NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, tests publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30.

Canada

The University of Alberta

A Study of Isolated Word Recognition Using the General Instrument SP1000 Chip

by

Firat Uludamar

A thesis
submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree
of Master of Science

Department of Computing Science

Edmonton, Alberta
Fall, 1987

ISBN 0-315-40933-9

THE UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR: Firat Uludamar

TITLE OF THESIS: A Study of Isolated Word Recognition Using the General Instrument SP1000 Chip

DEGREE FOR WHICH THIS THESIS WAS PRESENTED: Master of Science

YEAR THIS DEGREE GRANTED: 1987

Permission is hereby granted to The University of Alberta Library to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

(Signed) Firat Uludamar

Permanent Address:
Ileri Sokak 12/10
Anittepe, Ankara
Turkey

Date: Sept 24/1987

THE UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research, for acceptance, a thesis entitled **A Study of Isolated Word Recognition Using the General Instrument SP1000 Chip** submitted by **Firat Uludamar** in partial fulfillment of the requirements for the degree of **Master of Science.**

.....
Supervisor

.....

.....

.....

Date: Sept 24/1987

The quality of this microfiche is heavily dependent upon the quality of the thesis submitted for microfilming.

Please refer to the National Library of Canada target (sheet 1, frame 2) entitled:

CANADIAN THESES

NOTICE

La qualité de cette microfiche dépend grandement de la qualité de la thèse soumise au microfilmage.

Veuillez consulter la cible de la Bibliothèque nationale du Canada (microfiche 1, image 2) intitulée:

THÈSES CANADIENNES

AVIS

ABSTRACT

This thesis concerns the recognition of isolated words of a designated speaker using the General Instrument SP1000 chip. Previous research on isolated word recognition is reviewed. Digital signal processing concepts with an emphasis on linear prediction techniques are surveyed. The SP1000 chip which can perform linear predictive analysis of an analog speech signal is examined in detail. Design and implementation of **IWRT** (Isolated **W**ord **R**ecognizer **T**rainer) is discussed.

The hardware front-end used in the implementation of the IWRT is MICROMINT's LIS'NER 1000 voice recognition board designed around the SP1000 chip. The LIS'NER 1000 board is interfaced with modifications to a Sun-2 Workstation in the Department of Computing Science at the University of Alberta and the voice input device is a headset style microphone.

The IWRT operates in one of the two modes: training mode and testing (recognition) mode. In training mode, the IWRT is used to create a vocabulary of words of a designated speaker. In testing mode, the IWRT chooses the word in the reference vocabulary which most closely matches the input test word.

Acknowledgements

The author wishes to extend his sincere appreciation to his supervisor, Dr. William W. Armstrong, for his guidance, encouragement, and financial support during the development of this thesis; committee members, Drs. Xiabo Li, Anton J. Rozsypal and Wlodek Dobosiewicz, for their time and suggestions; Dr. Lee J. White for chairing the committee.

A sincere special thanks goes to Dr. M. Tamer Ozsu for his support; Ms. Corinne L. M. Geis for her understanding and encouragement; Mr. T. Sabri Oncu for his help in typing the manuscript.

Thanks are also extended to Mr. Steve Sutphen for his software and hardware support throughout all stages of the project.

The author greatly appreciates the the financial support provided by the Department of Computing Science at the University of Alberta.

List of Figures

Table of Contents

Chapter 1

Isolated Word Recognition (IWR)

The major objective of this thesis is to lay the foundations for further work on speech recognition by designing and implementing a speaker-dependent IWR system. The field of speech recognition is an interdisciplinary one and draws heavily upon computer science, electrical engineering, linguistics, etc. The organization and content of the paper reflect the interdisciplinary nature of the field.

Chapter 1 attempts to give a general yet far-from-comprehensive view of IWR before delving into the more fundamental subjects that form the groundwork for IWR technology. Chapter 2 reviews the underlying theory and techniques of IWR with references to the particular hardware system which will be described in greater detail in Chapter 3. We will discuss the design and implementation of a speaker-dependent IWR system in Chapter 4. The results of this project and further research topics will be elaborated upon in the last chapter.

The hardware system used in the implementation is MICROMINT's LISTNER 1000 voice recognition board [12] designed around the General Instrument (GI) SP1000 chip. The LISTNER 1000 board is interfaced with modifications to a Sun-2 Workstation running the Sun UNIX† 4.2 Release 3.2 operating system. The SP1000 chip is defined as a "special file" to UNIX and a driver provides a high-level interface to the chip. The IWR software is written in the C programming language.

† Registered trademark of AT&T in the USA and other countries.

1.1. Introduction

In this section we will roughly outline the operation of a typical speaker-dependent IWR system. The major logical components of such a system will be discussed in the following sections. Most IWR systems have two distinct modes of operation:

1. *Training*: In this mode, a speaker says the words in the vocabulary to an input device, which is a headset-style microphone in our implementation. Features are extracted from the input speech signal over short intervals (10-40 msec). Reference template(s) or pattern(s) for each word are created using the features extracted, and saved so that they can be retrieved when needed in testing mode. Figure 1.1 illustrates the training mode.

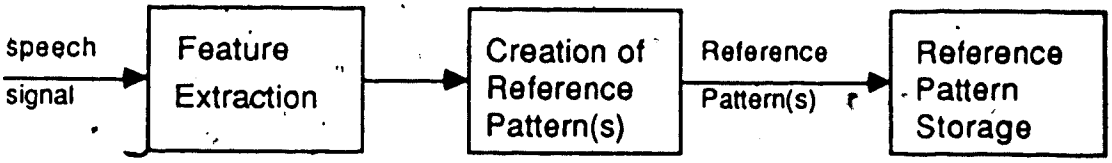

Figure 1.1 Training mode.

2. *Testing*: In this mode, the speaker utters a word, which is then analyzed, its features are extracted, and a test pattern is created. The component marked as *pattern matching* in Figure 1.2 computes the distance(s) between the test pattern(s) and the reference pattern(s). The output of the decision rule may be a list of candidates.

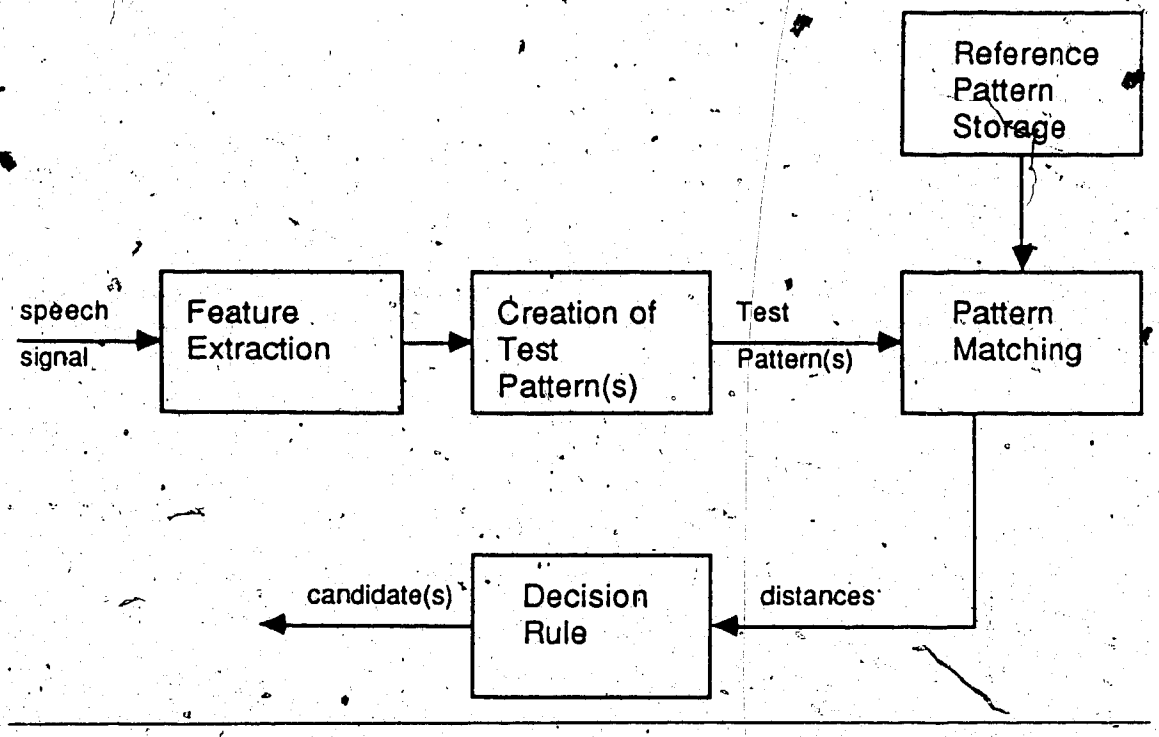

Figure 1.2 Testing mode.

1.2. Feature Extraction

Some features that can be used to characterize the speech signal are described below.

Zero-crossing density: This feature is simply the number of zero crossings of the speech signal in a given time interval. It has been used extensively in early research on speech recognition. The ease with which it can be implemented in hardware is the real advantage to using this feature.

Pitch period: Several algorithms have been developed for pitch extraction in both frequency and time domains. The importance of pitch extraction stems from the fact that fundamental frequency can be used to distinguish between voiced and unvoiced sounds. Equally importantly, fundamental frequency contours also carry prosodic information, such as stress and intonation.

Energy related parameters: Total energy can be used to determine the start of an isolated word. In fact, energy level is perhaps the most popular parameter employed in endpoint detection (see the following section). Pause, stop closure, or weak fricatives can also be determined from these parameters.

Spectral shape parameters: It has been shown that some of the speech events, such as production of the fricatives and the onset of plosive releases, are best characterized by gross spectral shape. Linear Predictive Coding (LPC) coefficients (see Chapter 2) can be used to estimate the spectrum of the speech signal.

Formant frequencies and trajectories: The first three formants for vowels carry important information about the articulatory configuration of the vocal tract. Steady-state values of formant frequencies have been used to classify vowels.

In summary, at the end of the first step, the input speech signal is transformed into a smaller set of information-carrying features which faithfully describe the salient properties of the input. The primary features that are used in our implementation are *reflection coefficients* (see Chapter 2), which indirectly determine the LPC filter coefficients and energy values since the SP1000 can extract these features from input speech.

As will be discussed in greater detail in Chapter 2, the shape of vocal tract changes during the course of an utterance. However, due to the relatively small speed at which the articulators can be moved, the vocal tract shape can be considered fixed for short time periods (10-40 msec), which are commonly called *frame periods*. Frame periods can be overlapping in time. Therefore, the feature extraction block can be thought of as producing a feature vector at each frame period. Feature vectors are then used in creating reference/test patterns. That process is discussed next.

1.3. Creation of Reference/Test Patterns

Three important factors should be considered in creating test/reference patterns:

1. Which frames belong to the utterance?
2. The size of patterns.
3. The number of repetitions of a word (in training mode) and the scheme to be used to cluster the patterns obtained after each repetition.

The first of the three is known as *endpoint detection*, and has been shown to be crucial to the performance of a speech recognition system [21]. The size of patterns, i.e. the number of frames in a a pattern has an impact on the design of the *pattern matching* block. The same is also true for the third factor listed above.

1.3.1. Endpoint Detection

IWR systems assume that an utterance is preceded and followed by silence or other background noise. Non-speech sounds may be generated by the speaker (lip smackings, heavy breathing, pops, etc.) or the transmission system (telephone line, etc.). Door slams and telephone ringing are also common sources of non-speech sounds if the IWR system is to operate in an office environment. The purpose of endpoint detection is to weed out these non-speech sounds and silent portions from the interval which is supposed to contain the utterance.

Most IWR systems use sound amplitude to perform this task. However, difficulties due to leading and trailing weak fricatives (/f, th, h/), weak plosive bursts (/p, t, k/), and final nasals are well known and make the endpoint detectors which depend solely upon sound amplitude susceptible to errors. However, due to the inherent redundancy of the speech signal, an error of few frames in the location of endpoints does not usually cause significant degradation in recognition accuracy.

There are three approaches to the endpoint detection problem:

1. In the *explicit* approach, the endpoint detection stage is isolated from the *pattern matching* and *decision rule* stages. A test/reference pattern consists of the frames between the estimated beginning and ending frames and this pattern is passed to *pattern matching* stage of the system.
2. In the *implicit* approach, there is no separate endpoint detection process. Endpoints are estimated by the *pattern matching* and *decision rule* phases of the system while making recognition decisions [35].
3. The *hybrid* approach combines the explicit and implicit approaches such that a set of endpoint pairs (beginning and ending frames) are estimated based on the features extracted from input signal. Depending on the recognition scores from *pattern matching* and *decision rule* stages, revised endpoint pairs can be chosen.

A comparative study of these approaches can be found in [13].

Features other than energy level are also used to refine the endpoints obtained by energy-level measurements. Rabiner and Sambur [27] use zero-crossing density for this purpose. A novel approach is to utilize spectral information in addition to energy measurements to estimate the endpoints of an utterance [28].

1.3.2. Size of Patterns

The number of frames in a pattern may be fixed or variable. Variable-size patterns are obtained by saving the frames between the beginning and ending points without further processing. Fixed-size patterns can be generated by means of a procedure known as *linear time normalization*.

Intuitively, one can think that variable-size patterns represent the input utterance more faithfully than fixed-size patterns. On the other hand, in the context of *pattern matching*, fixed-size patterns have some implementational advantages. In particular, Myers et al. [23] report improvements in recognition performance when fixed-size

patterns are used with the *dynamic time warping* (DTW) algorithm to perform pattern matching.

1.3.3. Clustering Patterns

It is almost impossible for a human speaker to repeat a particular word exactly the same way (i.e., to produce the same audio signal), due to mechanism of speech production. There will always be slight variations from repetition to repetition. In the context of speaker-independent IWR, because of differences between speakers (vocal tract length and shape, dialect, accent, etc.), these variations will be much greater, necessitating a clustering scheme. The purpose of clustering is to collect "close enough" versions of a particular word together so that the salient features of that cluster can be represented by one reference pattern. This "cluster representing" reference pattern can be obtained as the *minimax* center (i.e., the pattern whose maximum distance to all other reference patterns in the cluster is minimum) of the cluster or as an averaged version of all the elements in the cluster. Rabiner and Wilpon discuss this subject in greater detail in [31].

Clustering techniques are not exclusively employed by the speaker-independent IWR systems although the above discussion may suggest otherwise. In [30] Rabiner and Wilpon show that significant improvements in recognition accuracy can be achieved using the reference patterns obtained from cluster analysis of multiple replications of a word by a designated speaker. Another interesting result from this study is that cluster analysis could be omitted by randomly choosing large number (10 to 12) of reference templates to represent each word in the vocabulary. The major drawback of this approach is an increase in the response time, as it is directly proportional to the number of reference templates per word.

For both speaker-dependent and independent systems, averaging techniques can be used to obtain a reference pattern from multiple repetitions of a word. For fixed

size patterns, an averaged frame can be computed using weighted combinations of corresponding frames. Averaging after optimum time alignment which can be found by employing DTW might produce even better results. This method would work for both fixed and variable-size patterns.

The most straightforward method of obtaining reference patterns would be to create a reference pattern for each repetition of a word in training mode. The availability of reference pattern storage and response time considerations would limit the number of reference templates for each word regardless of the method used to create them.

1.4. Pattern Matching

At this point, feature vectors have been extracted from the input signal and a test pattern has been created. The objective of this step is to determine the similarity between the test and reference patterns. It is quite unlikely that reference and test patterns will have the same number of frames since the speaking rate can vary greatly. In order to compare two patterns, their lengths must be made equal. Two approaches can be taken for this purpose:

1. The simplest method is align the beginning points of the two patterns and then to discard those frames of the longer for which there are no corresponding frames in the shorter. This approach requires little computation but the recognition performance achievable through this method is rather poor.
2. *Linear time normalization* makes the patterns equal in length by stretching the shorter one (or compressing the longer one) to a certain length using some interpolation scheme. Implicit in this method is the assumption that the change of speaking rate is the same across the utterance.
3. The above-mentioned implicit assumption is certainly not true since the speaking rate can vary uniformly during an utterance. Therefore, a method capable of

performing non-linear time alignment is needed. *Dynamic time warping* (DTW) is one such technique which finds an optimal time alignment path for two patterns.

1.4.1. Dynamic Time Warping

We assume the test pattern, TP, and the reference pattern, RP, contain N and M frames, respectively.

$$TP = \{TP(1), TP(2), \ldots, TP(N)\} \tag{1.1}$$

and

$$RP = \{RP(1), RP(2), \ldots, RP(M)\} \tag{1.2}$$

where $RP(m)$ and $TP(n)$ are feature vectors. Then, the DTW problem is to find an optimal time alignment path

$$n = i(k), \quad 1 \le k \le K \tag{1.3a}$$

$$m = j(k), \quad 1 \le k \le K \tag{1.3b}$$

such that the total distance D

$$D = \sum_{k=1}^{K} d(TP(i(k)), RP(j(k)))W(k) \tag{1.4}$$

is minimized where K is the length of the common time axis, $d(TP(n), RP(m))$ is the local distance between frame n of the test pattern and frame m of the reference pattern and $W(k)$ is a weighting function. Figure 1.3 shows an example of optimal time alignment.

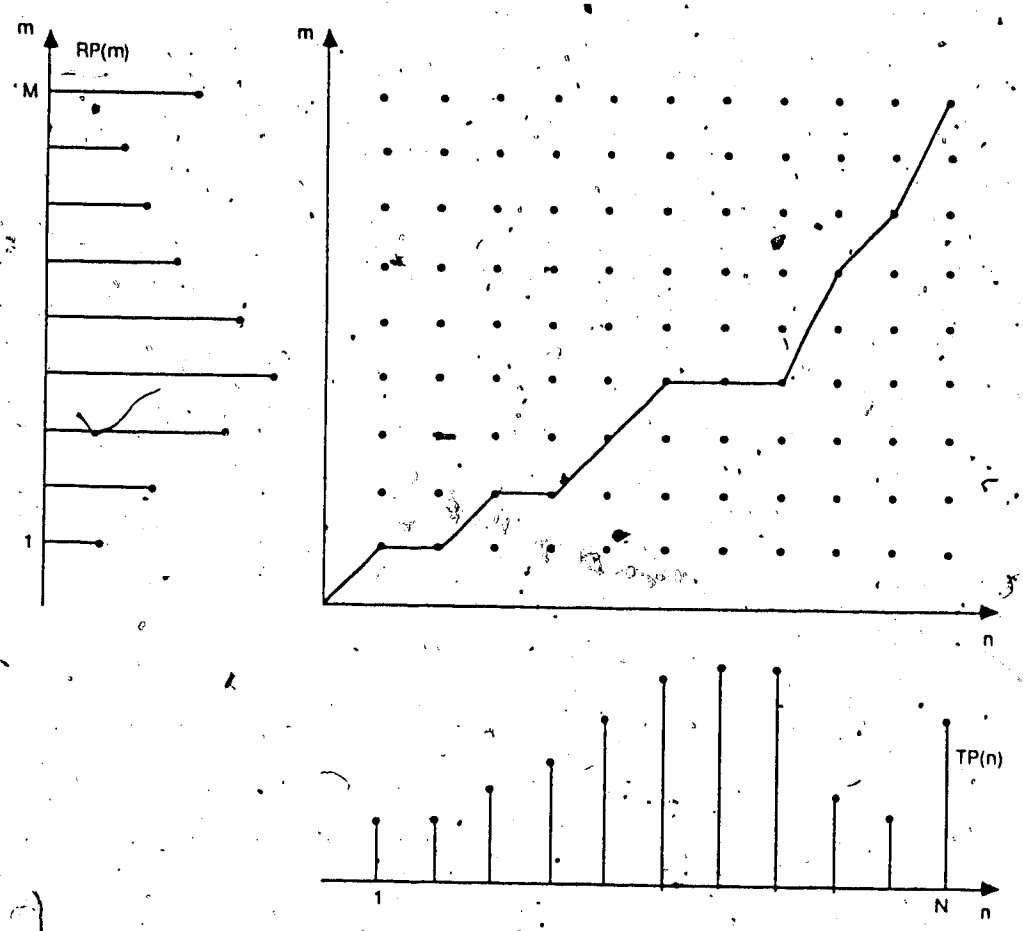

Figure 1.3 Optimal time alignment.

The following factors must be specified to tailor the DTW algorithm, which was discussed above in general terms, to the requirements of a specific application.

1. Endpoint constraints.
2. Local constraints.
3. Global constraints.
4. Axis orientation.
5. Distance measure.

Endpoint constraints

If the endpoints of reference and test patterns are determined, then

$$i(1) = 1, \quad j(1) = 1 \tag{1.5}$$

and

$$i(K) = N, \quad j(K) = M. \tag{1.6}$$

If the IWR system has no separate endpoint detection component (implicit approach), then there are no endpoint constraints.

Local constraints

Local constraints specify the points in a time-warp grid from which a valid path can be taken to a particular point (n,m). These constraints control the amount of compression or expansion of the time scales. The pragmatic advantage of the local constraints is that some points in the time-warp grid can be eliminated from consideration, resulting in a decrease in computation. Figure 1.4 illustrates three types of commonly used local constraints. The canceled path in Figure 1.4 c indicates that an optimal path cannot stay horizontal for two consecutive frames.

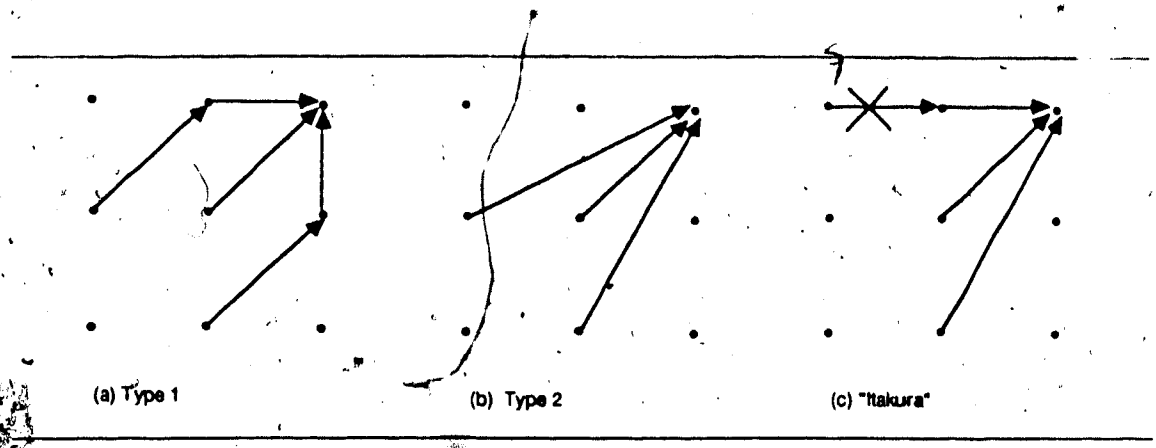

Figure 1.4 Local constraints.

Global constraints

As is pointed out earlier, endpoint constraints together with local constraints specify a region of the time-warp grid in which an optimal time alignment path can lie. Figure 1.5 shows one such region with Type-2 local constraints and $i(1)=j(1)=1$, $i(K)=N$, $j(K)=M$, $N=M=15$ endpoint constraints.

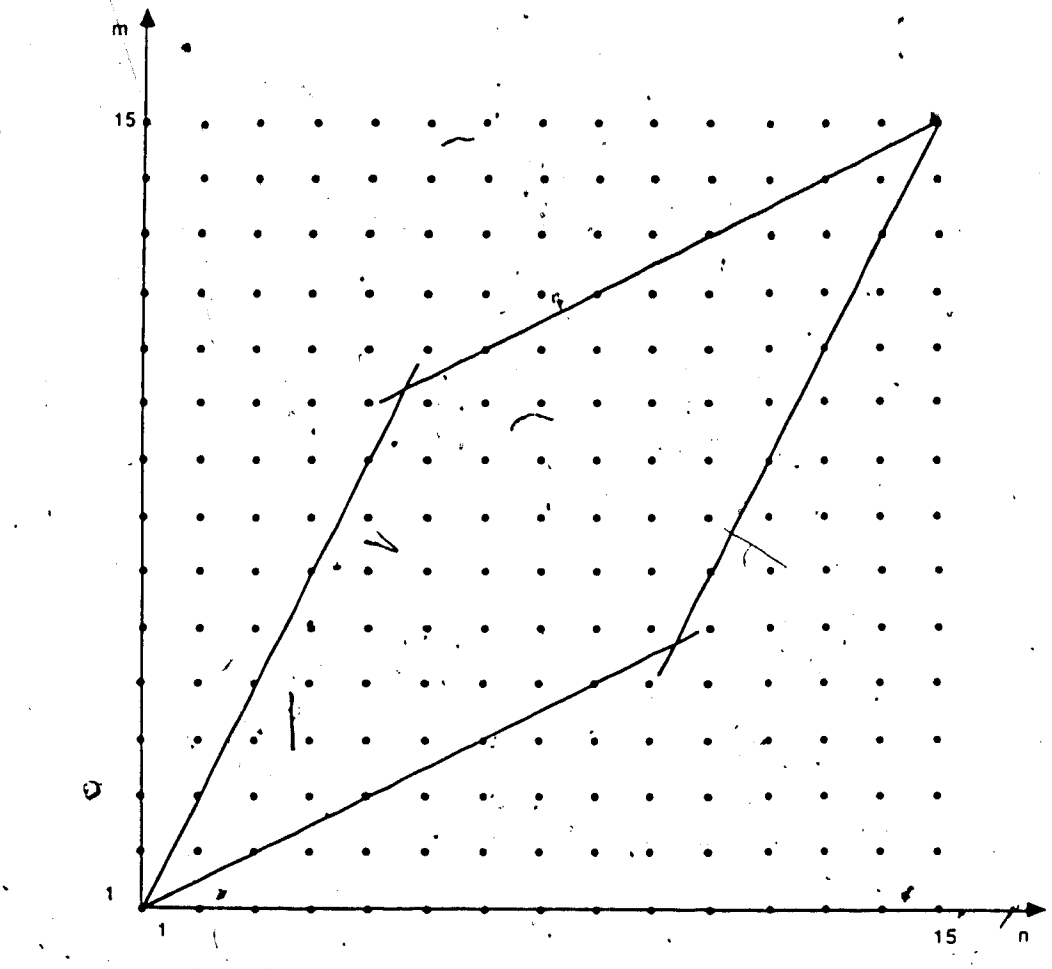

Figure 1.5 Global constraints.

Axis orientation

In the previous discussion, we considered the test pattern along the x-axis and the reference pattern along the y-axis. Another equally valid way of assigning the patterns to axes would be "test pattern along the y-axis" and "reference pattern along the x-axis", corresponding to eqs. (1.7).

$$m = i(k), \quad 1 \le k \le K \tag{1.7a}$$

$$n = j(k), \quad 1 \le k \le K, \tag{1.7b}$$

Eqs. (1.3) and (1.7) would define the same time alignment of test and reference pat-

terns if local constraints and the distance measure were symmetric. Otherwise, the choice between eqs. (1.3) and eqs. (1.7) has an effect on recognition accuracy.

Distance measure

There are two factors involved in the computation of global distance as given by eq. (1.4): the local distance measure $d(TP(n), RP(m))$ and the weighting function $W(k)$.

The local distance measure is inherently dependent on the features used in the recognition decision. Popular local distance measures include Euclidean distance on the LPC filter, reflection, or cepstral coefficients, [10], log area-ratios [7], covariance weighting, spectral distance, and LPC log likelihood measure [32].

The weighting function $W(k)$ specifies the weight of path taken to come to point $(i(k), j(k))$ from the previous point $(i(k-1), j(k-1))$. In other words, the contribution of local distance between frame $i(k)$ of the test pattern and frame $j(k)$ of the reference pattern to global distance depends upon the previous point.

As can be seen from the above discussion, finding an optimal time alignment path between test and reference patterns involves a sequence of decisions, namely deciding on which path to take to minimize the accumulated distance. The *principle of optimality* [11] applies for the DTW problem, making it possible to use the classical dynamic programming recursive relationship to define the accumulated distance function $D_A(n, m)$ which represents the accumulated distance along the best path from point (1,1) to point (n,m). Without loss of generality, for the special case of Type 2 local constraints with all three paths having the same weight of 1, $D_A(n,m)$ can be written as

$$D_A(n,m) = d(TP(n),RP(m)) + \min \begin{Bmatrix} D_A(n-1,m-1) \\ D_A(n-1,m-2) \\ D_A(n-2,m-1) \end{Bmatrix} \qquad (1.8)$$

1.5. Decision Rule

The inputs to the *decision rule* stage are the distance scores computed by the *pattern matching* stage. The output of this stage might be a word which most closely matches the test pattern, a list of words ordered by their distances to the test pattern, or a message stating that no word in the vocabulary matches the test pattern closely enough to justify declaring it to be the recognized word (i.e., test word is "rejected"). Employing rejection capability in an IWR system is closely related to the nature of the specific application for which the IWR system is intended.

The *nearest neighbor* (NN) rule, the *k-nearest neighbor* (KNN) rule, and the *nearest k-mean* (NKM) rule are three popular methods used in IWR systems. The NN rule simply chooses the word corresponding to the reference pattern for which the computed DTW distance is smallest. The KNN rule classifies the test pattern as the most frequently represented word among the k nearest reference patterns [8]. Clearly, these methods can also be used to choose a list of recognition candidates.

When each word is represented by two or more reference patterns, the NKM rule can be employed to make the recognition decision. The NKM rule can be described as follows:

```
for each word in the vocabulary {
    for each reference pattern for this word {
        compute the DTW distance between the reference pattern
        and the test pattern.
    }
    compute the average of the k smallest distances.
}
choose the word with the smallest average distance.
```

As with the NN rule, the NKM rule can also be used to obtain a list of recognition candidates. Rabiner and Wilpon [31] show that when from 6 to 12 reference patterns are kept for each word in vocabulary, a real statistical advantage is obtained using the NKM rule with $k=2$ *or* 3 over the NN rule.

Rejection capability can be incorporated using an absolute threshold value. If the lowest DTW score is not below a certain threshold then test pattern is rejected. Determination of a threshold value depends heavily on the specific application, users of the system, etc. This scheme can be further refined by employing a relative threshold value. If the difference between the DTW distances of the best and second best candidate is not less than the relative threshold, the IWR system can reject the test pattern on the grounds that a reliable decision cannot be made. Further refinements to this scheme are, of course, possible.

1.6. Applications

There are several advantages associated with using speech in man-machine communications [15] which make the number of applications for speech recognition limited only by the imagination of people interested in this field. Speech is the most *natural* mode of communication for humans. Also, as shown in previous studies [14], speech is the highest capacity output communication channel for humans. Since IWR systems require that words be separated by some amount of silence, some advantage with respect to rate of information transfer is lost, but speech is still superior to other forms of communication such as typing, manipulating knobs and buttons, etc. Using speech input does not prevent a user from simultaneously performing other tasks such as walking, manipulating objects with the hands, inspecting objects visually, etc. In this regard, the value of the speech input can be better appreciated by comparing it to using computer terminals which tie up both tactile and visual functions of the user.

Successful commercial applications for speech recognizers include [15] package sorting, quality control and inspection, programming of numerically controlled machines, voice-actuated wheelchairs, banking and credit card transactions, annotating integrated circuit photo masks, voice control of household appliances (especially for the blind), etc.

Speech recognizers have also enjoyed widespread use in military applications such as cartography in defense mapping, computer-assisted training of air traffic controllers, airplane and helicopter cockpit communications, etc.

The areas of *speaker identification* and *verification* and *word spotting* which are closely allied with the field of IWR offer other interesting applications including security and access control, monitoring conversations, etc.

Chapter 2

Background

Section 1 overviews digital signal processing (DSP) in the context of discrete-time processing of continuous-time signals since almost all speech processing systems employ DSP algorithms. In order to apply DSP algorithms to speech processing problems, it is extremely important to have an understanding of speech production processes as well as the principles of DSP. Introduced in Section 2 are digital models for the speech signal. The first two sections are far from complete and they are intended only to make this work more self-contained. Section 3 briefly discusses time-domain methods for speech analysis. The last section surveys linear predictive coding (LPC) of speech with an emphasis on lattice methods.

2.1. Overview of Digital Signal Processing (DSP)

This review is intended to serve as an easy reference for later sections and to establish the notation that will be used throughout the paper. Comprehensive treatments of this subject can be found in several excellent textbooks [16, 24, 26].

2.1.1. Discrete Time Signals and Systems

The acoustic wave produced in human speech is a continuously varying signal. These continuous-time (CT) signals can be represented using the notation of the form $x_a(t)$ where subscript a denotes the analog nature of the signal. It is also possible to represent the speech signal as a sequence of numbers. The notation $x(n)$ will be used to denote such sequences. A sequence can be obtained as a sequence of samples of an analog signal taken periodically with sampling period T. In other words,

$$x(n) = x_a(nT) \quad . \tag{2.1}$$

The special class of linear time-invariant (LTI) systems is used to a great extent in speech processing. Such systems are completely characterized by their response $h(n)$

to a unit sample input $\delta(n)$ [26]. The output of such a system in response to an arbitrary input $x(n)$ is computed using the **convolution sum** expression

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k) = x(n) * h(n) \tag{2.2}$$

where the symbol * stands for discrete-time (DT) convolution.

2.1.2. The z-Transform

The z-transform of a sequence $x(n)$ is defined as

$$X(z) \triangleq \sum_{n=-\infty}^{\infty} x(n)z^{-n} \tag{2.3}$$

where z is a complex variable. For convenience, the z-transform of $x(n)$ will sometimes be denoted as $Z[x]$. In general, eq.(2.3) will converge only for certain values of z. The set of values of z for which the eq. (2.3) converges defines a region in the z-plane known as the *region of convergence* (ROC).

The z-transform has some important features which we will encounter quite often in the analysis of digital speech models. Now, we will briefly state a few of these properties.

Linearity

If

$$Z[x_1(n)] = X_1(z), \quad ROC = R1$$

and

$$Z[x_2(n)] = X_2(z), \quad ROC = R2$$

then

$$Z[ax_1(n) + bx_2(n)] = aX_1(z) + bX_2(z), \quad ROC \text{ contains } R1 \cap R2 . \tag{2.4}$$

Time Shifting

If

$$Z[x(n)] = X(z), \quad ROC = R_x$$

then

$$Z[x(n-n_0)] = z^{-n_0}X(z), \quad ROC = R_x \tag{2.5}$$

Convolution Property

If

$$Z[x_1(n)] = X_1(z), \quad ROC = R_1$$
$$Z[x_2(n)] = X_2(z), \quad ROC = R_2$$

then

$$Z[x_1(n) * x_2(n)] = X_1(z)X_2(z), \quad ROC \text{ contains } R1 \bigcap R2 \tag{2.6}$$

2.1.3. Systems Characterized by Linear Constant-Coefficient Difference (LCCD) Equations

Consider an LTI system for which the input ($x(n)$) and output ($y(n)$) satisfy a LCCD equation of the form

$$\sum_{k=0}^{N} a_k y(n-k) = \sum_{k=0}^{M} b_k x(n-k) \tag{2.7}$$

This equation is a time-domain relationship. We can find the frequency-domain relationship characterizing this system by applying z-transform to both sides of the equation and using linearity and time-shifting properties. Then, we have

$$\sum_{k=0}^{N} a_k z^{-k} Y(z) = \sum_{k=0}^{M} b_k z^{-k} X(z)$$

We obtain

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^{M} b_k z^{-k}}{\sum_{k=0}^{N} a_k z^{-k}} \tag{2.8}$$

Also, note that the LCCD equation by itself does not provide information about the ROC to associate with the algebraic expression $H(z)$. An additional constraint such as causality or stability of the system is necessary to specify the ROC. For a stable system, all the poles must be *inside* the unit circle.

2.1.4. Sampling of Continuous-Time Signals

DT processing of CT signals such as speech is only possible with proper sampling of CT signals. If $x_a(t)$ is sampled uniformly with a sampling period of T, a DT signal $x(n)$ is obtained:

$$x(n) = x_a(t)|_{t=nT}$$

Sampling Theorem [26]: Let $x_a(t)$ be a bandlimited signal with $X(j\,\Omega) = 0$ for $|\Omega| > \Omega_M$, where $X(j\,\Omega)$ is the CT Fourier transform of $x_a(t)$. Then $x_a(t)$ is uniquely determined by its samples

$$x_a(nT),\ n = 0, \pm 1, \pm 2, \cdots$$

provided

$$\frac{1}{T} > 2f_M$$

where f_M is the highest frequency (in Hz) present in $x_a(t)$; i.e., $f_M = \frac{\Omega_M}{2\pi}$.

Given these samples, we can reconstruct $x_a(t)$ by generating a periodic impulse train in which successive impulses have amplitudes that are successive sample values. This impulse train is then processed through an ideal lowpass filter with gain T and cutoff frequency greater than f_M and less than $\frac{1}{T} - f_M$. The resulting output signal will exactly equal $x_a(t)$. The bound on the sampling rate $2f_M$ is referred to as the *Nyquist rate*.

If the sampling rate does not satisfy the above condition, *aliasing* (the situation where a high frequency component when sampled takes on the identity of a lower frequency component) can occur.

2.1.5. DT Processing of CT Signals

In many applications including speech processing, there is a significant advantage offered in processing a CT signal by first converting it to a DT signal and after processing, converting back to a CT signal. One of the reasons for this is the increasing availability of inexpensive, lightweight, programmable and easily reproducible digital and DT systems [26]. The DT signal processing can be implemented with a general- or special-purpose computer, with microprocessors or with any of the variety of devices (like the SP1000 chip) that are specially oriented toward DT signal processing.

Figure 2.1 shows an LTI system placed in an A/D-DT filter-D/A-LPF structure. At the output of the A/D converter we have

$$x(n) = x_a(t)|_{t=nT} = x_a(nT) \ .$$

The DT filter processes $x(n)$ and produces the output sequence $y(n)$ which is converted to a CT signal $y_a(t)$ by the D/A converter followed by the low-pass filter (LPF) with a cutoff frequency of half the sampling frequency.

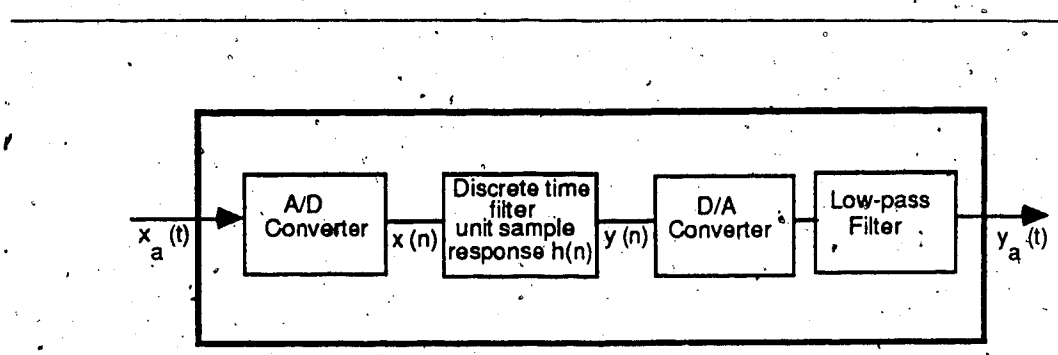

Figure 2.1 A/D-DT Filter-D/A-LPF structure.

The frequency response of the *equivalent analog filter* marked by thick solid lines in Figure 2.1 will be derived in terms of the frequency response $H(e^{j\omega})$ of the DT filter [16]. Consider the steady state response of the entire system to a sinusoid of an arbitrary frequency Ω_0 less than π/T ;

$$x_a(t) = A\cos\Omega_0 t, \quad \text{where } \Omega_0 < \pi/T .$$

At the output of the A/D converter we have

$$x(n) = x_a(nT) = A\cos[(\Omega_0 T)n] = A\cos\omega_0 n$$

where $\Omega_0 T = \omega_0$. Eq. (2.9) states the fundamental relationship between *digital radian frequency* and *analog radian frequency*.

$$\omega = \Omega T . \tag{2.9}$$

where ω, Ω, T are the digital and analog radian frequencies, and the sampling period, respectively. The steady state output $y(n)$ of the DT filter can be written as

$$y_{ss}(n) = A|H(e^{j\omega})|\Big|_{\omega=\omega_0=\Omega_0 T}\cos[(\Omega_0 T)n + arg H(e^{j\omega})\Big|_{\omega=\omega_0=\Omega_0 T}] \tag{2.10}$$

where *arg* denotes the *angle* or *phase* of its argument. Assuming an ideal D/A converter followed by an ideal LPF, the steady state output $y_a(t)$ can be written as follows:

$$y_a(t) = A|H(e^{j\Omega_0 T})|\cos[\Omega_0 t + arg H(e^{j\Omega_0 T})] \tag{2.11}$$

Therefore the net effect of equivalent analog filter is to scale the amplitude of the input signal by $|H(e^{j\Omega_0 T})|$ and to change the phase of input signal by $arg H(e^{j\Omega_0 T})$. Figure 2.2 (adapted from [16]) shows the magnitude of the frequency response of a DT filter along with the corresponding magnitude of the *equivalent analog frequency response* of the A/D-DT Filter-D/A-LPF structure.

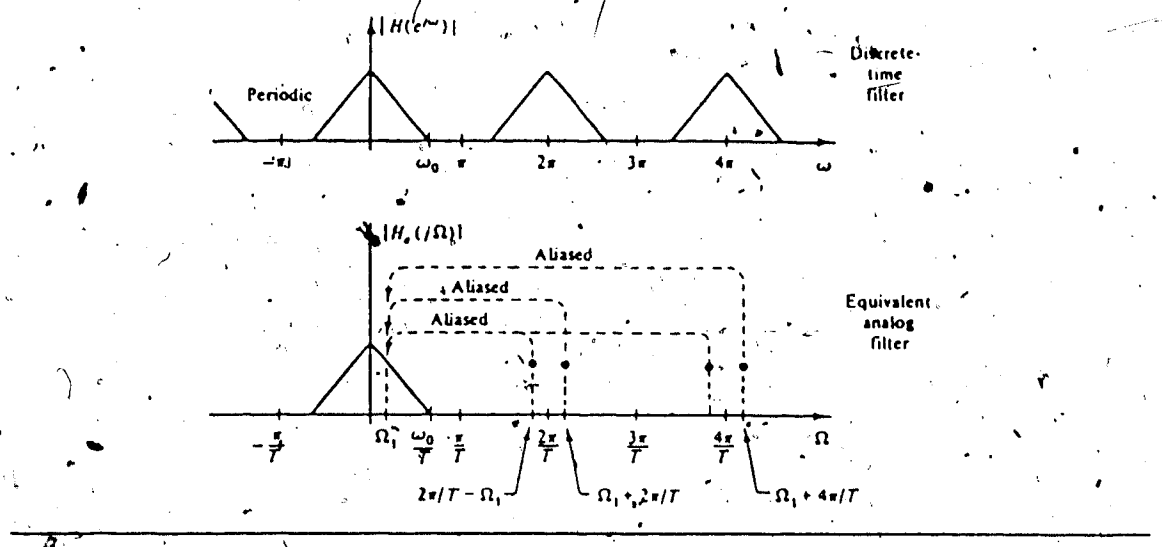

Figure 2.2 Equivalent analog filter frequency response for a D/A-filter-A/D-LPF structure.

In general, any input analog frequency greater than π/T will be aliased. One way to overcome this problem is to prefilter the input signal. Figures 2.3 and 2.4 (adapted from [16]) clarify this technique.

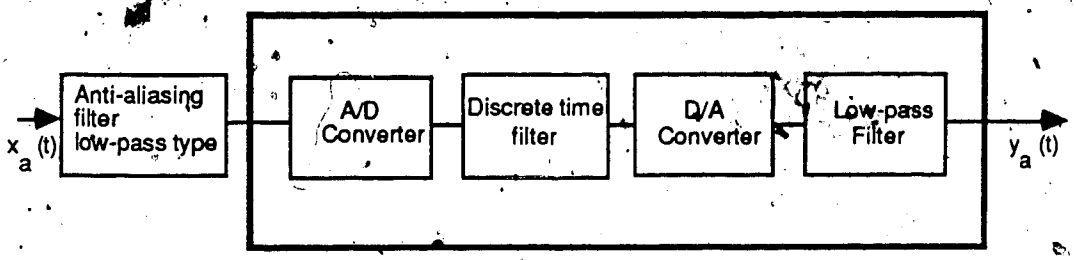

Figure 2.3 Use of prefilter to reduce aliasing in A/D-DT filter-D/A-LPF structure.

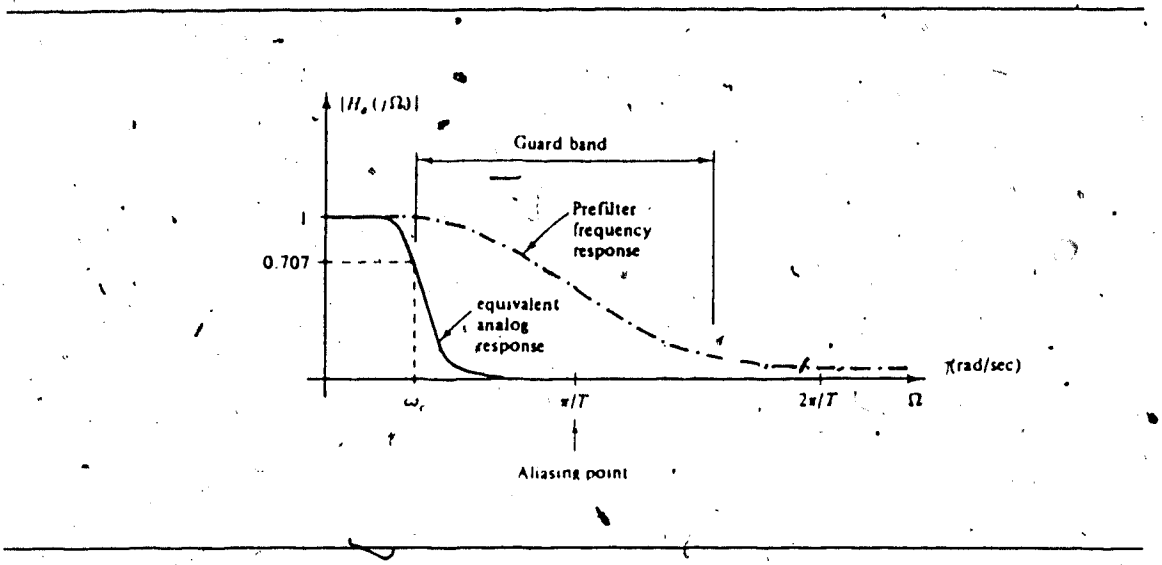

Figure 2.4 Guard band and prefilter frequency response.

Figure 3.5 shows how these ideas (and more) are realized in the LISNER 1000 recognition board. The anti-aliasing filter of the LISNER 1000 board cuts off at 3200 Hz. implying that the sampling rate should be greater than 6400 Hz to avoid aliasing.

In the above discussion, we have tacitly assumed that A/D (D/A) conversion is done with infinite precision. However, in reality, A/D converters have only finite number of bits to represent (encode) the quantized value of the input signal. In other words, the output of the A/D converter encodes the quantized value of the original input signal with finite precision. The difference between the original signal value and its quantized counterpart corresponding to a particular bit configuration produced by the A/D converter is known as *quantizing error* or *quantizing noise*.

Under certain assumptions about statistical properties of the input and quantizing error signal, it can be shown that [29] signal-to-quantizing-noise ratio (in decibels) is

$$SNR_Q(dB) = 6B - 7.2$$

for a uniform quantizing scheme where B is the word length (in bits) of the A/D converter. Roughly speaking, each additional bit used in the A/D converter improves the

SNR by 6 dB. The quantizing noise can be reduced by increasing the number of quantization levels or, equivalently, by increasing the word length of the A/D converter. This consideration gives rise to the "A/D converter cost versus SNR" tradeoff.

2.2. Models for the Speech Signal

2.2.1. Source-System Model for Speech Production

Speech output can be modeled as the response of a slowly time varying system to either a periodic or a noiselike excitation. More specifically, the speech-production mechanism consists essentially of an acoustic tube, the vocal tract, excited by an appropriate source to generate the desired sound. There are three major mechanisms of excitation [29]:

"1. Air flow from the lungs is modulated by the vocal cord vibration, resulting in a quasi-periodic pulse-like excitation.

2. Air flow from the lungs becomes turbulent as the air passes through a constriction in the vocal tract, resulting in a noiselike excitation.

3. Air flow builds up pressure behind a point of total closure in the vocal tract. The sudden release of this pressure causes a transient excitation."

These three different modes of excitation result in three categories of sounds. *Voiced* sounds are produced by type 1 excitation. Vowels, among other sounds, are in this category. Type 2 excitation produces *fricative* sounds. /f/, /s/, /sh/ are three of the sounds in this category. Type 3 excitation gives rise to *plosive* sounds. Examples of the sounds in this category are /b/, /d/, /g/. A cross-sectional view of of the vocal tract and source-system model are shown in Figures 2.5 [20] and 2.6 [29], respectively.

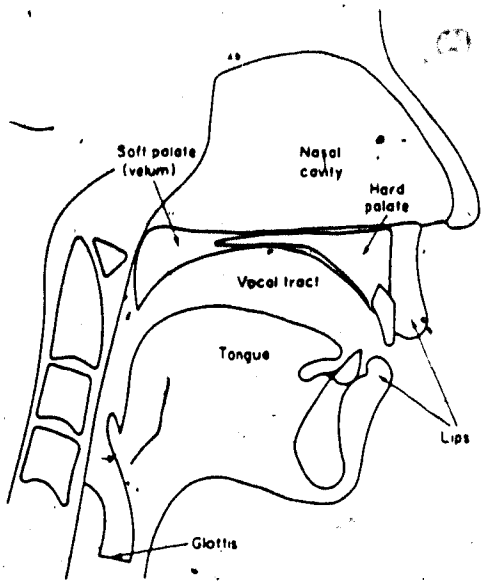

Figure 2.5 Cross-sectional view of the vocal mechanism.

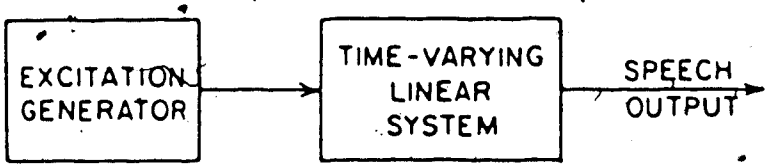

Figure 2.6 Source-system model of speech production.

If the vocal tract shape is fixed, as in the case of vowels, the output of the system is the convolution of the excitation and the vocal-tract impulse response. Some other sounds (e.g., diphthongs, semivowels), however, are produced by changing the shape of vocal tract. In that case, the vocal-tract shape shape can still be assumed fixed for short time intervals owing to relatively small speed at which the articulators can be moved. Under this assumption, the output can be approximated as a convolution of the excitation and vocal-tract impulse response over short time periods.

The spectrum of periodic excitation contains impulses in frequency domain corresponding to harmonics in input. The frequency response of the vocal tract exhibits peaks (Figure 2.7 [25]) corresponding to resonance frequencies of the vocal tract which are called *formant frequencies*. In the frequency domain, the output spectrum is the product of excitation spectrum and vocal tract spectrum. The envelope of the output spectrum (dotted lines in Figure 2.7) reflects the shape of the vocal tract spectrum.

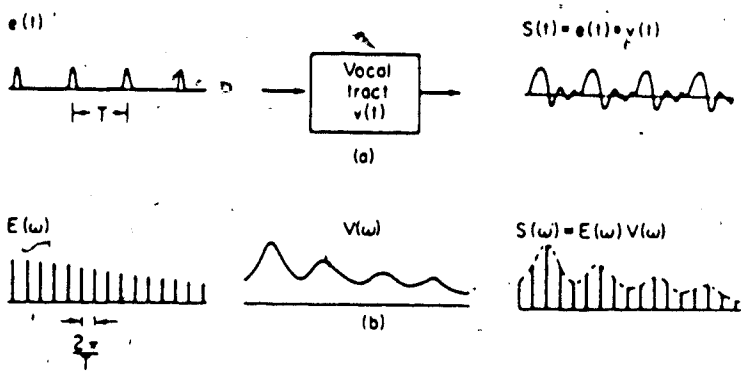

Figure 2.7 Speech production model (a) time-domain (b)frequency-domain.

The envelope of the output spectrum will vary with time as the vocal tract shape varies. Similarly a variation in pitch frequency (inverse of excitation period) will bring about a variation in the spacing of pitch harmonics.

2.2.2. Lossless Tube Models

As illustrated in Figure 2.8 [29], the vocal tract can be approximated by a concatenation of lossless acoustic tubes. A detailed analysis of lossless tube model is quite difficult unless some simplifying assumptions are made. These assumptions are [25] a) sound propagation through each section can be considered as a plane wave along the axis of the tube and b) losses due to viscous friction, thermal conduction and the effect

of the nasal tract and coupling between the vocal tract and glottis can be ignored. Under these assumptions, a relationship between the lossless tube model and a digital filter structure can be found (see Section 2.4.6.1). The lossless tube model can be characterized in terms of pressure or in terms of the volume velocity of air flow as a function of time and distance along the tube [25]. The volume velocity can be thought of as a superposition of two components: a) forward-traveling wave and b) reverse-traveling wave where the former moves from the glottis to the lips.

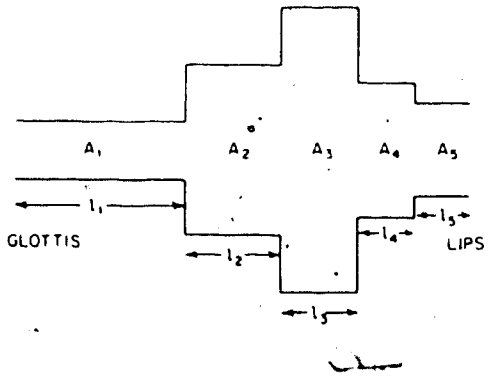

Figure 2.8 Concatenation of 5 lossless acoustic tubes.

If we consider the k^{th} tube with cross-sectional area A_k, the pressure and volume velocity in that tube have the form [29]

$$p_k(x,t) = \frac{\rho c}{A_k}\left[u_k^+(t-x/c) + u_k^-(t+x/c)\right] \qquad (2.12a)$$

$$u_k(x,t) = u_k^+(t-x/c) - u_k^-(t+x/c) \qquad (2.12b)$$

where x is distance measured from the left hand of the k^{th} tube ($0 \le x \le l_k$), $u_k^+()$ and $u_k^-()$ are forward-traveling and reverse-traveling waves in the k^{th} tube, ρ is air density, and c is the velocity of sound in air.

Since the pressure and volume velocity must be continuous across a boundary

between adjacent tubes, part of the forward-traveling wave that reaches the junction (see Figure 2.9 [29]) is propagated on to the right, while another part is reflected back to the left. A similar statement applies to the reverse-traveling wave. Eq. (2.13)

$$r_k = \frac{A_{k+1} - A_k}{A_{k+1} + A_k} \tag{2.13}$$

gives the amount of $u_{k+1}^-(t)$ that is reflected at the junction [29]. Thus, the quantity r_k is called the **reflection coefficient** for the k^{th} junction. It is obvious that since the areas are all positive

$$-1 \le r_k \le 1 .$$

With this definition, it follows that [29]

$$u^+_{k+1}(t) = (1+r_k)u_k^+(t-\tau_k) + r_k u^-_{k+1}(t) \tag{2.14a}$$

$$u_k^-(t+\tau_k) = -r_k u_k^+(t-\tau_k) + (1-r_k)u^-_{k+1}(t) \tag{2.14b}$$

where $\tau_k = l_k/c$ is the time for a wave to travel the k^{th} tube.

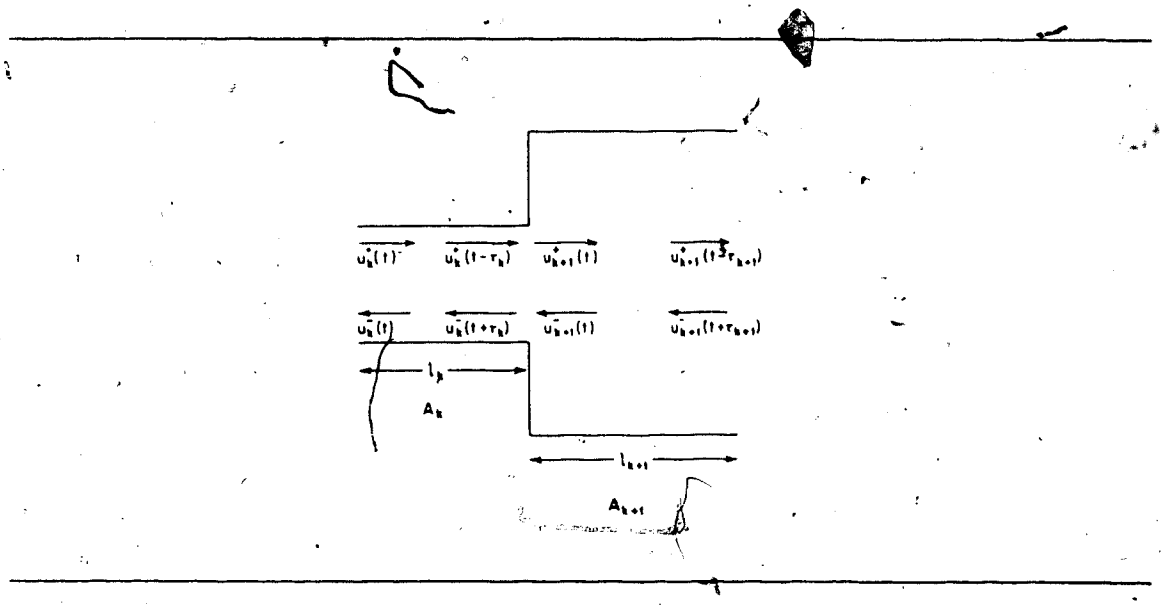

Figure 2.9 The junction between two lossless tubes.

2.2.3. The Complete Model

The transfer function of the tube model for the vocal tract can be stated as [29]

$$V(z) = \frac{G}{1 + \sum_{k=1}^{N} \alpha_k z^{-k}}$$

where G and α_k depend upon the area function and N is the number of tubes. By slowly changing the input parameters (i.e., excitation) and α_k, we can approximate the real human vocal tract and its output — speech.

Figure 2.10 (adapted from [29]) depicts the widely accepted and used DT model for speech production.

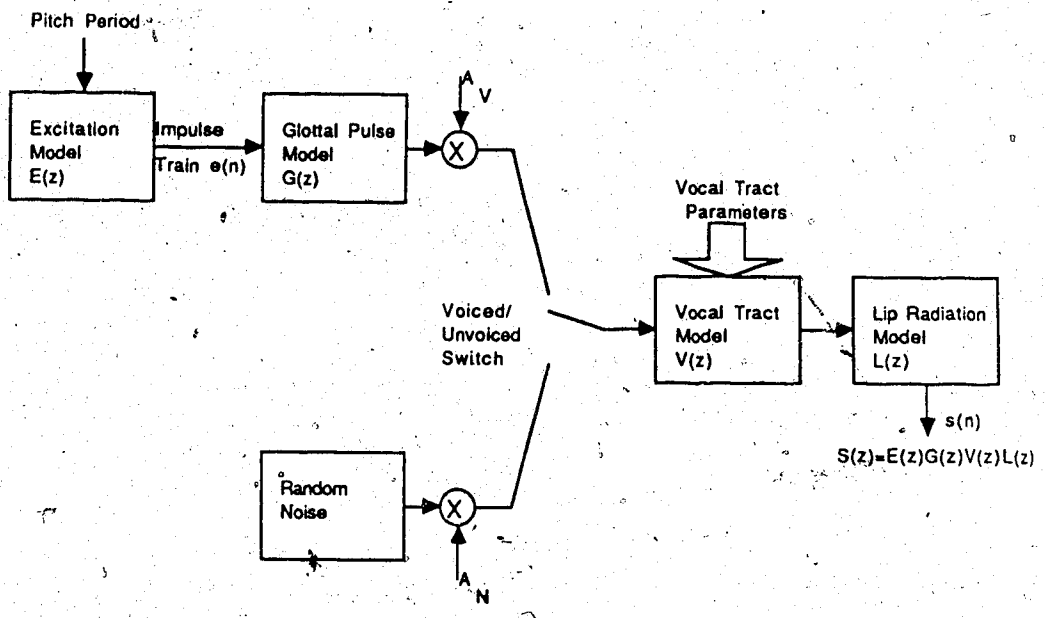

Figure 2.10 General DT model for speech production.

2.3. Speech Processing in the Time Domain

2.3.1. Short-Time Analysis of Speech

As indicated in the previous section, the fact that we can model the speech production process with a time-varying linear system forms the foundation for short-time processing methods. In these methods, short segments of the speech signal are extracted and processed under the assumption that vocal-tract shape is fixed during the short segment of the speech. These short segments, which are sometimes called *frames*, can overlap one another. The result of processing on each frame produces a new time-dependent sequence which can represent a useful feature of the speech signal, such as short-time energy and zero-crossing rate.

Most of the short-time processing techniques can be characterized mathematically in the form

$$Q_n = \sum_{m=-\infty}^{\infty} T[x(m)]\, w(n-m) \tag{2.15}$$

Eq. (2.15) describes a three-step procedure [29]:

1. Transformation $T[\,]$ is applied to the speech signal, resulting in a new sequence.
2. The result of first step is then multiplied by a window sequence positioned at a time corresponding to sample index n.
3. The product of step 2 is summed over all nonzero values.

Duration and properties of windows have significant effect on the results of processing. Figure 2.11 [29] illustrates these ideas for a specific window of finite length. As Figure 2.11 shows, the window slides along the sequence of $T[x(m)]$, selecting the interval to be used in the computation of Q_n.

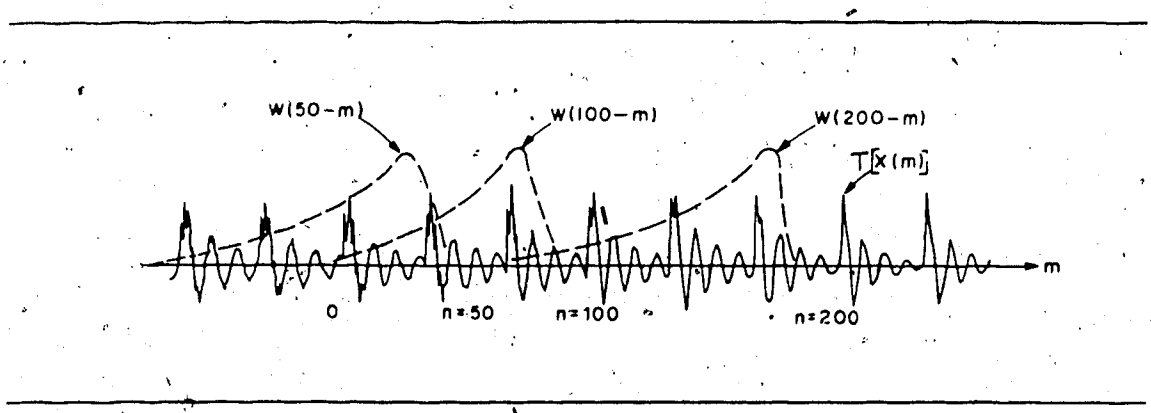

Figure 2.11 $T[x(m)]$ and $w(n-m)$ for several values of n.

2.3.2. The Short-Time Autocorrelation Function

One of the main uses of this function is in pitch-period estimation. It is also utilized extensively in LPC analysis of speech. Therefore it deserves some discussion. The autocorrelation function of a DT deterministic signal is defined as [29]

$$\phi(k) = \sum_{m=-\infty}^{\infty} x(m)x(m+k) \quad . \tag{2.16}$$

If the signal is periodic with period P samples, then the autocorrelation function of it is also periodic with the same period.

$$\phi(k) = \phi(k+P) \quad . \tag{2.17}$$

The autocorrelation function has some other important properties, namely [29]:

1. It is an even function; i.e., $\phi(k) = \phi(-k)$.
2. It attains its maximum value at $k=0$; i.e., $|\phi(k)| \leq \phi(0)$ for *all* k.
3. The quantity $\phi(0)$ is equal to the energy for deterministic signals or the average power for random or periodic signals.

The great utility of the autocorrelation function is in the way it displays the periodicity of the signal. Since it attains a maximum at samples $0, \pm P, \pm 2P, \ldots$, the period of the signal can be estimated by finding the location of the first maximum in the autocorrelation function, regardless of the time origin of the signal. Therefore it is

not surprising to see that the autocorrelation function has been used extensively to estimate pitch period for a speech signal.

The *short-time* autocorrelation function is defined as follows [29]:

$$R_n(k) = \sum_{m=-\infty}^{\infty} x(m)w(n-m)x(m+k)w(n-k-m) \qquad (2.18)$$

If the length of $w(n)$ is N, then it is clear that $w(n-m)$ and $w(n-k-m)$ are *both* nonzero for $m = n-N+1,\ldots,n-k$. In other words, only $N-k$ values of $x(m)x(m+k)$ are involved in the computation of $R_n(k)$. It is easily shown that

$$R_n(k) = R_n(-k)$$

Figure 2.12 [29] shows three examples of autocorrelation function for voiced and unvoiced speech, using a Hamming window with $N=401$. The peaks occurring at multiples of 72 indicate a period of 7.2 msec for the Figure 2.12 a. Similarly, peaks spaced 58 samples apart in Figure 2.12 b implies an approximate pitch period of 5.8 msec. Figure 2.12 c indicates a lack of periodicity.

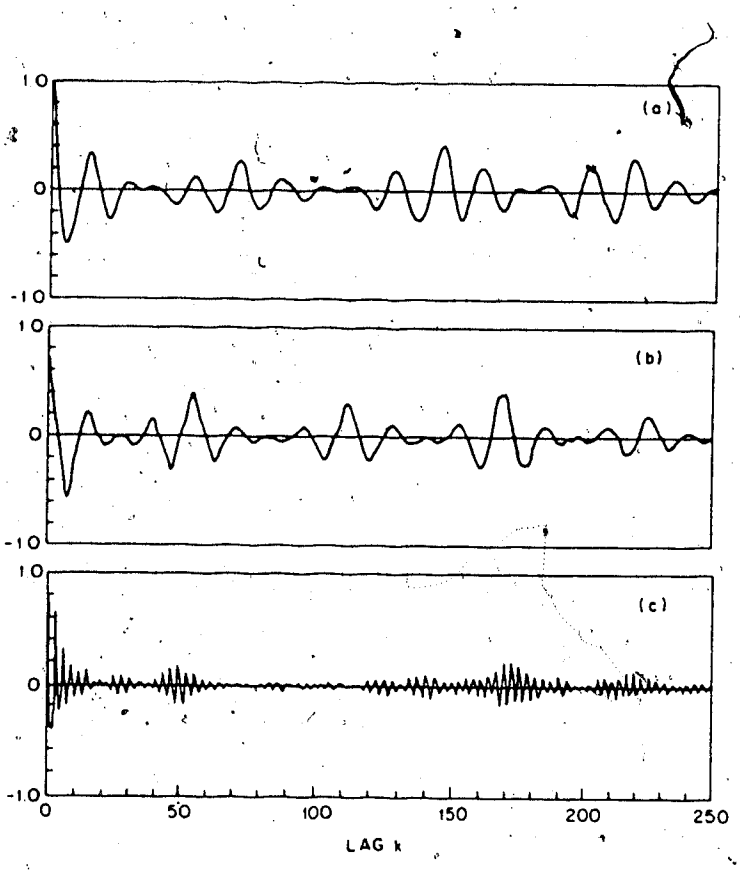

Figure 2.12 Autocorrelation functions for (a) and (b) voiced speech; (c) unvoiced speech.

2.4. Linear Predictive Coding of Speech

2.4.1. Introduction

The underlying idea behind linear prediction of speech is that a speech sample can be *predicted* from *linear* combinations of past speech samples and past and present samples of excitation. By minimizing the sum of squared differences (over a finite interval) between the actual speech samples and linearly predicted ones, a unique set of predictor coefficients can be determined.

The application of LPC is not limited to speech problems. In fact, it has been applied to problems ranging from neurophysics to geophysics. Linear prediction, as a

concept, has its roots in time series analysis which deals with the analysis of the outputs of dynamic systems.

In general, consider $s(n)$ which is the output of some system with input $e(n)$ such that the following relation holds [17]:

$$s(n) + \sum_{k=1}^{p} a_k s(n-k) = G \sum_{l=0}^{q} b_l e(n-l), \quad b_0 = 1 \tag{2.19}$$

where a_k, $1 \le k \le p$, b_l, $1 \le l \le q$, and the gain G are the parameters of the *hypothesized* system. Taking the z-transform of both sides of eq. (2.19), we have

$$H(z) = \frac{S(z)}{E(z)} = G \frac{1 + \sum_{l=1}^{q} b_l z^{-l}}{1 + \sum_{k=1}^{p} a_k z^{-k}} \tag{2.20}$$

$H(z)$ in eq. (2.20) is the general *pole-zero* model. There are two special cases of the model that are of interest:

1. all-zero model: $a_k = 0$, $1 \le k \le p$.
2. all-pole model: $b_l = 0$, $1 \le l \le q$.

In this section we will be concerned with the all-pole model. Three methods for computing predictor coefficients will be summarized. These methods are

1. the autocorrelation method
2. the covariance method
3. the lattice method.

Towards the end of this section we will discuss how other parameters of speech can be derived from LPC coefficients. These parameters will include pitch period and formant locations.

2.4.2. Principles of Linear Predictive Analysis

Consider the model illustrated in Figure 2.10. Lip radiation can be modelled as [20]

$$L(z) = 1 - z^{-1}$$

Glottal shaping model $G(z)$ † can be approximated as

$$G(z) \approx \frac{1}{L^2(z)}$$

Then,

$$S(z) = E(z)G(z)V(z)L(z) \approx E(z)V(z)/L(z)$$

where $V(z)$ has N poles (see Section 2.2.3). We now *assume* that we can represent the model depicted in Figure 2.10 with the following system function:

$$H(z) = \frac{S(z)}{E(z)} = \frac{G}{1 + \sum_{k=1}^{p} a_k z^{-k}} \tag{2.21}$$

where $p \geq N+1$. Then we have the following in the time domain

$$s(n) = -\sum_{k=1}^{p} a_k s(n-k) + Ge(n) \tag{2.22}$$

It has been shown that eq. (2.21) provides a good approximation to almost all sounds of speech. Now, our objective is to estimate the filter coefficients a_k and G.

2.4.2.1. The Least Squares Method

Here we assume that the input $e(n)$ is totally unknown. This implies that the signal $s(n)$ can be predicted only from a linearly weighted summation of past samples. If we define a **linear predictor** as a system whose output $\tilde{s}(n)$ and input $s(n)$ are related to each other via eq. (2.23) [29]

$$\tilde{s}(n) = -\sum_{k=1}^{p} \alpha_k s(n-k) \tag{2.23}$$

† Although the notation $G(z)$ (transfer function for glottal shaping model) and G (gain for the vocal tract model) may be somewhat confusing, we keep them to maintain compatibility with the literature on this subject.

then in the frequency domain we have $P(z)$ as its system function:

$$P(z) = -\sum_{k=1}^{p} \alpha_k z^{-k} \quad . \tag{2.24}$$

The **prediction error**, $f(n)$, is defined as

$$f(n) = s(n) - \tilde{s}(n) = s(n) + \sum_{k=1}^{p} \alpha_k s(n-k) \quad . \tag{2.25}$$

Taking the z-transform of both sides of eq. (2.25) we observe that the prediction error sequence is the output of a system with the system function $A(z)$ to the input $s(n)$ [29].

$$A(z) = 1 + \sum_{k=1}^{p} \alpha_k z^{-k} \quad . \tag{2.26}$$

If $\alpha_k = a_k$, then from eq. (2.22) and eq. (2.23) $f(n) = Ge(n)$. This implies that the **prediction error filter**, $A(z)$, will be an *inverse filter* for the system, $H(z)$, of eq. (2.21); i.e.,

$$H(z) = \frac{G}{A(z)} \quad .$$

We have to estimate a_k 's for short segments or frames of speech due to the time-varying nature of speech. This leads us to a short-time estimation of a_k 's that is consistent with our discussion in Section 1.2.

The **short-time average prediction error** is defined as [29]

$$F_n = \sum_m \left(s_n(m) + \sum_{k=1}^{p} \alpha_k s_n(m-k) \right)^2 \quad . \tag{2.27}$$

where $s_n(m)$ is a segment of speech that has been selected in the vicinity of sample n, i.e.,

$$s_n(m) = s(n+m) \quad .$$

The range of summation in eq. (2.27), which is not specified for the time being, is the *reason behind existence of two different computational methods, namely the autocorrelation method and the covariance method.* F_n can be minimized by setting

$$\frac{\partial F_n}{\partial \alpha_i} = 0, \quad 1 \le i \le p \tag{2.28}$$

Solving eq. (2.28) we obtain

$$\sum_m s_n(m-i)s_n(m) = \sum_{k=1}^{p} \alpha_k \sum_m s_n(m-i)s_n(m-k), \quad 1 \le i \le p \tag{2.29}$$

Defining

$$\phi_n(i,k) = \sum_m s_n(m-i)s_n(m-k) \tag{2.30}$$

eq. (2.29) can be rewritten as [29]

$$\sum_{k=1}^{p} \alpha_k \phi_n(i,k) = \phi_n(i,0), \quad 1 \le i \le p \tag{2.31}$$

What we have is p equations in p unknowns which can be solved for predictor coefficients α_k that minimize F_n for the frame $s_n(m)$.

Since we are doing short-time analysis, the range of summation in eq. (2.27) must be a finite interval. As is pointed out earlier, this range of summation gives rise to the autocorrelation method and the covariance method which are discussed in greater detail in [29].

2.4.2.2. Computation of the Gain (G)

Some important assumptions and results will be reported. A complete derivation of the results can be found in [17]. As pointed out previously, if we can exactly estimate a_k's; i.e., $a_k = \alpha_k$, then

$$f(n) = Ge(n) \tag{2.32}$$

However, eq. (2.32) is valid to the extent that the ideal and the estimated linear prediction parameters are identical [29]. It is more realistic to assume that the energy in the error signal is equal to the energy in the excitation input [17,29]; i.e.,

$$G^2 \sum_{m=0}^{N-1} e^2(m) = \sum_{m=0}^{N-1} f^2(m) = F_n \tag{2.33}$$

Our objective is to express G in terms of the known quantities, namely the α_k's and

the correlation coefficients. In [4] it has been shown that for both types of inputs (impulse input and white noise input) eq. (2.34) holds

$$G^2 = R_n(0) + \sum_{k=1}^{p} \alpha_k R_n(k) = E_n \qquad (2.34)$$

where $R_n(k)$ is defined in eq. (2.18). It has also been shown that the first $p+1$ coefficients of the autocorrelation function of the impulse response of the model are identical to the first $p+1$ coefficients of the autocorrelation function of the speech signal. Mathematically, if the autocorrelation function of $h(n)$ is defined as

$$\tilde{R}(m) = \sum_{n=0}^{\infty} h(n)h(m+n) \qquad (2.35)$$

then

$$\tilde{R}(m) = R_n(m) \quad 0 \le m \le p \ . \qquad (2.36)$$

2.4.3. Lattice Formulations for Computation of the Coefficients (α_k)

So far, we have tried to predict a speech sample from past speech samples; i.e., *forward prediction*. Lattice methods employ the idea of *backward prediction* as well as forward prediction. *Backward prediction error* is defined as [29]

$$b^{(i)}(m) = s(m-i) + \sum_{k=1}^{i} \alpha_k^{(i)} s(m+k-i) \qquad (2.37)$$

for an i^{th}-order predictor. This implies that $s(m-i)$ is predicted from the i samples of the input $\{ s(m-i+k),\ k=1,2,...,i \}$ that follow $s(m-i)$. It has been shown that i^{th}-order forward (backward) prediction error is related to $(i-1)^{th}$-order prediction errors via eq. (2.38) (eq. (2.39)) [29].

$$f^{(i)}(m) = f^{(i-1)}(m) + k_i b^{(i-1)}(m-1) \qquad (2.38)$$

$$b^{(i)}(m) = b^{(i-1)}(m-1) + k_i f^{(i-1)}(m) \qquad (2.39)$$

where k_i is calculated using eq. (2.40). Bacward and forward prediction errors for the zeroth stage are equal to the input speech sample; i.e.,

$$f^{(0)}(m) = b^{(0)}(m) = s(m) \ .$$

Figure 2.13 depicts eqs. (2.38-39) as a signal-flow graph.

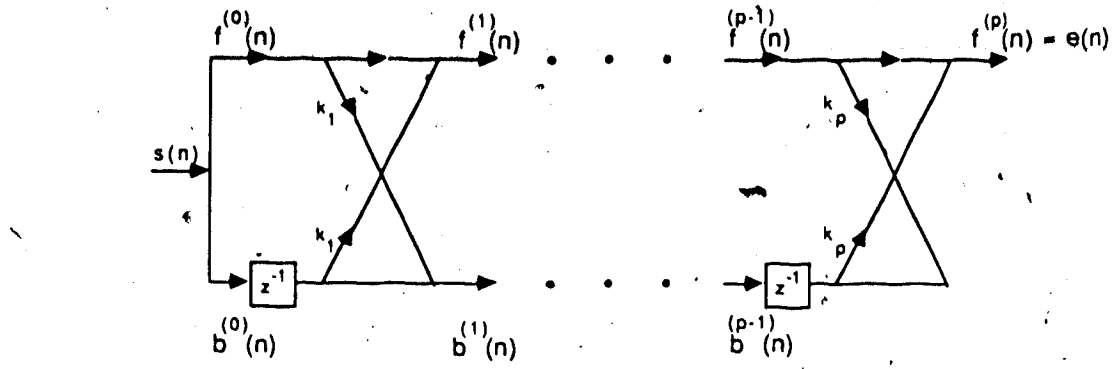

Figure 2.13 All-zero analysis lattice.

$$k_i = -\frac{\sum_{m=0}^{N-1} f^{(i-1)}(m)\,b^{(i-1)}(m-1)}{\left\{\sum_{m=0}^{N-1}(f^{(i-1)}(m))^2 \sum_{m=0}^{N-1}(b^{(i-1)}(m-1))^2\right\}^{1/2}} \tag{2.40}$$

Since the eq. (2.40) is in the form of a normalized cross-correlation function, the parameters k_i are called the *partial correlation coefficients* or PARCOR coefficients [29].

The quantities k_i can also be computed using Burg's method [29]

$$k_i = -\frac{2\sum_{m=0}^{N-1}\left[f^{(i-1)}(m)\,b^{(i-1)}(m-1)\right]}{\sum_{m=0}^{N-1}\left[f^{(i-1)}(m)\right]^2 + \sum_{m=0}^{N-1}\left[b^{(i-1)}(m-1)\right]^2} \tag{2.41}$$

which is obtained by minimizing the sum of the mean-squared forward and backward prediction errors.

In summary, α_i's and k_i's are computed as follows [29]:

1. Initially set $f^{(0)}(m) = s(m) = b^{(0)}(m)$.
2. Compute $k_1 = \alpha_1^{(1)}$ from eq. (2.41) or (2.40).
3. Determine $f^{(1)}(m)$ and $b^{(1)}(m)$ from eqs. (2.38-39).
4. $i \leftarrow 2$.
5. Determine $k_i = \alpha_i^{(i)}$ from eq. (2.41) or (2.40).
6. Determine $\alpha_j^{(i)}$, for $j = 1, 2, \ldots, i-1$ using

$$\alpha_j^{(i)} = \alpha_j^{(i-1)} + k_i \alpha_{i-j}^{(i-1)} \qquad 1 \le j \le i-1$$

7. Determine $f^{(i)}(m)$ and $b^{(i)}(m)$ from eqs. (2.38-39).
8. $i \leftarrow i + 1$.
9. If $i \le p$ go to step 5.
10. Stop.

2.4.4. Sequential Estimation Methods for Reflection Coefficients

Makhoul and Viswanathan in [18] distinguish two methods for computing a new set of reflection coefficients at each time instant n:

1. Block estimation, and
2. Adaptive estimation.

Block estimation

In this method, a set of reflection coefficients is computed using a "block" of sampled input values and the procedure defined in the previous section. When the next input sample arrives, a new block is defined and new coefficients are estimated. There is no simple functional relationship between the reflection coefficients of sample time n and those of sample time $n+1$. Forward and backward errors for each stage are re-computed for all time up to $n+1$, resulting in heavy computational requirements.

Adaptive estimation

In this method, the set of reflection coefficients to be used at the $(n+1)st$ sampling period are obtained by updating the set of reflection coefficients of n^{th} sample period. The amount of update depends (indirectly) on the input signal.

To exemplify this method, suppose the following window is used:

$$w(n) = \beta^n, \quad n \geq 0, \quad 0 < \beta \leq 1$$
$$w(n) = 0, \quad n < 0,$$

and the sum of forward and backward residual energies is minimized. $k_m(n+1)$ can be computed as

$$k_m(n+1) = k_m(n) - \frac{f_{m-1}(n)b_m(n) + b_{m-1}(n-1)f_m(n)}{D(n)}$$

where $D(n) = \beta D(n-1) + f^2_{m-1}(n) + b^2_{m-1}(n-1)$ [18].

As can be seen from the above equations, the forward and backward residuals are computed just once for each point in time. The SP1000 uses the adaptive estimation method with a different update formula. Adaptive estimation methods require less computation compared to block methods, but at the expense of noisier reflection-coefficients.

2.4.5. Frequency Domain Interpretation of Linear Predictive Analysis

So far, we have used linear prediction to estimate speech samples; i.e., we were working in the time domain. Another approach is to use linear prediction as a tool for *spectrum estimation*. This means that using predictor coefficients we can estimate the spectrum of our model. This can be accomplished by evaluating $H(z)$ on the unit circle.

2.4.5.1. Frequency-Domain Interpretation of Mean Squared Prediction Error

The mean-squared prediction error can be expressed in the time domain as [29]

$$E_n = \sum_{m=0}^{N+p-1} s_n^2(m) \qquad (2.42)$$

or in the frequency domain (using Parseval's theorem) as [29]

$$E_n = \frac{1}{2\pi}\int_{-\pi}^{\pi} |S_n(e^{j\omega})|^2 |A(e^{j\omega})|^2 d\omega \qquad (2.43)$$

where $S_n(e^{j\omega})$ is the DT Fourier transform of $s_n(m)$ and $A(e^{j\omega}) = A(z)|_{z=e^{j\omega}}$ (see eq. (2.25)). Substituting $\frac{G}{H(e^{j\omega})}$ for $A(e^{j\omega})$ we have

$$E_n = \frac{G^2}{2\pi}\int_{-\pi}^{\pi} \frac{|S_n(e^{j\omega})|^2}{|H(e^{j\omega})|^2} d\omega$$

Thus minimizing E_n is equivalent to the minimization of the integrated ratio of the $|S(e^{j\omega})|^2$ to its approximation $|H(e^{j\omega})|^2$. In Section 2.4.2.2 we noted that the autocorrelation function, $R_n(m)$, of $s_n(m)$ and the autocorrelation function, $R_n(m)$, of $h_n(m)$ are equal for the first $(p+1)$ values. As p increases, $|H(e^{j\omega})|^2$ becomes more similar to $|S(e^{j\omega})|^2$ and in the limit case, as $p \to \infty$, these two power spectra become identical; i.e.,

$$\lim_{p\to\infty} |H(e^{j\omega})|^2 = |S_n(e^{j\omega})|^2$$

What the above equation says is that any spectrum can be approximated arbitrarily closely by an all-pole model.

Figure 2.14 [29] illustrates the spectral modelling capability of linear prediction by showing $20\log_{10}|H(e^{j\omega})|$ and $20\log_{10}|S_n(e^{j\omega})|$. The signal spectrum was obtained by an FFT analysis of a 20 msec section of speech (sampled at 20 kHz), weighted by a Hamming window. The LPC spectrum was that of a 28-pole predictor obtained by the autocorrelation method.

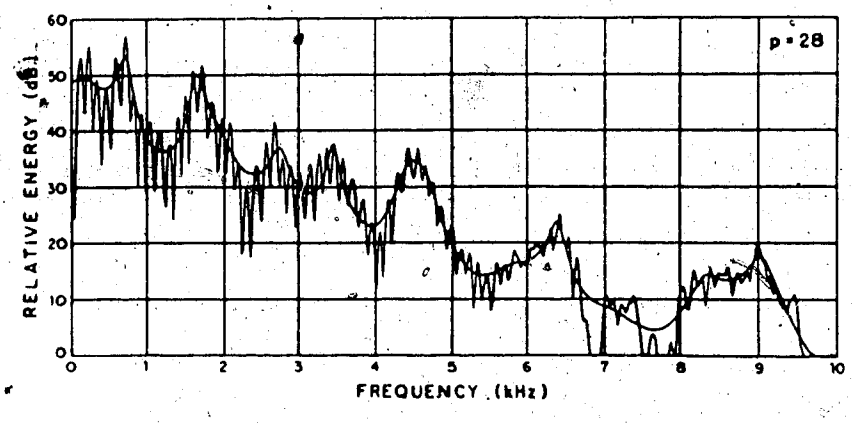

Figure 2.14 28-pole fit to an FFT signal spectrum.

2.4.6. Derivations of Some Other Speech Parameters from LPC Coefficients

2.4.6.1. Roots of the Predictor Polynomial

Poles of $H(z)$ or, equivalently, zeros of $A(z)$, z_k, $1 \leq k \leq p$ are either real or form complex conjugate pairs. Conversion of roots to the s-plane can be achieved by the following transformation

$$z_k = e^{s_k T} \tag{2.45}$$

where $s_k = \sigma_k + j\omega_k$ is the corresponding pole in the s-plane and T is the sampling period [29]. If the root $z_k = z_{kr} + jz_{ki}$ is in rectangular form, then it is easy to show that

$$\omega_k = \frac{1}{T} \arctan \frac{z_{ki}}{z_{kr}} \tag{2.46a}$$

and

$$\sigma_k = \frac{1}{2T} \log(z_{kr}^2 + z_{ki}^2) \ . \tag{2.46b}$$

Eqs. (2.46) can be utilized for formant analysis. The transformation defined by (2.45) has to do with a concept known as **impulse invariance** in DSP terminology and is

worth brief discussion.

Consider a system composed of N lossless tubes each of length $\Delta x = l/N$, where l is the total length of the vocal tract. In such a system, all the delays will be equal to $\tau = \Delta x/c$. Consider the response of the system to a unit impulse; i.e., $u_G(t) = \delta(t)$. The impulse propagates through the series of tubes. During propagation, the impulse is partially reflected depending on the reflection coefficients. The impulse response (i.e., the volume velocity at the lips) has been shown to be [29]

$$v_a(t) = \alpha_0 \delta(t - N\tau) + \sum_{k=1}^{\infty} \alpha_k \delta(t - N\tau - 2k\tau) . \tag{2.47}$$

Eq. (2.47) can be interpreted as follows: the soonest that an impulse can reach the lips is $N\tau$ seconds, hence the first term. Then successive impulses due to reflections at the junctions reach the lips at multiples of 2τ seconds later; hence the second term. Taking the Laplace transform of both sides of eq. (2.47), we have

$$V_a(s) = e^{-sN\tau} \sum_{k=0}^{\infty} \alpha_k e^{-s2\tau k} . \tag{2.48}$$

The second term,

$$V_a(s) = \sum_{k=0}^{\infty} \alpha_k e^{-sk2\tau} , \tag{2.49}$$

represents the resonance properties of the system. The impulse response for $V_a(s)$ is

$$v_a(t) = v_a(t + N\tau) . \tag{2.50}$$

Clearly,

$$V_a(j\Omega) = V_a(j\Omega + \frac{2\pi}{2\tau}) . \tag{2.51}$$

In words, the frequency response $V_a(j\Omega)$ is periodic in Ω with period $\frac{2\pi}{2\tau}$. This immediately reminds us of Figure 2.2 with T, the sampling period for A/D-DT filter-D/A-LPF structure, replaced by 2τ.

In summary, we can simulate the analog system characterized by $V_a(j\Omega)$, with a A/D-DT filter-D/A-LPF structure, where the sampling period $T = 2\tau$ and the

frequency response of the DT filter is $V(e^{j\omega})$ such that $V(z)|_{z=e^{sT}} = V_a(s)$.

We can approach the same problem from a slightly different point of view of impulse invariance. The unit sample response for the DT filter can be defined as

$$v(n) = v_a(t)|_{t=nT} \tag{2.52}$$

since $v_a(t)$ is nonzero only at the integer multiples of 2τ; i.e., the impulse response of the DT filter consists of the samples of the impulse response for the analog system, hence the term **impulse invariance**. It is straightforward to show that $V(z)|_{z=e^{sT}} = V_a(s)$.

2.4.6.2. PARCOR Coefficients

The LPC coefficients can be transformed into the PARCOR coefficients using the following recursive relations [29]:

$$k_i = a_i^{(i)} \tag{2.53}$$

$$a_j^{(i-1)} = \frac{a_j^{(i)} + k_i a_{i-j}^{(i)}}{1 - k_i^2} \quad 1 \le j \le i-1 \tag{2.54}$$

where i varies from p to 1, decreasing by 1 at each iteration. Initially, $a_j^{(p)} = \alpha_j$, $1 \le j \le p$.

2.4.6.3. Log Area Ratio Coefficients

The log area ratio coefficients are defined as [7]

$$g_i = \log\left[\frac{A_{i+1}}{A_i}\right] = \log\left[\frac{1+k_i}{1-k_i}\right], \quad 1 \le i \le p \tag{2.55}$$

The g_i's have been found to be especially appropriate for quantization [17].

Chapter 3

General Instrument SP1000 Speech Processor

This chapter takes a close look at the General Instrument (GI) SP1000 speech chip around which the recognition hardware is designed. The first section examines architectural aspects of the SP1000 in order to make the so-called "rituals" that must be followed to control the chip seem less mysterious and arbitrary. At the heart of the SP1000 is a lattice filter which can be reconfigured for recognition as well as synthesis. A good understanding of the lattice filter, which is the subject of section 3.2, is essential to fully exploiting the capabilities of the SP1000. Sections 3.3 and 3.4 briefly discuss software and hardware interfaces for the SP1000. A sample block-level logical design of a vocoder incorporating the SP1000 is presented in the last section to further clarify the discussions in the first four sections and to demonstrate the versatility of the chip.

3.1. SP1000 Architecture

SP1000 is a 28 pin, NMOS chip combining speech analysis and synthesis functions in a single dual in-line package (Figure 3.1 [1]).

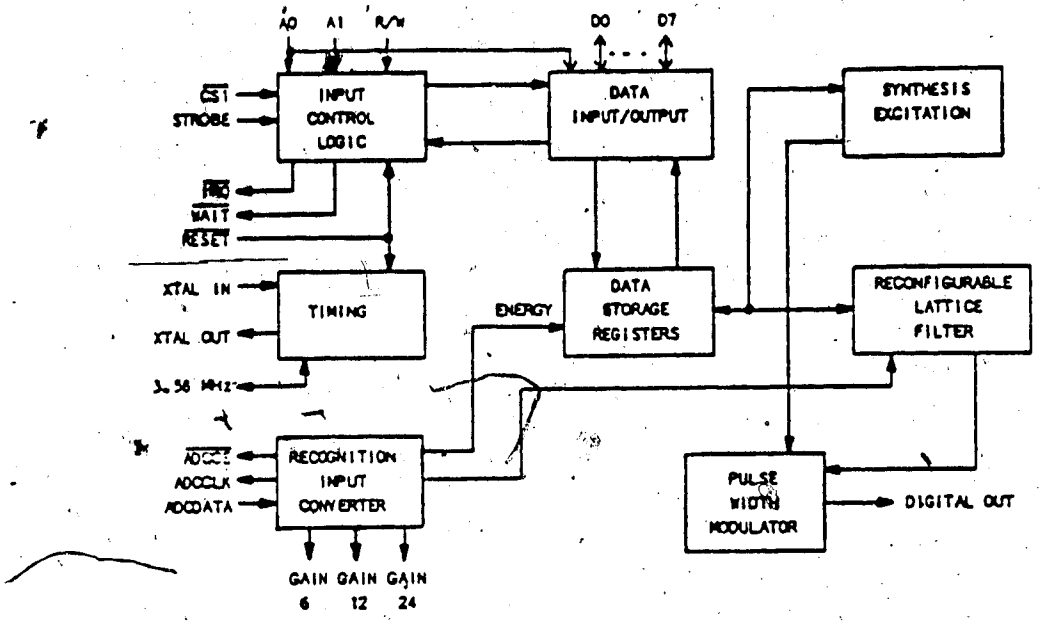

Figure 3.1 SP1000 block diagram.

The SP1000's major architectural feature is that all of the data paths in it are serial. Figure 3.2 [3] shows the functional blocks of the SP1000 as configured for analysis.

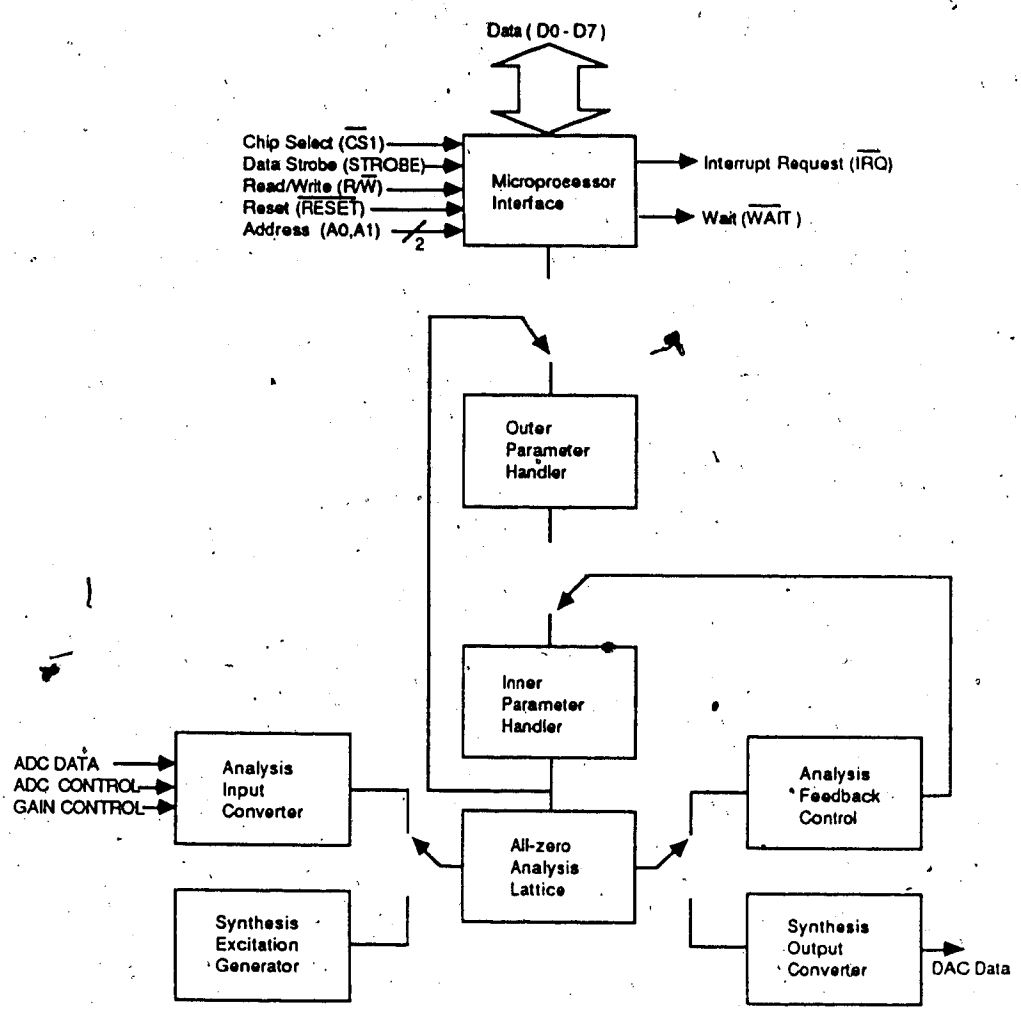

Figure 3.2 SP1000 configured as analyzer.

The position of switches in Figure 3.2 would be reversed to use the SP1000 as a synthesizer. Outer and inner parameter handlers are essentially large recirculating shift registers. The recirculation period is equal to the sampling period. The lattice filter gets its coefficient data from the inner parameter handler.

Analysis input converter (Figure 3.2) controls the analog-to-digital converter (ADC) via ADCCE and ADCCLK. It also controls the *automatic gain control* (AGC) circuit using GAIN 6, GAIN 12, and GAIN 24 lines. The analysis input converter receives the digitized speech sample from an external ADC chip via ADCDATA line (in serial fashion). The digitized speech sample is further processed and then fed into the lattice filter. The analysis feedback control implements the communication between filter stages.

In recognition mode, the lattice filter computes 8 reflection coefficients in each sample period and sends these estimates to the inner parameter handler. Sample-by-sample estimates of reflection coefficients can be averaged over a number of samples specified by one of the two timers on the chip. These averaged reflection coefficients are stored in the outer parameter handler and can be read by a microprocessor. Although one would normally write to (or read from) the outer parameter handler, it is also possible to write directly to (or read from) the inner parameter handler, bypassing the outer parameter handler. This fact is not shown in Figure 3.2 due to rare usage of this possibility.

In addition to reflection coefficients (8 for recognition, 10 for synthesis), the following parameters are kept in the parameter handlers:

Recognition mode: timer1, timer2, energy, and the sample rate.

Timer1 and *timer2* hold the number of samples to be counted down for each and are decremented by one after each sample period. Two bits in the status register indicate time-out condition for timer1 and timer2. The user can specify which timer can trigger

an interrupt. When an interrupt occurs, the user can read averaged coefficient values and energy from the outer parameter handler. Clearly, polled operation is also possible for the same purpose. Additionally, when timer2 runs out, gain control lines are updated. Therefore, timer2 is said to specify the *gain period*. *Energy* accumulates the absolute value of the input speech samples during a gain period.

Sample rate indirectly specifies how often the input audio signal should be sampled (see next section for details).

Synthesis mode: timer1, timer2, sample rate, excitation type (EXCTYP), and excitation amplitude (EXCAMP). Although our emphasis is on the recognition side of the SP1000, we will briefly discuss the above parameters to make the treatment complete. In synthesis mode, user can specify which timer should trigger the transfer of parameters from the outer handler to the inner handler. Instead of transfer, it is also possible to add the parameters in the outer handler to corresponding parameters in the inner handler which can be used to achieve a reduction in the rate of data transfer to the SP1000. Sample rate determines the frequency at which synthetic speech samples will be put out by the SP1000. EXCTYP determines which one of the eight different excitation sources will be input to the lattice filter. It is possible to provide external excitation to the filter on a sample-by-sample basis by writing directly to the EXCAMP field of the *inner parameter handler* and using EXCTYP code "1". This implies that the SP1000 can be used to perform any other signal processing task that can be implemented using a lattice filter. The EXCAMP field simply determines the amplitude of excitation.

3.2. Lattice Filter

The lattice filter can be configured as having 8 stages for analysis and 10 stages for synthesis (Figure 3.3).

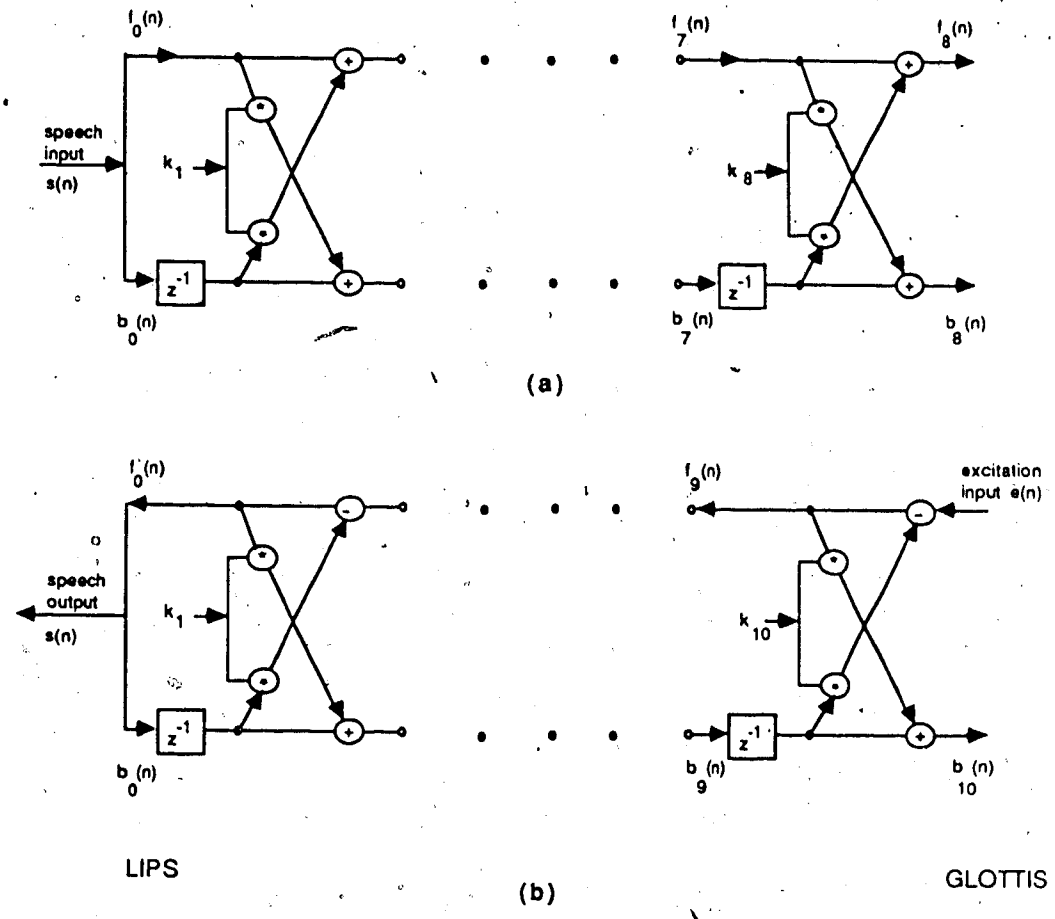

Figure 3.3 SP1000 lattice filter configured for (a) analysis (b) synthesis.

In analysis mode, the SP1000 estimates a new set of reflection coefficients at each sample period. Specifically, assume we have $k_i(n)$, $b_{i-1}(n-1)$, $1 \le i \le 8$, and $s(n)$. Then, forward and backward errors for n^{th} sample period and reflection coefficients to be used in $(n+1)$st sample period are computed using eqs. (3.1) [3] :

$$f_0(n) = b_0(n) = s(n) \quad , \tag{3.1a}$$

$$f_i(n) = f_{i-1}(n) + b_{i-1}(n-1) * k_i(n) \quad , \tag{3.1b}$$

$$b_i(n) = b_{i-1}(n-1) + f_{i-1}(n) * k_i(n) \quad , \tag{3.1c}$$

$$k_i(n+1) = k_i(n) - C f_i(n) SGN[b_{i-1}(n-1)] \quad , \tag{3.1d}$$

where $SGN[\,]$ is the sign of the argument inside the square brackets and C is a positive power-of-two constant. This scheme requires less computation than standard methods of adaptive estimation [18]. The price to be paid for the substantial savings in computation time and simplicity of implementation is less accurate reflection coefficients. However, for a limited-word, speaker-dependent IWR system, the loss in accuracy can be acceptable.

In the SP1000, only one filter stage is physically implemented. The structure of the computations as discussed above makes it possible to time-multiplex this filter stage in a sample period to achieve the result that can be produced by 8 "real" cascaded filter stages in analysis mode (10 stages in synthesis mode). As a result, only one filter stage needs to be reconfigured for the analysis/synthesis function. Figure 3.4 [3] depicts this time-multiplexed stage.

Time multiplexing of one filter stage gives rise to a concept that we will subsequently call the *stage period* . A sample period is divided into 8 stage periods in analysis mode (10, in synthesis mode). The value of the sample parameters in handlers determines the actual sample period as follows. In the current implementation the SP1000 is clocked at $f_c = 3.579545$ MHz; i.e. a clock period takes $1/f_c$ second. The sample rate (SR) parameter determines the number of clock periods during which the filter stage will be idle. Regardless of sampling rate, it takes 28 clock periods for a

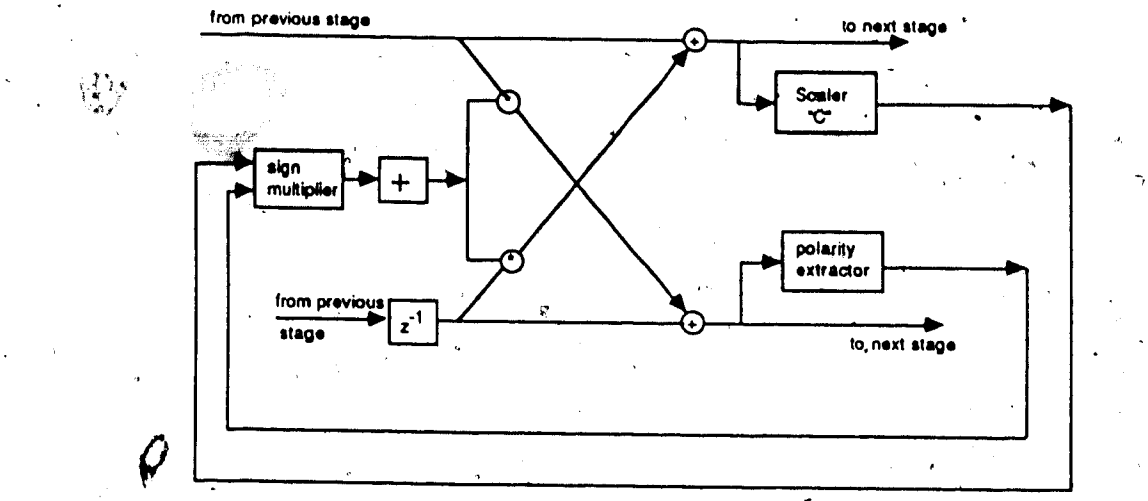

Figure 3.4 Time-multiplexed filter stage.

stage to perform the necessary computations. Thus, a stage period consists of $28+SR$ clock cycles. As a result, a sample period is

- $T_s = 8*(28+SR)/f_c$ seconds in analysis mode, and
- $T_s = 10*(28+SR)/f_c$ seconds in synthesis mode.

The SR can take on values in the range of [0,63] since the low-order 6 bits of this field are used. The highest and lowest possible values for sampling frequency can be computed substituting $SR=0$ and $SR=63$, respectively in the above formulae, giving the following ranges for both modes:

- analysis: [4916, 15980]Hz
- synthesis: [3933, 12784]Hz.

It should be noted that not all the frequencies in the above ranges are realizable because of the way the T_s is computed using SR.

Access to internal filter data path

It is possible to access the upper input path of a stage using the *test option*, which can be set by a bit in the control register of the SP1000. When this option is set, pin 11 of the SP1000 is disconnected from synthesis output and connected to the upper input path of the time-multiplexed filter stage. As a result, in analysis mode, $f_0(n)$, ..., $f_7(n)$ and in synthesis mode, $f_9(n)$, $f_8(n)$, ..., $f_0(n)$ appear on pin 11. Section 3.5 will present a design which makes use of the test option.

3.3. Microprocessor Software Interface for Recognition

When dealing with the SP1000, user sees four registers:

1. control register (write-only)
2. status register (read-only)
3. parameter address register(write-only)
4. parameter data register(read/write).

The operation to be performed on these registers is specified by R/W, A0, and A1 lines which, together with chip select and STROBE lines, make it easy to control the SP1000 as a memory-mapped I/O device [2]. Control and status registers share the same address in the microprocessor's address space since A0 and A1 are both 0 for write-to-control-register and read-status-register operations. Descriptions of the fields in control and status registers can be found in [1], so we will not duplicate them here.

The parameter address register specifies the handler (bit 5 is 0 for the outer handler and is 1 for the inner handler) and the parameter in it. The parameter data register holds the data that is read from or to be written to the SP1000.

The parameter data register differs from control and status registers in that any operation on control and status registers takes effect immediately. For example, a read-status operation returns the current state of the SP1000 and a write-control operation sets new control parameters "immediately". On the other hand, the contents

of the parameter data register after a read or write operation does not always reflect the current value of the parameter specified by the address register. This difference is completely due to the serial architecture of the SP1000 as mentioned in section 3.1. The parameter data register can be seen as a parallel-to-serial converter for write-data operation and serial-to-parallel converter for read-data operation. This register can tap both handlers, which are essentially recirculating shift registers with recirculation period of one sample period, in one place. The ability of tapping handlers in just one place means that to access a particular parameter in a handler one has to wait until that parameter passes by the tap and this waiting may take as long as a sample period if the desired parameter happens to have just passed by the tap when access is requested.

This synchronization problem is solved by the use of the BUSY bit of the status register. The BUSY bit is set when a transfer between the data register and a parameter handler is requested and remains set until the transfer is complete. The operations that set the BUSY bit are both write commands (codes 2 and 3) and read-data-and-fetch command (code 7) [1]. The read-data (code 6) and read-data-and-fetch operations both return whatever is already in the parameter data register. In addition to that, the read-data-and-fetch operation also requests data transfer between the parameter data register and the parameter designated by the parameter address register.

In the light of above discussion, the steps that must be taken to write to or read from a particular parameter of a particular handler are:

1. Specify the handler and the parameter using a write-to-parameter-address-register operation (unless it is known for sure that the parameter address register already contains the desired address).
2. To write eight bits of the parameter data register followed by a 0 (1) as the least significant bit (LSB), the write-data operation with code 2 (3) is issued.

To read, read-data-and-fetch (code 7) is used. The eigth-bit value returned by read-data-and-fetch is the current contents of the parameter data register and is discarded.

3. Wait until the BUSY bit is cleared which indicates that data transfer is complete. Now the 8-bit value obtained from the specified handler is in the parameter data register which can be retrieved by either read-data (code 6) or read-data-and-fetch (code 7) operation. If the latter is used, the low-order 4 bits of the parameter address register is automatically incremented and a fetch operation is initiated. It should be noted that the bit specifying the handler remains unchanged.

In the case of write operation, the parameter address register is similarly updated. The automatic update of the parameter address register is especially useful for accessing the consecutive parameters. References [1] and [2] should be consulted for more detailed information on the software interface for the SP1000.

3.4. The Hardware Interface for Recognition

A typical hardware interface for the SP1000 is illustrated in Figure 3.5 [2] which is also followed in the LIS'NER 1000 voice recognition board used in our implementation [12].

Rumble filter

The purpose of this filter is to eliminate the low-frequency noise related to power-line frequency. This filter is usually implemented as a high-pass analog filter. The cutoff frequency of the high-pass filter implemented in the LIS'NER 1000 recognition board is 250Hz.

Pre-emphasis filter

Distribution of energy to frequencies of interest for speech recognition is not uniform. The energy content of lower frequencies is higher than that of higher frequencies. A

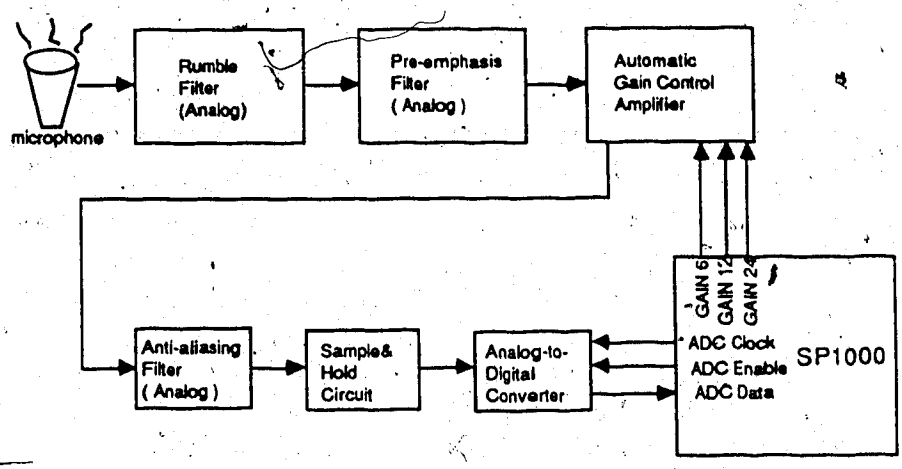

Figure 3.5 Typical hardware interface for the SP1000 for speech recognition.

pragmatic justification for the use of a pre-emphasis filter is to boost the higher frequencies so that reflection coefficients can capture more information about the shape of higher frequency portion of the spectrum of the input speech signal than would be possible with no pre-emphasis. In other words, reflection coefficients obtained from the hardware configuration depicted in Figure 3.5 do not model the spectrum of the original speech signal since the signal seen by the SP1000 is the pre-emphasized version of the original signal. For an IWR system based on pattern matching, this is of no consequence because both training and testing utterances will pass through the same front-end. A popular way of implementing pre-emphasis filter is to use a first-order analog high-pass filter with a frequency response such that higher frequencies of interest are boosted while at the same time lower frequencies are attenuated. Pre-emphasis can also be achieved by a first-order high-pass digital filter with a frequency response of $1-\mu z^{-1}$ where μ is in the range of [0.9, 1.0] [20].

From a more theoretical viewpoint, pre-emphasis can be used to eliminate the spectral contributions of glottal waveform $G(s)$ and lip radiation $L(s)$ so that the reflection coefficients characterize the vocal tract frequency response [20]. More specifically, by choosing $P(s) = L(s)$ and recalling that $S(s) \approx V(s)E(s)/L(s)$ (see Chapter 2), after analog-to-digital conversion we have the following as z-transform of the pre-emphasized speech signal which is seen by the SP1000:

$$P(z)S(z) \approx V(z)E(z) \quad . \tag{3.2}$$

The reflection coefficients estimated by the SP1000 from the pre-emphasized speech signal uniquely and indirectly (see chapter 2) determine an inverse filter $A(z) = 1/V(z)$. As a result the z-transform of the residual signal output from the last stage (but not accessible, see section 3.2) is (Figure 3.6)

$$P(z)S(z)A(z) \approx V(z)E(z)/V(z) \approx E(z) \quad . \tag{3.3}$$

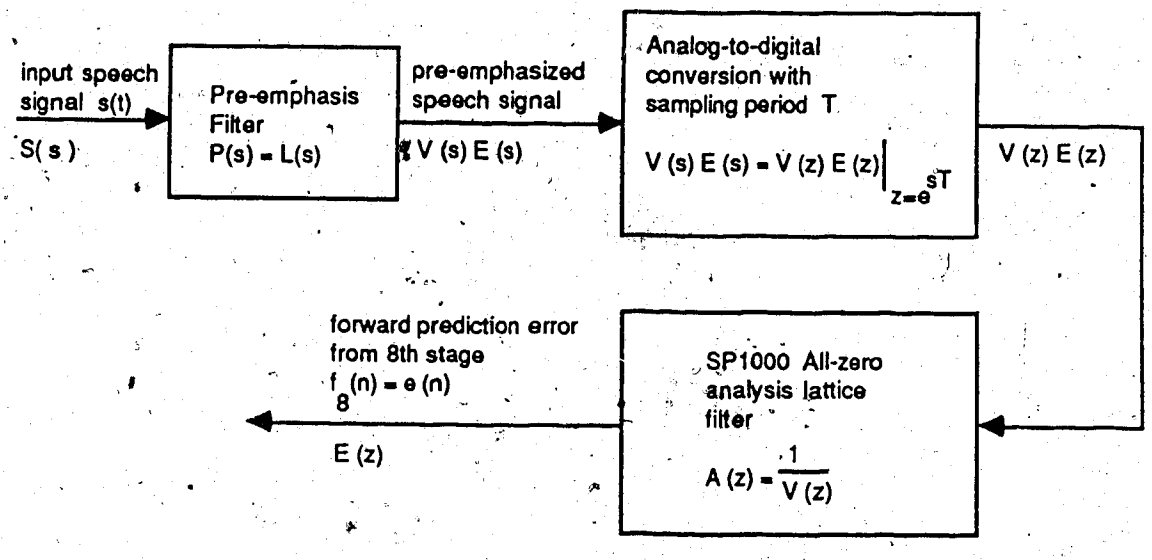

Figure 3.6 Pre-emphasis for the SP1000-based speech analysis.

Automatic Gain Control Amplifier (AGC)

The purpose of the AGC amplifier is to keep the average input speech amplitude at a fairly constant level so that the adaptation time of the filter for low-amplitude portions of input speech is close to the adaptation time of the filter for high-amplitude portions. This can be seen from eq. (3.1d) by observing that the amount of update on $k_i(n)$ is proportional to $f_i(n)$ which, in turn, is dependent on the input speech signal (eq. (3.1a)). A side benefit of the AGC scheme is that the dynamic range of the input speech signal is limited and reduced, making the use of an inexpensive 8-bit analog-to-digital converter feasible [22]. GAIN 6, GAIN 12, and GAIN 24 lines which determine the gain of the amplifier are updated based on the accumulated absolute value of the speech samples taken in a gain period which is specified by timer2 of the SP1000.

Anti-aliasing filter

As discussed in Chapter 2, the purpose of this filter is to prevent aliasing. In the LIS'NER 1000 voice recognition board, this filter is implemented as a two-pole, active, low-pass filter which cuts off at 3200 Hz.

Sample-and-hold circuit and analog-to-digital conversion

The sample-and-hold circuit consists of a switch and a capacitor and its input is the output of the anti-aliasing filter. Normally, the switch is closed and the voltage on the capacitor follows the input signal. When the ADCCE becomes active low, the switch is opened and the capacitor is disconnected from the input signal. The sample value is the voltage stored on the capacitor which feeds the ADC. ADCCLK provides clock signal to ADC and digital value of the sample is serially transferred from ADC to the SP1000 via the ADCDATA line. The ADC used in the LIS'NER 1000 board is an 8-bit ADC0831 manufactured by National Semiconductor [12].

3.5. An Example Design: SP1000-based Vocoder

In this section, we will discuss a block-level logical design of an SP1000-based vocoder. Our design is not meant to be detailed and complete. Our purpose in this section is to illustrate, by way of this example design, how the SP1000 can be used for other speech processing tasks. In doing so, several aspects of the SP1000 which might go unnoticed in the context of IWR as described in Chapter 4 will be clarified. This design will also involve the use of the synthesis capability of the SP1000 and access to the internal lattice filter data.

The *vocoder* (**Voice Coder**) is a device to reduce the data rate for digitized speech well below the levels achievable with waveform coding techniques [9]. Although the speech quality obtained by the use of vocoders is not usually acceptable for public telephone conversations, vocoders have been successfully used in military applications. Figure 3.7 illustrates the use of vocoders in voice communication.

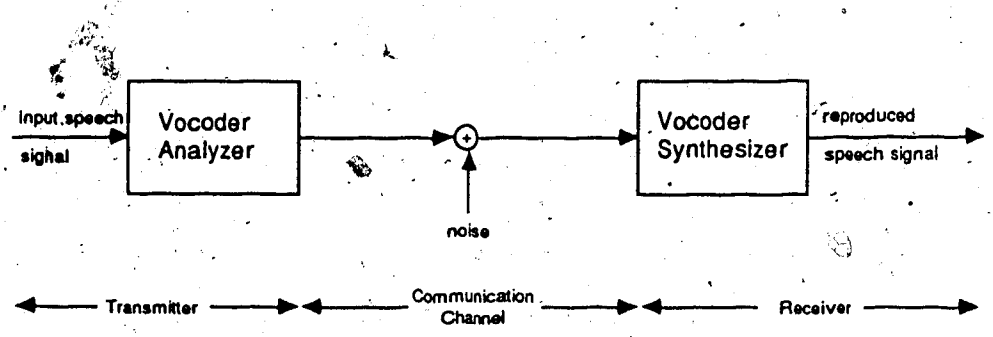

Figure 3.7 Vocoders in voice communications.

In our design, eight reflection coefficients, the voiced/unvoiced decision, the pitch period (if voiced), and the sampling rate will be transmitted over the channel.

Vocoder analyzer

Figure 3.8 shows the major blocks of the vocoder analyzer.

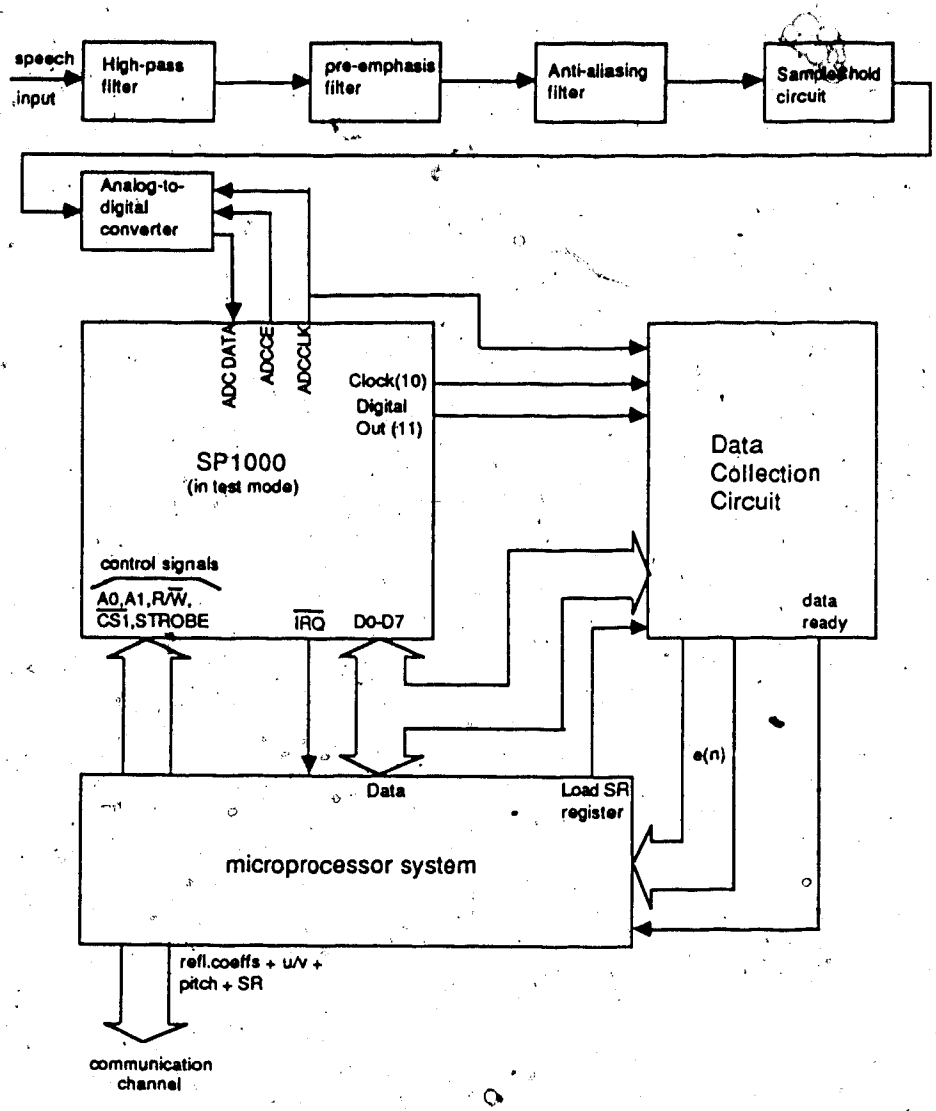

Figure 3.8 Vocoder analyzer.

The eight reflection coefficients are easily obtained from the SP1000. The residual output signal can be used to make a voiced/unvoiced decision and to compute the pitch period. Since this signal is not accessible (see Section 3.2), the residual from the 7^{th} stage will be used as an approximation to it. The data collection circuit (Figure 3.9) extracts this signal from pin 11 (digital out) of the SP1000 which must be in the test mode [1].

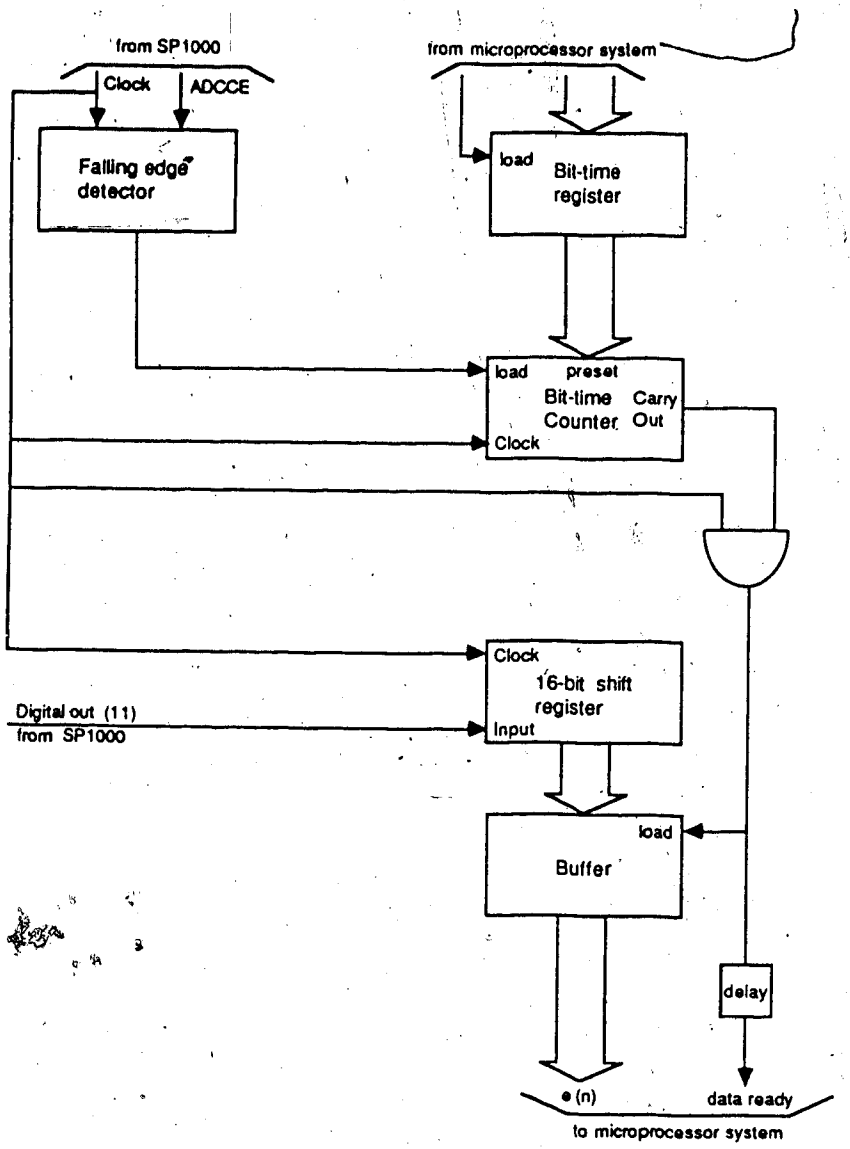

Figure 3.9 Data collection circuit.

The LSB of the 16-bit forward prediction error or residual from a particular filter stage appears on pin 11 in the first clock cycle of a stage period. Then, for a sampling rate SR, the most significant bit (MSB) of the residual from filter stage n, $1 \leq 7 \leq n$, will be on pin 11 after $(n-1)*(28+SR) + 16$ clock periods passed since the start of a sample period which is indicated by the positive-to-negative transition of ADCCE line. The microprocessor computes this time-offset and writes it to bit-time register.

The bit-time counter is loaded from the bit-time register at the start of each sample period. The bit-time counter is decremented at each clock pulse (pin 10) and when it reaches zero a load pulse is generated and the contents of the shift register is transferred to a 16-bit buffer which is read by the microprocessor system. The mechanism (e.g., DMA channel) used by the microprocessor system to retrieve and buffer the forward prediction error samples is beyond the scope of this design.

The microprocessor system can perform autocorrelation analysis (see Section 2.3) on the buffered values of the residual signal to make the voiced/unvoiced decision and to compute the pitch period for voiced parts [29]. The SIFT algorithm of Markel [19] can also be adapted to this situation for pitch period estimation. The microprocessor system also gets reflection coefficients from the SP1000 at each frame period and transmits them along with voiced/unvoiced decision, the pitch period (if voiced), and the sampling rate over the channel. The sampling rate is not to be transmitted at each frame period.

Vocoder synthesizer

As shown in Figure 3.10 , vocoder synthesizer is comprised of three major blocks, namely a microprocessor system, the SP1000 in synthesis mode, and the external output circuitry.

The microprocessor system receives eight reflection coefficients, the voiced/unvoiced decision, the pitch period, and the sampling rate and controls the SP1000. The syn-

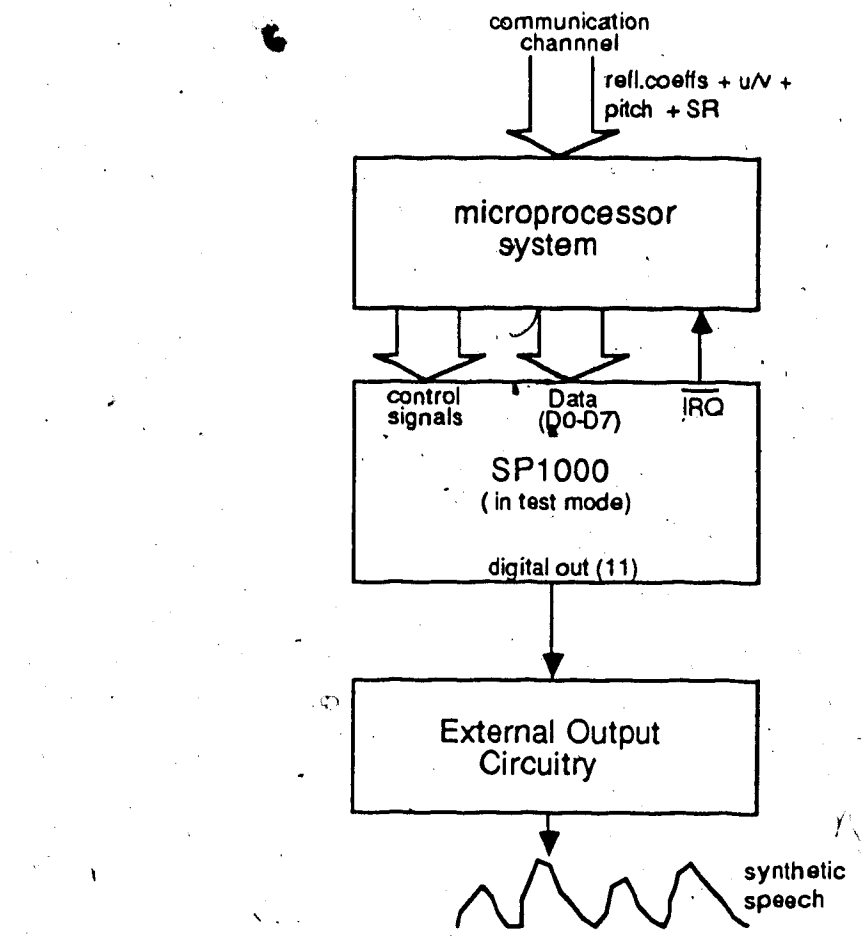

Figure 3.10 Vocoder synthesizer.

thesis lattice filter of the SP1000 has ten stages, but only eight reflection coefficients are transmitted by the analyzer. Therefore, the microprocessor system sets k_9 and k_{10} to zero, effectively cancelling out the ninth and tenth synthesis stages (see Section 3.2). The EXCTYP parameter is set according to voiced/unvoiced decision and the pitch period is written to timer1. The sampling rate transmitted by analyzer determines the corresponding parameter of the SP1000.

Two different methods can be used to implement the external output circuitry.

1. If the SP1000 is in *normal* mode [1], the pulse-width modulated (PWM) output

appears on pin 11. The recovery of synthetic speech from the PWM output occurs by means of the spectrum PWM signals [5,34]. Basically, the spectrum of a PWM signal consists of *carrier* lines at the harmonics of the sampling frequency f_s, which are accompanied by a symmetrical pattern of *sideband* lines that are due to the modulating signal (i.e., synthetic speech samples). As the name implies, the carrier lines are introduced into the spectrum by the unmodulated pulse train which is the "carrier" of the modulation.

The structure of the spectrum suggests that the modulating signal can be recovered by passing the PWM signal through a low-pass filter, cutting off at $f_s/2$. However, the output of the LP filter would not be an exact replica of the modulating signal since the sidebands extend indefinitely outward from each carrier line, with decreasing amplitude at greater distances. As a result, any LP filter with a cutoff frequency of $f_s/2$ must include some of the lower sideband components of the carrier line of f_s and to a lesser extent the lower sideband components of $2f_s$, etc. The output of the LP filter is of lower quality but still intelligible despite these distortions [5]. On the positive side, the LP filter is easier and cheaper to implement compared to the second method which will be discussed next.

2. This method can produce better quality speech by exploiting the full internal precision of the SP1000 but is more costly and complicated to implement. The basic ideas behind this scheme are to use 16-bit synthetic speech samples instead of PWM output and to use a digital-to-analog converter (DAC) followed by an LP filter to produce analog speech output. In order to get 16-bit speech samples from the SP1000, it must be in test mode. Figure 3.11 illustrates the external output circuitry which in many respects is similar to the data collection circuit of the analyzer. The 16-bit speech sample can be obtained from the upper filter path of

the first stage (one which uses k_1 as reflection coefficient) via pin 11. In synthesis mode, ALACCE makes a positive-to-negative transition at the start of the seventh stage period of a sample period. As a result, the microprocessor system computes the number of clock periods to be written to the "bit-time register 0" as $3*(SR+28)+16$. This design also differs from that of the data collection circuit of the analyzer in that there are two bit-time registers. The reason for this is that the parameters written to the SP1000 in response to an interrupt in, say, the n^{th} pitch period are used in the $(n+1)$st pitch period.

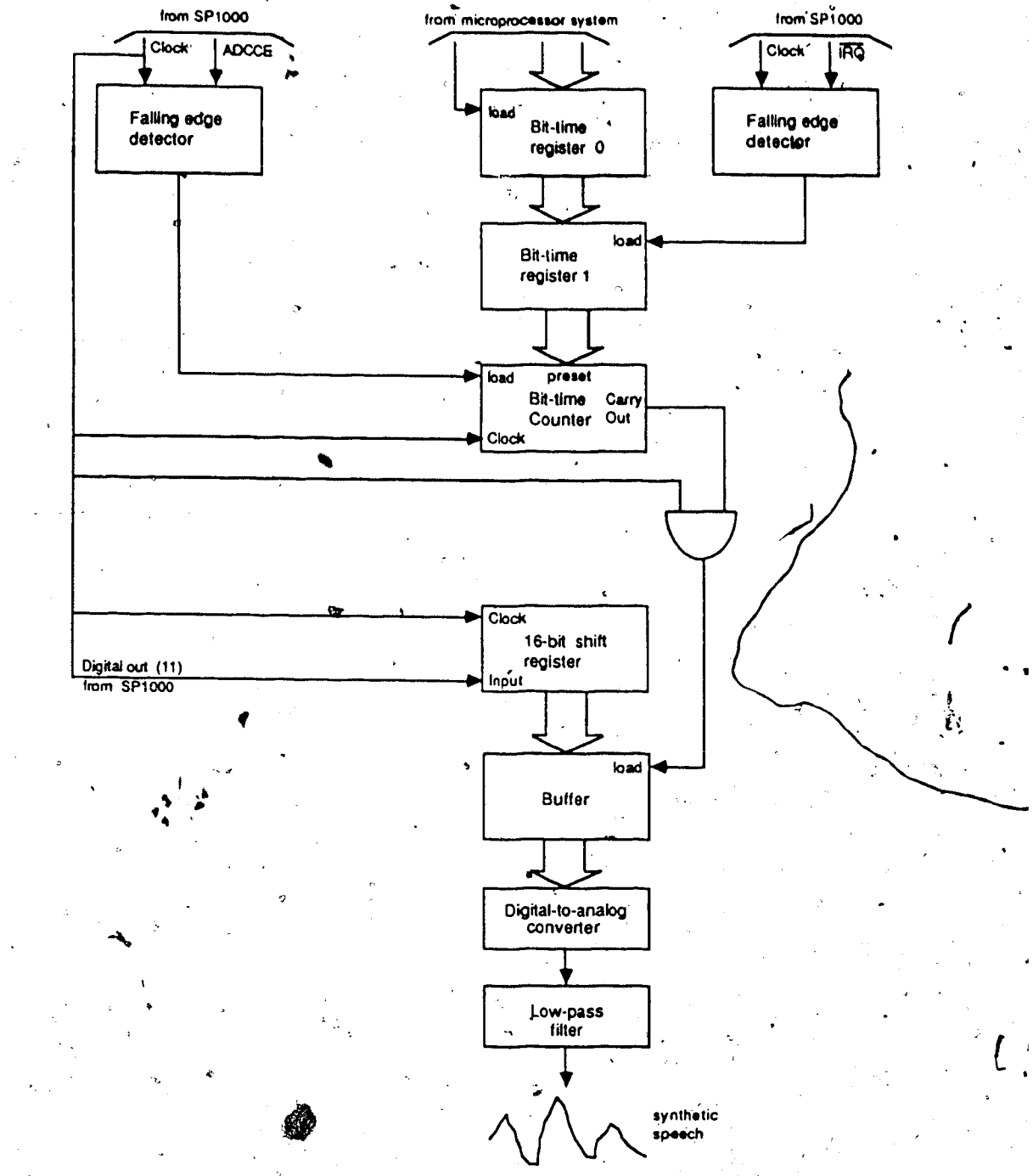

Figure 3.11 External output circuitry for vocoder synthesizer using test mode.

Chapter 4

Design and Implementation of an Isolated Word Recognizer

This chapter discusses the design and implementation of **IWRT** (Isolated Word Recognizer Trainer). Our discussion parallels Chapter 1. Section 4.1 reports how feature extraction is done in the IWRT. Section 4.2 examines the IWRT's endpoint detection scheme. Section 4.3 discusses the linear time normalization process which compresses or stretches all the reference/test patterns to a standard length. The subject of how the reference patterns are saved is elaborated upon in Section 4.4. The DTW algorithm used for pattern matching and the decision rule employed for making recognition decisions are treated in the last two sections.

4.1. Feature Extraction

The features of the speech signal that are extracted by the IWRT are eight reflection coefficients and the energy level. This decision is imposed on the design by the capabilities of the SP1000 as discussed in Chapter 3.

The sampling rate for the input audio signal and the frame period can be determined by the user at run time. If these parameters are not specified, default values of 6500 Hz and 20 msec. are assumed for sampling rate and frame period, respectively. The gain period is fixed as half of the frame period for the sake of quick response to changes in the input signal.

4.2. Endpoint Detection

Estimation of the endpoints is based solely on the energy level of the input frames. An implicit approach to endpoint estimation is ruled out because too much freedom in determination of the endpoints tends to degrade the recognition performance [13]. IWRT's endpoint detector was designed with a *hybrid* approach (see Chapter 1) in mind, and accordingly it generates an ordered set of endpoint pairs.

However, after experimentation with the system, it has been observed that the best endpoint pair estimated by the endpoint detector was quite accurate most of the time. Also, considering the response time and the high recognition accuracy obtained using only the best endpoint pair, we decided to use only the best endpoint pair. Relatively clean transmission conditions and use of a close-talking headset-style microphone contribute to better determination of endpoints. Endpoint detection is done "on-the-fly": i.e., the IWRT continuously "hunts for" an utterance in the incoming frames. The utterance is assumed to have ended if a silence period of more than 300 msec. is detected.

Average background noise (expressed in the decibel or dB scale) is an important factor in endpoint detection. When the training or recognition mode is entered, an initial average noise level is computed. The user can specify at run-time how many frames should be used in this computation. The IWRT also continuosly monitors the energy level of background noise and updates the average level when a frame classified as silent is found before the start of utterance is detected. The user can specify how many of the most recent frames will be involved in the average noise level computation.

An utterance is assumed to contain one or more *pulses*. As a result, the first step in endpoint estimation is to detect *valid* pulses in the input signal. *Pulse detection* is followed by *forming* all possible endpoint pairs. In the last step these pairs are *ordered*.

Pulse detection

Three thresholds, th_1, th_2, and th_3 are used in the process of determining which frames belong to a pulse. th_1, th_2, and th_3 have the values of 4 dB, 8 dB, and 6 dB, respectively. Let f_i, e_i, f_b, and f_l denote the i^{th} frame, the energy level (in dB) of f_i, the starting (first), and the ending (last) frame of a pulse, respectively.

Now assume that f_n is the first frame in time whose energy e_n exceeds th_1. Then

f_n is recorded as f_s. If the energy level falls below th_1 for f_m, $m > n$ before rising above th_2, then $f_n, f_{n+1}, \ldots f_m$ are classified as a false start and are discarded. Otherwise, $f_n, f_{n+1}, \ldots, f_m$ are considered part of pulse, unless the *rise time* (time taken by the frames f_i, $n \leq i \leq m$) is greater than 80 msec. in which case they are also discarded and f_s is recorded as f_{m+1}.

Suppose f_l is the first such frame that comes after the starting frame f_s (i.e., $l > s$) and its energy level is below th_3 (i.e., $e_l < th_3$). IWRT's endpoint detector also keeps track of the latest frame which comes before f_l satisfying $e_l < th_2$. Let us call this frame f_f. If the *fall time* taken by the frames $f_f, f_{f+1}, \ldots f_l$ is greater than 80 msec., then these frames are discarded and f_l is recorded as f_{f-1}. A fall time greater than 80 msec. may be caused by heavy breathing at the end of utterance.

In order to be considered valid, the pulse that has been just detected must pass two tests. If the highest energy level in the frames making up the pulse is less than 10 dB or the pulse duration is less than 80 msec., then the pulse is considered invalid and all the frames in it are discarded. Such an invalid pulse might be produced by a door slam, lip smacking, etc.

If a valid pulse is followed by a silence period of at least 300 msec., then an utterance which may contain all the valid pulses detected so far is assumed over.

Forming possible endpoint pairs

Having detected valid pulses, the endpoint detector proceeds to form all possible endpoint pairs using the following assumptions:

1. An utterance contains one or more valid pulses.
2. Every endpoint pair should include the pulse which contains the highest energy level.

Ordering endpoint pairs

In this step, endpoint pairs are ordered according to the following criteria.

1. Pulses separated by more than a silent gap of 150 msec. are less likely to be part of the same utterance.

2. Endpoint pairs containing more pulses are given preference in the ordering process.

3. Endpoint pairs with less overall silent gap duration are given preference in the ordering process.

In summary, IWRT's endpoint detector detects valid pulses, forms all possible endpoint pairs, and orders them. The ordered set of endpoints can be used to create test/reference patterns and to make recognition decisions.

4.3. Linear Normalization of Reference/Test Patterns

The purpose of linear normalization is to make all reference and test patterns contain the same number of frames. It has been shown in [23] that DTW algorithms perform best when the ratio of the length of the reference pattern to the length of the test pattern approaches 1, and worst when the ratio is near the bounds of the interval [1/2, 2]. This is because the area of the region of the time-warp grid in which the optimal path can lie is maximum when the ratio is 1, assuming a maximum compression/expansion factor of 2 [32].

Linear normalization is performed as follows. Let us assume the pattern to be normalized has N frames and the standard length is S frames. First, fixed upsampling with a ratio of 1:S is applied to the pattern, yielding an upsampled version of $N*S$ frames. In other words, each frame of the pattern is replicated S times.

This upsampled version is then divided into S groups so that each group contains N consecutive frames. Each group is averaged to obtain a frame of the normalized pattern. It should be noted that averaging is done across the same parameter of N con-

secutive frames.

User can specify the standard length at run time. A default value of 30 is assumed if it is not specified.

4.4. Storage of Reference Patterns

The scheme for storage of reference patterns is designed with two objectives in mind, namely future inclusion of clustering capability and ease of use.

Reference patterns for each word in a particular vocabulary are stored in a *template file*. Also, for each vocabulary, a *link file* containing the frame period, sampling rate, and length of normalized patterns that are used in training for the vocabulary is created. Names of template files for a particular vocabulary follow the format

<link file name>.<word identification>

A template file may contain more than one reference pattern for a word. A new reference pattern created during the training phase is simply appended to the end of a template if one already exists. Also, words in a particular vocabulary do not have to have the same number of reference patterns. A new word can be added to a vocabulary at any time. However, all the reference patterns in a particular vocabulary should be obtained using the same values for sampling rate, frame period, and standard length. A reference pattern consists of $9 * S$ bytes where S is the standard length (in frames) of patterns and 9 is the number of bytes in a frame, containing eight reflection coefficients and the energy level.

In recognition mode, the user is prompted to enter the name of a link file for the vocabulary to be tested. Values of the sampling rate, frame period, and standard length parameters of the test run should match those recorded in the link file. Identifications of the words in the vocabulary are derived from the file names in the directory in which the IWRT is run. This enables the user to use standard UNIX commands to rename and remove template files at any time and obviates a special

interface for dealing with the template files.

4.5. Dynamic Time Warping Algorithm

Endpoint constraints

Since the first and last frames of an utterance are estimated by the endpoint detector of the IWRT, endpoint constraints are

$$i(1) = j(1) = 1$$

and

$$i(K) = j(K) = S$$

where K is the length of common time axis for particular test and reference patterns (see Chapter 1) and S is the standard length of patterns for a particular vocabulary.

Local constraints and axis orientation

Assuming test pattern is along the x-axis, Figure 4.1 shows the local constraints used in the IWRT. Numbers on each arc indicate the weight attached to that path. Myers et al. [23] have shown that the combination of a test pattern along the x-axis and the weighting function illustrated in Figure 4.1 provides significant improvement in recognition accuracy.

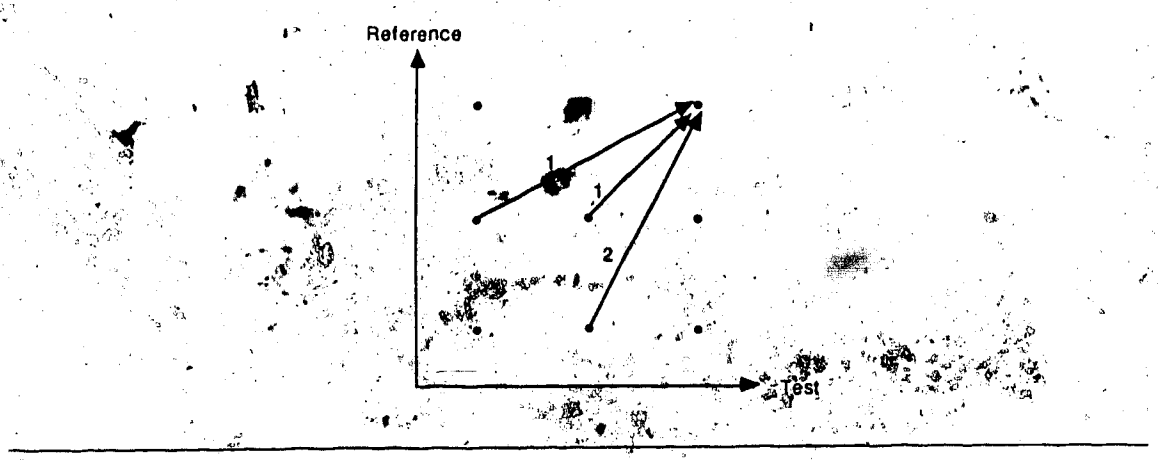

Figure 4.1 Local constraints and weighting function used in the IWRT.

The recursive formulation for the DTW algorithm with these local constraints is given below.

$$D_A(n,m) = \min\begin{Bmatrix} D_A(n-1,m-1) + d(TP(n),RP(m)) \\ D_A(n-1,m-2) + 2d(TP(n),RP(m)) \\ D_A(n-2,m-1) + d(TP(n),RP(m)) \end{Bmatrix}$$

where n, m are frame indices for test and reference patterns, respectively. Definitions for $D_A(n,m)$ and $d(TP(n),RP(m))$ can be found in Chapter 1.

Global constraints

Local and endpoint constraints discussed above define a parallelogram in which the optimal path can lie (see Figure 1.4). Therefore, accumulated distances are calculated only for those points that are inside the parallelogram.

Distance measure

Two methods for computing the local distance between two frames are implemented in the IWRT:

1. Euclidean distance on reflection coefficient values as read from the SP1000.
2. Euclidean distance on log area-ratios.

Since computation of log area-ratio is much more complicated than a simple subtraction required for the first method, the second method results in much longer response time compared to the first method. On the other hand, theoretically, the distance computed via the second method is more reliable. Since the IWRT is conceived as a research tool, emphasis is placed upon easy incorporation of new methods for local distance computation. Which method to use depends heavily on the response time and accuracy requirements of a specific application.

4.6. Decision Rule

A modified form of NKM rule is implemented in the IWRT. The value of K (see Chapter 1 for definition) is the lesser of 3 and the number of reference patterns for a particular word. The IWRT lists the four best candidates and scores in testing mode for thorough evaluation of performance.

Chapter 5

Results and Further Research

5.1. Results

A speaker-trained IWR system, **IWRT**, has been implemented successfully. The IWRT can be trained to recognize isolated words of a designated talker. As discussed in Chapter 4, various parameters of the IWRT can be set at run time, making the fine tuning of the system for a particular speaker an easy task.

There is no universally accepted test to evaluate the performance of an IWR system. As a result, designers of isolated word recognizers (in general, speech recognition systems) have developed their own testing strategy and recognition accuracies reported by them cannot be compared to each other on a common basis. Consequently, any such score of accuracy must be qualified with the particular way the experiment is conducted.

It should be noted that "confusability" of a vocabulary has a greater impact on recognition accuracy than the size of the vocabulary. For instance, it is quite likely that a specific IWR system can achieve a higher score on a vocabulary consisting of, say, 100 acoustically rich, polysyllabic words than on a vocabulary consisting of, say, 30 monosyllabic, "similar-sounding" words.

Also, experience of a particular talker affects the recognition accuracy to a large extent. Some speakers are problematic; i.e., they produce sounds extraneous to what is to be said. For example, "um"s, "err"s, heavy breathing, etc. are common impediments to accurate endpoint detection. Also, some speakers have the ability to repeat a particular word consistently whereas some produce quite different versions for it on different occasions. The talker's mood and stress also have an impact on recognition performance.

In summary, considering the characteristics of vocabulary, speaker-related factors, background noise conditions etc., it is difficult to obtain an objective and quantitative measure of performance of an IWR system.

Some experiments have been conducted with the IWRT. In these experiments, the talker was the author of this paper and the vocabulary consisted of 10 Turkish numbers and 28 Turkish words that could be of use in a text editor. The sampling rate, frame period, and standard pattern length had values of 6500 Hz, 20 msec, and 15 frames, respectively. In the training phase, some of the words were said twice while some were said only once. Two reference patterns are created for words that have been observed confusable with other words in the vocabulary. In testing phase, each word in the vocabulary has been said four times over three days. The IWRT failed twice to recognize the correct word resulting in %98.6 (=150/(4*38)) recognition accuracy. However, the correct word was still in the best four recognition candidates.

Two failures were due to the Turkish number four which sounds like English "dirt". The final stop /t/ was the cause of failures. Such a stop is characterized by a stop gap (time interval during which the pressure in vocal tract is built up) with very little energy followed by a burst with fair amount of energy. Depending on the weakness/strength and duration of the burst, the endpoint detector may detect it as a valid pulse or discard it. Two reference patterns for this word had only one valid pulse corresponding to a short and weak burst of /t/. Two test repetitions of this word having stronger and longer bursts caused the two failures.

In fact, weak final stops are well-known as a common source of errors in endpoint detection since it is very difficult to distinguish them from background noise using only energy measurements. Another feature of speech, such as zero-crossing density, can be used to distinguish final stops from background noise. Highest energy and duration conditions for a valid pulse might be loosened in order to include the weak plosive

bursts in the utterance; but this increases the risk of detecting a burst of background noise as a valid pulse. This tradeoff can be solved by considering the background noise conditions and accuracy requirements for a specific application. Having two reference patterns corresponding to "weak-burst" and "strong-burst" versions of this word might be helpful to a certain extent depending on the existence of similar words in the vocabulary.

5.2. Further Research

The IWRT can be made speaker-independent by incorporating a clustering scheme into the current implementation. As was mentioned in Chapter 4, the foundations for such an improvement have already been laid.

The design and implementation of voice input system for a specific task using the IWRT as a basis would be an interesting project. The IWRT can be modified so that it accepts strings of isolated words produced according to a task-imposed grammar. The use of syntactical constraints would limit the number of words that can be selected at "branching points" of the grammar. This would result in higher overall recognition accuracy and faster response than would be possible without such constraints.

Another improvement to the IWRT might be to design and implement a connected-word recognition capability, which is more difficult than the IWR problem. However, a two-level [33] or one-pass [6] dynamic programming approach for pattern matching can be utilized for that purpose. A moderate amount of modification to the IWRT's DTW algorithm would permit connected-word recognition.

References

[1] General Instrument (GI), SP1000 Preliminary Data Sheet, 1983.

[2] General Instrument (GI), Speaker dependent speech recognition using the SP1000, 1983.

[3] General Instrument (GI), private communication, 1987.

[4] B. S. Atal and S. L. Hanauer, "Speech analysis and synthesis by linear prediction of the speech wave", *JASA*, Vol. 50, 1971 , pp. 637-655.

[5] H. S. Black, *Modulation theory*, D. Van Nostrand Co. Inc., 1953.

[6] J. S. Bridle, M. D. Brown and R. M. Chamberlain, "Continuous connected word recognition using whole word templates", *The Radio and Electronic Engineer*, Vol. 53, 1983, pp. 167-175.

[7] G. Bristow, ed., *Electronic speech synthesis techniques, technology, and applications*, McGraw Hill, 1984.

[8] R. O. Duda and P. E. Hart, *Pattern classification and scene analysis*, Wiley-Interscience, 1973.

[9] F. Fallside and W. A. Woods, *Computer speech processing*, Prentice Hall International, 1985.

[10] A. H. Gray and J. D. Markel, "Distance measures for speech processing", *IEEE Trans. ASSP*, Vol. ASSP-24, No. 5, October 1976, pp. 380-391.

[11] E. Horowitz and S. Sahni, *Fundamentals of computer algorithms*, Computer Science Press, 1978.

[12] Micromint, Inc., LIS'NER 1000 voice recognition, 1984.

[13] L. F. Lamel, L. R. Rabiner, A. E. Rosenberg and J. G. Wilpon, "An improved endpoint detector for isolated word recognition", *IEEE Trans. ASSP*, Vol. ASSP-25, No. 4, August 1981, pp. 777-785.

[14] W. A. Lea, "Establishing the value of voice communication with computers", *IEEE Trans. Audio and Electroacoustics*, Vol. AU-16, 1968, pp. 184-197.

[15] W. A. Lea, ed., *Trends in speech recognition*, Prentice Hall, 1980.

[16] L. C. Ludeman, *Fundamentals of digital signal processing*, Harper&Row 1986.

[17] J. Makhoul, "Linear prediction: a tutorial review", *Proc. of IEEE*, Vol. 63, No. 4, April 1975, pp. 561-580.

[18] J. Makhoul and R. Viswanathan, "Adaptive lattice methods for linear prediction, *Intl. Conf. ASSP*, 1978, pp. 83-86.

[19] J. D. Markel, "The SIFT algorithm for fundamental frequency estimation", *IEEE Trans. on Audio and Electroacoustics*, Vol. AU-20, No. 5, December 1972, pp. 367-377.

[20] J. D. Markel and A. H. Gray, *Linear prediction of speech*, Springer-Verlag, 1976.

[21] T. Martin, Applications of limited vocabulary recognition systems, *Rec. 1974 Symp. Speech Recognition*, 1975, pp. 55-71.

[22] T. [illegible], K. Skoge, D. Vetter and P. Ahrens, A combination of speech synt[illegible] recognition Integrated Circuit, *Intl. Conf. ASSP*, Boston, 1983, pp. [illegible].

[23] C[illegible]rs, L. R. Rabiner and A. E. Rosenberg, "Performance tradeoffs in dy[illegible] time warping algorithms for isolated word recognition", *IEEE Tr[illegible] ASSP*, Vol. ASSP-28, December 1980, pp. 622-635.

[24] A. [illegible] Oppenheim and R. W. Schafer, *Digital Signal Processing*, Prentice [illegible], 1975.

[25] A. V. Oppenheim, ed., *Applications of digital signal processing*, Prentice Hall, 1978.

[26] A. V. Oppenheim, A. S. Willsky and I. T. Young, *Signals and systems*, Prentice Hall, 1983.

[27] L. R. Rabiner and M. R. Sambur, "An algorithm for determining the endpoints of isolated utterances", *Bell Systems Technical Journal*, Vol. 54, February 1975, pp. 297-315.

[28] L. R. Rabiner and M. R. Sambur, "Application of LPC distance measure to the voiced-unvoiced-silence detection problem", *IEEE Trans. ASSP*, Vol. ASSP-25, No. 4, August 1977, pp. 338-343.

[29] L. R. Rabiner and R. W. Schafer, *Digital processing of speech signals*, Prentice Hall, 1978.

[30] L. R. Rabiner and J. G. Wilpon, "Application of clustering techniques to speaker-trained isolated word recognition", *Bell Systems Technical Journal*, Vol. 58, No. 10, December 1979, pp. 2217-2233.

[31] L. R. Rabiner, and J. G. Wilpon, "Considerations in applying clustering techniques to speaker independent word recognition", *Journal of Acoustical Society of America*, Vol. 66, No. 3, September 1979, pp. 663-673.

[32] L. R. Rabiner and S. E. Levinson, "Isolated and connected word recognition - theory and selected applications", *IEEE Trans. on Communications*, Vol. COM-29, No. 5, May 1981, pp. 621-659.

[33] H. Sakoe, "Two-level DP-~~matching~~ a dynamic programming based pattern matching algorithm for connected-word recognition", *IEEE Trans. ASSP*, Vol. 27, 1979, pp. 288-295.

[34] H. Taub and D. L. Schilling, *Principles of Communication systems*, McGraw Hill, 1971.

[35] G. M. White, Dynamic programming, the Viterbi algorithm, and low cost speech recognition, *IEEE Intl. Conf. on ASSP*, Tulsa, Oklahoma, 1978.