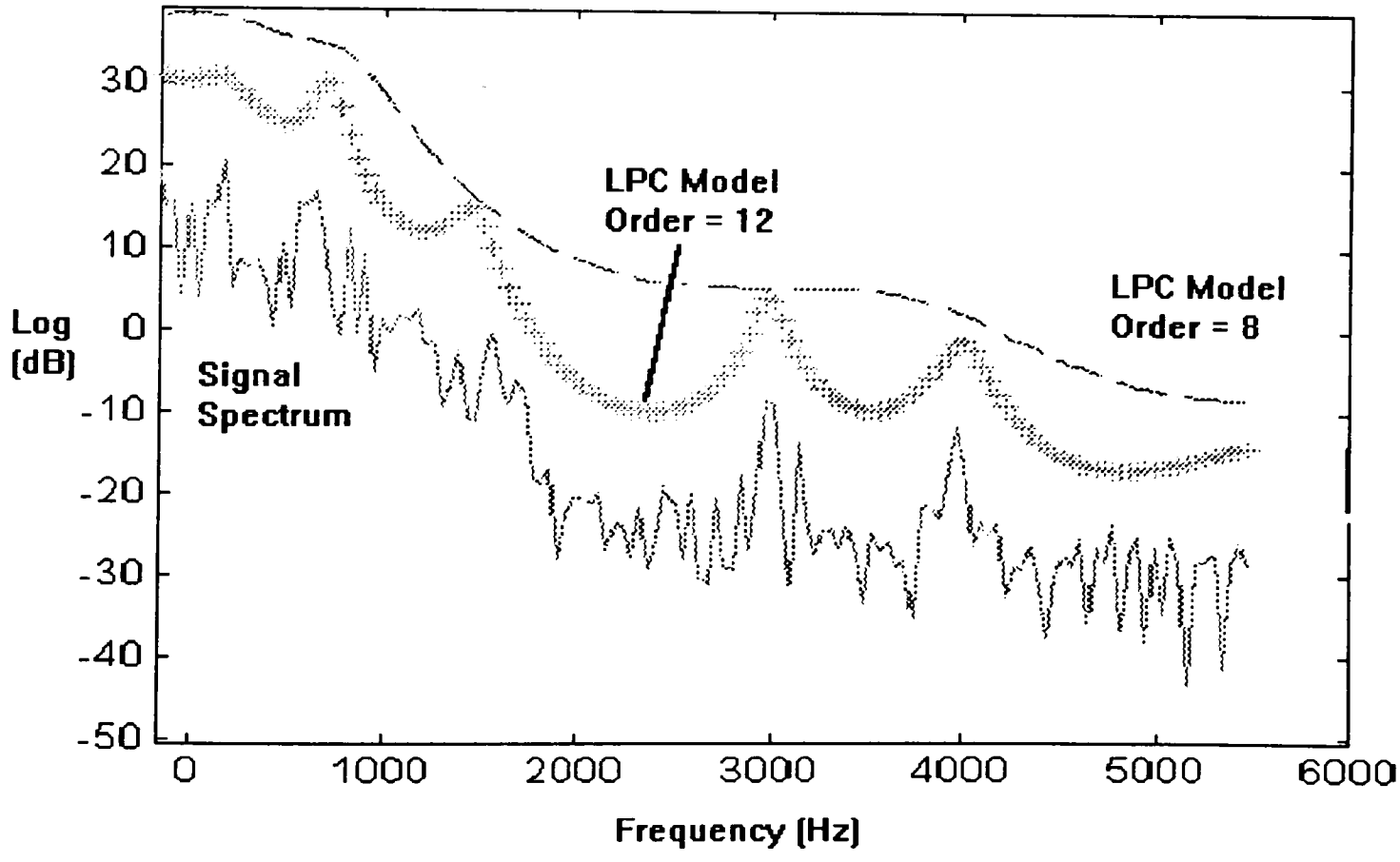
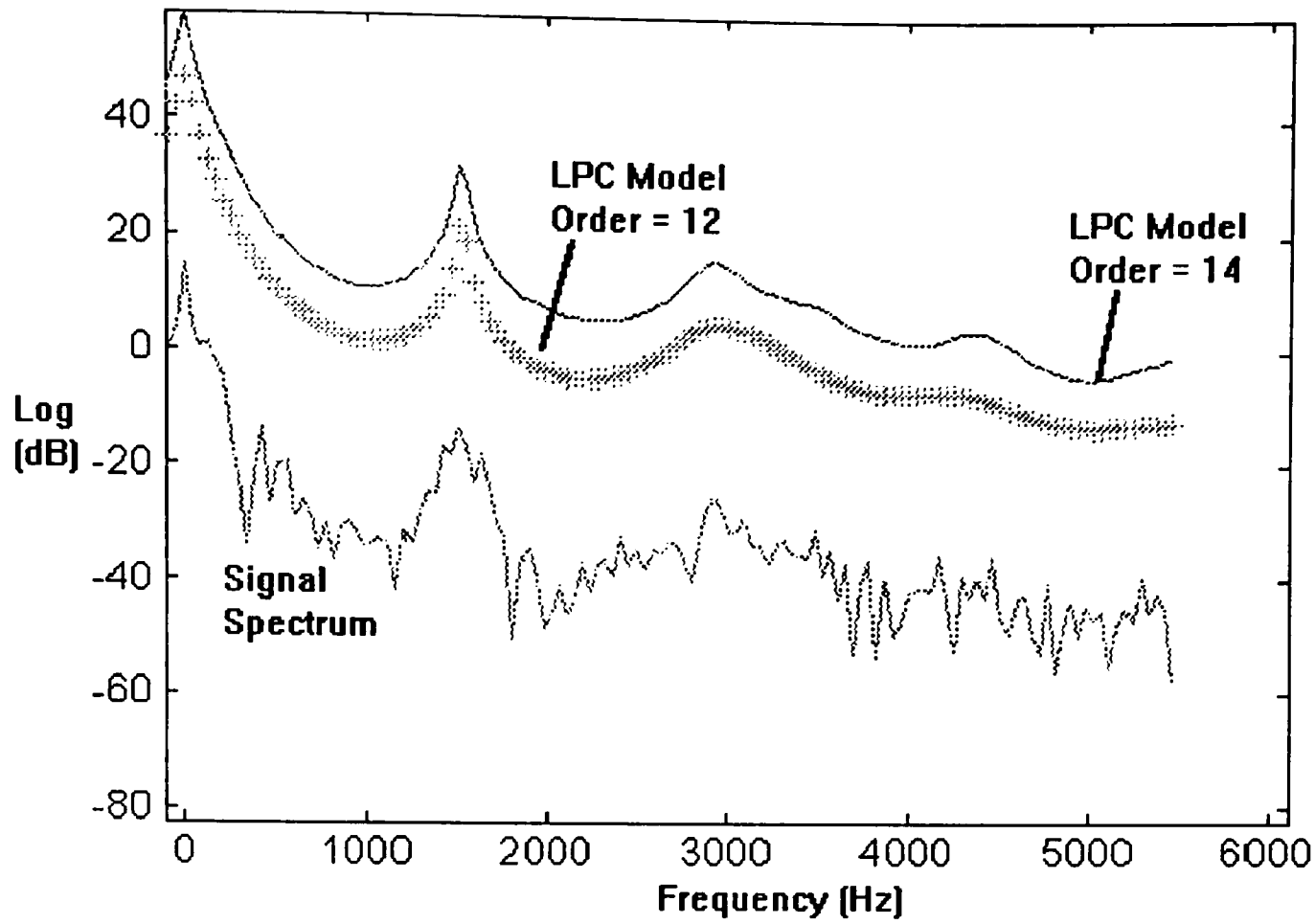


Fig. 5.6 Approximation of signal spectrum by LPC parameters



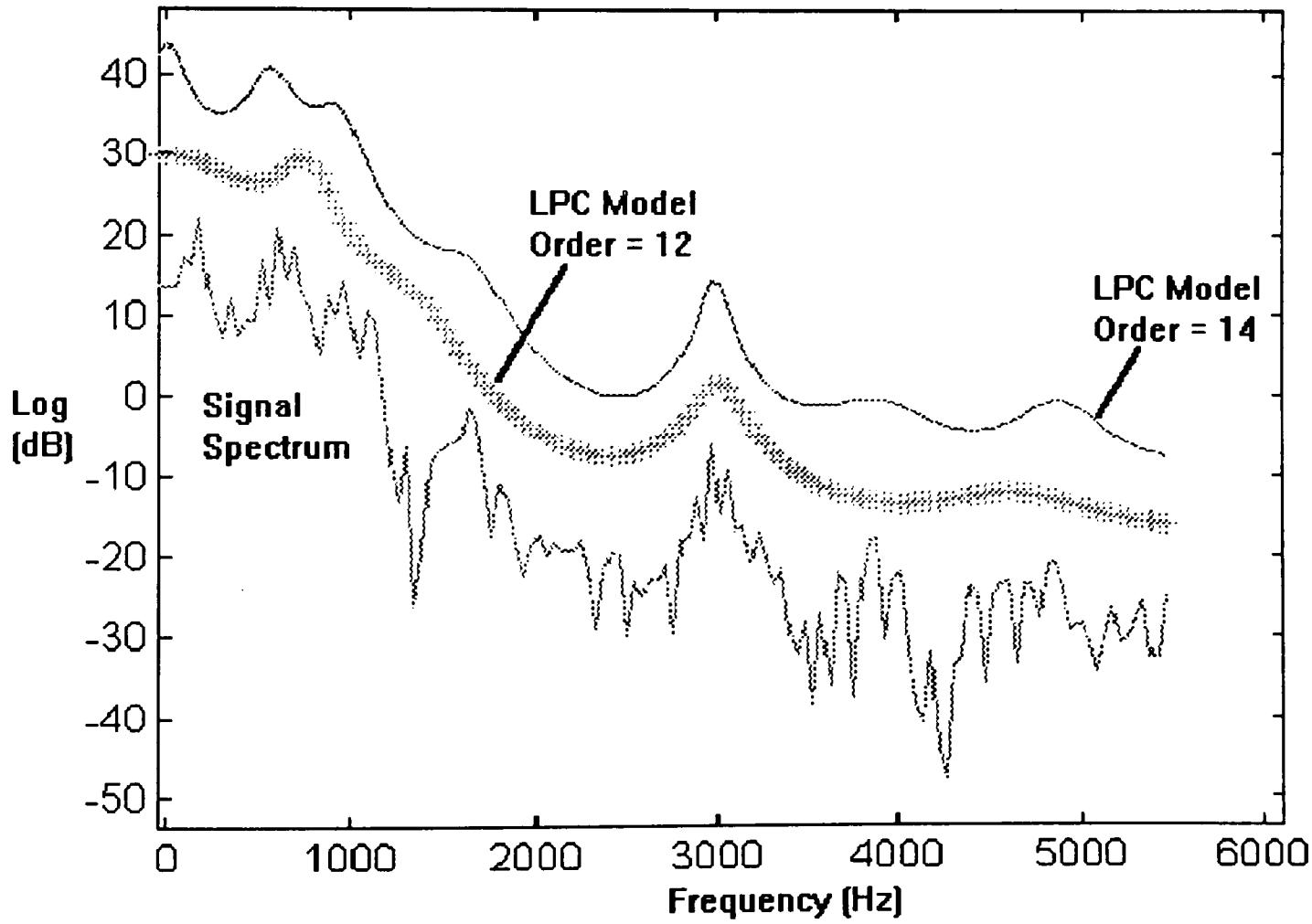


Fig. 5.7 Approximation of signal spectrum by LPC parameters

5.5.1 Training of Speech Data

Codebook generation was performed using various sets of training data from the original database. The original database consists of eight words namely: Close, Forward, Go, Start, Open, Turn, Right, and Left, each spoken 3 times by eight speakers. The above set of utterances was used to generate "classes". A "class" involves LPC coefficients of several repetitions of a word arranged in a back-to-back fashion. Thus, in this case, there are eight classes for eight words. The size of the "class" depends on the number of words in that class and the number of speakers used for training. The training sets for several experiments performed in this project depended upon the vocabulary size, the speaker population, and the type of VQ (i.e., Normal or Segmental). A codebook size of 32 codewords and a distortion value of 0.05 was chosen for the Vector Quantization technique implemented in the thesis.

The five main sets of training data that were used throughout this thesis are described below:

SET I. This set of data has been used for the purpose of speaker dependent speech recognition. In this set the entire vocabulary is classified on the basis of individual speakers, thus arriving at eight subsets of training data.

SET II. This set had all the words in the vocabulary including every utterance of each of the eight speakers. The codebook generation for this set of data was performed using normal Vector Quantization.

SET III. This set eliminates 2 words from the vocabulary of 8 words. This elimination is performed using a zero-crossing based algorithm, that selects six words from a vocabulary of eight words. The selected words provide the least distortion, when compared to the two words that were dropped out of the vocabulary. The number of repetitions of each word is the same as in the first set. The codebook generation for this set of data was performed using normal Vector Quantization.

SET IV. This set had all the words in the vocabulary, but in this set, speech data from a few of the speakers is left out of the training phase, so that it could be used during the recognition phase.

SET V. This set, unlike set III eliminates 2 words from the vocabulary of 8 words. The number of repetitions of each word is the same as in the first set. In this set, data from two of the speakers is left out from the training phase.

Sets II and III were used for Multiple Speaker speech recognition, while Sets IV and V were employed for Speaker Independent speech recognition. The need for the kind of training sets derived in Sets III and V is shown by analyzing the results of Sets II and IV. For each of the words included in each of the above sets, one or more repetitions of each word were left out, so that they could be used during the recognition phase.

After developing these sets of training data, the system training, as explained in earlier chapters, was initiated. The result of the training phase was a set of codebooks that was stored in a *reference database.* For each set of data, the vector quantization algorithm was executed and the results were stored for use in the recognition phase. Fig.5.8 through

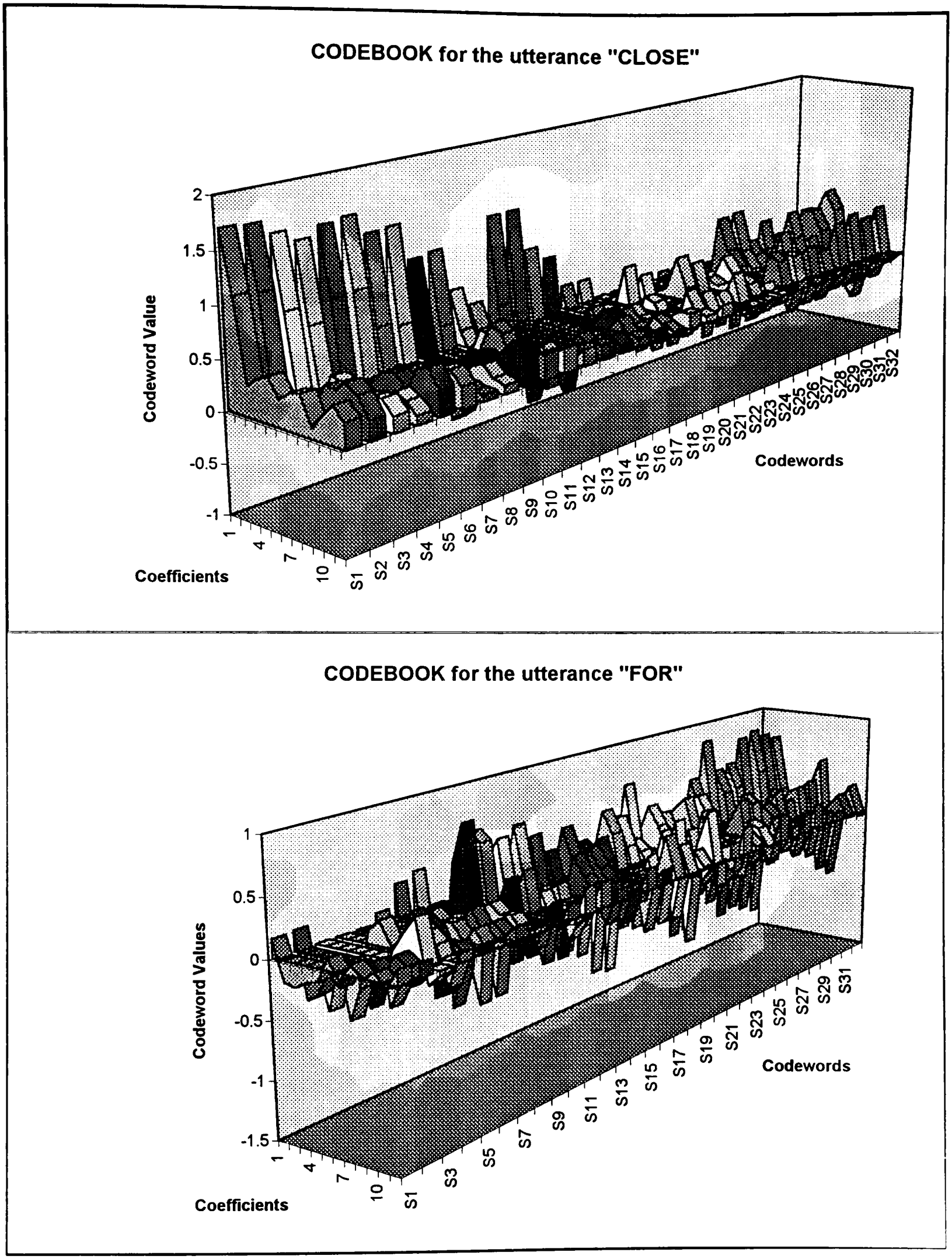


Fig. 5.8 Codebooks for tthe words "CLOSE" and "FOR"

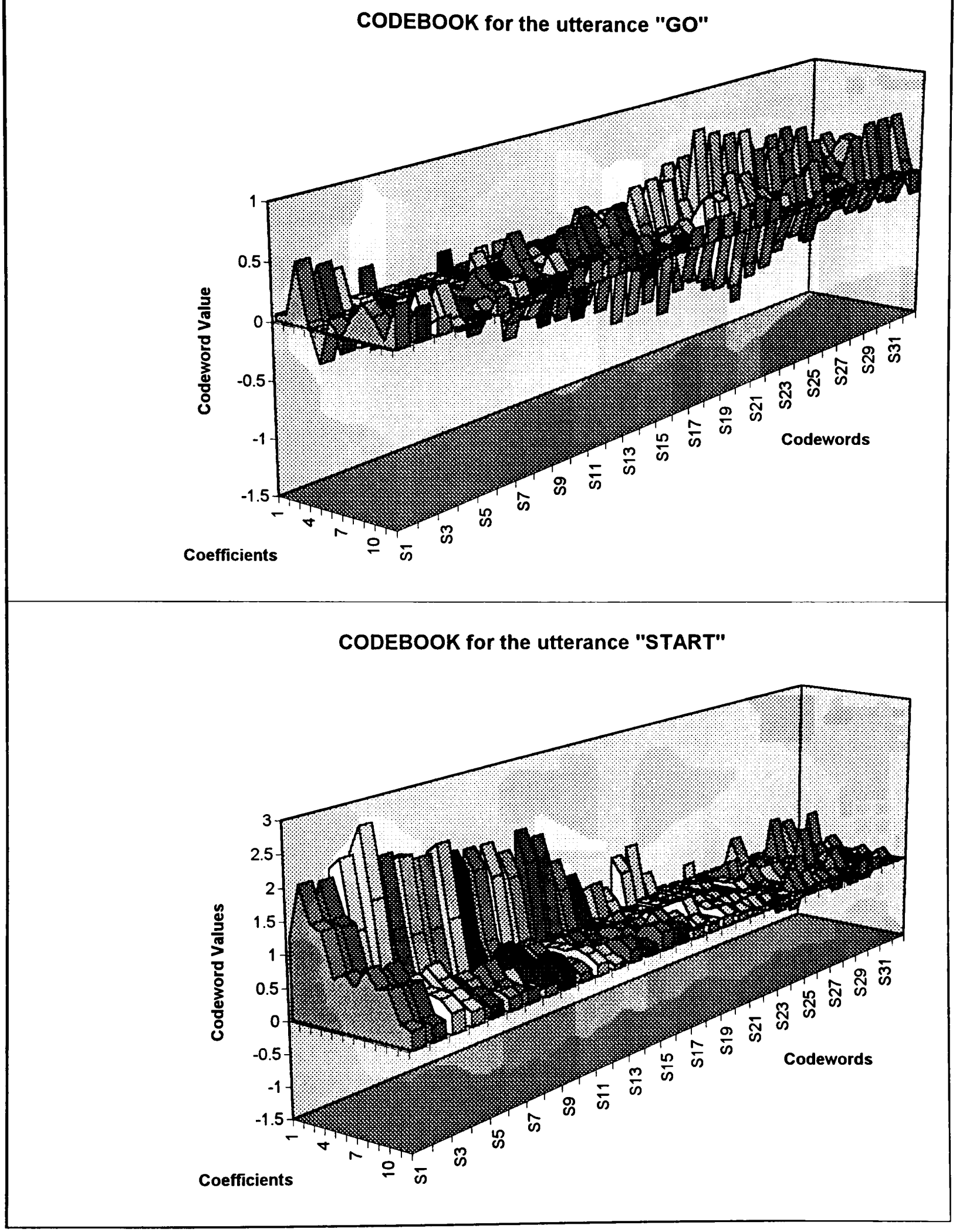


Fig. 5.9 Codebooks for the words "GO" and "START"

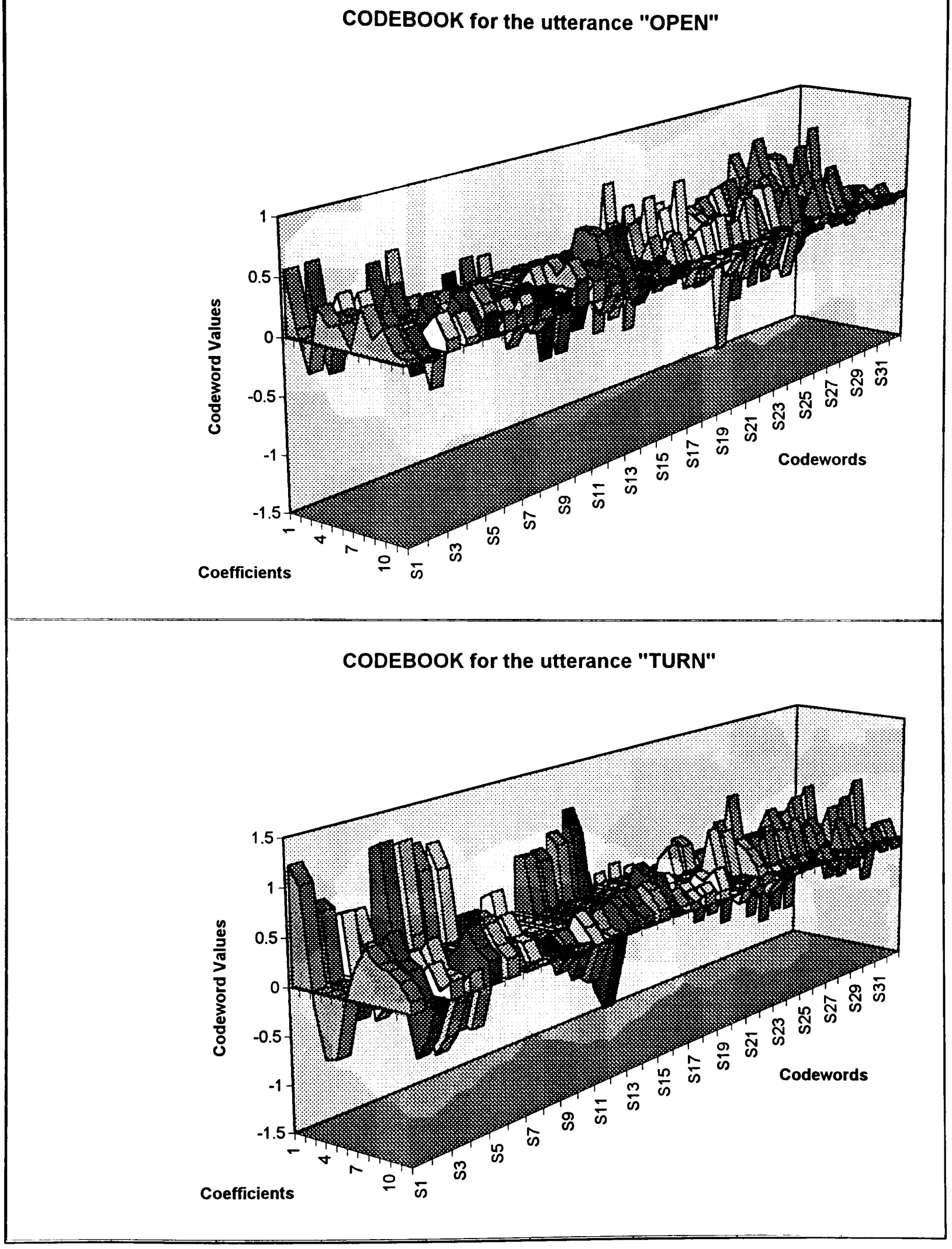


Fig. 5.10 Codebook for the words "OPEN" and "TURN"

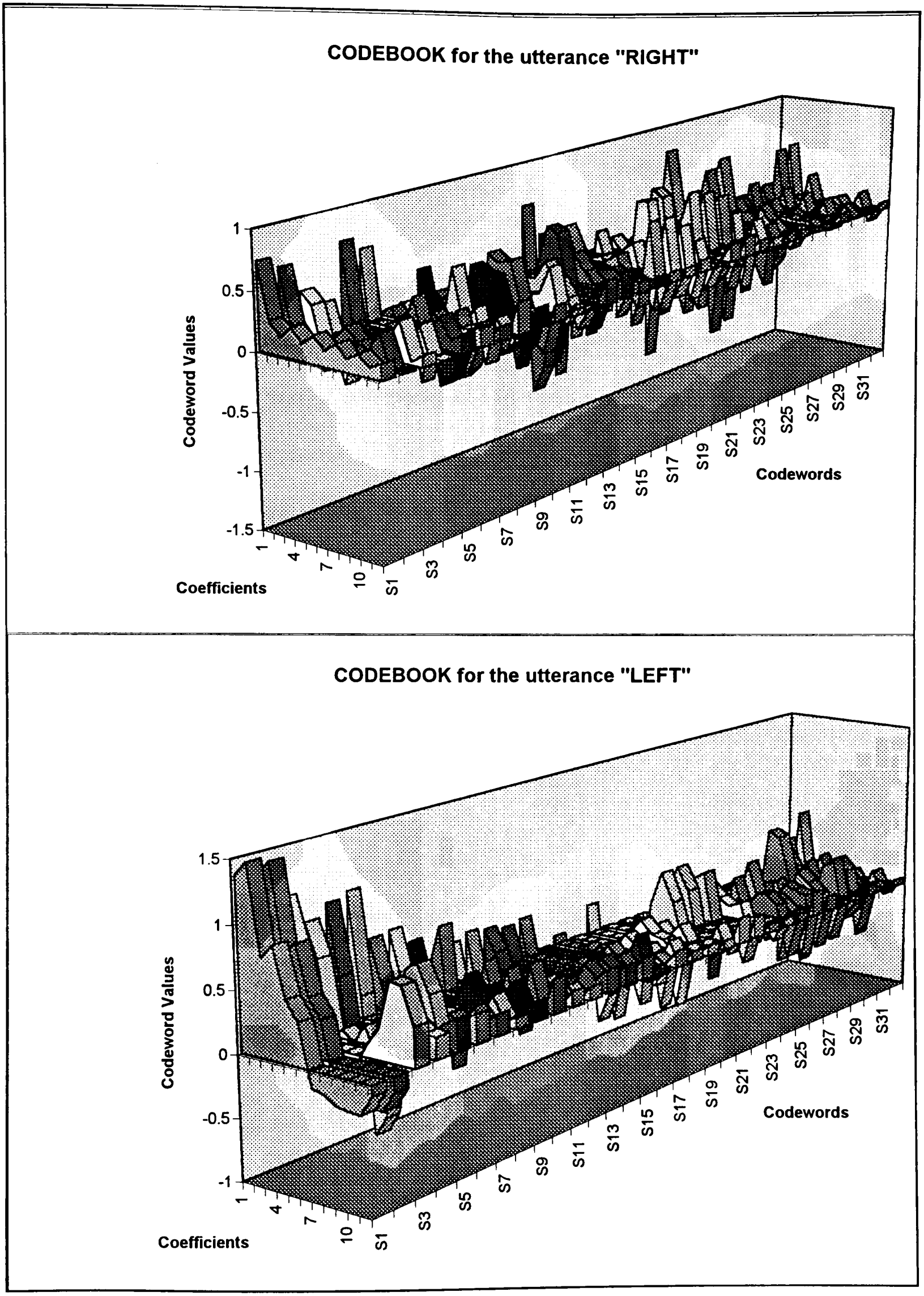


Fig. 5.11 Codebooks for the words "RIGHT" and "LEFT"

Fig. 5.11. indicate how the codebook for each of the eight words appears in 3 dimensions. These particular codebooks were generated using the training Set V.

5.5.2 Recognition Phase

Once the codebook for each of the five training sets is ready, the next step is to perform speech recognition. This is where an unknown utterance (i.e., the utterance left out during the training phase) is provided as an input to the system of Fig. 5.2. The system performs data acquisition and signal processing on the speech data and then performs Vector Quantization on the set of LPC coefficients. The Vector Quantizer operates here in the recognition mode, i.e., a comparison is performed between the LPC coefficient set of the input word and each of the codebooks in the *reference database.* The details of the recognition mode have been discussed in an earlier chapter. The result is a recognition index that is indicative of the classification of the input word on the basis of the *reference database.*

5.5.3 Results of Speech Recognition

The Speech Recognition system was implemented in phases, on each of the five sets of training data starting with the first set.

5.5.3.1 Recognition for Set I

The results of SR performed on set I are shown in Table 5.1. This table is actually applicable for each of the eight speakers since all of them achieved the same results. Table 5.1 is actually a confusion matrix, which indicates the number of successful hits for each word and the word that it was mistaken for if it missed. As the results indicate, a 100% accuracy has been achieved for Speaker Dependent SR recognition. This means that each of the 24 words spoken by all of the eight speakers were recognized accurately.

5.5.3.2 Recognition for Sets II and III

The recognition performed using Sets II and III is a case of Multiple Speaker speech recognition. This means that all the words spoken by the entire speaker population are included in the set. The codebook thus generated from the training phase includes representations of utterances from each of the six speakers. This should aid in the recognition process for obvious reasons. Actually recognition using the training Sets II and III were performed successively since both the sets were based on multiple speaker criteria. The results of speech recognition on both the training Sets II and III are indicated in Table 5.2 and Table 5.3. As seen from Table 5.2, the Set II generated an overall accuracy of 92% where six of the eight words had an individual accuracy above 99%, which is a more than satisfactory result. The interesting part is that the results of Table 5.3. indicate an overall accuracy of 97%. This improvement in accuracy can be attributed to the elimination of two words, which produced the most distortion when compared to

Table 5.1 Confusion matrix for speaker independent recognition

Speaker Dependent Recognition Vocabulary: Extensive VQ: Normal

	CL	FO	GO	ST	OP	TU	RI	LE	% Accuracy
CL	**3**								**100**
FO		**3**							**100**
GO			**3**						**100**
ST				**3**					**100**
OP					**3**				**100**
TU						**3**			**100**
RI							**3**		**100**
LE								**3**	**100**

Trained Individually for speakers : 1-8 Overall Recognition Accuracy: 100%
Testing performed individually speakers : 1-8

Table 5.2 Confusion matrix for Multiple Speaker recognition using extensive vocabulary

	CL	FO	GO	ST	OP	TU	RI	LE	% Accuracy
CL	**23**		**1**						**95**
FO		**24**							**100**
GO		**2**	**22**						**92**
ST				**19**			**4**	**1**	**79**
OP	**1**		**1**		**22**				**92**
TU				**1**		**23**			**95**
RI				**1**			**20**	**3**	**83**
LE			**1**					**23**	**95**

Overall Accuracy: 92%

Table 5.3 Confusion matrix for Multiple Speaker recognition using limited vocabulary

	CL	FO	ST	OP	TU	LE	% Accuracy
CL	**24**						**100**
FO		**24**					**100**
ST			**22**		**2**		**92**
OP	**1**	**1**		**22**			**92**
TU			**1**		**23**		**95**
LE						**24**	**100**

Overall Accuracy: 96%

the remaining six. In this case, the words turned out to be 'Go' and 'Right'. Incidently, it can be seen from Table. 5.2 that these were two of the few words that had a relatively low accuracy and were most mistaken for some other word. Thus the system performed a good task of Multiple Speaker speech recognition.

5.5.3.3 Recognition for Set IV

The next recognition task is one of the most challenging ones for any speech recognition system. The task was to perform Speaker Independent speech recognition using the Sets IV and V. To start with, the training phase is executed on training Set IV, using the *normal* VQ technique. The number of speakers used in the training set were 1 through 4. Once the system was trained, it was tested with all the words spoken by speakers 5 through 8. The results are tabulated in Table 5.4 in the form of a confusion matrix. The results indicate that system had an accuracy of 66% for the speaker independent case. Table 5.5 used the same setup except for the number of speakers used for training, which was speakers 1 through 6. Even though the number of speakers for the training set was increased, the results did not show any great improvement. This probably has to do with the inherent quantization structure which renders the addendum not useful.

Substantial changes can be seen in the above results when the the same data set IV was used for training with the *Segmental* VQ approach rather than the *normal* approach. The results for the above implementation are shown in Table 5.6. Tables 5.6 and 5.7 show an 8% increase in recognition accuracy, over the normal VQ approach, when speakers 1

Table 5.4 Confusion matrix for recognition using Normal VQ with limited training speakers

	CL	FO	GO	ST	OP	TU	RI	LE	% Accuracy
CL	**12**								**100**
FO		**8**			**1**		**3**		**66**
GO			**5**		**3**		**4**		**43**
ST				**9**	**1**	**2**			**75**
OP			**4**		**5**		**3**		**43**
TU			**3**	**1**		**5**		**3**	**43**
RI							**12**		**100**
LE						**3**	**1**	**8**	**66**

Overall Recognition Accuracy: 66 %

Table 5.5 Confusion matrix for recognition using Normal VQ with maximum training speakers

	CL	FO	GO	ST	OP	TU	RI	LE	% Accuracy
CL	**6**								**100**
FO		**5**	**1**						**83**
GO			**3**		**1**		**2**		**50**
ST				**3**		**2**		**1**	**50**
OP			**2**		**2**	**2**			**33**
TU						**4**		**2**	**66**
RI							**5**	**1**	**83**
LE							**1**	**5**	**83**

Overall Recognition Accuracy: 68%

through 4 are used for training, while a 10% increase in recognition accuracy is observed for the case where speakers 1 through 6 were used for training. This increase in accuracy can be attributed to the analysis technique of the segmental VQ, which splits the speech data into segments, thus preserving the spectral characteristics in a better way [27] compared to the normal VQ approach.

5.5.3.4 Recognition for Set V

For Set V of training data, two words from the vocabulary were eliminated using the elimination algorithm explained earlier. The training phase of the system was implemented using the Segmental VQ approach. The drastic effects of the elimination of two words can be seen in Table 5.8, where data from speakers 1 through 4 was used for training the system and the remaining data was used for recognition. An accuracy of 91% is definitely a good recognition rate for a Segmental VQ based SR system. This accuracy increases to 95%, as shown in Table 5.9, when data from speakers 1 through 6 is used for training, while speakers 7 and 8 are used for recognition purposes.

Table 5.6 Confusion matrix for recognition using segmental VQ with maximum test speakers

	CL	FO	GO	ST	OP	TU	RI	LE	% Accuracy
CL	**12**								**100**
FO		**11**					**1**		**90**
GO			**5**		**3**		**4**		**43**
ST				**9**	**1**	**2**			**75**
OP			**3**		**5**		**4**		**43**
TU						**9**		**3**	**75**
RI							**12**		**100**
LE						**3**	**1**	**8**	**66**

Overall Recognition Accuracy: 72%

Table 5.7 Confusion matrix for recognition using segmental VQ with minimum test speakers

	CL	FO	GO	ST	OP	TU	RI	LE	% Accuracy
CL	**6**								**100**
FO		**5**					**1**		**83**
GO			**3**		**1**		**2**		**50**
ST				**5**		**1**			**83**
OP			**2**		**4**				**66**
TU						**5**		**1**	**83**
RI							**5**	**1**	**83**
LE							**1**	**5**	**83**

Overall Recognition Accuracy: 78%

Table 5.8 Confusion matrix for recognition using segmental VQ with limited vocabulary and maximum test speakers

	CL	FO	ST	OP	TU	LE	% Accuracy
CL	**12**						**100**
FO		**9**		**2**		**1**	**86**
ST			**9**		**3**		**75**
OP				**12**			**100**
TU					**10**	**2**	**86**
LE						**12**	**100**

Overall Recognition Accuracy: 91%

Table 5.9 Confusion matrix for recognition using segmental VQ with limited vocabulary and minimum test speakers

	CL	FO	ST	OP	TU	LE	% Accuracy
CL	**6**						**100**
FO		**6**					**100**
ST			**5**		**1**		**84**
OP				**6**			**100**
TU					**5**	**1**	**84**
LE						**6**	**100**

Overall Recognition Accuracy: 95%

5.6 Training and Recognition of Speech Data using Zero-Crossing

The second algorithm used to perform Speech Recognition was the Zero-Crossing algorithm. A detailed explanation of the Zero-Crossing technique is given in section 4.5.2 of Chapter 4. During the training phase of the system, a reference database was created for three sets of training data. The sets were generated on the basis of the speakers that were going to train the system and the speakers that were going to test the system.

Set I: Speakers used to generate reference database : 1,6

Speakers used to perform recognition : 2,3,4,5

Set II: Speakers used to generate reference database : 2,5

Speakers used to perform recognition : 1,3,4,6

Set III: Speakers used to generate reference database : 3,4

Speakers used to perform recognition : 1,2,5,6

5.6.1 Training and Recognition Phase

The system was trained using the training sets described above. Training the system in this case meant generating several sets of parameters such as zero-crossing, mean, energy and standard deviation for each word in the reference database. These parameters were then utilized to generate the "evaluation vectors" that would represent the entire class of words. During the recognition phase, the word to be recognized is split into frames and the parameters described during the training phase are determined for each

frame. Using these parameters, the "evaluation vector" for the test word is generated. This vector is then compared to the vectors in the reference database. The reference database vector that generates the smallest Euclidean distance with the test vector is declared as the "Recognized Word." Since the speakers used during the testing are different than those used for generating the reference database, the system thus developed is a Speaker-Independent speech recognition system. The results for the three sets of training data are displayed in Tables 5.10 through 5.12. Table 5.13 gives results for the recognition using set I with limited vocabulary.

The results derived during the above implementation indicate that the zero-crossing approach had an overall recognition accuracy of 69% for extensive vocabulary and 89% for limited vocabulary approach. Thus the zero crossing approach was as effective as the VQ approach as far as the recognition accuracy is concerned.

5.7 Comparison of the Vector Quantization Approach to the Zero-Crossing Approach for Speech Recognition

The results in the previous two sections give an indication of how certain variations of the vector quantization algorithm can prove to be better than their zero crossing counterpart. As evident from Tables 5.4 through 5.6 and Tables 5.10 through 5.12, the results of Normal VQ implementation are pretty much at par with that of the Zero Crossing approach. The significant change in accuracy occurs when a variation of the VQ approach, the segmental VQ, is implemented on the training and recognition sets of data. We get an improvement of almost 10% in the recognition accuracy over the zero

Table 5.10 Recognition Results for SET I

	CL	FO	GO	ST	OP	TU	RI	LE	% Accuracy
CL	**10**					**1**	**1**		**83**
FO	**2**	**4**	**2**			**3**		**1**	**33**
GO			**12**						**100**
ST				**12**					**100**
OP					**12**				**100**
TU				**1**		**6**	**2**	**3**	**50**
RI				**4**			**5**	**3**	**42**
LE							**2**	**10**	**83**

Overall Accuracy: 73%

Table 5.11 Recognition Results for SET II

	CL	FO	GO	ST	OP	TU	RI	LE	% Accuracy
CL	**9**			**1**			**2**		**75**
FO	**1**	**6**				**4**		**1**	**33**
GO			**12**						**100**
ST				**9**		**2**		**1**	**75**
OP					**12**				**100**
TU	**1**				**3**	**4**	**1**	**3**	**33**
RI				**4**			**6**	**2**	**50**
LE						**1**	**1**	**10**	**83**

Overall Accuracy: 68%

Table 5.12 Recognition Results for SET III (with extensive vocabulary)

	CL	FO	GO	ST	OP	TU	RI	LE	% Accuracy
CL	**11**				**1**				**91**
FO	**2**	**4**	**2**			**3**		**1**	**33**
GO			**12**						**100**
ST				**8**			**2**	**2**	**66**
OP					**12**				**100**
TU						**4**	**5**	**3**	**33**
RI				**2**		**2**	**6**	**2**	**50**
LE						**1**		**11**	**91**

Overall Accuracy: 70%

Table 5.13 Recognition Results for SET III (with limited vocabulary)

	CL	FO	ST	OP	TU	LE	% Accuracy
CL	**8**				**1**		**88**
FO	**2**	**7**					**78**
ST			**8**		**1**		**88**
OP	**1**		**1**	**7**			**78**
TU			**1**		**8**	**1**	**88**
LE			**1**			**8**	**88**

Overall Accuracy: 89%

crossing approach, which is evident from Table 5.7. This increase in accuracy can be attributed to the fact that in segmental VQ we are generating the codebook for smaller sets of frames rather than all the frames of the entire word. This helps in generating a codebook that resembles the speech waveform more closely than it did earlier. The recognition accuracy for segmental VQ after implementing the "Elimination algorithm" ends up at a respectable 94%. This indicates a 5% increase over the zero crossing approach (with limited vocabulary) to recognition.

The second factor used to compare the two algorithms for speech recognition has been the speed of recognition. The time required for the zero-crossing, the normal VQ and the segmental VQ algorithms to perform speech recognition when implemented on the PC and the DSP have been tabulated for better readability.

Finally, the two algorithms were compared on the basis of the memory space the algorithms occupied for storage of the reference database and the algorithm code. The Vector Quantization and the Zero-Crossing techniques were implemented both on a PC and the DSP. The results shown below were the same for both these implementations (Table 5.14 and Table 5.15).

Table 5.14 Time requirements for system implementation

Type of Algorithm	Time required for SR when implemented on a PC	Time required for SR when implemented on a DSP
Zero Crossing	**10 seconds**	**1 seconds**
Normal VQ	**2 minutes and 30 seconds**	**3.5 seconds**
Segmental VQ	**3 minutes and 45 seconds**	**6 seconds**

Table 5.15 Memory requirements for system implementation

Type of Algorithm	Memory space occupied	
	Reference Database	Algorithm Code
Zero Crossing	**32 Kbytes**	**16 KBytes**
Vector Quantization	**200kbytes**	**30KBytes**

CHAPTER 6

CONCLUSION

This chapter summarizes the results of the speaker independent speech recognition system and gives some future directions that could make this system more versatile and robust.

6.1 Results

The system has been implemented using a zero crossing algorithm and variations of the Vector Quantization algorithm, the results of which have been indicated in Chapter 5. Summarizing the results for the recognition algorithm implemented on the DSP leads to Table 6.1.

As far as recognition accuracy is concerned, the above results suggest that the VQ algorithm for SR gives satisfactory results and the results with SVQ (limited vocabulary) seem to be at par with the ZCT. Nevertheless, comparing the recognition time and memory requirements for VQ and SVQ with that of the ZCT, we observe two facts

-- The memory required for data and module storage in the ZCT approach is 5 times less than that required for VQ and 12 times less than that required for SVQ.

-- As far as recognition speed is concerned, the ZCT is twice as much faster that its VQ counterpart and four times faster than the SVQ algorithm.

Table 6.1 Results of the Speech Recognition System

Algorithm Type	Accuracy %	Time (seconds)	Memory Requirements
Vector Quantization (VQ)			
- Speaker-dependent	100	3.5	226 Kbytes
- Multiple Speaker	92	3.5	226 Kbytes
- Speaker independent	68	3.5	226 Kbytes
Segmental VQ			
- Speaker independent	78	6	624 Kbytes
- Speaker independent (Limited Vocabulary)	94	6	624 Kbytes
Zero-crossing technique (ZCT)			
- Speaker independent	73	1	48 Kbytes
- Speaker independent (Limited Vocabulary)	89	1	48 Kbytes

6.2 Selection of SR Module

The goal of this thesis was to develop a "small, fast, speaker independent and cost efficient Speech Recognition System." Of the three options for the speech recognition module, i.e., VQ, SVQ, and ZCT, the one that gives good accuracy is the SVQ approach. Nevertheless, given the above figures, the VQ and SVQ approach relatively take up a lot of memory space, which in turn reduces the cost-effectiveness of the system. Second, the

time required for an utterance to be recognized is quite more than what one would expect from an SR system installed in a daily use consumer appliance (e.g., microwave).

Finally, the zero crossing approach resulted in an accuracy of 73% (with extensive vocabulary) while the same approach with limited vocabulary provided an accuracy of 89%, which is an improvement of 16%. Speech recognition using this approach, resulted in a recognition time of approximately 1 second. The memory required to implement this algorithm is just 46Kbytes.Thus for the present SR system, where space is a premium commodity, cost-efficiency, imperative, and reduced recognition time, essential, the ZCT approach seems to be an appropriate choice.

6.3 Future Work

One of the areas where future work can be concentrated is the vocabulary generation. As evident from the results of both the VQ and ZCT algorithms, the proper choice of words from an vocabulary can prove to be the key for improved recognition accuracy of the system. The word pairs 'GO-RIGHT' and 'GO-OPEN' do not seem to have any similar features or phonetics, however, one was misrepresented for the other. The reason for this can be the similar set of spectral features, not evident in the audio regime, that each of these words could have in common. The elimination of these words from the vocabulary resulted in an increase in the accuracy. A way to perform efficient elimination would be to add a few synonyms for each word in the current vocabulary.

Then the "elimination algorithm" would have more flexibility in eliminating quite a few of the "problem" words.

In the current approach of Vector Quantization, the time axes for the two utterances are not aligned. This means that the length of spoken word does not matter. A technique that could be tried for recognition on the present data would be the Dynamic Time Warping (DTW) technique, discussed in Chapter 2. In this technique the time axes of the test and reference utterance are aligned or "warped" and a path corresponding to the similarity between the two words is plotted. Thus while comparing two words, the time axis of one is shrunk or extended to match with that of the other word. The DTW has proven to give good results [27] when employed for isolated word recognition.

As far as VQ is concerned, a codebook with energy information about the word could provide a better template of the word. This means that along with the LPC parameters, energy information on a frame to frame basis would form the input to the codebook generator. A better codebook for each word in the vocabulary could increase the recognition rate. The distortion measure could also be changed to improve the codebook. Measures such as Log likelihood ratio and Itakura-Saito can be employed in the VQ algorithm. These distortion measures could also help in developing a better codebook and inturn improve the accuracy.

Finally, the options for implementing the system on a different DSP should be considered. This could lead to a faster and more cost-effective SR system.

REFERENCES

[1] Juang B. H. " On the Hidden Markov Model and Dynamic Time Warping for Speech Recognition - A Unified View, " *The Bell System Technical Journal*, AT&T, 1984

[2] Baker J. K. " The DRAGON System - An Overview," *IEEE Transactions on Acoustics,Speech, and Signal Processing*, Vol. 23, February 1975b.

[3] Lowerre B. T. " The HARPY Speech Recognition System," PhD dissertation, Carnegie Mellon University, April 1976.

[4] Rabiner L. R., Levinson S. E. and Sondhi M. M., "On the application of Vector Quantization and Hidden Markov Models to Speaker-Independent, Isolated Word Recognition, " *The Bell System Technical Journal,* AT&T, 1983.

[5] Rabiner L. R., Wilpon J. G. and Soong, F. K., "High Performance Connected Digit Recognition Using Hidden Markov Models," *IEEE International Conference on Acoustics, Speech and Signal Processing*, April 1988.

[6] IBM Speech Recognition Group, "A Real-Time, Isolated-Word, Speech Reognition System for Dictation Transcription," *IEEE International Conference on Acoustics, Speech and Signal Processing,* March 1985.

[7] Deller John R. , John G. Proakis, and John H. L. Hansen. *Discrete Time Processing of Speech Signals.* New York: Macmillan Publishing Company, 1992.

[8] Parsons Thomas. *Voice and Speech Processing*. New York: McGraw-Hill, 1987.

[9] Haddad Richard A., Thomas W. Parsons. *Digital Signal Processing Theory, Applications and Hardware*. New York: Computer Science Press, 1991.

[10] Syrdal A., Bennett R., Greenspan S. *Applied Speech Technology.* Boca Raton, Florida: CRC Press, 1994.

[11] Dautrich Bruce A., Rabiner L. R. "On the effects of varying bank parameters on isolated word recognition," *IEEE Transactions on Acoustics, Speech and Signal Processing.* Vol ASSP-31, No. 4, pp 793-806, August 1983.

[12] Ito M. R., " Relationship between Zero crossing measurements for speech analysis and recognition," *The Journal of Acoustical Society of America*, vol. 51, no. 6, pp 2061-2, 1982.

[13] Kavaler R. A., Menahem L. Hy Murveit, Brodersen R. W., "A Dynamic Time Warp Integrated Circuit for a 1000-word Spech Recognition System." *IEEE Journal of Solid State Circuits*, Vol. sc-22, No. 1, pp3-14, February 1987.

[14] Sakoe Hiroaki, Seibi Chiba. "Dynamic Programming Algorithm Optimization for Spoken Word Recognition." *IEEE Transactions on ASSP*, Vol. ASSP-26, No.1, pp 43-49, February 1978.

[15] Furui Sadaoki. "Speaker-Independent Isolated-Word Recognition Using Dynamic features of spectrum." *IEEE Transactions on ASSP, Vol ASSP-34,* No. 1, pp 52-59, February 1986.

[16] Chen Wang, "Automatic Speech Recognition System Via Dynamic Time Warping" Rose-Hulman Institute of Technology, Master's Thesis, March 1995.

[17] Donaldson W. Robert and Cheong K. Gan "Adaptive Silence Detection for Speech Storage and Voice Mail Applications" *IEEE Transactions on ASSP*, vol 13 ,pp 924-25, 1988.

[18] Jesus Savage and Alistair Holden, "Combination of two methods for isolated word recognition," *IEEE Workshop on Robot and Human Communication*, Vol-12, pp 129-36, 1992.

[19] Jelinek F. , "The Development of an Experimental Discrete Dictation Recognizer," *Proceedings of the IEEE*, November 1985, pp 1616-24.

[20] Pallett D., Fiscus J., Garofolo J., "DARPA Research Management Benchmark Test Results," *Proceedings of the Speech and Natural Language Workshop*, February 1990, pp 298-305.

[21] Murveit H., Butzberger J., Weintraub M., " Speech Recognition in SRI's Resource Management and ATIS System, *Proceedings of the Speech and Natural Language Workshop*, February 1991, pp 94-100.

[22] Waibel A., Lee K. *Stochastic Approaches, Introduction.* San Mateo, CA: Morgan Kaufmann Publishers Inc., 1990, pp 263-65.

[23] Lesser V., Fennel R., Erman L., "The HEARSAY II Speech Understanding System," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol ASSP-23, February 1975, pp 11-24.

[24] Waibel A., Lee K. *Knowledge Based Approaches, Introduction.* San Mateo, CA: Morgan Kaufmann Publishers Inc., 1990, pp 198-202.

[25] Sakoe H. "Two level DP- Matching, a dynamic programming based pattern matching algorithm," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol ASSP-27, December 1979, pp588-595.

[26] Rabiner L. R. et al. "Speaker Independent Recognition of Isolated Words using clustering techniques," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol ASSP-27, August 1979, pp 336-49.

[27] Rabiner L. and Juang B.H. *Fundamentals of Speech Recognition.* New York: Prentice Hall Publishers, 1993.

[28] Levinson S.E. and Roe D.B., "A perspective on speech recognition," *IEEE Communications magazine*, Vol. 28, no.1, January 1990, pp 28-34

[29] Chan Y.T. et al. "A microcomputer based speaker independent isolated word recognition system," *Microcomputer Applications*, vol. 6, no. 1, pp 25-28.

[30] Chan C.K. and Lau Y.K., "Speech Recognition based on Zero Crossing Rate and Energy," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 33, no. 1, February 1985, pp 320-323.

[31] Cole R.A. et al. "Feature based speaker independent recognition of isolated english letters," *ICASSP*, vol. 6, pp 731-733.

[32] Itakura F, "Minimum prediction residual principle applied to speech recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 23, February 1985*, pp 57-72

[33] Shore J.E. and Burton D.K., "Discrete utterance speech recognition without time alignment," *IEEE Transactions on Information Theory*, vol. 29, no. 4, July 1983, pp 473-492.

[34] Linde Y., Buzo A., and Gray R., "An algorithm for Vector Quantization Design," *IEEE Transactions on Communications*, vol. 28, no. 1, January1980, pp 85-95

[35] Lee K.F., "Context dependent phonetic hidden markov models for speech recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 38, April 1990, pp 599-609

[36] Waibel A. et al., "Phoneme recognition using time delay neural networks," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, May 1989.

[37] Gray A. et al. "Distortion measures for speech recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing,* Vol.28, no. 4, August 1980, pp 367-376

[38] C. Rowden. *Speech Processing.* New York: McGraw Hill Book Company, 1991.

[39] *TMS floating point DSP / Optimizing C compiler.* Dallas, TX: Texas Instruments, 1995

[40] *TMS floating point DSP / Assembly language tools.* Dallas, TX: Texas Instruments, 1995

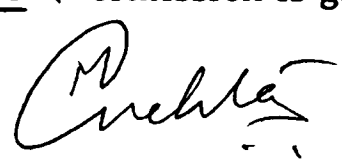
[41] *TMS floating point DSP / Peripheral control library.* Dallas, TX: Texas Instruments, 1995

[42] *Spirit-30 Users Manual.* Santa Clara, CA: Spirit Inc., 1993

PERMISSION TO COPY

In presenting this thesis in partial fulfillment of the requirements for a master's degree at Texas Tech University or Texas Tech University Health Sciences Center, I agree that the Library and my major department shall make it freely available for research purposes. Permission to copy this thesis for scholarly purposes may be granted by the Director of the Library or my major professor. It is understood that any copying or publication of this thesis for financial gain shall not be allowed without my further written permission and that any user may be liable for copyright infringement.

Agree (Permission is granted.)



Student's Signature

8/12/96

Date

Disagree (Permission is not granted.)

Student's Signature

Date