Old Dominion University

ODU Digital Commons

Signal Modeling with Non-Uniform Time Sampling of Features for Automatic Speech Recognition

Montri Karnjanadecha
Old Dominion University

SIGNAL MODELING WITH NON-UNIFORM TIME

SAMPLING OF FEATURES FOR AUTOMATIC SPEECH

RECOGNITION

by

Montri Karnjanadecha
M.Eng. 1995, Prince of Songkla University, Thailand
B.Eng. 1990, Prince of Songkla University, Thailand

A Dissertation Submitted to the Faculty
of Old Dominion University in
Partial Fulfillment of the
Requirement for the Degree of

DOCTOR OF PHILOSOPHY

ELECTRICAL ENGINEERING

OLD DOMINION UNIVERSITY

August 2000

Approved by:

Stephen A. Zahorian (Director)

John W. Stoughton (Member)

Oscar R. González (Member)

Wu Li (Member)

ABSTRACT

SIGNAL MODELING WITH NON-UNIFORM TIME SAMPLING OF
FEATURES FOR AUTOMATIC SPEECH RECOGNITION

Montri Karnjanadecha
Old Dominion University, 2000
Director: Dr. Stephen A. Zahorian

This dissertation presents an investigation of non-uniform time sampling methods for spectral/temporal feature extraction in speech. Frame-based features were computed based on an encoding of the global spectral shape using a Discrete Cosine Transform. In most current "standard" methods, trajectory (dynamic) features are determined from frame-based parameters using a fixed time sampling, i.e., fixed block length and fixed block spacing. In this research, new methods are proposed and investigated in which block length and/or block spacing are variable. The idea was initially tested with HMM-based isolated word recognition, and a significant performance improvement resulted when a variable block length and variable block method were applied. An accuracy of 97.9% was obtained with an alphabet recognition task using the ISOLET database. This result is by far the highest reported in the literature. The variable block length method was then adapted to accommodate the complexity of continuous speech. Three methods were proposed and each was tested with the TIMIT and NTIMIT databases using HMM recognizers. Phone recognition experiments were conducted using the standard 39 phone set. Tuning of parameters was achieved with monophone models using a simple

HMM configuration. The methods were also evaluated with more complex models, such as models with more mixture components, models with a full covariance matrix and right-context biphone models. Experimental results indicated that none of the proposed methods perform significantly better than the standard method. However, the absolute best result obtained with the proposed front end is comparable to those obtained with current state-of-the-art systems. Also, the performance achieved with monophone models is favorable to many context-dependent systems which are more complex.

This dissertation is dedicated to my parents.

ACKNOWLEDGMENTS

I especially acknowledge the Thai government for financially supporting me during my entire graduate program of study in the United States. My Ph.D. would not have been possible without this support.

I wish to express my sincere gratitude to my dissertation advisor, Dr. Stephen A. Zahorian, for his invaluable advice and patience throughout the research work. His expertise in the area of speech processing allowed me to work with confidence and expand my knowledge over the last four years of my working in the Old Dominion University Speech Communication Lab. I would also like to acknowledge the National Science Foundation, grant BES-9977260, which partially supported some of this work during the final stages.

I would like to thank Dr. John Stoughton, Dr. Oscar González and Dr. Wu Li for serving on my dissertation committee and for their time and assistance.

I would like to thank all my friends in the Speech Communication Lab for establishing a positive working environment. Special thanks go to Matt Zimmer for maintaining the fine computer systems in the lab. His assistance enabled my experiments to run smoothly.

I wish to thank all my friends for their friendship and support and for making my living in this country a joy.

Finally, I wish to thank my family for their support. My deepest thanks go to my wife, Phansiri, for her hard work, encouragement and patience.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

The ultimate automatic speech recognition (ASR) system should be a system that transcribes any input utterance as well or better than humans under the same conditions. At the present time, such an ASR system does not yet exist. Although, speech scientists and engineers have been working on ASR problems for almost half a century with great success, there remain many issues for researchers to still explore.

Many ASR systems available today are, nevertheless, capable of achieving a useful level of performance. Several of them are integral components of computer software products for general computer users. Speech dictation software is a popular application for ASR that can be purchased at a reasonable cost. A user can create and edit a text document using his/her speech with the aid of ASR [1]-[4]⋮. ASR can also be used to help non-native speakers to speak a language [5] or to teach hearing-impaired people to speak [6].

Speech recognition is accomplished in two main steps: front-end processing and recognition. The front-end analyzer takes samples of speech as the input and computes a sequence of feature vectors that represent important characteristics of the speech data. In the recognition step, the sequence of feature vectors is segmented and each segment is classified.

⋮ The journal model used for this dissertation is IEEE Transactions on Speech and Audio Processing.

The outcome of this step is called a hypothesis transcription of the input speech.

1.1 Importance of Front-end Analysis

Front-end analysis is the first step in the speech recognition process whereby the digitized speech signal is analyzed and a sequence of multidimensional feature vectors is created. Ideally, these features are extracted such that all information needed for speech recognition is preserved while irrelevant information, such as background noise, loudness, speaker-specific characteristics, etc., are suppressed. This goal should be accomplished using as few parameters as possible.

Currently, Mel-frequency Cepstral Coefficients (MFCC) are regarded as the de facto standard acoustic features for ASR. Most high-performance automatic speech recognizers employ MFCC or a close variation.

Feature extraction is a crucial part of a high-performance speech recognition system. It has been a research topic since the beginning attempts to automatically recognize speech. It is impossible to construct a state-of-the-art recognizer using low quality features, no matter how well the recognizer performs. On the other hand, if speech features were able to capture all necessary speech characteristics such that each speech unit can be correctly classified, a sophisticated statistical model would be unnecessary for the recognizer. In reality, speech is not pronounced very clearly and the environment may not be quiet. Also some distortions may be introduced during the process. Furthermore, there are no perfect feature extraction methods. Hence good higher-level speech modeling

is still necessary. The other point that should be noted is that, for non-clearly uttered speech, even human listeners have difficulties in recognizing the speech. Humans heavily rely on higher-level modeling in the form of contextual information to identify the most likely word sequence they are listening to.

In summary, speech recognition cannot totally rely on acoustic features because there are no features that fully separate every category of speech. A powerful statistical classifier is required for a high performance system. In some way, contextual information should be included in the signal modeling and should be efficiently incorporated within the framework. Task grammar is another major constraint that limits the search space and can be used to improve recognition accuracy.

1.2 Related Work

This work mainly focuses on the issue of spectral/temporal information encoding for speech. Since the introduction of the combination of static and dynamic spectral features for speech recognition by Furui [7] and [8], there has been a tremendous amount of work in speech recognition that utilizes such techniques. Basically, "dynamic" features are extracted from static features using a linear regression formula over a time interval on the order of 50 to 200 ms. This basic technique is widely used today.

Even though adding dynamic terms to the feature set does increase the dimensionality of the feature space, and thus requires more training data, most speech recognition systems gain recognition accuracy with this method. Several feature transformations, such as Linear Discriminant

Analysis (LDA) and Principal Components Analysis (PCA), can be applied to reduce the dimensionality without significant performance degradation.

Ever since the early 1980s, Hidden Markov Model (HMM) [9] based recognizers have become popular due to the elegant mathematical foundation of the HMM and its power to model the variability in speech durations. However, an HMM has a well-known problem in that it lacks the ability to model temporal information within each state. Also, an HMM is based on an independence assumption which means the probability of a certain symbol being generated does not depend on any previously generated symbols but rather on the current state. This assumption is invalid. Explicit inclusion of temporal information into the feature set, e.g., to combine static and dynamic features, has proven to greatly improve recognition performance of HMM-based speech recognition [8],[10],[11] and [12], at least partially because dynamic features violate the independence assumption to a lesser extent than do the static features.

Researchers have long realized the usefulness of temporal information. Thus, a significant amount of effort has been spent trying to improve ASR with the use of such knowledge. So far, research has been conducted in three different directions. The first direction has been to modify the front-end analyzer to better capture the speech information and incorporate the temporal characteristics in the feature set. The second approach has been to improve the HMM foundation to take the temporal information of the observation into consideration and to relax the independence assumption. The third method has been to avoid using HMMs by creating a new framework that corrects the HMM's flaws. The next part of this section briefly discusses these three approaches.

Furui [7] successfully applied dynamic spectral features to speaker-independent isolated word recognition. The task was to recognize 100 Japanese city names using a Dynamic Programming (DP) matching technique. Dynamic features were computed from intervals of 56 ms of Linear Prediction Cepstral Coefficient (LPCC) parameters using linear regression methods. The combination of static and dynamic terms was reported to be robust and superior to using static features alone. Dynamic terms computed directly from static terms are usually referred to as delta (or velocity) coefficients. In fact, even higher order coefficients can be obtained from the delta coefficients by applying the linear regression formula to the delta coefficients. These are called delta-delta (or acceleration) coefficients. The combination of static, delta and delta-delta terms resulted in performance improvements in ASR with normal speech [8] and with Lombard speech [13].

If computational efficiency is a major concern of a system, linear regression techniques can be replaced by a spectral difference method where a difference coefficient is obtained from the subtraction of the features of the two frames around a center frame. Difference coefficients are beneficial in speech recognition [13],[14].

Milner [12] tested and compared several methods for temporal transformations. A formula to compute spectral/temporal coefficients is generalized, where the transformation matrix can be determined in various ways. Experiments were conducted to compare the performance among the transformations based on the Discrete Cosine Transform (DCT), the Discrete Legendre Transform (DLT), the Discrete Rectangle Transform (DRT), the Karhunen-Loeve Transform (KLT), the Linear Discriminant Transform (LDT) and also using identity transforms and regression analysis. LDA, DCT,

DLT, DRT and KLT gave excellent results, but with best performance obtained with the KLT. In contrast, spectral/temporal features computed using regression analysis were not as well suited for ASR.

Much of the work done at the Old Dominion University Speech Communication Lab uses the DCT as a way to include temporal information into speech features with great success. Zahorian and Jaghaghi [16] and Zahorian and Nossair [17], successfully applied a DCS expansion over time ("dynamic" features) to spectral shape features for speaker-independent stop consonant recognition and for vowel classification.

Multi-resolution cepstral features have been proposed as an alternative to MFCC features for a speech representation [18]-[21]. Coefficients obtained from a DCT of the spectrum over the full bandwidth of speech were combined with coefficients obtained from the DCT of subbands of several levels. The proposed features were shown to perform better than MFCCs. Although the main purpose of the works was to explore the multi-resolution features, it also emphasized the inclusion of temporal information into the feature set.

Some researchers [22]-[24] have made attempts to extend the HMM framework to account for temporal modeling and to overcome the inherent poor within-state temporal modeling of the "standard" HMM. New training formulas and recognition algorithms were derived to accommodate the new acoustic models. Incorporating a neural network framework into the HMM has been a topic of interest for many researchers [25]-[27].

The acoustic segment model, an alternative to the HMM, is a technique that models speech at the segmental level.

This contrasts with an HMM which models speech on a frame-by-frame basis. These segment models have a capability to capture the temporal information of speech better than a conventional HMM. Many studies were focused on the same concept in which speech segments were explicitly determined and a statistical framework was used to model each segment [28]-[34]. The independence assumption was relaxed and temporal information was incorporated in the segment model. However, this framework has two major problems. First, segment boundaries need to be explicitly determined and second--because the length of each segment may be different--scoring must be done properly. Searching for segment boundaries is the most difficult and time-consuming part of the segment modeling technique. Hence, an efficient search technique is required to allow the recognition to run in real time [35],[36]. There are some Neural Networks-based and Wavelet Transform-based speech segmentation techniques that would fit in such segment model frameworks [37]-[39]. In summary, at the present time, none of these acoustic segment models perform as well overall as the HMM methods, and thus the segment models remain at a research stage.

1.3 Scope of Dissertation

The main objective of this dissertation is to investigate several approaches for computing spectral/temporal features using various approaches to implement a variable segment length (referred to as a "block" in this work). Discrete Cosine Transform Coefficients (DCTCs) extracted from a Discrete Cosine Transform (DCT) of the log magnitude spectrum are used as the "starting" static features. The "standard" HMM framework with standard training and recognition algorithms was used throughout to statistically model the speech. To

test our proposed methods, one series of experiments was conducted with isolated words using the ISOLET database. However, the main investigation was performed with continuous speech recognition using the TIMIT and NTIMIT databases.

The real objective of this work is to investigate variable length segments and segment spacings (i.e., non-uniform time samplings and time resolutions of speech features) for the purpose of improving ASR. This should be placed in a mathematical framework to help find segments which minimize a global mean square representation error. However, this general case is too difficult to tackle unless sufficiently supported by knowledge in the general area. For example, determination of segments so as to minimize a mean square error reconstruction may or may not be helpful in improving ASR accuracy. In the end, it is system performance in the actual environment that counts. The goal of this dissertation was to contribute some basic knowledge in the field which could later be augmented in order to achieve the real objective.

The specific objectives of this work were to:

1) Study variable block length techniques for feature extraction. The length of the block was determined from local Mean Squared Error (MSE) to measure differences between an original frame-based feature matrix and a matrix reconstructed from block features.

2) Investigate a variable block spacing technique, where the spectral derivative of frame-based parameters was used to determine optimum block spacing.

3) Investigate a technique based on combining blocks of different lengths using PCA.

Note that the standard method is a method whereby features are computed using a fixed block length and fixed block spacing. This served as a control method in this work.

1.4 Outline of Dissertation

Chapter 2 briefly describes several speech representations including DCTC. Computation of higher order features and feature transformations are explained. The basic concepts of HMMs, and the HTK toolkit used to implement HMMs, which was used for the speech recognition reported in this work, are also summarized. Chapter 3 discusses isolated word recognition with the ISOLET database using Discrete Cosine Series Coefficients (DCSCs). The feature computation techniques based on variable block length are presented. The task and database used for continuous speech recognition experiments are described in chapter 4. Chapter 5 is concerned with various signal modeling methods proposed in this dissertation. Chapter 6 discusses experiments conducted using proposed signal modeling methods. Finally, Chapter 7 presents conclusions of the research and suggestions for future work.

CHAPTER II

BACKGROUND

2.1 Representation of the Speech Signal

It would be impractical for a speech recognizer to directly model samples of the speech signal. Much of the detail of the speech signal is not relevant or only marginally relevant to speech message information. For example, the speech waveform samples are highly dependent on overall amplitude and short-time phase. Speech information has been shown to depend mainly on short-time spectral characteristics, primarily independent of overall gain and phase. Thus, the first step in representing the speech signal invariably consists of a short-term spectral magnitude computation, approximating the analysis of the human ear. However, even the spectral magnitude contains much redundancy and seemingly irrelevant information, and therefore this spectrum is generally reduced to a small number of parameters or "features." There are two kinds of approaches which guide the feature calculations of speech: articulation based and perceptual based. Articulation-based feature extraction attempts to model the transfer function of the human vocal tract using a method such as linear prediction. Perceptual based methods attempt to model that part of the human ear which is used to perceive speech.

A very good introductory paper that discusses many aspects of signal modeling of speech signals is authored by Picone [40]. In the paper, an overview of many techniques involved in signal modeling which have been developed over

the last 30-40 years are summarized. Interested readers are also directed to read a textbook by Rabiner and Juang [9].

2.1.1 Linear Prediction Analysis

Linear prediction (LP) is a least mean squared error algorithm that can be applied to and has been successfully used in many areas, including speech processing. There are several discussions of LP theory in the literature [41],[42],[43]. In linear prediction analysis of human speech, an all-pole filter is used to model the vocal tract. In other words, the vocal tract is considered as a filter and the goal is to determine the transfer function of the vocal tract. The transfer function of the filter is defined as

$$H(z) = \frac{G}{\sum_{i=0}^{p} a_i z^{-i}}$$

(2.1)

In the equation, G is the overall gain, p is the number of poles and $a_0 = 1$. The $\{a_i\}$ are filter coefficients that are selected to minimize the overall mean squared error over the analysis frame. The transfer function of the vocal tract is not unique, but rather is slowly time varying depending on the vocal tract "positions" during the course of the utterance. Thus, the LP coefficients, or more commonly some transformation of these coefficients, can be used for ASR features.

There are three ways to determine LP coefficients [41],[43]: the covariance method, the harmonic method and the autocorrelation method. The autocorrelation method is mostly used in speech recognition because it yields stable filters and is very computationally efficient. Note that in all of these methods, the mean square error minimization is

with respect to time-domain waveforms, as opposed to the frequency domain features, as considered in this work.

2.1.2 Digital Filter Bank Analysis

Filter bank analysis is the first method of spectral analysis which was used (and still is to some extent) for speech processing. In the past, the filters were implemented with analog circuits but now filters are implemented digitally with software running on a digital computer. Digital filter bank analysis is an alternative to LP analysis. Its purpose is to simulate the perceptual properties of the human ear. Filter bank analysis uses a set of band-pass filters whose center frequencies spread non-linearly across the frequency range of speech (typically approximately 100 Hz to 5 KHz). Each filter has a triangular shape frequency response and overlaps adjacent filters. The center frequencies of the filters are chosen to distribute non-linearly because the human ear does not resolve frequencies in a linear fashion.

There are two well-known perceptual frequency scales, the Bark scale and the Mel scale, used to determine filter spacings. The two scales are quite similar in that the filters are closely spaced in the low frequency area and coarsely spaced at high frequencies. The Mel scale is more commonly used for automatic speech recognition. It is approximated as a linear scale from 0 to 1,000 Hz and a logarithmic scale beyond 1,000 Hz.

FFT-based filter bank analysis is the most popular technique used to implement the Mel-scaled filter bank. The magnitude of the spectrum of a frame of speech is computed with an FFT and then summed appropriately to obtain each filter output. Each filter gives one parameter indicating

the magnitude of the spectrum in that particular filter channel. The filter outputs (typically 15 to 30) can be directly used to form a feature vector for each speech frame, simply called Mel Frequency Coefficients (MFC), or further transformed to MFCC parameters as described below.

2.1.3 Cepstral Coefficients

Cepstral coefficients, defined as the Inverse Fourier Transform (IFT) of the log-spectrum [44], can be computed using the Discrete Cosine Transform (DCT). Generally, cepstral coefficients are preferable to LPC or digital filter bank parameters. MFCCs can be obtained by applying the DCT to the log energy of each filter [45]. Not only is the number of parameters reduced by this transformation but also the resulting features are less correlated. The data reduction helps to reduce modeling complexity. Uncorrelated features are also beneficial for an HMM recognizer. Linear Prediction Cepstral Coefficients (LPCCs) can also be determined, by first performing an LP analysis followed by a cosine transform operation.

2.1.4 "Dynamic" Features

A great deal of research has shown that inclusion of spectral/temporal information, in addition to static (frame based) features, improves overall recognition performance of the system [7],[8],[10]-[13],[46]. These "dynamic" features, as they are often called, are determined from a collection of features of several adjacent frames. The purpose is to capture the change of each feature from frame to frame. This whole approach is motivated by the observation that humans recognize speech not only by its short-time spectrum but also by the change of its spectrum over time [46].

The standard method of computing dynamic features is to use a simple regression formula:

$$D_t = \frac{\sum_{i=-W}^{W} i C_{t+i}}{\sum_{i=-W}^{W} i^2}$$ (2.2)

where D_t is a delta coefficient at time t, C_t is a static coefficient at time t and W is the time span (in number of frames) in each direction around the center time t.

Acceleration coefficients (or delta-delta coefficients) can be obtained by applying (2.2) to the delta coefficients. Note, however, that the window size may be set differently. As discussed in [40], many systems use a window size of 3 ($W=1$) [14],[47],[48], and many systems use a much longer window size [8],[49].

Usually, the static features are augmented by these delta and/or acceleration terms. These augmented parameters are sometimes called spectral/temporal features. Notice that the dimensionality of the resulting feature vectors will be doubled or tripled. Further feature transformation may be required for dimensionality reduction.

2.1.5 Acoustic Representation based on the Encoding of Global Spectrum Shape

Though LP analysis for spectral analysis was used widely in the 1970's and early 1980's [40], most current systems rely on FFT-based spectral analysis. Filter bank analysis is usually later applied to compute MFCs and MFCCs. At the Old Dominion University Speech Communication Lab, a technique for feature extraction based on the encoding of global spectral shape has been used for many years

[10],[11],[16],[50]. While MFCC analysis applies the DCT to the energy of the spectrum across all filter bands, our method applies the DCT directly to the log-scaled spectrum. The resulting features are called DCTCs (Discrete Cosine Transform Coefficients). To account for the non-linearity of the human ear in speech perception, a modified DCT is used. The basis vectors of the standard DCT were modified by a bilinear warping function [16]. Use of warped basis vectors results in an operation that gives more resolution at low frequency and less resolution at high frequency. This concept is similar to a non-uniform distribution of the filters in a Mel-scale filter bank analysis.

We have also proposed a technique to compute spectral/temporal features by again using a Discrete Cosine Series Expansion but applying it to feature trajectories over time. The resulting parameters are called DCSCs (Discrete Cosine Series Coefficients). Instead of using the regression formula, this technique applies the DCT over a **block** or stack of frame features. One fundamental difference between DCSCs and MFCCs with delta and delta-delta terms is that the first DCSC is the smoothed version of the corresponding DCTC. Smoothing can cause loss of fine detail but it makes the features less noisy and more robust.

The following equations were presented by Zahorian and Nossair [16] to show the derivation of DCTC and DCSC parameters. They are provided here for the convenience of interested readers.

First, let $X(f)$ be the magnitude squared spectrum represented with linear amplitude and frequency scales and let $X'(f')$ be the magnitude spectrum as represented with perceptual amplitude and frequency scales. Let the relations

between linear frequency and perceptual frequency, and linear amplitude and perceptual amplitude, be given by:

$$f' = g(f), \qquad X' = a(X). \tag{2.3}$$

For convenience of notation in later equations, f and f' are also normalized, using an offset and scaling, to the range [0,1]. The acoustic features for encoding the perceptual spectrum are computed using a cosine transform,

$$DCTC(i) = \int_0^1 X'(f') \ cos(\pi \ if')df', \tag{2.4}$$

where $DCTC(i)$ is the ith feature as computed from a single spectral frame. Making the substitutions

$$f' = g(f), \ X'(f') = a(X(f)), \text{and}$$

$$df' = \frac{dg}{df} \ df, \tag{2.5}$$

the equation can be rewritten as

$$DCTC(i) = \int_0^1 a(X(f)) \ cos[\pi \ ig(f)] \ \frac{dg}{df} \ df. \tag{2.6}$$

We therefore define modified basis vectors as

$$\phi_i(f) = cos[\pi \ ig(f)] \ \frac{dg}{df}, \tag{2.7}$$

and rewrite the equation as

$$DCTC(i) = \int_0^1 a(X(f)) \ \phi_i(f) \ df. \tag{2.8}$$

Thus, using the modified basis vectors, all integrations are with respect to linear frequency. In practice, (2.8) can be implemented as a sum, directly using spectral magnitude squared values obtained from an FFT. Any differentiable warping function can be precisely implemented,

with no need for the triangular filter bank typically used to implement warping.

The terms computed with (2.8) *(DCTC(i))* are equivalent to cepstral coefficients. However, to emphasize the underlying cosine basis vectors and the calculation differences relative to most cepstral coefficient computations, we call them *DCTCs*, consistent with terminology in previous related work [16],[51].

In this dissertation, DCTC parameters were computed with (2.8) using a logarithmic amplitude scale (i.e., a() is the log function) and bilinear warping with a coefficient α = .45.

$$f' = f + \frac{1}{\pi} tan^{-1} \left\{ \frac{\alpha \ sin(2\pi \ f)}{1 - \alpha \ cos(2\pi \ f)} \right\}. \qquad (2.9)$$

The first three basis vectors, incorporating the bilinear warping, are shown in Fig. 2.1.
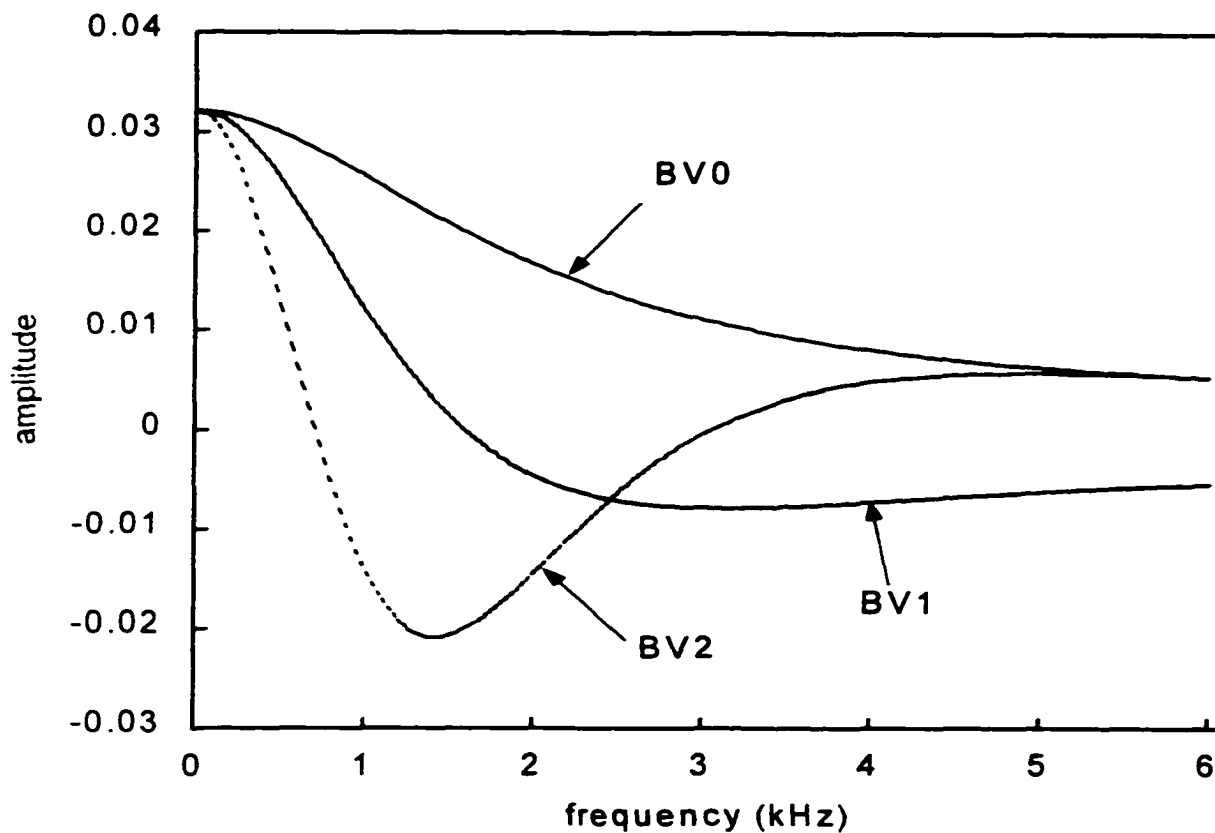
Fig. 2.1. The first three DCTC basis vectors, with a warping factor of 0.45 (after Zahorian and Nossair [16])

DCSC features were computed so as to encode the trajectory of the short-time spectra. Using the processing as described above, P DCTCs (typically 10 to 15) were computed for equally-spaced frames of data spanning a segment of each token. Each DCTC trajectory was then represented by the coefficients in a modified cosine expansion over the segment interval. The equations for this expansion, which are of the same form as for (2.7), allow non-uniform time resolution as follows.

Let the relation between linear time and "perceptual time" (i.e., with resolution over a segment interval proportional to estimated perceptual importance) be given by

$$t' = h(t).\qquad(2.10)$$

For convenience, t and t' are again normalized to the range [0,1]. The spectral feature trajectories are encoded as a cosine transform over time using

$$DCSC(i,j) = \int_o^1 DCTC'(i,t')\cos(\pi jt')\,dt'.\qquad(2.11)$$

The $DCSC(i,j)$ terms in this equation are thus the new features which represent both spectral and temporal information ("dynamic") over a speech segment. Making the substitutions

$$t' = h(t),\ DCTC'(i,t') = DCTC(i,t),$$

and

$$dt' = \frac{dh}{dt}\,dt,\qquad(2.12)$$

the equation can be rewritten as

$$DCSC(i,j) = \int_o^1 DCTC(i,t)\cos[\pi jh(t)]\,\frac{dh}{dt}\,dt.\qquad(2.13)$$

We again define modified basis vectors as

$$\theta_j(t) = \cos[\pi jh(t)]\,\frac{dh}{dt}\qquad(2.14)$$

and rewrite the equation as

$$DCSC(i,j) = \int_o^1 DCTC(i,t)\,\theta_j(t)\,dt.\qquad(2.15)$$

Using these modified basis vectors, feature trajectories can be represented using the static feature

values for each frame, but with varying resolution over a segment consisting of several frames. The terms computed in (2.15) are referred to as *DCSCs* to emphasize the underlying cosine basis vectors and to differentiate between expansions over time *(DCSC)* versus *DCTC* expansions over frequency. In general, each *DCTC* (index i in (2.15)) was represented by a multi-term *DCSC* (index j in (2.15)) expansion.

The function $h(t)$ was chosen such that its derivative, dh/dt, which determines the resolution for t', was a Kaiser window. By varying the Kaiser beta parameter, the resolution can be changed from uniform over the entire interval *(beta = 0)*, to much higher resolution at the center of the interval than the endpoints (beta values of 5 to 15). Fig. 2.2 depicts the first three *DCSC* basis vectors, using a coefficient of 5 for the Kaiser warping function. The motivation for these features is to compactly represent both spectral and temporal information with considerable data reduction relative to the original features. For example, 12 *DCTCs* computed for each of 50 frames (600 total features) can be reduced to 48 features if 4 *DCSC* basis vectors are used for each expansion.
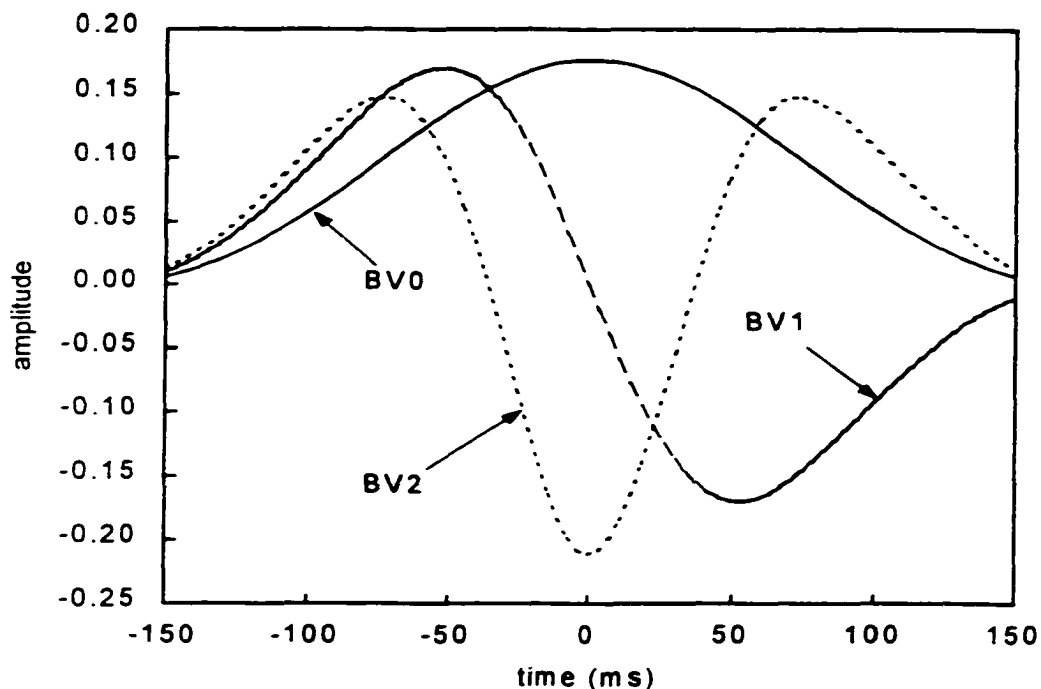
Fig. 2.2. The first three DCSC basis vectors, with a coefficient of 5 for the Kaiser warping function (After Zahorian and Nossair [16]).

2.2 Transformations of Speech Features

Two main reasons that motivate researchers to perform a feature transformation as a post-processing step in front-end analysis are: 1) to reduce dimensionality of the feature set and 2) to improve separability (classification power) of the features. There are several techniques for feature transformations, but only the two most popular methods will be discussed in this dissertation.

The transformations discussed here are linear. Let X = [x_1, x_2, x_3, ... x_N] be an M x N feature matrix of an utterance, where M is the number of features per frame, N is the number of frames in the utterance and x_i = [x_1, x_2, x_3,

..., $x_M]^T$ is the feature vector of the ith frame. Let **A** be an M x P transformation matrix. The transformed feature matrix **Y** can be determined from

$$Y = A^T X, \qquad (2.16)$$

where $Y = [y_1, y_2, y_3, ..., y_N]$ is a P x N matrix containing the features after transformation, and $y_i = [y_1, y_2, y_3, ..., y_P]^T$ is the transformed feature vector of the ith frame. If P is less than M, then the transformed feature vectors have lower dimensionality. In other words, dimensionality reduction is obtained when P is less than M.

The way the DCSCs are computed can be regarded as a feature transformation, where the **A** matrix is constructed from DCT basis vectors. Milner [12] has generalized this idea to work with many transformations.

2.2.1 Principal Components Analysis

PCA is a well-known technique for dimensionality reduction and feature decorrelation. Its objective is to find a lower-dimensional representation that accounts for the variance of the features. PCA reduces feature dimensionality by forming linear combinations of the features, as discussed above. To compute a PCA transformation matrix (**A** matrix as in equation 2.3), first a covariance matrix of the features must be estimated. The eigenvectors are then determined from the covariance matrix. Large eigenvalues correspond to features with large variance (more important features) and small eigenvalues correspond to features with small variance (less important features). If we sort the eigenvalues in descending order and sort the eigenvectors accordingly, we can use only the first P vectors ($P < M$, where M is the dimension of the original

feature vectors) for the transformation. Hence, dimensionality reduction results. The columns of the transformation matrix, **A,** are the eigenvectors.

Prior to applying PCA, there is some degree of correlation among features. Since a PCA transformation effectively removes highly correlated features (by combining or grouping them), the resulting features will be less correlated. This, in turn, results in performance improvements in speech recognition, especially with HMM based recognizers [12]. Sometimes the only advantage is that no degradation occurs even with a reduced dimensionality space.

2.2.2 Linear Discriminant Analysis

LDA can be regarded as an extension of PCA. Not only can dimensionality reduction and feature decorrelation be achieved with LDA, but also better separation between classes is achieved (or at least is an explicit goal). Unlike PCA, a covariance matrix of each class must also be estimated. The objective of the LDA is to maximize the ratio of between-class covariance to within-class covariance. In other words, LDA attempts to transform the features into a new feature space such that "within-class" tokens are well clustered but tokens from separate classes are well separated.

As mentioned previously, a PCA transformation is simply a matrix multiplication. Thus, there is no significant loss of computing speed in the recognition phase. However, there is more additional computational overhead in the training phase. For the case of LDA, the training overhead is even more, but the recognition phase again requires only one additional matrix multiply for each frame of features. Let

B be a between-class covariance matrix which is estimated across features of all classes (the same as in PCA). Let *W* be a within-class covariance matrix which is an average covariance obtained by averaging all of the covariance matrices, each of which is individually computed for a single class (typically a class is a phone[:] for the case of speech). A simple method to compute the LDA transformation matrix (*A*) is to determine the eigenvectors of $W^{-1}B$, and then form an *A* matrix such that its columns consist of those eigenvectors. The eigenvectors are sorted according to the magnitude of their corresponding eigenvalues. By putting eigenvectors associated with large eivenvalues in low-indexed columns, dimensionality reduction can be achieved simply by ignoring the last columns of the transformation matrix. This transformation rotates the feature space such that the features are ordered in terms of discriminability with respect to the defined classes.

The other method to compute *A* is an extension of the simple method discussed above. A whitening step is included in this step. The input feature matrix, *X*, is first rotated and scaled using

$$Z = EX, \qquad\qquad (2.17)$$

where

$$E = CD^{-1/2}. \qquad\qquad (2.18)$$

The columns of *C* are the eigenvectors of *W*, the within-class covariance matrix. *C* is a rotation which results in a diagonal within-class covariance matrix. *D* is a diagonal

_____

[:] A phone is a distinct unit of spoken sound while a phoneme is a distinct linguistic unit of a given language.

matrix with the ith diagonal element equal to the ith eigenvalue of W. Therefore, the within-class covariance matrix for Z is an identity matrix, i.e., this is a whitening transformation. This step is followed by another rotation

$$Y = FZ. \qquad (2.19)$$

The columns of F are the eigenvectors of the rotated B matrix, i.e., the eigenvectors of $D^{-1/2}E^{T}BED^{-1/2}$. The second transformation assigns the axes of the whitened space with maximum discriminability. Because of the whitening step, Euclidian distance should be a better measure of distance between classes than for the first LDA method explained above. This method has also been used in the IMELDA transform [52]-[54]. Note that A^{T} in (2.3) is equivalent to FE in this case.

LDA has been successfully used in many speech classification/recognition problems [10],[11],[55]-[59] Our previous work has shown that LDA improves recognition under noisy condition [10],[11]. However, we found that use of LDA is not beneficial when the original features are already quite good. We also found that dimensionality can be reduced without a significant performance degradation. This is a valuable technique if high recognition speed is of concern.

LDA was used for some of the experiments reported in Chapter 3 for isolated word recognition. The main issues for LDA are the details for computing the W and B matrices. In our implementation, the between class covariance B is estimated as the grand covariance matrix of all the training data (the same as for a principal components analysis). The estimate of W is more complex and is explained as follows.

The within-class covariance W is estimated by computing the average covariance of time-aligned frames of data which belongs to the same class. Time alignment is accomplished using Dynamic Time Warping (DTW), [60], to first determine a "target" for each word by successively aligning and averaging all tokens of that word in pairs until only one token remains. Covariance contributions are then computed as variations about the target, after another time alignment to that target. These two matrices are then used to create an LDA transformation which maximizes the ratio of between-to-within class covariance.

2.3 Statistical Acoustic Modeling

After speech has been encoded with a small number of parameters called features, a speech model has to be established in order to model characteristics and variations of speech and to further reduce the number of parameters. There are currently three popular acoustical modeling methods used in ASR: HMM, Neural Networks (NNs) [61] and Acoustic Segment Modeling (ASM) (or trajectory model). Experiments conducted in this dissertation were obtained using only HMM based recognizers. Thus, only a brief discussion of an HMM and its application to speech recognition will be presented.

2.3.1 Hidden Markov Model

A Hidden Markov Model is a statistical signal model that has been widely and successfully used in many ASR applications. One of the features that make HMMs popular is the existence of efficient and powerful training and recognition algorithms [62]. This section briefly discusses fundamentals of HMMs and the application of HMMs to ASR.

Interested readers are recommended to read further in [9] and [63].

An HMM is a finite state machine that is governed by two stochastic processes. An HMM consists of a collection of states connected with transitions. There are two kinds of transitions: self-transitions and transitions to other states. Both types of transitions are random. Transitions among states are controlled by a transitional probability matrix. Each time a state is visited, a symbol is emitted. A symbol or an observation is an outcome of the stochastic process within each state.

For the speech recognition application, a left-to-right HMM is usually used. A left-to-right model has transitions that never go back to preceding states. In other words, only self-transitions and transitions to the next states are permitted. This configuration is consistent with speech production. In most HMM based systems today, symbols are modeled as mixtures of a Gaussian probability density function (PDF) in each state. This type of model is called continuous-density HMM. Fig. 2.3 shows an example of a 3-state, left-to-right HMM.
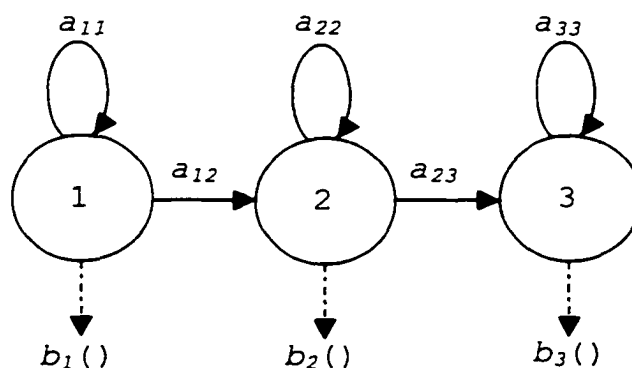
Fig. 2.3. A 3-state left-to-right HMM.

As illustrated in Fig. 2.3, each transition is associated with a probability a_{ij}. Practically, these probabilities are organized in matrix form, **A**, as depicted below

$$
\mathbf{A} \;=\; \begin{bmatrix} a_{11} & a_{12} & 0 \\ 0 & a_{22} & a_{23} \\ 0 & 0 & a_{33} \end{bmatrix},
$$

where a_{ij} is the probability of the transition from state i to state j.

Note from Fig. 2.3 that each state j has an associated observation probability distribution $b_j(\mathbf{o}_t)$. This probability distribution determines the probability of generating symbol $\mathbf{o}_t$ at time t.

For continuous density HMMs, which were used in this work, the observation probability is modeled by a mixture of Gaussian densities. Thus, the probability $b_j(\mathbf{o}_t)$ of the symbol $\mathbf{o}_t$ is given by

$$
b_j(\mathbf{o}_t) \;=\; \sum_{m=1}^{M} c_{jm} N(\mathbf{o}_t;\, \boldsymbol{\mu}_{jm};\, \boldsymbol{\Sigma}_{jm}), \tag{2.20}
$$

where M_j is the number of mixture components in state j, c_{jm} is the mixture weight of the mth component and $N()$ is a multivariate Gaussian with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. Either a full covariance or diagonal covariance matrix can be used.

Choice of the covariance matrix depends on the statistics of the observations (features). Usually, there is correlation among features. However, the degree of correlation varies depending on the type of front end analysis and in many

cases, uncorrelated features have been assumed to allow the use of a diagonal covariance matrix to model the data distributions. Although this is inaccurate, the computational overhead is greatly reduced. For better modeling of the speech data within a state, a full covariance matrix should be applied. However, for this case, much more storage and computing power are required.

An HMM can be used to model a unit of speech where the unit of speech could be a phone, a syllable or a word. Therefore, the number of HMMs needed has to be sufficient to model all the realizations of all the utterances in the vocabulary. For example, 39 HMMs are needed to model all realizations of the American English language.

Theoretically, each state of an HMM represents the statistics of an acoustically similar (by means of a spectral distance measurement) sub-unit of speech. For an HMM of N states, the unit of speech being modeled is divided into N segments. Each segment is characterized by its corresponding state.

In the recognition phase, we need to find among competing models the most likely word sequence W' that best represents a given acoustic evidence, or

$$W' = arg\ \max_{W}\ P(W \mid O),\quad\quad\quad (2.21)$$

where $P(W|O)$ denotes the probability that the word string was spoken given that the acoustic evidence O was observed. The term $P(W|O)$ is not directly determinable but it can be rearranged with Bayes's rule as follow

$$P(W \mid O) = \frac{P(O \mid W)P(W)}{P(O)},\quad\quad\quad (2.22)$$

where $P(O|W)$ is the probability that the observation O will be generated when the word sequence W is pronounced. $P(W)$ is the probability that the word string will be uttered. This can be determined from the statistics of training data. $P(O)$ is the probability that the observation sequence O will be observed. This term can be ignored if equi-probable symbols are assumed. Thus, the most likely word string can be determined from

$$W' = arg \max_{W} P(W)P(O \mid W).\qquad(2.23)$$

There are three main problems associated with the implementation of the HMM [9]. The first problem is to determine the $P(O|W)$ as described above. This is the evaluation problem that can be solved effectively with the "forward" algorithm. The second problem is to find an optimum state sequence given an observation sequence. The solution to this problem is useful for transcribing continuous speech. The Viterbi decoding algorithm is a standard method used to determine the optimum state sequence. The last problem is to compute the model parameters that maximize $P(O|W)$. This is known as the training problem. The Baum-Welch or forward-backward algorithm is an extremely efficient method that has been widely used for HMM training [62],[63].

For phonetic-based continuous speech recognition, several phone models are connected to form a word model and several word models are connected to form a sentence. In the recognition process, a hypothesized phone transcription is determined given an unknown utterance. Word sequences are determined from the phone transcription. Thus, a sentence is constructed.

A language model can be easily integrated into the HMM framework. That is probabilities of word pairs or word trigrams can be incorporated as part of the $P(W)$ term in (2.23). This is another reason why HMMs are so popular.

2.3.2 Hidden Markov Model Toolkit

The Hidden Markov Model Toolkit (HTK) [64] is a software package for building automatic speech recognition systems using HMMs. HTK provides several tools that support speech processing, HMM initialization, HMM training and recognition and data management. HTK tools are flexible in that users are allowed to configure functionality of each tool as desired. Also, many standard file formats are recognized, thus making it possible to interface to other software packages. Although the HTK is primarily designed for continuous speech recognition, it can also be adapted to work with isolated word recognition.

The HTK was chosen for this work because it is a powerful and a widely used software package. Using mostly default parameters provided with the HTK to build a speech recognizer, one can easily achieve a high performance system. The HTK allows researchers to substitute HTK's tools with their own ones. For example, in this work, we substituted HTK front-end analysis tool with our program but we did use the HTK front-end for comparison purposes. Use of the HTK toolkit permits us to concentrate more on feature extraction issues which are the main concern of this dissertation.

CHAPTER III

ISOLATED WORD RECOGNITION

3.1 Introduction

Continuous speech recognition systems have been developed for many applications, with low-cost PCs now having sufficient computing power for speech recognition software. However, high performance and robust isolated word recognition is still useful for many applications such as recognizing telephone numbers, spelled names and addresses, ZIP codes, and as a spelling mode for use with difficult words and out of vocabulary items in a continuous speech recognizer.

Because of the potential applications, as mentioned above, many isolated word recognizers are optimized for the digits or alphabet or both (alpha-digit). The alphabet recognition task is particularly difficult because there are many highly confusable letters in the alphabet set, for example the letters of the E-set (B, C, D, E, G, P, T, V, Z) and the (M, N) pair. Also, since language models cannot generally be used, the alphabet recognition task is a small, challenging and potentially useful problem for evaluating acoustic signal modeling and word recognition methods.

This chapter discusses feature extraction techniques for a high performance, HMM based alphabet recognition system using DCTC and DCSC parameters obtained as described in Chapter 2. A series of experiments were conducted to test the techniques. The experiments and results are reported.

3.2 ISOLET Database

The Oregon Graduate Institute (OGI) ISOLET database [66] was used for all experiments reported in this dissertation for the alphabet recognition task. This database contains 26 English alphabet letters pronounced in isolation by 75 male and 75 female speakers. Each speaker uttered the same letter twice. The database is divided into 5 equal groups: ISOLET1 - ISOLET5. Thus, each group comprises the speech utterances spoken by 15 males and 15 females. Speech was digitized under a controlled environment at a sampling rate of 16 KHz with 16-bit quality. The speech signal-to-noise ratio (SNR) reported by OGI is 31.5 dB with a standard deviation of 5.6 dB. In some experiments, Gaussian noise of various levels was added to the speech signal to test the robustness of our recognizer.

The ISOLET database was intended for evaluation of isolated word recognizers, and it has therefore been used in several studies [10],[11],[67],[68]. Thus, it is possible to directly compare results. All of the alphabet recognition experiments reported were performed in a speaker-independent fashion using all files from the database.

The original speech was endpoint detected with 50 ms of silence added to both ends. However, we found that the endpoints provided are not sufficiently accurate. Therefore, we applied another endpoint detection algorithm to refine the endpoints. After this additional endpoint detection, 30 ms of silence was included at each end. We found that the error rate was reduced by half with use of the new endpoint-refined data.

The endpoint detection algorithm was adapted from the one given in [69]. Endpoints were extended 30 ms in each

direction (i.e., backward in time for the onset and forward in time for the offset) to allow for some inaccuracies in the original detection, and also to include a small amount of silence at the beginning and end of each utterance. The primary difference between the method given in [69] and our implementation was that we used 20 ms frames for the first pass of endpoint detection and 10 ms frames for the second pass, as opposed to the longer frames used in the original method. This endpoint detection method resulted in approximately a 34% decrease in errors for an analysis condition corresponding to the best case reported here. However, use of the endpoint algorithm but without the 30 ms silence added at each end resulted in over a doubling of the error rate under the same analysis condition.

3.3   Recognizer Architecture

Our task is to accurately recognize all 26 letters of the English alphabet. We could employ a sub-word based topology for this task but it would be very complex to implement. For example, if we find all phones that build up all words in the vocabulary, then we could use the training data to estimate each phone model. In the recognition phase, a hypothesis phone sequence would be compared against all possible phone sequences in the vocabulary. The major problem is that in the training process time labeling of each letter is required to accurately initialize the phone models. This time labeling is not commonly provided with speech databases. If needed, it must be determined manually.

In this work, we chose a word-based approach in which each alphabet was modeled by one HMM. Continuous density HMMs with Gaussian PDFs were used throughout. More specifically, there were 26 continuous density HMM models,

each of which has 6 states with 3 Gaussian mixtures per state. For state transitions, only a self-transition and transition-to-the-next-state were allowed. A full covariance matrix was used for each Gaussian mixture.

In the training phase for each letter of the alphabet, every training utterance was segmented into equal lengths and then initial model parameters were estimated. Next, the Viterbi decoding algorithm was applied to determine an optimum state sequence for each training token. Every token was re-segmented based on its corresponding optimum state sequence. Model parameters were re-estimated repeatedly until the estimates were unchanged or the maximum number of iterations was reached. Note that no Baum-Welch iterations were performed, as they were not found to improve accuracy on test data.

Once all models are trained, the recognition step is carried out using the Viterbi algorithm to determine the most likely model that best matched each test utterance. We evaluate system performance by its percentage of accuracy in recognizing the test data. Accuracy can be determined as follows:

$$Accuracy = \frac{N_{tot} - N_{err}}{N_{tot}} \times 100\%, \qquad (3.1)$$

where N_{tot} is the total number of test tokens and N_{err} is number of tokens that are mis-recognized.

3.4 Experimental Verifications

Several speaker-independent recognition experiments were conducted with the alphabet set to evaluate our algorithms and also to determine the effects of variations in some of the parameter values. Except were noted

differently, all experiments used ISOLET-1 through ISOLET-4 for training and ISOLET-5 for testing.

3.4.1 Baseline Experiments

Recognition results were carried out using DCTC and MFCC coefficients. DCTC coefficients were computed using the method described in Chapter 2 with the following basic parameters: Frame size = 25 ms, Frame spacing = 10 ms, Kaiser window beta = 6, Bilinear warping factor = 0.45, 512-point FFT, 10 DCTC terms extracted from each frame.

The standard features--MFCC coefficients--were used as a control. Thirteen MFCC terms (12 + 1 energy term) were extracted from each 25-ms frame. The frame spacing was 10 ms, and 24 filter channels were used. The speech signal was pre-emphasized with a high pass filter whose transfer function was $H(z) = 1-0.95z^{-1}$. Note that these MFCC parameters were supplied by HTK's front end.

The first baseline experiment was based on single frame features ("static" features) for both DCTCs and MFCCs. Typical test results in terms of percent accuracy were 82.3% for DCTC terms and 82.6% for 13 MFCC terms. In general, for all conditions tested (various frame lengths, numbers of terms, etc.) results for these two static feature sets were quite similar.

The next baseline experiment was to use a fixed block length for both DCSC features (as explained in Chapter 2) and MFCC features. Typical results were 94.6% for the DCSC features and 95.8% for the MFCC features. These results were based on 50 terms for the DCSC features (10 DCTCs, each represented with 5 terms, and computed over a block length of 115 ms), and 39 MFCC terms, with delta and delta-delta

parameters computed using 5 frames (65 ms). Although several other parameter settings were tried (block length, total number of terms, etc.), the conditions mentioned were the best (by a small amount) of the ones tried for both the DCSC terms and the MFCC terms.

3.4.2 Variable Block Length Experiments

The technique explored was to vary the block size depending on the location of the block in the speech signal. Frame level features (DCTCs) were computed as described in Section 3.4.1, except a frame spacing of 5 ms was chosen. Then, at the beginning of an analyzed token, a block size of 6 frames (i.e., 45 ms total duration, including end effects of the analysis frames) was used. As the analysis window moved forward, the block size increased until a maximum of 40 frames (215 ms) was reached. The block size was then fixed at this maximum until the end region of the utterance was reached. At this point the block length was again gradually reduced until, for the very final block, it again reached 6 frames. Time "warping" was also applied to each block, with the amount of warping controlled by the beta value for a Kaiser window, and with this beta value linearly interpolated from 0, for the shortest length windows, up to a maximum value of 5 (approximately a Hanning window), for the longest length blocks.[1] Thus, the features gave better time resolution for the onset and offset portions of each word and less time resolution in the central portions of each word. The block features were re-computed every 2 frames (10 ms). No manual segmentation or phonetic labeling was required or used. The primary modification, relative to

---

[1] The values for minimum and maximum block length, and degree of time warping, were varied and tested. The values mentioned were the ones used for our experimental results.

[11], is that the block length was varied at both ends of each analyzed utterance, rather than only for the beginning section.   See Fig. 3.1 for an illustration of this variable block length method.

Sequence of frame based features

(a)

(b)

(c)
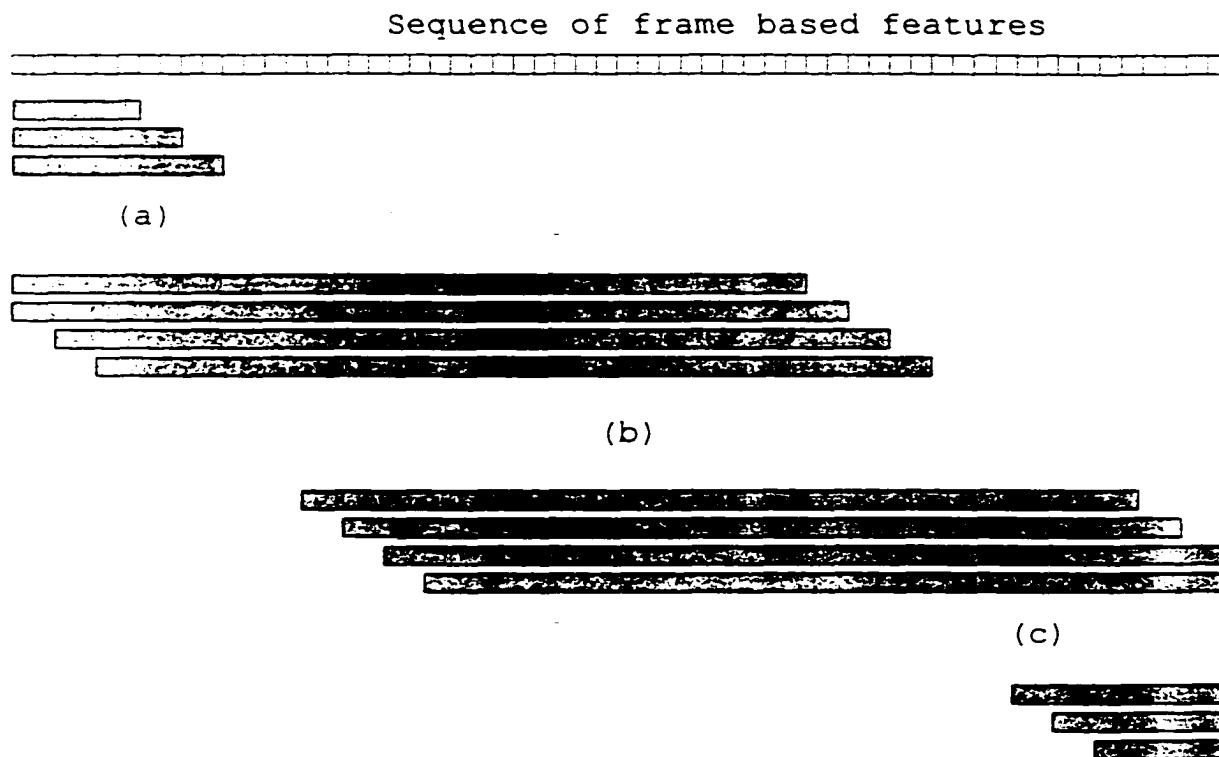
Fig. 3.1. An illustration of variable block length method.

Note from Fig. 3.1 that each grayed rectangle represents a block from which spectral/temporal features are computed. In region (a) the block length is increasing (6 to 40 frames), in region (b), the block length is fixed at the maximum (40 frames), and in region (c), the block length is decreasing (from 40 to 6 frames.)

As mentioned previously, the endpoint detection program included 30 ms of silence at both ends of each utterance in the database. However, as noted in the next section, the best recognition accuracy on test data (ISOLET-5) of 97.9%

was achieved using the 30 ms of initial silence, and 25 ms of silence at the final endpoint. Therefore, unless otherwise noted, this configuration (30 ms initial silence, 25 ms final silence) was used in this experiment and all the following experiments.

The variable block length method was evaluated, using 10 DCTCs, each represented with 5 terms in the DCSC expansion (50 features), with the block length varied from 6 frames (45 ms) at the beginning and end of each utterance up to 40 frames (215 ms) at the center of each utterance. As a verification, this test was repeated in round robin fashion, using ISOLET-1 through ISOLET-5 as test data, one at a time (and the remaining four sets for training in each case). Results of the round robin test were then averaged. Similar testing was done with the 39 MFCC parameters mentioned above. Results for the four cases are given in Table 3.1. Note that the DCSC and results obtained from the robin tests are very close to those obtained with ISOTLET-5 tests, thus implying that the parameters are not overly "tuned" to ISOLET-5.

	Recognition accuracy (%)	
Feature type	ISOLET-5 for test	Round robin test
DCSC	97.9	97.7
MFCC	95.8	95.9

Table 3.1. Alphabet test recognition rates for "best" DCSC and MCFF parameters.

3.4.3 Endpoint Examination Experiment

Since the block length in the variable block length method depends on the position of the block relative to the determined starting and final endpoints of each word, we hypothesized that performance might depend heavily on the accuracy of the endpoint algorithm. This notion was reinforced when we repeated the test mentioned above with the variable block length using the data as distributed without the benefit of the refined endpoint algorithm and found that accuracy dropped from 97.9% to 96.8% (about a 52% increase in errors). To test our hypothesis more systematically, we independently varied the starting and final endpoints over a range of ±30 ms from the automatically computed location, with the other endpoint fixed (30 ms for starting endpoint and 25 ms for final endpoint.) A negative amount of silence means that a portion of speech is discarded. Except for these variations in endpoints, all other signal processing was identical to that used above in the variable block length experiment. The recognition results for these tests are depicted in Fig. 3.2.

The results clearly illustrate that, at least for this data, the beginning endpoint is much more critical than the final endpoint. It also appears that adding even more than 30 ms of silence before the onset of each word would have been beneficial; however, this could not be done since the original tokens did not include sufficient silence. The absolute best result of 97.9% was obtained with 30 ms of initial silence and 25 ms of ending silence. These values of silence were thus used in the other experiments reported in this chapter.
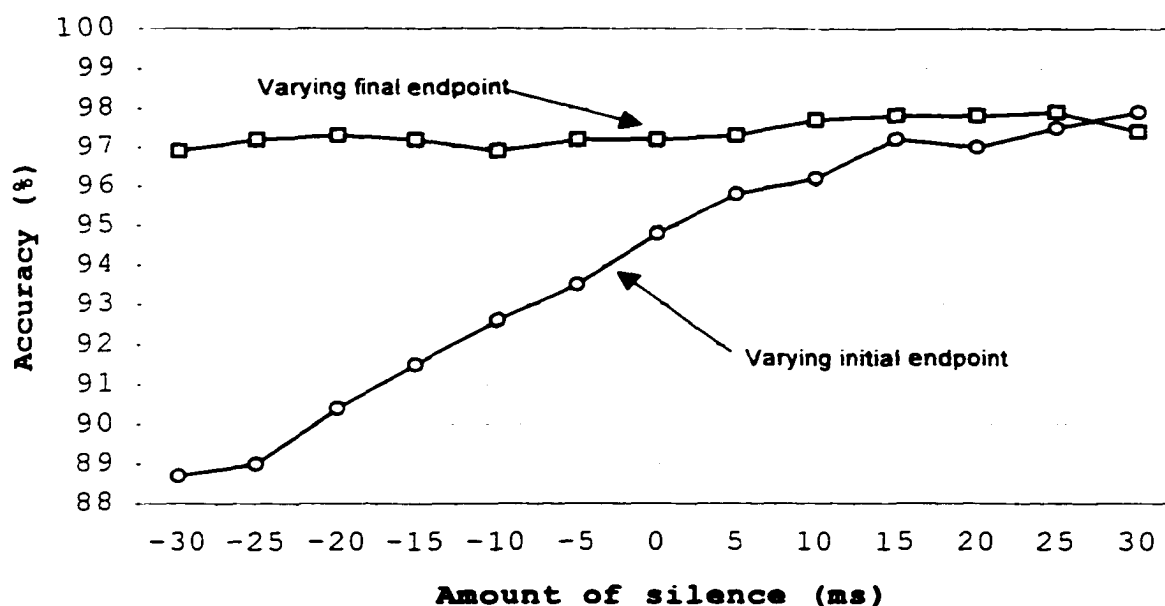
Fig. 3.2. Effects of variations in endpoint locations on recognition accuracy.

3.4.4 Signal-to-Noise Ratio Experiment with and without LDA with and without Band Limiting

To test the robustness of the signal features more thoroughly, experiments were done with additive Gaussian white noise over a range of SNRs from -10dB to +30 dB. These tests were done both with the 50 DCSC features as used above and the 39 MFCC features also previously mentioned. All conditions were identical to those reported for Section 3.4.2, except for the additive noise. In addition, tests were made using LDA for each noise level, with the hypothesis that the effects of LDA might depend on noise level. Based on pilot experiments, we extracted 30 LDA terms for the DCSC case and 25 LDA terms for the MFCC case

(about 60% of the size of the original feature vector in each case). Results are given in Fig. 3.3.
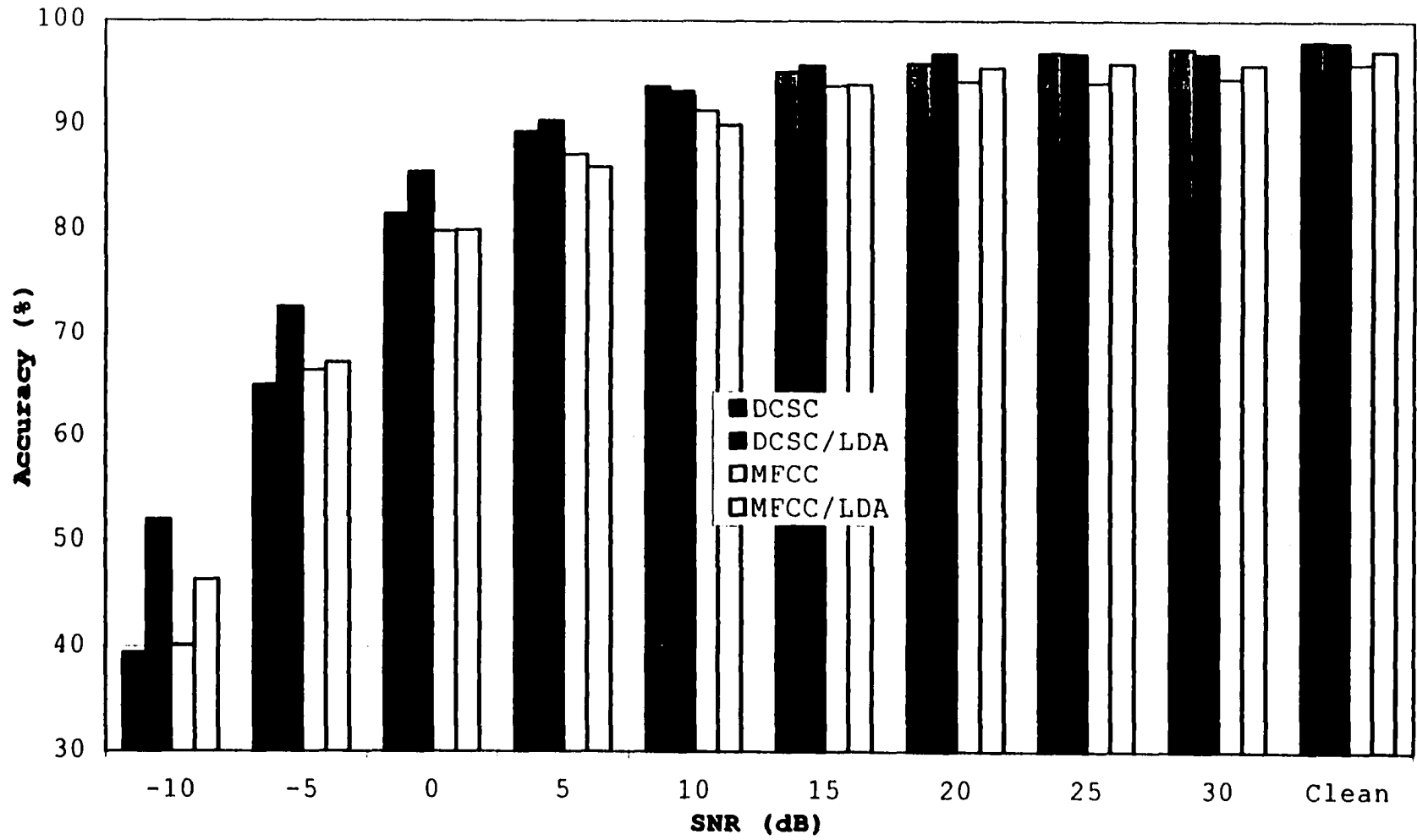
Fig. 3.3. Variations in recognition accuracy versus SNR, with and without LDA.

Note that for all cases except SNRs of -10 and -5 dB, the 50 DCSCs perform better than the 39 MFCC parameters. For both features sets, LDA is beneficial with SNRs of 0 dB or worse. For higher SNR values, the effect of LDA is quite small for both feature sets, sometimes slightly degrading performance and sometimes slightly improving performance. For "clean" speech, SNRs of 25 dB or higher, LDA degrades the DCSC features--but only by a small amount--whereas LDA improves the MFCC features nearly to the level of DCSC features.

As one final "robustness" test, tests were made with band-limited speech (300 Hz to 3,200 Hz) to simulate telephone bandwidth for each noise level. Results are shown in Fig. 3.4.

In every case without the use of LDA, the DCSCs resulted in higher accuracy than the MFCCs. Except at an SNR of 30 dB for the DCSCs, the performance of DCSCs and MFCCs were improved somewhat with LDA. In general the performance increase due to LDA was higher at lower values of SNR.

The effects of LDA applied to features extracted from band-limited speech are quite different depending on DCSC versus MFCC parameters. For almost all noise levels, the LDA results in very little change for the DCSC case, whereas the LDA results in larger improvements with the MFCC parameters. The performance obtained with LDA transformed MFCCs very closely matches that obtained with the DCSCs, with or without LDA.
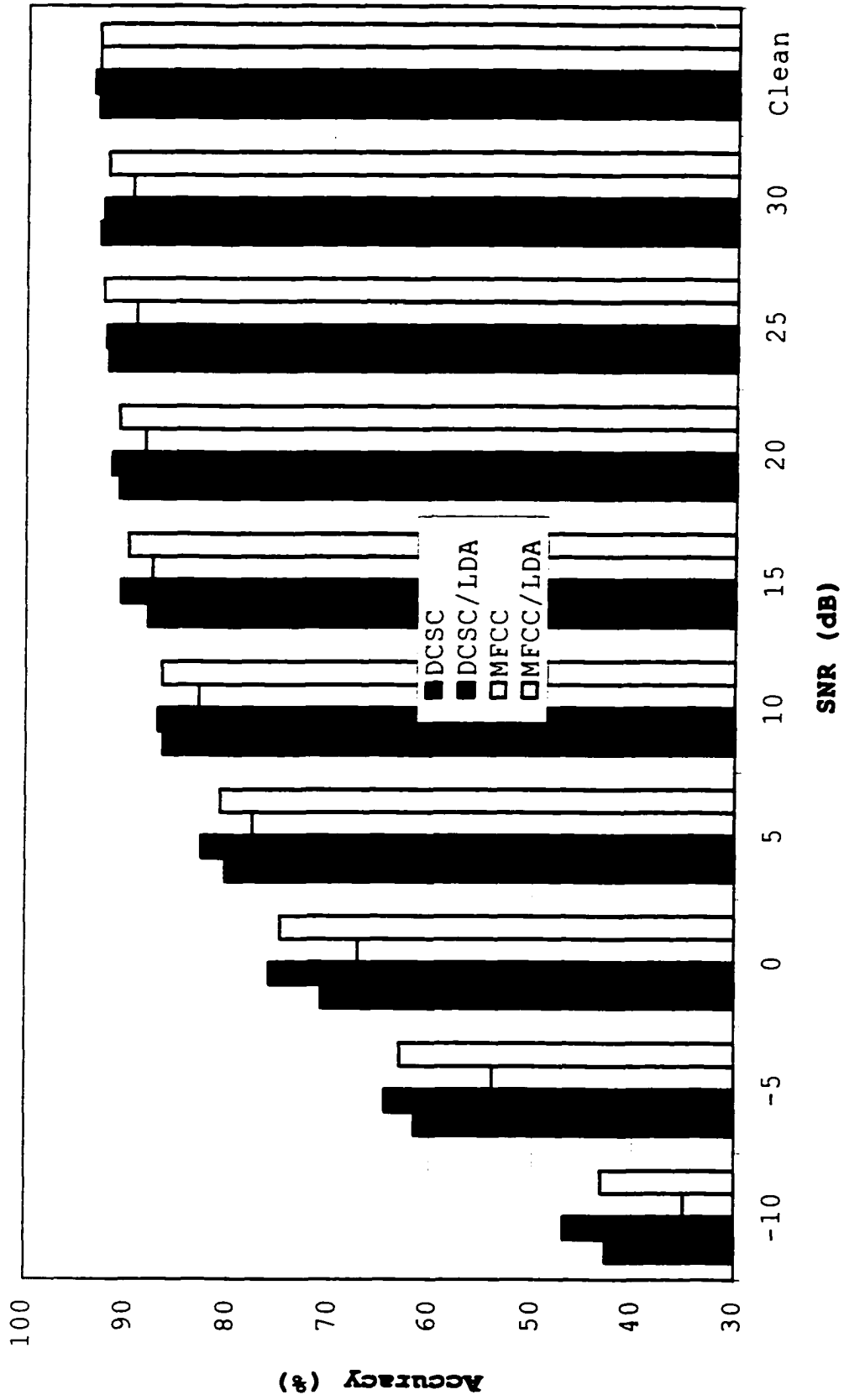
Fig. 3.4. Variations in recognition accuracy versus SNR, with and without LDA with band limited data.

Note that for the full bandwidth speech, the rates based on DCSC signal modeling are typically about 1.5% higher than the rates based on MFCC signal modeling. For the telephone bandwidth case, the DCSC method averages about 3.4% higher than for the MFCC method. Additionally, the typical degradation between full bandwidth and telephone bandwidth is also less for the DCSC case versus the MFCC case (average degradation of 5.3% versus 7.3%).

3.4.5 Error Analysis

Fig. 3.5 shows the confusion matrix of the best case (97.9% accuracy on clean and full bandwidth test data with 50 DCSCs). The confusion matrix indicates that there are 33 errors and that the (M,N) pair is the most confusable set and contributes approximately 25% of all the errors.

The number of errors was small enough to be inspected visually as plots of time waveforms and spectrograms on a computer, and by listening. Of these 33 errors, 18 were due to confusions within the E set, eight were due to confusions between letters M and N and only seven confusions were for the remaining letters. After listening to all of the error tokens, we concluded there are four situations for which tokens were misrecognized. Six tokens appeared to have a severe endpoint detection problem. There were nine tokens pronounced in an unusual way, mostly by a single speaker. Also there were four tokens misrecognized because they are so similar to other letters, that, even with careful listening, they were difficult to recognize. Finally, fourteen tokens sounded intelligible and distinct with no obvious reasons for the errors in machine performance, although some of these may have had endpoint detection inaccuracies.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	59							1																		
B		59																			1					
C			59																							1
D				58		1												1								
E	1				59																					
F						60																				
G							56	2										2								
H								60																		
I									60																	
J										60																
K											60															
L												60														
M													58	2												
N													6	53							1					
O													1		59											
P			1													59										
Q																	60									
R																		60								
S						1													59							
T						2			1	1										56						
U																					59	1				
V																						59		1		
W																							60			
X																								60		
Y									1																59	
Z		3																		1						56

Fig. 3.5. Confusion matrix corresponding to the best test conditions.

To test the hypothesis that several of the errors were due to endpoint problems, we manually corrected the endpoints for all the error tokens. A recognition test performed on this corrected test data results in 19 fewer errors, or 99.1% correct. Although this result does not really "count," since it was not done fully automatically, it does help to illustrate the importance of good endpoint detection.

3.5   Comparison of Results

The highest recognition accuracy (97.9%) obtained with the variable block length method is favorable to that obtained with other systems under the same test data. For example, in [68], a recognition accuracy of 97.37% resulted with a two-tier, phoneme-based HMM recognizer. Another system used a Neural Network-based system with 617 inputs [67] and was able to achieve 96.0% accuracy.

3.6   Chapter Conclusions

The variable block length signal analysis method with 50 DCSC terms and a "standard" HMM recognizer results in 97.9% accuracy for the alphabet set. This represents a 19% reduction in errors when compared with the previously reported best result in the literature for the same database. More importantly, the method introduced in this study is straightforward to implement and extend to other speech recognition tasks. Also, the signal modeling used is generally more robust to noise and band limiting than MFCC terms augmented by delta and delta-delta terms.

The best recognition rate (97.9%) obtained with our methods corresponds to 19% fewer errors than the best result reported in the literature for this task from other labs [68], and 47% fewer errors than the next best reported result [67]. More importantly, the method introduced here is easier to implement and duplicate than these previous state-of-the-art systems. Some of work on this task has been reported in Karnjanadecha and Zahorian [10],[11].

The primary contribution of this chapter is the demonstration that the calculation of signal trajectories over intervals with lengths dependent on position within the

utterance can improve performance. The advantage of this method is that it is able to capture rapid transients at the beginning and ending of each word, while simultaneously using longer, more noise resistant averaging intervals in the center of each word. We also showed that the benefit of using LDA for automatic speech recognition depends heavily on the features and noise level. The improvements in ASR performance due to LDA tend to be the largest under noisy conditions, and with signal parameters which are sub optimal in terms of recognizer performance. The only real benefit of LDA for "good" parameters, such as the DCSC used in this study, is that a reduction in dimensionality by about a factor of two is possible with very little change in performance.

An extension of this method, in order to apply it to continuous speech recognition, would be to base the block length on a measure of spectral derivative rather than simply position with respect to endpoints of an utterance.

An extension of this method, in order to apply it to continuous speech recognition, would be to base the block length on an optimization criterion, such as minimum reconstruction error, rather than simply position with respect to endpoints of an utterance. A spectral transition measurement (e.g., spectral derivative) of frame level observations could also be used to identify the temporal characteristics in a particular area of the utterance. Thus, an appropriate block length can be applied.

The results from this chapter have confirmed the importance of temporal information for speech recognition. Although different characteristics over time are captured using different block sizes, a similar effect could be achieved with different block spacing. For example, the

advancing rate of the processing block could be determined form the amplitudes of the spectral derivative in the interval around the block's center.

CHAPTER IV
CONTINUOUS SPEECH RECOGNITION: TASK AND DATABASE

4.1   Introduction

The previous chapter has shown an application of our spectral/temporal signal processing methods to isolated word recognition. In this chapter, these techniques are extended to the more difficult problem of continuous speech recognition. The basic technique presently in the last chapter is modified to suit the task. The TIMIT database [70] (and also the telephone version called NTIMIT [71]) was used for experimental verification, focusing on phonetic recognition. The underlying assumption for the experimental work is that a high recognition accuracy at the phonetic level will also lead to high word accuracy. As in the previous chapter, the HTK toolkit was used to implement the recognizer.

4.2   TIMIT Database

TIMIT [70] is a well-known speech corpus that has been used by many researchers to evaluate their phonetic classification/recognition techniques. The database contains 6300 sentences of speech recorded from 630 male and female speakers from 8 major dialect regions of the United States. Each speaker uttered 10 sentences in a citation form in a quiet environment. TIMIT is a phonetically rich database that is suitable for phonetic classification/recognition tasks.

Speech was sampled at 16,000 samples per second with 16-bit resolution and stored using a NIST Sphere format. Each sentence on the database CD-ROM has 4 associated files:

1) .wav file—speech waveform file in NIST format,

2) .txt file—associated orthographic transcription of the words the person said,

3) .wrd file—time-aligned word transcription,

4) .phn file—time-aligned phonetic transcription.

Only .wav and .phn files were used in this work. Labeling information given in the .phn files were used to estimate initial HMM models for each phone and to compute recognition accuracy.

4.3 NTIMIT Database

The NTIMIT database [71] was obtained by playing all TIMIT waveform files through telephone channels and then re-digitizing them. All transcriptions and filenames are identical to TIMIT's. However, the speech waveform in NTIMIT is band limited by the telephone channel and also has channel distortion and noise due to the overall process. Thus, NTIMIT is useful to examine ASR robustness with respect to noise and signal degradation. NTIMIT was used for this purpose in this work. Note that NTIMIT was created in a controlled systematic way as a research project and is available through the Linguistic Data Consortium (LDC).

4.4 Phonetic Model

There are a total of 61 phone categories defined in TIMIT. Since many of these phones are not phonemically significant in English, generally not all phones are used. Some phones can be removed without loss of information for

higher level recognition such as words. Some phones can be collapsed to a group of similar phones. A 39-phone set designated by Lee [14] has become the de facto standard set for ASR work. For approximately the past twenty years, many ASR researchers have used this phone set to evaluate their systems. In this arrangement, all glottal stops ('q') are removed and 15 allophones are folded into the corresponding phone resulting in the list in Table 4.1.

Phone	Folded	Phone	Folded
iy		en	
ih		ng	eng
eh		ch	
ae		jh	
ix		dh	
ax		b	
ah		d	
uw	ux	dx	
uh		g	
ao		p	
aa		t	
ey		k	
ay		z	
oy		zh	
aw		v	
ow		f	
l		th	
el		s	
r		sh	
y		hh	hv
w		cl	pcl,tcl,kcl,qcl
er	axr	vcl	bcl,dcl,gcl
m	em	epi	
n	nx	si	h#,pau

Table 4.1.   List of the TIMIT phones.

The folding results in 48 different phone categories. As described by Lee [14],[15] that there are 7 phone groups where confusions are not counted as errors, i.e. {ah,ax}, {aa,ao}, {ih,ix}, {en,n}, {el,l}, {sh,zh}, {si,cl,vcl,epi}.

Hence, only 39 effective phones remain in different categories.

In this dissertation, one HMM model was created for each phone category. As a result, there were 48 HMM models. These 48 phone models were trained and recognized but the final evaluation was performed using only 39 phones as mentioned before. Since confusions primarily occur within groups, we found that evaluating the system with 39 phones yields higher accuracy than with 48 phones. By measuring our system performance this way, we can directly compare our results with current systems developed by other researchers in the field. Although, some systems try to model all 61 TIMIT phones separately [29], usually they eventually collapse phones and evaluate them with 39 phones.

4.5 Use of HTK Toolkit

Although it can be used for isolated word recognition, the HTK toolkit was mainly designed for continuous speech recognition. Since the TIMIT database provides a phonetic transcription with time labeling, the HTK can employ this information to increase performance of the system.

Initially, 48 prototypes of continuous density HMMs (one model for each phone) with a Gaussian PDF were created. The next step was to estimate the initial parameters of each model. Basically, these parameters (means, covariances, mixture weights, and transitional probabilities) can be estimated from global statistics of all the training data regardless of category. This could result in a fair estimate of initial models for a task in which the phonetic labeling is absent. However, the HTK can make use of time labels provided with the TIMIT database to estimate initial model parameters. For most experiments reported in this

dissertation, the time labels were used for training, since performance was typically substantially higher with this method (see experiments Chapter). Note that time labels were not used in the recognition phase.

To initialize the model of a phone, the HTK scans all occurrences of the phones from training data and then extracts the features of the phone using the time labeling provided. After all occurrences of the phone are collected, Viterbi training is used to estimate the model parameters. This process is repeated for all phones.

The next phase of the initialization process is to apply the Baum-Welch formula (forward-backward algorithm) to train each phone in an isolated word mode. This is similar to the previous step where time labeling is used.

The next step of training is to apply the Baum-Welch algorithm to train the initial models in the embedded mode. The sequence of feature vectors and the phonetic transcription (ignoring the time labels) of each sentence are read. Then a lattice of HMM models is constructed given the phone transcriptions of the sentence. The Baum-Welch algorithm is then applied and the statistics obtained are accumulated. After all sentences have been processed, the accumulated statistics are used to update the models. The whole process is repeated for several iterations.

For the HTK, the Viterbi algorithm is used in the test phase to generate the hypothesis phone sequence for each test sentence. Percent accuracy can then be determined.

In this dissertation, a 3-state continuous density HMM was used to model each phone. Only self-transitions and transitions to the next state were allowed in the model. A

Gaussian PDF was used throughout. The covariance matrix used was diagonal, unless otherwise stated. Specific details of the HMM configuration used are explained in the experimental chapter.

See Appendix A for more information on the use of HTK toolkit for this task.

4.6 Training and Test Data

The training data used in this work were the 3696 SX and SI sentences. There are 462 (326 males, 136 females) speakers in this set. This training set is recommended by NIST as training data.

192 SX and SI sentences were used as test data. This set is recommended by NIST and is known as the NIST core test set. It contains sentences pronounced by 16 male and 8 female speakers. Many researchers using the TIMIT database did use this recommended test set, thus making it easier to compare results.

The above training and test set are identical for the TIMIT and NTIMIT databases, since the only different between both databases is the quality of the speech signal.

CHAPTER V
CONTINUOUS SPEECH RECOGNITION: SIGNAL MODELING METHODS

5.1 Introduction

This chapter discusses all proposed segment-based signal modeling methods that are applicable for the phonetic recognition task described in the previous chapter. By "signal modeling" we refer to the calculation of spectral/temporal features using DCTC based spectral features as a starting point. The basic idea is to compute higher order terms of each feature by observing the change of the feature values within a time window. Note that the time is measured in discrete units indexed by frame number. For example, if a block of K adjacent frames is formed, then changes of each feature in these frames over time can be observed. The value of K (an integer) represents the size of the time window. Other block can be formed in the same manner. Usually, blocks are overlapped and advanced forward in time. In this work, block spacing (in number of frames), L, is used to represent the block advancing rate. In this study, K and/or L were chosen to be fixed or variable.

The first method presented in this chapter is a control method where K and L are fixed--as is used in nearly all ASR systems. The focus of this dissertation is to examine methods for varying K and L: K is varied, and L is fixed; K varies and L is fixed. The goal is to investigate methods for non-uniform time sampling of speech features. We also examined another method in which spectral/temporal features

of short and long blocks are first computed and then combined using PCA.

5.2 Control Method

This method uses a fixed block length (K) and fixed block spacing (L) scheme which is similar to the standard method used by most researchers. It was noted previously that the time labels, provided with the TIMIT database, were used to estimate initial parameters of each phone model. Furthermore, our pilot experiments have shown that the DCTC based DCSC parameters yielded excellent results when a large block size was used. If each block is formed with respect to a center frame, then a problem will arise for the first few frames and for the last few frames of each analysis token because of "end" effects. We can avoid this problem by shifting the center of the block forward and ending the processing before reaching the last frame. The amount of shifting must equal half of the block length. Consequently, the resulting number of blocks will be less than the number of frames when the block length is greater than one frame. This causes feature vectors to be out of synchronization with time labeling, which is done in terms of frame indices. Thus, this "artifact" results in performance degradation.

The synchronization problem can be solved in three ways: 1) to ignore time labeling for training, 2) to add some extra blocks after the block processing and 3) to add some extra frames before the block processing. The first option—although technically an option for HMM methods—in practice causes substantial performance degradations. The addition of extra blocks in the second method is achieved by replicating the feature vector for the first and the last block a number of times equal to half of the block length.

If the block length is odd, this method will yield a total number of blocks equal to the number of frames in the utterance. This method is viable to restore synchronization, but the beginning and ending features vectors for each utterance do not reflect the temporal characteristics of the beginning and end of each utterance. The third method is similar to the second method in a way that it adds some extra feature vectors at both ends of the sequence except that this method functions at the frame level. If the block length is known to be K, then $K/2$ occurrences of the features of the first frame are added. Similarly at the final frame, $K/2$ feature vectors identical to the last feature vector are added. Fig. 5.1 illustrates this concept for the case of $K=5$ and block spacing $L=1$.
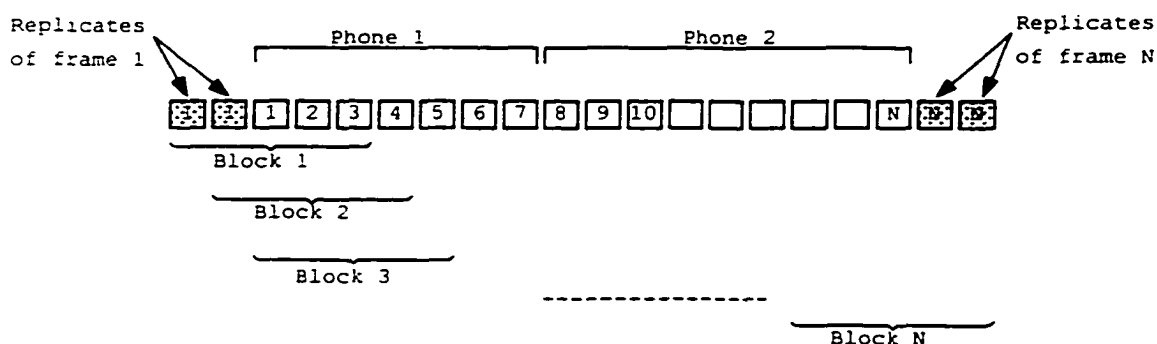
Fig. 5.1. Illustration of the fixed block length and fixed block spacing method.

Fig. 5.1 shows an example of the fixed block length and fixed block spacing method with block length, $K = 5$ frames. The feature vector of each frame is indicated by a rectangle with its corresponding frame index marked inside. The figure also indicates the boundaries of phone 1 and phone 2. The original data has N frames, but, with our method, two extra

frames are added at both ends prior to the block processing. The first block consists of the first two extra frames and frames 1 to 3. The second block consists of the second extra frame and frame 1 to 4. Extra frames are not in use from block 3 and beyond. The same logic applies to the processing at the end of the utterance.

As seen in Fig. 5.1, phone 1 starts at frame 1, and ends at frame 7. After block processing with this method, the boundaries of phone 1 will begin at block 1 and end at block 7. Hence, the out-of-synchronization problem is solved. However, block 7 and 8 contain the initial portion of phone 2 which is potentially a problem. This effect is negligible since: 1) phone boundaries may be off by a small amount and 2) phonetic context is modeled with partial inclusion of the information of adjacent phones. This effect also occurs at the first 2 blocks of phones 2 where part of phone 1 is included.

For the same scenario, if no extra frames were added, the first block would comprise frame 1 to frame 5, where the center of the block would be frame 3. Since the duration of phone 1 is 7 frames, after block processing the boundaries of phone 1 would be from the block centered at frame 3 and the block centered at frame 10. Obviously, there would be two problems with this scheme. First the last block of phone 1 would be entirely in phone 2. This is significant when the block size is large. The second problem is that there would be insufficient data (too few blocks) for the last phone because the number of blocks is always less than the number of frames when the block length is greater than 1. Extra blocks can be appended at both ends after the block processing is completed to regain synchronization. However, our pilot experiments indicated that the method based on

adding extra frames performed better, so this method was generally used.

Once the boundaries of each block are known, we can apply the DCT to the feature vectors in the block using a pre-computed set of DCT basis vectors. The length of each basis vector must match the length of the block. Thus, in this case, where the block size is fixed, only one set of basis vectors are required. There are just two free parameters to tune in this control method--block size and block spacing.

5.3 Variable Block Length Method

This method is similar to the control method in terms of block encoding and maintaining synchronization of the time labels of each phone and the block indices. The major difference is that the block length is not fixed. Block length depends on the temporal characteristics of the features around the center of the block.

Let I and J be two integers representing the minimum and maximum number of frames to extend about the block's center, respectively. Therefore, the minimum block length is $2I+1$ and maximum block length is $2J+1$. Suppose the block centered at frame M is to be encoded. The following pseudocode was used to determine the block length:

```
Loop n = I to J

    BLOCK = Resampling (block of frame M-n to frame M+n)

    RECONSTRUCTEDBLOCK = EncodeDecode (BLOCK)

    MSE = ComputeMSE(BLOCK, RECONSTRUCTEDBLOCK)

    If MSE > THRESHOLD
```

Half_width = *Maximum* (*I, n-1*)

      Exit Loop

    End IF

  End Loop

Half_width = *Minimum* (*J, n*)

Resampling() is a function to linearly interpolate a block of $2n+1$ frames to a block of fixed size. This step normalizes the size of each block to ensure that the features extracted from each block belong to the same feature space.

EncodeDecode() is a function that takes the resampled block and encodes it using a DCT. The encoded output, a matrix of DCSC coefficients, is obtained. Then block decoding begins by applying the Inverse Discrete Cosine Transform (IDCT) to the coefficients. The reconstructed block, **RECONSTRUCTEDBLOCK**, obtained with this function is then compared to against the original block, **BLOCK**, using the *ComputeMSE*() function from which a normalized mean-squared error (MSE) is computed. This error is returned by this function to **MSE**. The following shows how MSE is computed:

$$MSE = \sqrt{\frac{1}{MN} \sum_{m=1}^{M} \sum_{n=1}^{N} (X_{mn} - X'_{mn})^2} ,$$
(5.1)

where X_{mn} is the mth feature of the nth frame of the original block X and X'_{mn} is the mth feature of the nth frame of the reconstructed block X'. Note that the error is normalized with respect to the total number of features (MN).

The MSE of each selected block size is compared against a predefined threshold value, **THRESHOLD**. If MSE is greater

than the threshold, the implication is that the reconstructed signal is considerably different from the original signal. Thus, the block length is too long for an accurate representation. In the code, the loop is broken and the block size is reduced by two for signal modeling. However, if MSE is less than the threshold, the block size is increased by two (expanded at both sides) and the process is repeated.

Minimum() and *Maximum*() are functions to limit minimum and maximum length of the block.

Fig. 5.2 illustrates block length candidates when the block is centered at frame 6 (M=6, I=1, J=5).
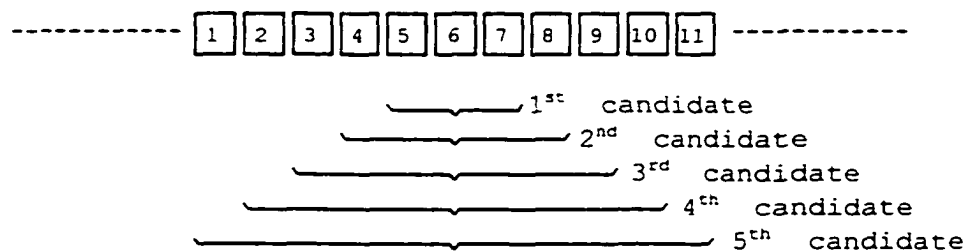
Fig. 5.2. Example of the variable block length method when the center of block is at frame 6, and minimum and maximum block length is 1 and 5 frames, respectively. There are possibly 5 block length candidates.

This method of determining block length is based on an assumption that a short block gives a lower reconstruction error than a longer block. This is due to the effect of data reduction where a low degree of data reduction yields low distortion. Also, decoding with a short block insures that a fast changing spectrum is well captured. With this algorithm, a short block will be used in the area where the

spectrum changes rapidly and a longer block will be used in the area where the spectrum changes slowly.

In order to speedup the computations for the processing, DCT basis vector sets are pre-computed for each set of basis vectors that could be used. Knowing I and J, one can determine the number of sets of basis vectors needed and length of each set.

Once a block has been processed, the center of the block moves forward and the whole process repeats. Note that the number of frames that the center of the block has to be moved is a free parameter.

Altogether, there are four free parameters associated with this method: minimum and maximum block length (I and J), MSE threshold, and block spacing. Again, for synchronization purposes, J replicates of the first frame are added at the beginning and J replicates of the last frames are added at the end of the utterance. J is equal to half of the maximum block length.

Fig. 5.3 illustrates the algorithm using real speech data. Horizontal dark bars, overlaid on the spectrogram, represent the optimum block length at each position. The time is given in terms of frame indices. The figure clearly shows that block size is small in transitional areas and large in steady areas.
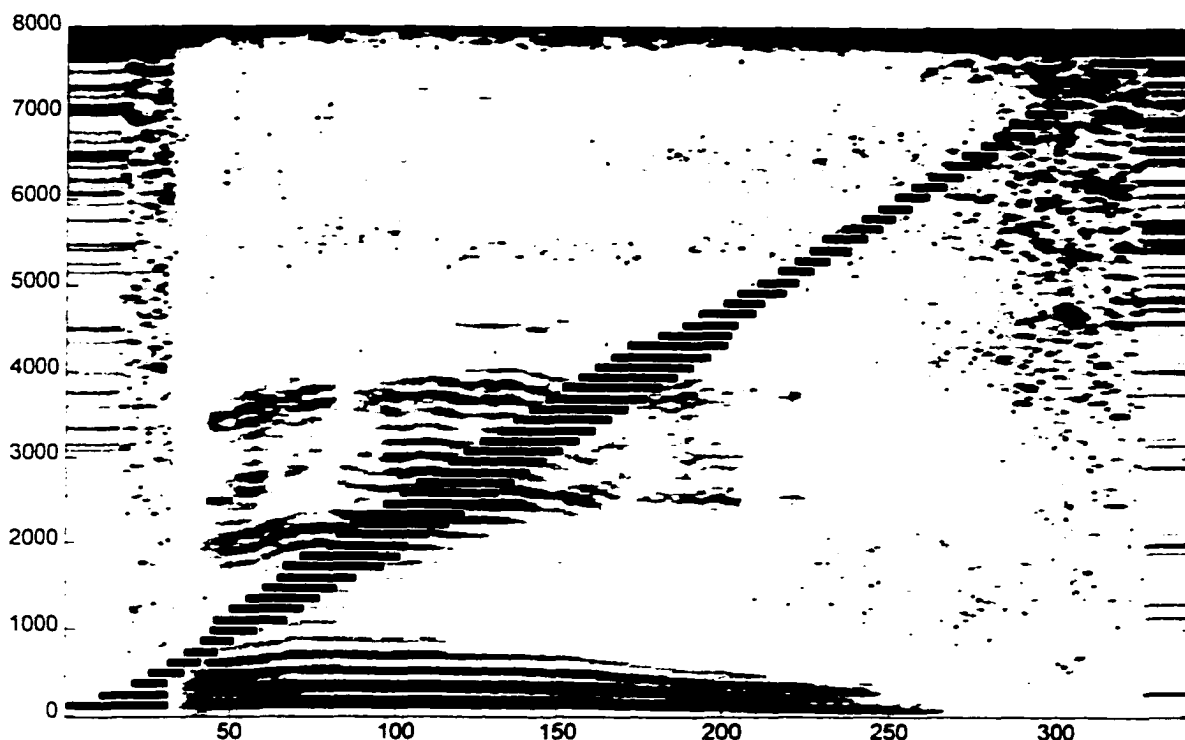
Fig. 5.3. Variable block length technique applied to an utterance of the letter "b," Threshold = 2, Block length = 11-31 frames, block spacing = 5 frames, frame size = 25 ms, frame spacing = 2 ms.

5.4 Variable Block Spacing Method

This method uses a variable block spacing scheme to capture characteristics of a fast changing spectrum. It relies on the fact that a finer sampling is needed for a higher bandwidth signal. In this case we sample the signal closely when its spectrum changes rapidly. Basically, frame-based features have to be extracted with a small frame spacing (fast frame rate) so that most information will be preserved. Conversely, in slowly changing areas of the

spectrum, redundancy can be reduced, by using a large block spacing.

Note that for this method, only the block spacing is variable---block length is fixed for all blocks. From a sequence of frame-based features, we compute a spectral transition measure (spectral derivative) using the method described in [72]

$$SPCDRV\ (t)\ =\ \frac{\sum_{i=1}^{p} D_i^2(t)}{p}. \tag{5.2}$$

In this equation, $D_i(t)$ is the delta coefficient of the ith feature at time t computed using Equation (2.2) and p is the number of coefficients used to compute the spectrum transition measure. Hence, the spectral derivative at time t is equal to the mean of the square of all p coefficients. Notice that the summation index starts at 1, which means the 0th coefficient is not used, since, for cepstral coefficients, it represents the overall level of the spectrum of the frame.

As indicated in Equation (2.2), the window size used to compute delta terms is controlled by the variable W. In this case, use of a large window results in a smooth spectral derivative while use of small window results in a noisy one.

Fig. 5.4 and Fig. 5.5 show the spectrogram and spectral derivative plots of the utterance "Even then, if she took." The spectral derivative was computed with a window size of 7 frames ($W=3$) for Fig. 5.4 and 21 frames ($W=10$) for Fig. 5.5. The frame spacing was 2 ms.
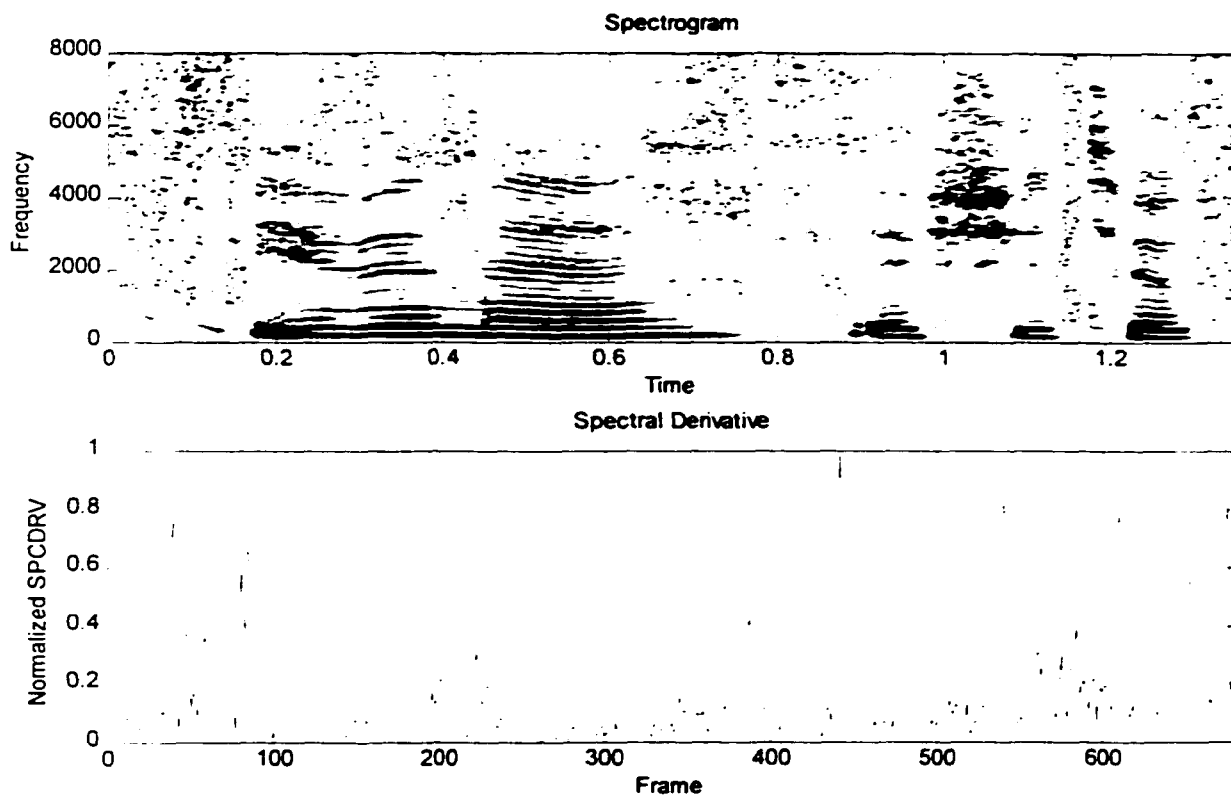
Fig. 5.4. Spectrogram and spectral derivative plot (W=3).

Notice that the peaks of spectral derivative are a good match to the acoustic transitional regions indicated in the corresponding spectrogram. The peaks are broader and overall the spectral derivative plot is smoother with a larger window in Fig. 5.5 as compared to those in Fig. 5.4.

The block processing requires a sequence of frame-based feature vectors and the corresponding spectral derivative to begin with. The length of each block and also the minimum and maximum allowable block spacing must be specified. Extra frames are added to both ends of the sequence as for the other methods. The number of frames that need to be added is equal to half of the block length.
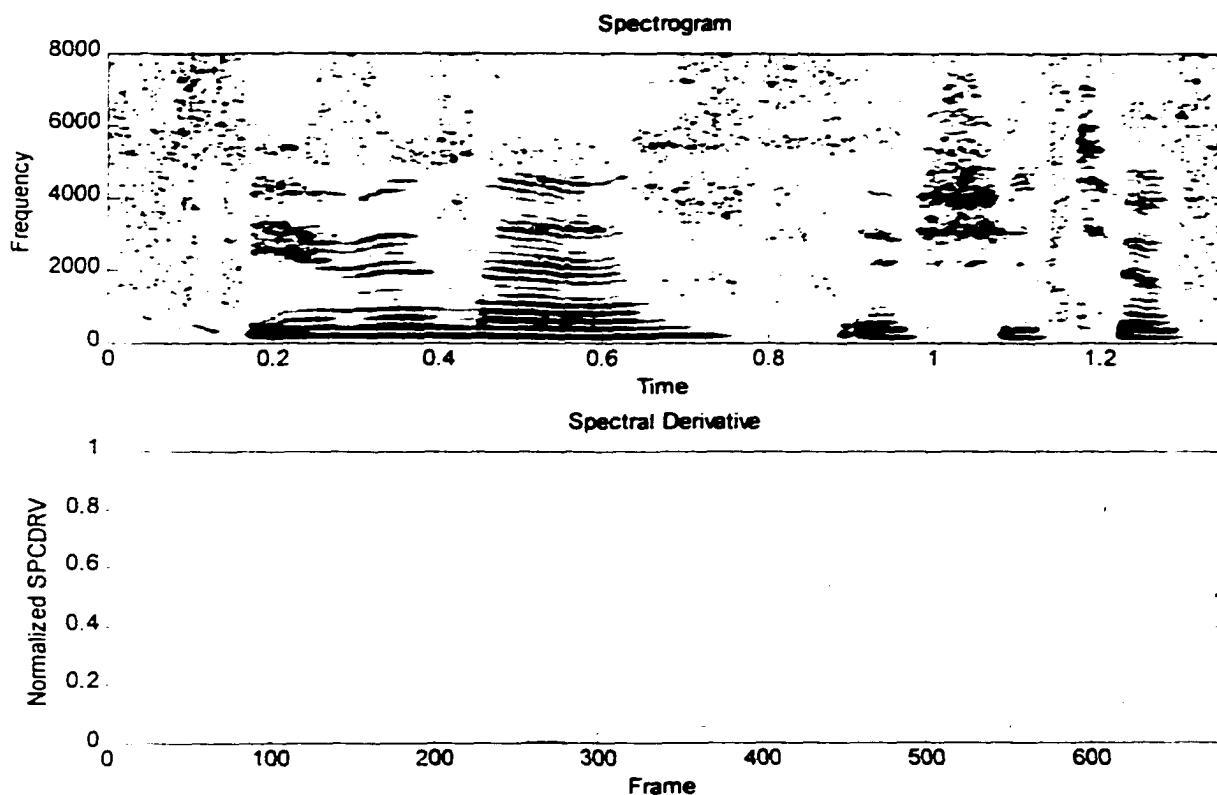
Spectrogram

Fig. 5.5. Spectrogram and spectral derivative plot (*W*=10).

The method can be further explained as follows. Suppose the center of block M is located at frame *M*. A set of DCT basis vectors is used to encode the block. To further process block *M*+1, we need to find an appropriate block spacing. In other words, we need to determine how far block *M*+1 has to be from block *M* in order to properly capture a rapidly changing signal in transitional areas and ignore redundant signals in steady state areas. To do so, first the spectral derivative value at frame *M* (center of the block) is computed and examined. The basic idea is that a small block spacing will be used when the spectral derivative amplitude is high and a large block spacing will be used when the amplitude is low. The amplitude of the spectral

derivative of frame *M* is uniformly divided into a number of levels that equals to the number of possible block spacings. For example, if the block spacing can range from 3 to 5 frames, then the spectral derivative will be divided into (5-3 + 1) = 3 levels. Each frame spacing candidate is assigned to one level. A frame spacing is picked according to the level of the spectral derivative of frame *M*.

Once the block spacing is found, the center of block *M*+1 can be determined and the block is encoded and so on. Fig. 5.6 illustrates the idea of how the block spacing is determined.

From Fig. 5.6, the spectral derivative is divided into 3 levels. Block spacing is minimal when the spectral derivative falls in Level 3, moderate in Level 2 and maximal in Level 1. Clearly, the blocks are closely spaced in the region where the spectral derivative is large.
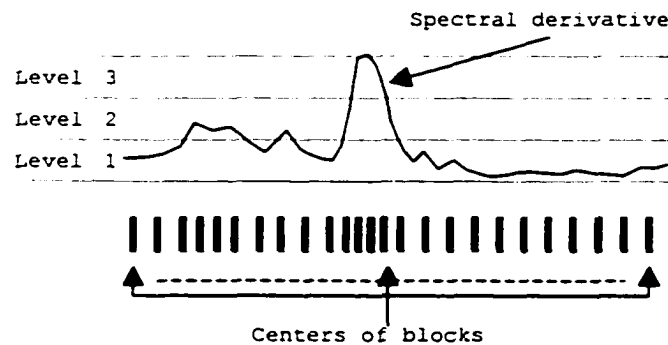
Fig. 5.6. Illustration of variable block spacing method. A small spacing is used when the spectral derivative is high and vice versa.

There are some refinements applied to the basic algorithm of the variable block spacing method described above. First, the spectral derivative can be normalized to the range of 0-1 by dividing every element by the maximum value of the sequence or by limiting the maximum value to not exceed a threshold value. In the latter case, the spectral derivative is normalized to the range from 0 to the threshold value. Note that the spectral derivative is always positive.

The second refinement is to consider the neighboring spectral derivative amplitude around the center of the block. In this work, we chose to search for the maximum value of the spectral derivative of frame M to frame $M+S$, where S is maximum block spacing. The maximum value in the range is used to determine the block spacing. This ensures that the algorithm reliably captures the transition regions because the peak may not exactly occur at frame M but rather a nearby frame. If the look-ahead method is not used, the block spacing may be too large and the rapid transitional area may be missed

As stated previously, the HTK-based HMM recognizer is expecting input features at a constant frame rate. However, the method presented here results in observation sequences of a variable data rate. Thus, we will "pretend" that the data rate is fixed so that it will comply with the HMM concept. Unfortunately, the synchronization problem will occur because the duration of a phone may have changed. For example, phones with steady spectral characteristics will become shorter in length and phones with rapidly changing spectra will become longer. In this context, longer or shorter is based on a comparison of the number of blocks for each phone versus the number of blocks for that phone using the standard method of block processing (fixed block length

and fixed block spacing). A phone that is 5 blocks longs when computed with the standard method may become 3 blocks long when computed with the variable block spacing method. The phone length depends on the characteristics of the phone.

To solve this synchronization problem, the time labeling of each phone was modified to match the boundaries of the phone after block processing.

This method has spectral derivative window size, minimum and maximum block spacing, block length and spectral derivative threshold as free parameters (5 parameters).

Fig. 5.7 depicts an implementation of the variable block spacing method on real speech data. The horizontal dark bars were placed on top of the spectrogram plot to illustrate how block spacing is varied. Note that blocks stay close together in the area where the spectrum is rapidly changing.

5.5 Combined Block Length Method

The idea behind this method is to try to capture fast changing and steady spectrum at the same time by combining the results of block encoding with a short block and a long block. Similar to the variable block length method, block spacing is constant. This work only examined a combination of two blocks of different size. The two blocks are linearly adjusted by interpolation to have the same size. This is to ensure that the dynamic features obtained from both blocks are in the same feature space.
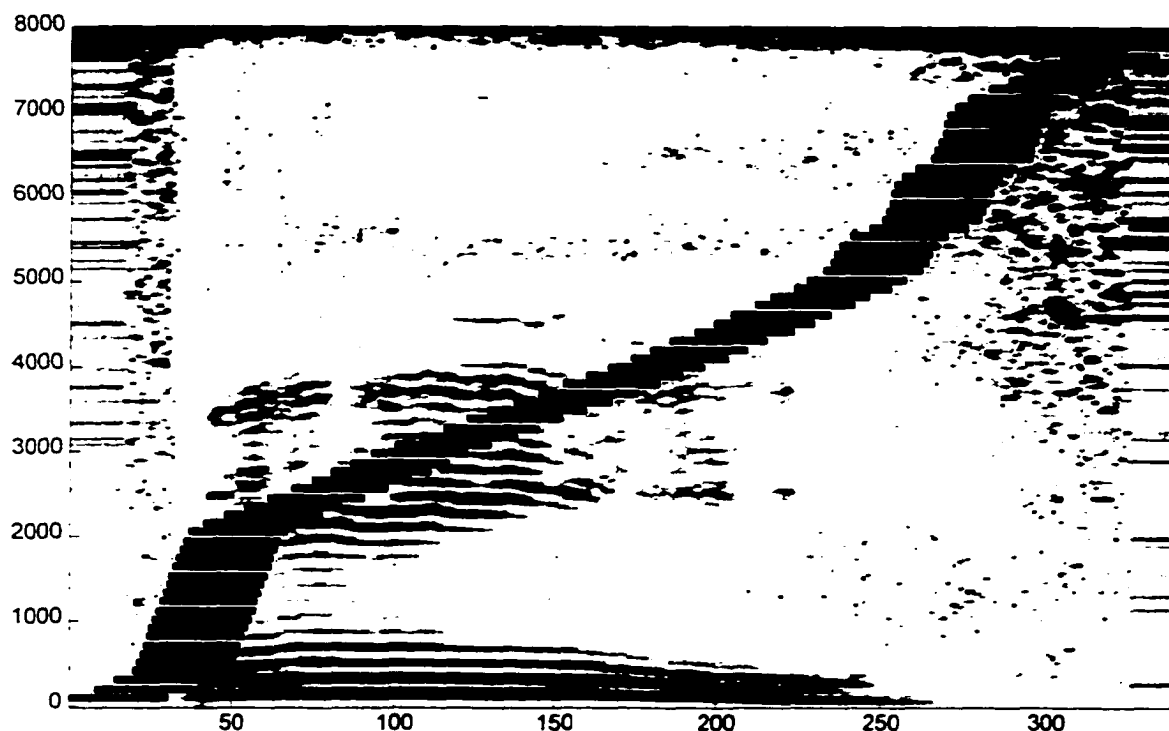
Fig. 5.7. Variable block spacing for the letter "b," block size = 31, spacing = 1-8 frames, SpcDrv window = 10.

The combination process is simply a feature vector concatenation. Note that, in block encoding process, having N DCT basis vectors results in N times more coefficients. In other words, the dimensionality of the final feature vector becomes N times of that of the input vector. For example, if frame-based features have a dimensionality of P, and N DCT basis vectors over time are used, then the feature matrix extracted from each block will be an N-by-P matrix or a NP dimensional vector[1].

---

[1] Usually such a matrix will be arranged into a vector form.

Since this method combines or concatenates the block level features of two blocks together, then the resulting feature vector has a doubled dimensionality. Even though having high dimensionality features may result in an acoustic model that is well matched with training data, it does not guarantee that the recognition performance on unseen utterances will be improved. In other words, the model trained with too many features trends to lack generalization or suffers from the "curse of dimensionality" (see Duda and Hart [73]). This method uses PCA for dimensionality reduction. We have no knowledge of whether features from small or large blocks are more important. However, PCA tends to automatically select the more important features.

Minimum and maximum block length, resampling length and block spacing are the free parameters for this method.

5.6   Chapter Summary

This chapter explained several techniques that will be investigated in this dissertation. A fixed block length and fixed block spacing technique serves as a control method for comparisons.

A variable block size technique, where the size of a block is chosen depending on local MSE measurements, was proposed. Short blocks are used to capture rapidly changing spectra and longer blocks are used to capture slowly changing spectra.

A technique called variable block spacing was proposed to capture temporal information by non-linearly adjusting the spacing between blocks. Block spacing is determined using a spectral derivative measure.

The last technique proposed was to use a combination of short and long blocks. For one output feature vector, this method concatenates the feature vectors extracted from a short block and a long block and reduces the dimensionality with PCA.

CHAPTER VI

CONTINUOUS SPEECH RECOGNITION: EXPERIMENTS

6.1   Frame Based Features

The aim of this study is to investigate alternative methods of inclusion of temporal information into the feature set. There are two major steps involved in this process. Initially, spectral characteristics of each frame in the utterance are extracted. This results in a sequence of frame-based feature vectors or a frame-based observation sequence. Then block processing, which incorporates the temporal characteristics of speech segments, is performed. In this section, the parameters used to compute frame-based observations are described.

A second order highpass IIR filter, with cutoff frequency at 3,200 Hz, was used to pre-emphasize the digitized speech input (16-bit resolution, 16 KHz sampling rate). The filtered signal was then divided into frames of 25 ms with a frame spacing of 2 ms (e.g., 23 ms overlapping). Each frame was windowed with a Kaiser window with beta=6 and width=400. The windowed signal was analyzed with a 512-point FFT, and the squared magnitude of the FFT spectrum in the range of 70 Hz to 7,000 Hz was selected for further processing. The spectrum floor was set to 60 dB below the spectrum peak. Thirteen DCT basis vectors, computed with bilinear warping (with warp factor =.45) were computed with respect to the selected frequency range. The selected spectrum was encoded with these basis vectors resulting in 13 DCTC features for each frame. Note that no spectrum smoothing was performed and no orthornormalization

was applied to the basis vectors, i.e., the method given in Zahorian and Nossair [17] was used partially track spectral peaks.

Unless otherwise stated, the above parameters were used to obtain frame-based observations of all TIMIT data. The parameters were slightly different for NTIMIT data in that the frequency range was limited to 70 Hz - 4,000 Hz and the spectrum of each frame were floored at 35 dB below the peak. Frame-based parameters were computed once, and then used for all block processing methods reported in this dissertation. Thus, performance can be fairly compared among block processing techniques.

6.2   Context-Independent Phone Model

As explained in Chapter 4, the 61 TIMIT phones were collapsed to 48 and used for context-independent phone modeling. Each phone was modeled with a 3-state, 3-gaussian mixture diagonal covariance matrix, left-to-right HMM model. Only self transitions and transitions to the next state were allowed.

The training process began with model initialization where each model was initialized for 20 iterations using the Viterbi algorithm. The time labeling of each phone was used. The parameters of each model were estimated from the data that belong to its class. Each initialized model was then re-estimated with the Baum-Welch algorithm for another 20 iterations. These two steps are similar to the training used for the isolated word recognizer. Next, all models were trained with the Baum-Welch algorithm in the embedded mode for 5 iterations. Note that the time labeling was not used in the embedded mode.

Bigram statistics of the phones were estimated from the training data. These were used in the Viterbi recognition process. Accuracy evaluation was done after the 48 phone sets had been collapsed to 39 phone sets. Results are reported in terms of percent accuracy on the test data. Accuracy was determined as follow:

$$Accuracy \quad = \frac{H - I}{N} \times 100 \text{ \%}, \qquad (6.1)$$

where H is the number of correct phones, I is the number of insertion errors and N is the total number of phones available in the test data. If insertion errors are ignored (an insertion is not counted as an error), i.e., I is removed from (6.1), then the result becomes percent correct.

6.3 Control Experiments

Experimental results reported in this section serve as baseline information. Experiments were conducted either with MFCCs or DCTCs as basic features.

All experiments conducted in this section, except those with MFCC parameters, employed segment-based (block-based) features computed with fixed block length and fixed block spacing method. Each block was decomposed with 3 DCT basis vectors over time resulting in 39 spectral/temporal coefficients. No orthornormalization was applied to the basis vectors.

For each utterance, spectral/temporal features were computed from frame-based features of the utterance. Frame based parameters were obtained as described in Section 6.1.

6.3.1 Determining Best Block Length and Block Spacing

Some of our preliminary investigation has shown that the reasonable block length for the DCTC features is between 50 ms to 100 ms and a block spacing of around 10 ms should be used. The experiments conducted in this section were based around these parameters. Table 6.1 shows experimental results when the block size was varied from 20, 25, 30, 35, and 40 frames and block spacing was varied from 6, 8, 10, and 12 ms.

Block length (frames)	Accuracy (%)			
	Blk. Spc. 6 ms	Blk. Spc. 8 ms	Blk. Spc. 10 ms	Blk. Spc. 12 ms
20	61.7	62.8	**63.1**	62.7
25	62.5	**63.3**	**63.5**	**63.4**
30	62.6	**63.5**	**63.4**	**63.5**
35	62.5	62.6	**63.1**	**63.1**
40	62.0	62.5	62.6	62.4

Table 6.1. Test results with fixed block length and fixed block spacing method at various block lengths and various block spacings.

The shaded area in Table 6.1 indicates optimum parameters for high performance. Best recognition results were obtained when the block size was between 25 to 35 frames (e.g. 73 ms to 93 ms) and block spacing was in the range of 8 ms to 12 ms. Highest accuracy of 63.5% was

obtained with a block length of 25 frames and block spacing of 10 ms. By careful examination of the results in terms of deletion, insertion and substitution error, a conclusion can be drawn that use of a short block spacing introduced more insertion and substitution errors, but the number of deletion errors was reduced. On the other hand, use of a long block spacing caused less insertion and substitution errors, but deletion errors were increased.

6.3.2 Investigation of Very Short and Very Long Blocks with and without Time Warping

Experiments were further conducted to demonstrate effects of using very short and very long block length. In addition to the previous experiments, block length of 5, 10, 15, 45, 50, 55, and 60 frames were tried. Note that a block spacing of 10ms was used throughout. It was expected that use of very short and very long block should result in significant performance degradation; it was the objective of these experiments to investigate this point. For the case of very long blocks, applying warped basis vectors over time, which effectively emphasizes the center section of the block, was expected to be beneficial. Thus, experiments were carried out with and without warping. Recognition accuracies on test data are summarized in Fig. 6.1.
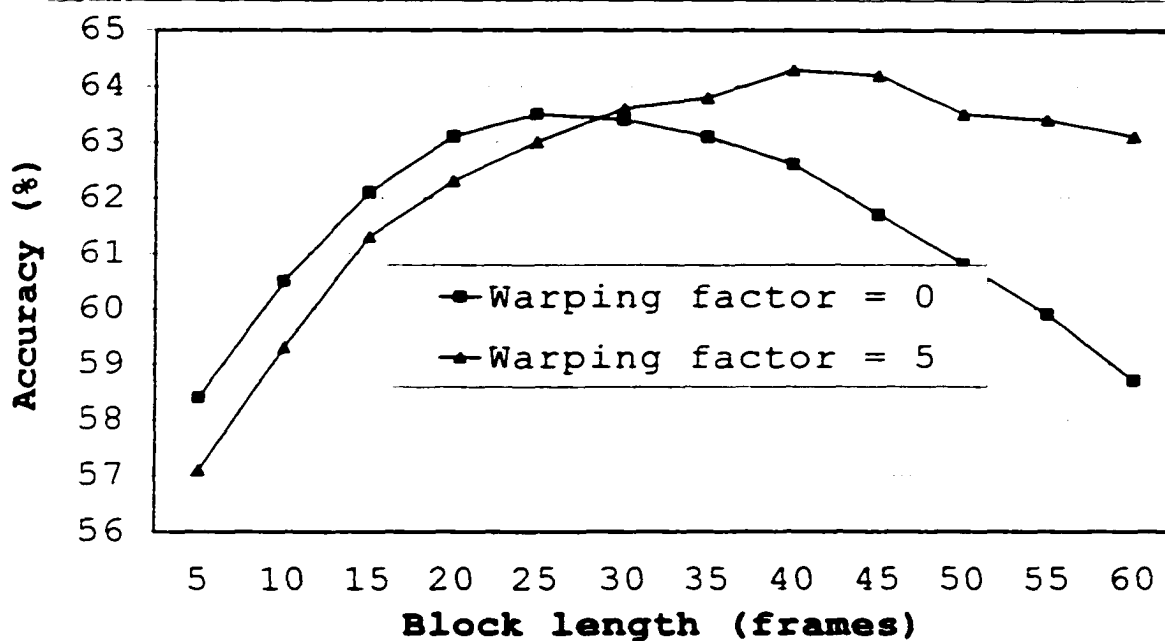
Fig. 6.1. Results at various block lengths with and without time warping (Block spacing = 10 ms).

From Fig. 6.1, there is significant degradation in accuracy, for very short or very long blocks, as expected. By applying basis vector warping, performance is improved for the large block size. However, warping slightly degrades recognition performance with short blocks. Best accuracy was obtained with a block length = 40 frames and warping factor = 5.

Experimental results with standard methods have given us a good starting point for further investigation of other techniques. This ensures that the use of a block length around 73 ms and a block spacing of around 10 ms should work reasonably well with other methods.

6.3.3 Time Label Synchronization

To investigate effects of time label synchronization, experiments were carried out using the best parameters obtained previously (block length = 40 frames, block spacing = 10 ms, warping factor = 5).

To study the importance of the use of the time labeling for model initialization, a phonetic recognition was conducted while the time labeling provided with the TIMIT database were totally ignored (no time labeling were used to estimate the initial phone models.) The initial parameters of every phone were estimated from the global mean and global covariance of the training data. Thus, all HMM models were initialized with the same set of parameters. Obviously, each initial model does not accurately represent the statistics of its data. Highest recognition accuracy obtained after 15 training iterations was 58.2%. Note that more training iterations were tried but there was no improvement after 15 iterations.

A series of experiments were conducted to test the sensitivity of the recognizer to the accuracy of the time label. Original time labeling was perturbed by uniformly distributed noise (with zero mean) such that the time indices of each phone were off by as much as ±20 ms. The recognizer was initialized with these modified labeling. Table 6.2 shows recognition performance at various perturbation levels.

Perturbation (ms)	Accuracy (%)
0	64.3
±2	64.2
±4	64.1
±6	64.1
±8	64.3
±10	64.4
±15	64.0
±20	63.7

Table 6.2. Recognition results at various levels of noises added to the time labeling.

In Table 6.2, there was no significant performance degradation if the labels were not off by more than 15 ms in both directions. However, there was a slight loss of accuracy when the time labeling was modified by as much as ±20 ms. Note that it was impractical to consider time perturbations larger than this, since some of the phones would then have zero length.

Also recall from Chapter 5 that in our method of computing spectral/temporal features, some extra frames must be appended to both ends of the utterance to maintain the synchronization between the feature vectors and the phone boundaries. The first two series of experiments described in this subsection verified the importance of the time labeling but did not clearly illustrate the actual problem which occurs when no extra frames are added. Hence, an experiment which used the fixed block length and fixed block spacing method was performed to clarify this point. However, there were no extra frames added. As a result, the number of

processed blocks were less than the number of frames. The parameters used for this experiment were block length = 40 frames, block spacing = 10 ms, warping factor = 5. The accuracy obtained was 57.6%. Note that the result was much lower than that obtained when extra frames were included (64.3%). Notice that the result achieved was slightly lower when compared to the result obtained without the use of labeling information.

In conclusion, labels boundaries are essential but need not be extremely accurate. A recognizer that has been initialized with reasonable training samples performs much better than the one that has not been initialized or has been initialized with false data.

6.3.4 Baseline Experiment with HTK's MFCC Features

HTK provides a tool that can extract several kinds speech representations, including MFCC which is a standard feature set for many current systems. An experiment was therefore also conducted using MFCC parameters to serve as a baseline result. The parameters used to compute the MFCC features were mostly default for the HTK and are rather standard. Theses parameters can be summarized as follow:

- Pre-emphasize filter: $H(z) = 1-0.95z^{-1}$
- Use Hamming window: Yes
- Frame size: 25 ms
- Frame spacing: 10 ms
- Frequency range: 70-7,000 Hz
- Spectral floor: 50 dB
- Number of filter channels: 20
- Number of Cepstral Coefficients: 12

Twelve MFCC terms plus one normalized-energy term were extracted from each frame. Delta and delta-delta coefficients, both obtained over a 5-frame window, were included resulting in a total of 39 terms. The highest accuracy obtained with MFCC based features was 64.0% on test data. This result is slightly lower than that obtained with DCSC features.

6.4 Variable Block Length Experiments

We first tested the variable block length method with the ISOLET database believing that if a technique does not work well with isolated word recognition, then it will not work well with continuous speech recognition. Unfortunately, we did not find this technique to be beneficial for isolated word recognition. Best recognition result with the variable block length technique was not better than that obtained with the standard method (fixed block length and spacing). Nevertheless, some tests were performed with continuous speech, as reported in this section, for the sake of completeness.

Frame-based features obtained as described in Section 6.1 were used throughout in this section.

This method involved normalization of the block using a resampling technique. In order to make certain that resampling does not itself cause significant performance degradations, a series of experiments were carried out. Based on the best parameters experimentally obtained in Section 6.2 (block size of 40 frame, block spacing of 10 ms, BVT warping factor of 5), each block was resampled. The resampling length was varied from 20 to 60 frames. Table 6.3 shows recognition results for these cases.

Resampling length (frames)	Accuracy (%)
20	64.1
30	64.1
40	64.3
50	64.4
60	64.3

Table 6.3. Recognition results with various resampling lengths.

Note that the resampling length of 40 frames indicated in Table 6.3 implies no resampling. The results of this experiment leads to a conclusion that resampling does not degrade recognition as long as the new length is longer than the original block length (up sampling). Resampling with a smaller number of frames (down sampling) degrades performance only slightly for down sampling by up to a factor of two.

Use of warped basis vectors over time to compute spectral/temporal features has been shown to be beneficial. However, for the variable block length technique, the aim is to find an optimum block length that yields a MSE below a specified threshold. The effect of warping is to emphasize the signal in the center area of the block. Thus, the use of warping does not fit well with this method. As a consequence, warping was not used for the experiments in this section.

Experiments were conducted to find the best parameters for this method. Some preliminary experiments have indicated that the use of a threshold around 4.0 to 5.0 should result in a promising performance. Based on the best parameters

obtained in Section 6.2, several experiments were conducted using several combinations of parameters. Block size in the range of 11 to 41 frames, block spacing of 8 to 10 ms and resampling length of 20 and 50 frames were examined. Table 6.4 shows recognition performance on test data using various combinations of parameters with a resampling length of 20 frames. Table 6.5 illustrates the same experiments with the resampling length of 50 frames.

Block Spacing (ms)	Block length (frames)	Accuracy at various threshold and block length (%)					
		4.0	4.2	4.4	4.6	4.8	5.0
8	11 to 25	63.4	63.6	63.5	63.4	63.4	63.3
8	11 to 31	63.9	63.6	63.6	63.5	63.6	63.4
8	11 to 41	63.1	62.9	62.8	62.9	62.9	62.8
10	11 to 25	63.6	63.4	63.3	63.2	63.4	63.7
10	11 to 31	63.5	63.7	64.0	63.8	63.5	63.4
10	11 to 41	63.1	62.7	62.5	62.7	62.8	62.8
12	11 to 25	63.2	63.5	63.2	63.1	63.4	63.4
12	11 to 31	63.2	63.3	63.5	63.4	63.3	63.1
12	11 to 41	62.1	63.4	62.3	62.3	62.1	62.3

Table 6.4. Recognition accuracy for test data when all blocks were resampled to 20 frames.

	Accuracy at various threshold and block length (%)						
Block Spacing (ms)	Block length (frames)	4.0	4.2	4.4	4.6	4.8	5.0
8	11 to 25	63.6	63.5	63.3	63.5	63.4	63.2
8	11 to 31	63.8	63.9	63.5	63.5	63.4	63.6
8	11 to 41	63.1	63.3	63.0	62.8	62.7	63.2
10	11 to 25	63.2	63.3	63.2	63.4	63.4	63.8
10	11 to 31	63.5	63.9	63.5	63.8	63.6	63.5
10	11 to 41	62.9	63.0	62.6	62.4	62.8	62.9
12	11 to 25	63.3	63.4	63.2	63.0	63.4	63.6
12	11 to 31	63.2	63.3	63.3	63.2	63.5	63.4
12	11 to 41	62.7	62.7	62.6	62.3	62.6	62.6

Table 6.5.   Recognition accuracy for test data when all blocks were resampled to 50 frames.

From the tables, best performance was 64.0% achieved with a block length varying from 11 to 31 frames and each block normalized to 20 frames. Block spacing was 10 ms and the error threshold was 4.4. The accuracy obtained was slightly better than that obtained from the fixed block length method with no warping. In general, there were no significant performance differences among the various parameter settings tried.

6.5 Variable Block Spacing Experiments

This method was also first tested on the alphabet recognition task as described in Chapter 3. The highest accuracy achieved was 97.8%, which is comparable to the 97.9% obtained in Chapter 3. However, much simpler HMM models (5 states, 3 mixtures with a diagonal covariance matrix) were used for the results reported in this Chapter. In the remainder of this section, we will focus on the use of this method for phonetic recognition for the case of continuous speech.

As noted in Chapter 5, this method requires modification of the original time labeling to correct the synchronization problem. Also, it was shown in control experiments that loss of synchronization can cause a severe performance degradation. To support this reasoning and to partially test the new labels, we conducted an experiment that forced the algorithm to behave like the fixed block length and fixed block spacing method. This was achievable by setting the minimum block spacing to be equal to the maximum block spacing. In fact, features obtained this way are identical to those obtained with the standard technique. However, the difference lies in the time labeling used for HMM initialization where the original timing was used with the standard method and the modified timing was used with the variable block spacing method. Experimental results verified that there was no significant difference in performance.

To more thoroughly investigate the performance of the variable block spacing technique, a series of phone recognition experiments were performed to determine the range of block spacings that will give optimum results. Several block spacing combinations and spectral derivative

windows of 5, 10 and 15 frames were tried. Every block has a length of 25 frames and was encoded without warped basis vectors. Unless otherwise noted, 3 DCT basis vectors over time were used for each block. Neither warping nor orthonormalization were applied to the basis vectors. This resulted in a spectral/temporal feature vector of 39 dimensions for each block. Fig. 6.2 is a bargraph illustrating experimental results obtained using the variable block spacing technique for feature extraction.

From the figure, low performance was obtained when the range of block spacing was large. For example recognition accuracies were low when a block spacing of 2-8, 2-10 and 2-12 ms were used. On the other hand, high accuracies were obtained when block spacing was not varied by much. Since the best result of 64.0% was achieved with block spacing varying in the range of 8-10 ms and a spectral derivative window length of 30 frames was used. As a result, further experiments were conducted based on these parameters, as summarized below.
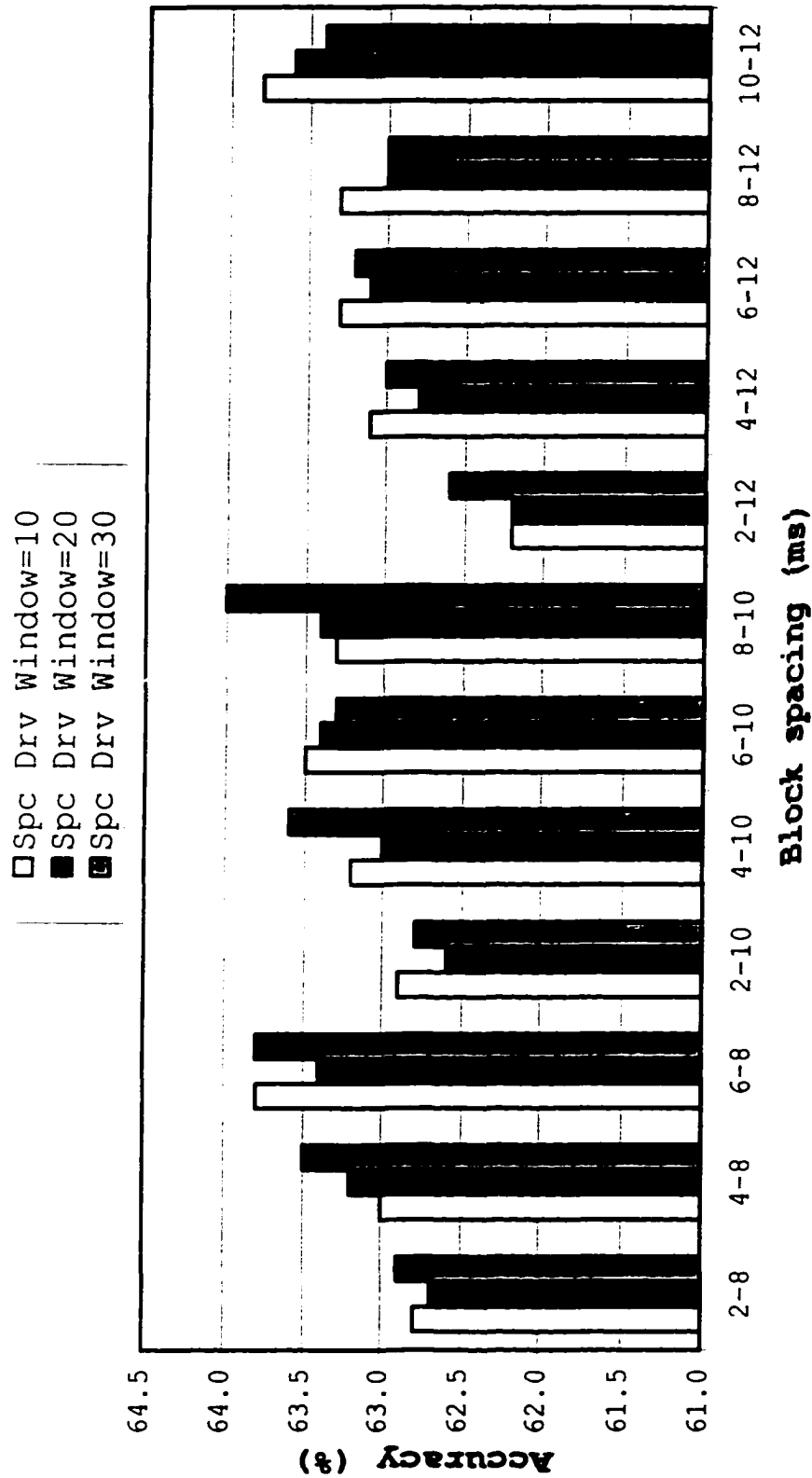
Fig. 6.2. Recognition results at various block spacing ranges and spectral derivative window lengths.

The next series of tests was designed to find a block length and a warping factor that works best with this technique, given that the block spacing is in the range of 8-10 ms and the spectral derivative window size is 30 frames. Warping was used in some cases here because it has shown to be useful in control experiments, especially when the block length is large. Fig. 6.3 shows recognition accuracies on test data when the block length varied from 25 to 61 frames and the warping factor was 0, 5 or 10.
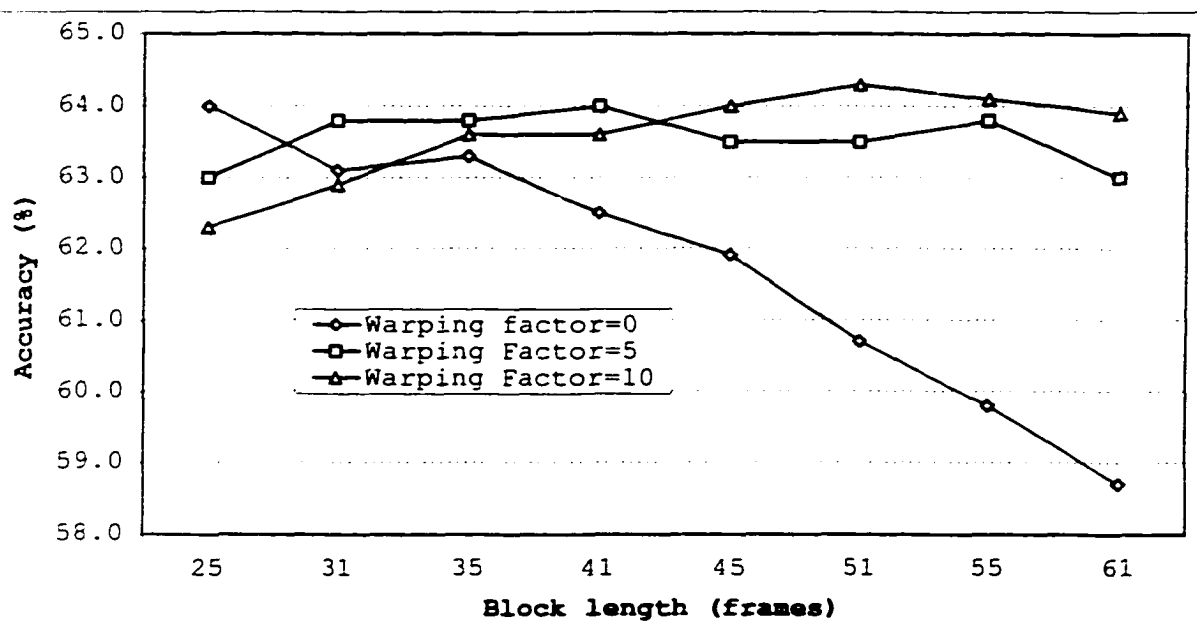
Fig. 6.3. Results at various block lengths and various warping factors.

Fig. 6.3 shows that the technique was not very sensitive to block length. However, the use of a warping factor of 5 performed better with a block length shorter than 45 frames (113 ms) and the use of warping factor of 10 performed best with the longer block length. Again,

performance degradation occurred when no warping was used at a large block length. Highest accuracy of 64.3% was obtained with a block length = 51 frames, and a warping factor = 10.

6.6 Investigation of PCA Combination of Two Blocks

At each position in the utterance, this method computes two sets of spectral/temporal parameters, each with a different block length. Then the two sets of parameters are combined with PCA.

The frame-based observations obtained as described in Section 6.1 were used throughout. Since a frame based feature vector has 13 components, each block, when expanded with 3 DCS, results in 39 terms. Thus, the total number of parameters is 78 if the two blocks are directly combined. However, for these tests, PCA was used to reduce the dimensionality of the features to 39, so that the total feature numbers would be the same as for the other tests reported in this study. All experiments presented in the section were conducted with these 39 PCA terms.

This method of feature computation has 4 free variables to adjust. Based on our pilot study and previous experimental results, the minimum block length was chosen to be 11, 15, 21 or 25 frames and the maximum block length was chosen to be 31, 35 or 41 frames. A block spacing of 8 or 10 ms and a resampling window size of 50 frames was used. All combinations of these values were tried and the results obtained are reported in Fig. 6.4 and Fig. 6.5.
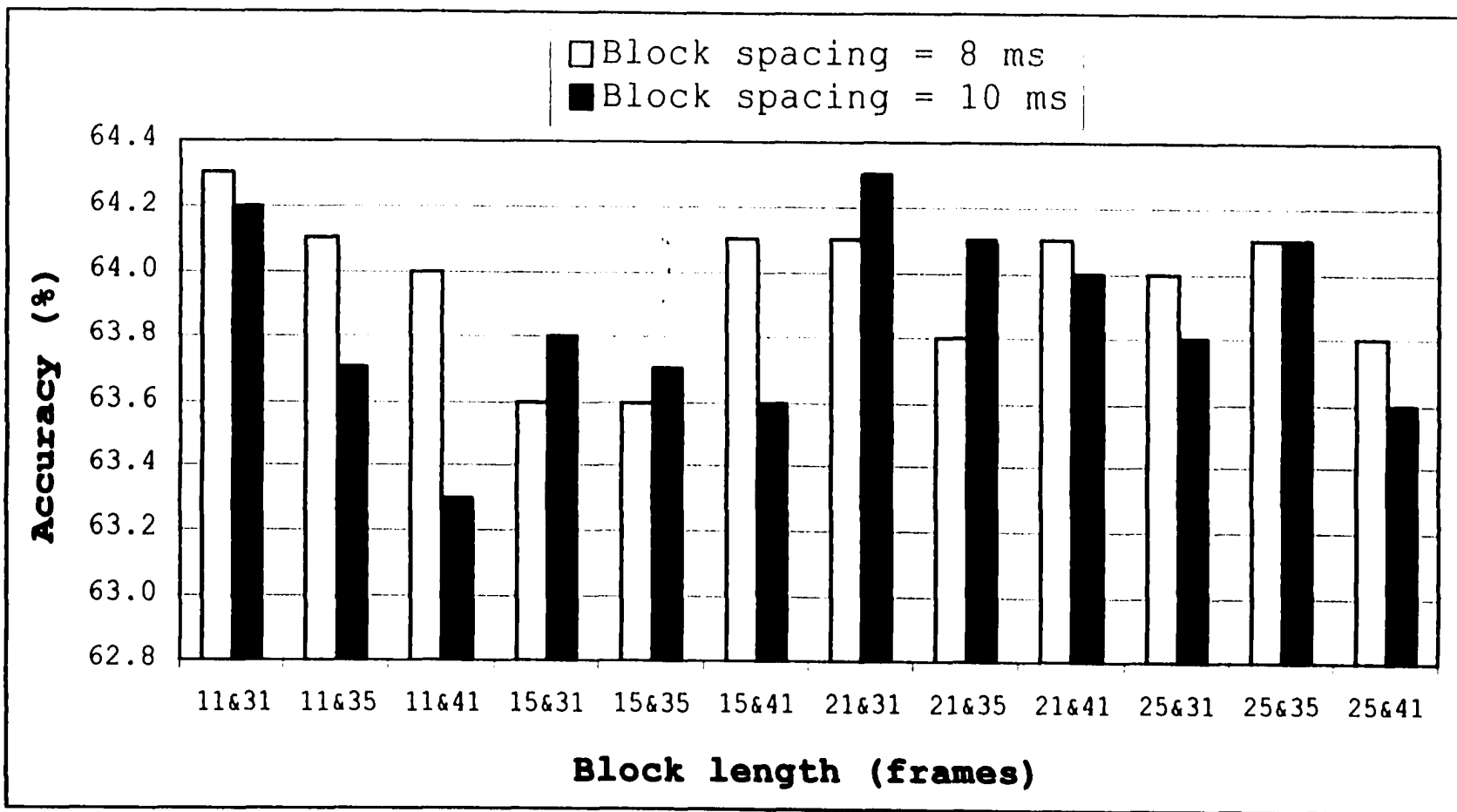
Fig. 6.4. Results with various minimum and maximum block sizes (warping factor = 0, and resampling window = 50 frames).
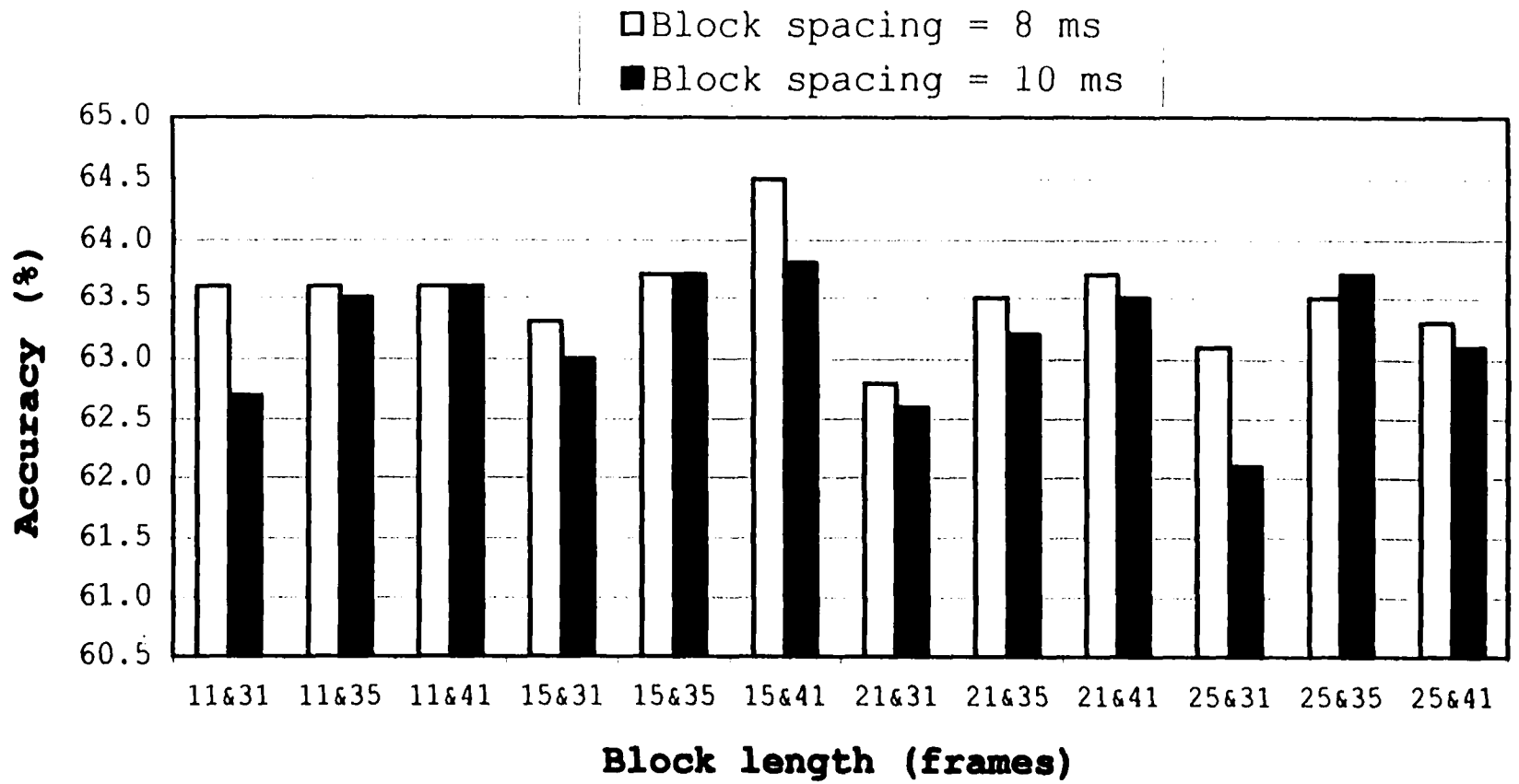
Fig. 6.5. Results with various minimum and maximum block sizes (warping factor = 5, and resampling window = 50 frames).

From Fig. 6.4 and Fig. 6.5, there were no significant differences in performance among all cases. In general, use of a block spacing of 8 ms is slightly better than use of block spacing of 10 ms. The best result (64.5%) was obtained with a block spacing of 8 ms, block length of 15&41 frames and warping factor of 10.

6.7   Increasing Model Complexity

The experimental results reported thus far in this section were obtained with 3-state 3-mixture HMMs. Low complexity models were used because training and recognition could be completed in a short amount of time. This allowed us to try many sets of parameters for each technique. Once optimum parameters were found, higher complexity HMM models could be applied to achieve higher recognition accuracy. In this section, the best case for each method was taken and some experiments with HMMs of a higher number of mixture components were performed. Table 6.6 compares recognition results obtained with standard method (DCSCs and MFCCs) and all proposed methods when the number of mixture increases from 3 to 18 mixtures (diagonal covariance) per state and 1 to 4 mixtures (full covariance) per state.

Number of mixtures	Accuracy with different methods (%)				
	DCSC	MFCC	Var. Block Length	Var. Block Spacing	Combined Block Size
3 Diag.	64.3	64.0	64.0	64.3	64.5
6 Diag.	66.2	66.3	65.6	65.6	65.7
9 Diag.	67.9	67.1	67.0	67.5	67.3
12 Diag.	68.3	67.8	68.2	68.9	67.8
15 Diag.	68.9	67.9	68.6	68.9	68.2
18 Diag.	69.4	68.4	69.2	69.0	69.1
1 Full	64.8	63.6	65.0	65.4	65.3
2 Full	68.7	67.5	68.6	68.7	69.2
3 Full	70.1	68.7	N/A*	70.4	70.5
4 Full	N/A*	69.7	N/A*	N/A*	71.2

*No result obtained due to training problems.

Table 6.6.   Results with various numbers of mixtures per state (diagonal and full covariance matrix)

As seen from the table above, the recognition accuracy improves significantly with higher complexity HMMs. However, training and recognition times are also longer. For example, the 18-mixture model takes about 5 times longer for recognition than the 3-mixture model. Note that "N/A" means that the model could not be successfully trained. This was due to the complexity of a model with a full covariance matrix in that some distribution of the training data may result in a non-invertible covariance matrix.

For systems with a diagonal covariance matrix, performance was not significantly improved when the number of mixtures was more than 12. Best results overall for 18 mixtures were slightly below 70%. Use of one mixture per state with a full covariance matrix showed slight improvement over the use of 3 mixtures with a diagonal covariance matrix. In general, use of 3 or 4 mixtures with a full covariance matrix outperformed use of 18 mixtures with a diagonal covariance. However, the 3 mixture model with a full covariance matrix takes about 10 times longer for recognition than the 3 mixture model with a diagonal covariance matrix and about 2 times longer as compared to the 18 mixture model with diagonal covariance matrix.

There is an issue of the difference in memory requirement between the use of diagonal and full covariance matrix. Only diagonal terms must be stored for the diagonal case. For the full covariance case, all the diagonal terms and all the upper (or lower) triangle terms of the covariance matrix must be stored. Memory usage for a one-mixture full covariance model is slightly more than that of a 9-mixture diagonal covariance model. On the other hand, systems with 9 diagonal mixtures performed much better than systems with one full mixture, as seen in Table 6.6.

6.8 Context Dependent Experiments

A popular method for increasing modeling accuracy is to use context-dependent HMM models such that context is taken into consideration. The underlying reason for using context-dependent models is the coarticulation which occurs in speech production because of the inertia of the vocal tract of the speaker. This results in a different realization of the same phone depending on the adjacent phones. For example, the realization of the phone /ae/ that is followed by the phone /m/ (as in the word 'am') is different from the realization of phone /ae/ that is followed by the phone /t/ (as in the word 'at'). Monophone systems do not really take in account these coarticulation effects.

There are two techniques for adding contextual information into the HMM-based recognizer: biphone and triphone methods. The first one models the coarticulation effects, taking into account only pairs of adjacent phones. There are right-context biphones and left-context biphones. Right-context biphones model the current phone and coarticulation with the succeeding phone. On the other hand, left-context biphones model the current phone and coarticulation with the preceding phone.

Contextual information can be better observed with triphone context-dependent models. In this case, coarticulation effects of the preceding and succeeding phones are taken into consideration. Although there are several implementation problems that have to be accounted for, many of the state-of-the-art results in continuous speech recognition were achieved using triphones.

The big drawback to the use of either biphones or triphones is the large number of models needed. Theoretically, the number of models is N^2 for biphones and N^3

for triphones, where N is number of monophone models. For example, in our case with 48 monophone models, there will be 2304 biphone models and 110592 triphone models. Although some triphones and biphones may not be present in the vocabulary, the number of models is still relatively large. This leads to the problem of insufficient training data. As a result, a model that has only a few training samples will not be accurately and robustly estimated. Use of context-dependent models will not improve overall system performance if this issue is not properly addressed.

To build a context-dependent system, one has to first build a context independent system, i.e., monophone models must be trained. Then each biphone or triphone model is cloned from its corresponding monophone. After that each context-dependent model is further trained with its associated training data.

There are many techniques to overcome the lack of training data problem. One solution is to use a sharing mechanism. A simple but effective example would be for all models of the same base phone to share the same transitional probability matrix. Although the transitional probability matrix is a crucial part of a HMM model, it is not sensitive to context. The sharing of the transitional probability matrix greatly reduces the number of parameters to estimate. Further sharing is still needed to make these models practical.

Recall that each state of a HMM has a mixture of Gaussian PDFs to capture the statistics of the speech data. Even though it is a good practice and it is possible to share parameters among several states, care must be taken about which parameters are shared because the performance of the recognizer depends crucially on how accurate a state can

model the speech. The HTK provides tools to perform model cloning and very effective parameter tying (sharing) [64],[65].

In this work, simple right-context biphone models were used. First, all occurrences of a biphone in the training data were counted. Then, the top 400 phone pairs which occurred the most were chosen. This will ensure that each biphone will be sufficiently trained. Furthermore, all 48 monophones were included to account for the less frequently occurring phone pairs and for the last phone of each utterance which has no succeeding phone. As a consequence, there were a total of 448 HMM models for the context-dependent system.

The original phone transcriptions provided with the TIMIT database was updated with the 448 context-dependent models. The HTK provides a tool that can modify the TIMIT label files and such a tool was used in this work.

Since the use of a Gaussian mixture with a full covariance matrix introduces a system with great complexity, use of context-dependent phone models results in an extremely complex and computationally demanding system. Therefore, context-dependent models were used only with diagonal covariance matrices. Table 6.8 depicts experimental results obtained with CD models using the best parameters of each method for feature computations. The results should be compared against those in Table 6.7 where monophone models were used.

Results from Table 6.7 indicate that context-dependent models improve performance significantly over the low complexity models. However, adding context did not perform significantly better than monophones with a large number of mixtures. Thus, it appears that the additional mixture

components capture most of the contextual information. Therefore, introducing explicit context modeling is not necessarily beneficial. However, since only relatively simple CD models were used in this dissertation, it is possible that monophone models with a large number of mixtures could be further improved with a carefully designed biphone or triphone system. Time limitations prevented a more detailed investigation of this issue, which is essentially a secondary point for the present work.

The best result of 72.6% obtained with the PCA method is not the absolute highest in the literature. However, it does compare favorably. In the concluding chapter (Section 6.11), a table is given which compares results of the present study to results reported in the literature for the same task.

Note that the highest accuracy obtained with monophone models was 71.2% with full covariance Gaussian PDF. The 1.4% improvement in accuracy due to the introduction of the biphone models is quite small, considering the increasing of recognition time by about a factor of three.

Number of mixtures	Accuracy with different methods using context-dependent models (%)				
	DCSC	MFCC	Var. Block Length	Var. Block Spacing	Combined Block Size
3 Diag.	69.2	69.1	69.2	69.2	69.1
6 Diag.	70.9	70.4.	70.1	70.5	70.2
9 Diag.	71.4	58.3[*]	71.6	71.1	70.8
12 Diag.	71.5	63.7[*]	71.6	71.5	70.9
15 Diag.	71.3	63.9[*]	71.4	71.5	71.5
18 Diag.	71.4	68.3	71.3	71.4	72.6

[*] Low result obtained due to training problems.

Table 6.7.   Results with right-context biphone model for various numbers of mixtures per state (diagonal covariance matrix only).

6.9 Experimental Results with NTIMIT

The experiments reported in the previous sections were carried out to test the proposed techniques with high quality data. Parameters were "optimized" for each method, at least subject to the constraints given. In this section, these same methods were evaluated with the telephone-grade NIMIT database. With band-limiting and lower SNR data, the robustness of various feature extractions was tested. There were no parameter tunings performed in these tests. Rather, the parameters obtained previously with the TIMIT data were used. The only difference when working with NTIMIT data was that the frequency range was reduced (70 Hz to 4,000 Hz) and the spectrum floor was reduced (to 35 dB) to match the database.

Table 6.8 summarizes recognition results obtained under the same conditions as for Table 6.6 except that the NTIMIT database was used. In general, the results with MFCC parameters were better than other methods when the number of mixtures was small and when a full covariance matrix was used. In many cases, there were no performance improvements over the use of a full covariance matrix.

Table 6.9 shows results with NTIMIT when context-dependent models were applied. There were slight performance improvements with the use of context-dependent models. Best accuracy of 59.2% was achieved with PCA features.

Number of mixtures	Accuracy with different methods (%)				
	DCSC	MFCC	Var. Block Length	Var. Block Spacing	Combined Block Size
3 Diag.	50.8	51.4	50.9	50.9	50.9
6 Diag.	54.0	54.4	53.4	53.4	53.2
9 Diag.	57.2	55.2	54.6	54.6	54.6
12 Diag.	N/A*	55.4	56.2	56.2	55.1
15 Diag.	56.0	56.0	55.6	55.6	55.9
18 Diag.	56.5	56.8	N/A*	N/A*	56.3
1 Full	51.6	53.3	51.8	51.8	52.5
2 Full	54.5	56.9	54.5	54.5	55.7
3 Full	56.1	58.4	56.2	56.1	57.4
4 Full	N/A*	N/A*	N/A*	N/A*	N/A*

*No result obtained due to training problems.

Table 6.8. NTIMIT results with various numbers of mixtures (diagonal and full covariance matrix).

Number of mixtures	Accuracy with different methods (%)				
	DCSC	MFCC	Var. Block Length	Var. Block Spacing	Combined Block Size
3 Diag.	55.7	48.4**	54.9	54.9	55.2
6 Diag.	56.7	49.9**	57.6	57.6	56.5
9 Diag.	57.8	53.9**	57.6	57.6	57.7
12 Diag.	N/A**	53.5**	58.2	58.2	58.4
15 Diag.	58.0	53.6**	58.0	58.0	59.2
18 Diag.	57.7	42.1**	N/A*	N/A*	58.7

* No result obtained due to training problems.
** Low result obtained due to recognition problems.

Table 6.9. Results on NTIMIT database with context-dependent models (diagonal covariance matrix only).

6.10 Discussion of Results

On the TIMIT database, none of methods with the non-uniform time sampling of features performed significantly better than the standard method. One explaination is that continuous speech is so complex that small inconsistencies in the processing steps may degrade overall performances even though the method is overall theoretically sound. Another reason could be that that the HMM is already so highly tuned that further improvements are just very difficult to achieve. The HMM framework may need more substantial modifications, with a better statistical model, in order to obtain any significant benefits from the approach attempted in this work.

One possibility is that the amount of training data is not sufficient. This can be tested by comparing the accuracy on training data versus test data for some cases. If the recognition rate on the training data is much higher than that on the test data, then it implies that more training data is needed for better model estimation. The recognition rate of 65.7% on training data versus 64.3% on test data (best case of the fixed block length and fixed block spacing method, 3 states and 3 mixtures per model) indicates sufficient training data for this case. On the other hand for our best case (combined block length method with biphone models), the training results were about 89.4% versus test results of 72.6%, indicating lack of training data for this case.

6.11 Comparison with Other Systems

The absolute best result on the NIST core test set (TIMIT) was 74.4% reported by [74]. Table 6.10 summarizes the methods used and accuracy achieved by other systems.

System	Accuracy on NIST core test set(%)
Lamel and Gauvain, 1993, [75], Bigram, Triphone, CDHMM.	69.1
Goldenthal, 1994, [29], Trigram, Triphone, STM.	69.5
Robinson, 1994, [76], Bigram, Recurrent Nets.	73.9
Deng and Sameti, 1996 [77], Polynomial state HMM.	73.4
Mari et al., 1996, [78], Bigram, 2nd order HMM.	68.8
Chang and Glass, 1997, [79], SUMMIT, Bigram, Diphone.	73.4
Ming and Smith, 1998, [74], Bayesian triphone models.	74.4
Karnjanadecha, Bigram, Monophone, Diag. Cov., Non-uniform time sampling.	69.4
Karnjanadecha, Bigram, Monophone, Full Cov., Non-uniform time sampling.	71.2
Karnjanadecha, Bigram, Diphone, Diag. Cov., Non-uniform time sampling.	72.6

Table 6.10. Phonetic recognition accuracies on NIST core test set for various ASR systems.

There were several systems that achieved an accuracy greater than 70%. The best result reported in this work was 72.6%, achieved with context-dependent models and PCA features.

Interestingly, our monophone system with a full covariance matrix outperforms 3 out of 7 context-dependent systems. The accuracy obtained from such a system was 71.2%. Moreover, the performance was not much lower (69.4% accuracy) when an 18-mixture with diagonal covariance matrix was used.

There are not many results reported for NTIMIT data for this task. The results reported here should serve as a baseline for further investigation.

6.12 Chapter Conclusions

This chapter contained a summary of all experiments and results for every proposed method using the TIMIT and NTIMIT databases. Although none of the proposed non-uniform time sampling methods were significantly better than the fixed block length/fixed block spacing DCTC method, the best result obtained was very close to the performance of the state-of-the-art systems presently available.

CHAPTER VII

CONCLUSIONS AND FUTURE WORK

Many aspects of the use of variable block length and/or variable block spacing method for spectral/temporal feature computation have been investigated. Even though the proposed non-uniform time resolution methods have not proved themselves to be beneficial for continuous speech recognition over fixed time resolution methods, they should serve well as a starting point for further study in this field. Furthermore, there is room for improvement for each method which may lead to significant performance improvement.

Use of triphone models with careful parameter tying should result in a much better performance. Since the aim of this work was concerned with the non-uniform time feature computation issue, we did not test our features with a more advanced statistical modeling method.

7.1 Contributions

In Chapter 3, a simple variable block length technique has been proposed for use with alphabet recognition. There, the block length and block spacing was varied depending on the position within the utterance. Our recognition accuracy on test data was the highest result reported in the literature on the same task. The method was also robust in the presence of noise.

The idea of variable block length was adapted to continuous speech recognition. In Chapter 4, several methods were proposed. The first method was to choose a block length

based on the reconstruction error, while the block spacing was fixed. The second method used the spectral derivative obtained from frame-based observation sequence to determine appropriate block spacing at each position in the utterance. Block length was fixed in this case. The third method was to apply a small block to capture rapid change of spectrum and a long block to capture the steady state portion at the same time. Then, PCA was used to reduce the dimensionality of the combined parameters.

Chapter 6 illustrates these methods with a collection of experiments that were conducted using features computed from various methods. Performance was very similar for almost all cases. Best accuracy achieved in this work was comparable to most high-performance systems while the model complexity of our model was lower.

7.2 Future Work

There are several suggestions for further research as follow.

1. All methods proposed in this dissertation are applicable to any frame-based parameters. Thus, it would be interesting to test them with MFCC coefficients.

2. Linear feature transformations such as LDA or PCA can also be applied to the final features after the block processing.

3. Better context-dependent model should be introduced.

4. Variable block spacing based on the variable block length method and variable block length based on

the variable block spacing method should be investigated.

5. Instead of an HMM, another framework should be employed.

6. The HMM may need to be modified to more directly incorporate the idea of non-uniform time sampling of the observation vector.

REFERENCES

[1]   Dragon NatuallySpeaking, http://www.dragonsys.com/.

[2]   IBM ViaVoice, http://www.4.ibm.com/software/speech/millennium/.

[3]   Chen, S., Eide, E., Gales, M., Gopinath, R. Kanevsky, D. and Olsen, P., "Recent Improvements to IBM's Speech Recognition System for Automatic Transcription of Broadcast News," *Proc. ICASSP 99*, pp. 37-40, Phoenix, AZ, March 1999.

[4]   Billa, J., Colhurst, T., El-jaroudi, A., Iyer, R., Ma, K., Matsoukas, S., Quillen, C., Richardson, F., Siu, M., Zavaliagkos, G. and Gish, H., "Recent Experiments in Large Vocabulary Conversational Speech Recognition," *Proc. ICASSP 99*, pp. 41-44, Phoenix, AZ, March, 1999.

[5]   Language Connect Institute, *Accent Coach, English Pronunciation Trainer*, Syracuse Language, 1999.

[6]   Zimmer, A., Dai, B. and Zahorian, S., "Personal Computer Software Vowel Training Aid for the Hearing Impaired," *Proc. ICASSP 98*, pp. 3625-3628, Seattle, WA, May 1998.

[7]   Furui, S., "Speaker-Independent Isolated Word Recognition Using Dynamic Features of Speech Spectrum," *IEEE Trans. ASSP*, vol. 34, pp. 52-59, February 1986.

[8]   Furui, S., "On the Use of Hierarchical Spectral dynamics in Speech Recognition," *Proc. ICASSP 90*, pp. 789-792, 1990.

[9]   Rabiner, L. and Juang B., *Fundamentals of Speech Recognition*, Prentice Hall, New Jersey, 1993.

[10]   Karnjanadecha, M., and Zahorian, S. A., "Robust Feature Extraction for Alphabet Recognition," *Proc. ICSLP 98*, Sydney, Australia, vol. 2, pp. 337-340, 1998.

[11]   Karnjanadecha, M., and Zahorian, S. A., "Signal Modeling for Isolated Word Recognition," *Proc. ICASSP 99*, vol. 1, pp. 293-296, Phoenix, AZ., March 1999.

[12]   Milner, B., "Inclusion of Temporal Information into Features for Speech Recognition," *Proc. ICSLP 96*, pp. 256-259, 1996.

[13]   Hanson, B. and Applebaum, T., "Robust Speaker-Independent Word Recognition Using Static, Dynamic and Acceleration Features: Experiments with Lombard and Noisy Speech," *Proc. ICASSP 90*, pp. 857-860, month 1990.

[14]   Lee, K. F., *Large Vocabulary Speaker-Independent Continuous Speech Recognition: The SPHINX System*, Ph.D. dissertation, Computer Science Department, Carnegie Mellon University, 1988.

[15]   Lee, K. and Hon, H., "Speaker-independent Phone Recognition using Hidden Markov Models," *IEEE Trans. ASSP*, vol. 37, no. 11, pp. 1641-1648, November 1988.

[16]   Zahorian, S. and Jagharghi, A. "Spectral-shape Features versus Formants as Acoustic Correlates for Vowels," *J. Acoust. Soc. Amer.*, vol. 94, pp 1966-1982, 1992.

[17]   Zahorian S. A., and Nossair Z. B., "A Partitioned Neural Network Approach for Vowel Classification Using Smoothed Time/Frequency Features," *IEEE Trans. Speech and Audio Processing*, vol. 7, pp. 414-425, July 1999.

[18]   Vaseghi, S., Harte, N. and Milner, B., "Multi-Resolution Phonetic/Segmental Features and Models for HMM-Based Speech Recognition," *Proc. ICASSP 97*, pp. 1263-1266, Munich, Germany, April 1997.

[19]   McCourt, P., Harte, N. and Vaseghi, S., "Multi-Resolution Cepstral Features for Phoneme Recognition Across Speech Sub-Bands," *Proc. ICASSP 98*, pp. 581-585, Seattle, WA, 1998.

[20]   McMahon, P., Harte, N., Vaseghi, S. and McCourt, P., "Discriminative Spectral-Temporal Multi-Resolution Features for Speech Recognition," *Proc. ICASSP 99*, pp. 581-585, Phoenix, AZ, March 1999.

[21]   Bourlard, H. and Dupont, S., "Subband-Based Speech Recognition," *Proc. ICASSP 97*, Munich, Germany, April 1997.

[22]   Gales, M. and Young, S., "Segmental Hidden Markov Models," *Proc. EuroSpeech 93*, pp. 1579-1582, 1993.

[23]   Russell, M., "A Segmental HMM for Speech Pattern Modeling," *Proc. ICASSP 93*, pp. 499-502, Minneapolis, MN, 1993.

[24]   Kapadia, K., Veltchev, V. and Young S. J., "MMI Training for Continuous Phoneme Recognition on the TIMIT Database," *Proc. ICASSP 93*, pg. 491-494. Minneapolis, MN, 1993.

[25]   Morgan, N. and Bourlard, H., "Continuous speech recognition using Multilayer Perceptrons with Hidden Markov Models," *Proc. ICASSP 90*, pp. 413-416, Albuquerque, NM, April 1990.

[26]   Morgan N. and Bourlard, H., "Continuous speech recognition," *IEEE Signal Processing Magazine*, Vol. 12, pp.25-42, May 1995.

[27]   Riis, S. and Krogh, A., "Hidden Neural Networks: A Framework for HMM/NN Hybrids," *Proc. ICASSP 97*, pp. 3233-3236, Munich, Germany, April 1997.

[28]   Glass, J., Chang, J. and McCandless, M., "A Probabilistic Framework for Feature-Based Speech Recognition," *Proc. ICSLP 96*, pp. 2277-2280, Philadelphia, PA, October 1996.

[29]   Digilakis, V., *Segment-based stochastic model of spectral dynamics for continuous speech recognition*, Ph.D. thesis, Boston University, 1992.

[30]   Goldenthal, W., *Statistical trajectory models for phonetic recognition*, Ph.D. dissertation, MIT, Cambridge, September 1994.

[31]   Ostendorf, M. and Roucos, S., "A Stochastic Segment Model for Phoneme-based Continuous Speech Recognition," *IEEE Trans. ASSP*, vol. 37, no. 12, pp. 1857-1869, December 1989.

[32]   Ostendorf, M., Kannan, A., Kimball, O. and Rohlicek, J., "Continuous Word Recognition based on the Stochastic Segment Model," *DARPAR Proc. On Continuous Speech Recognition Workshop*, September 1992.

[33]   Ostendorf, M., Bechwati, I. And Kimball, O., "Context Modeling with the Stochastic Segment Model," *Proc. ICASSP 92*, San Francisco, CA, March 1992.

[34]   Ostendorf,   M.,   Digalakis,   V.   and   Kimball,   O.   A.,
         "From   HMMs   to   Segment   Models: A   Unified   View   of
         Stochastic   Modeling   for   Speech   Recognition,"   *IEEE*
         Trans.   SAP,   vol.   4,   no.   5,   pp.   360-378,   September
         1996.

[35]   Hetherington,   I.,   Phillips,   M.,   Glass,   J.   and   Zue,
         V.,   "A*   Word   Network   Search   for   Continuous   Speech
         Recognition,"   *Proc.   Eurospeech   93*,   pp.   1533-1536,
         Berlin,   Germany,   1993.

[36]   Lee,   S.   and   Glass,   J.,   "Real-time   Probabilistic
         Segmentation   for   Segment-based   Speech   Recognition,"
         Proc.   ICSLP   98,   pp.   1803-1806,   Sydney,   Australia,
         1998.

[37]   Schuster,   M.,   "Acoustic   Model   Building   based   on   Non-
         uniform   Segments   and   Bidirectional   Recurrent   Neural
         Networks,"   *Proc.   ICASSP   97*,   pp.   3249-3252,   Munich,
         Germany,   April   1997.

[38]   Figueiredo,   F.   and   Violaro,   F.,   "An   Isolated   Word
         Speech   Recognition   System   Based   on   Kohonen   Neural
         Network,"   *Proc.   ICASSP   98*,   pp.   151-154,   Seattle,   WA,
         May   1998.

[39]   Janer,   L.,   Marti,   J.,   Nadeu,   C.   and   Lleida-Solano,
         E.,   "Wavelet   Transforms   for   Non-uniform   Speech
         Recognition   Systems,"   *Proc.ICSLP96*,   pp.   2348-2351,
         New   York,   NY,   October   1996.

[40]   Picone   J.,   "Signal   Modeling   Techniques   in   Speech
         Recognition,"   *IEEE   Proceedings*,   vol.   81,   pp.   1215-
         1247,   September   1993.

[41]   Rabiner,   L.   and   Schafer,   R.,   *Digital   Processing   of*
         Speech   Signals,   Prentice-Hall,   Englewood   Cliffs,   NJ,
         1978.

[42]   Markel, J. and Gray, A., *Linear Prediction on Speech*, Springer-Verlag, New York, NY, 1980.

[43]   Marple, S. Jr., *Digital Spectral Analysis With Applications*, Prentice-Hall, Englewood Cliffs, NJ, 1987.

[44]   Oppenheim, A. V. and Johnson, D. H., "Discrete Representation of Signals," *Proc. of the IEEE*, vol.60, no. 6, June 1972.

[45]   Davis S. B., and Mermelstein P., "Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-28, pp.357-366, 1980.

[46]   Kuwabara, H., "Temporal Effect on the Perception of Continuous Speech and a Possible Mechanism in the Human Auditory System," *Proc. Eurospeech 93*, pp. 713-717, Berlin, Germany, September 1993.

[47]   Paul, D., "The Lincoln Robust Continuous Speech Recognizer," *Proc. ICASSP 89*, pp 556-559, Glasgow, Scotland, May 1989.

[48]   Shirai, K., Hosaka, N., Kitakawa, E. and Endou, T., "Speaker Adaptable Phoneme Recognition Selecting Reliable Acoustic Features based on Mutual Information," *Proc. ICSLP 90*, pp. 353-356, Kobe, Japan, November 1990.

[49]   Wilpon, J., Lee, C. and Rabiner, L., "Application of Hidden Markov Models for Recognition of a Limited Set of Words in Unconstrained Speech," *Proc ICASSP 89*, pp. 254-257, Glasgow, Scotland, May 1989.

[50]   Zahorian, S. A., Silsbee, P. L., and Wang, X., "Phone Classification with Segmental Features and a Binary-Pair Partitioned Neural Network Classifier," *Proc. ICASSP 97*, pp. 1011-1014, Munich, Germany, April 1997.

[51]   Nossair, Z. and Zahorian, S., "Dynamic Spectral Shape Features as Acoustic Correlates for Initial Stop Consonants," *J. Acoust. Soc. Amer.*, vol. 89, pp. 2978-2991, 1991.

[52]   Parsons, T., *Voice and Speech Processing*, Mc-Graw Hill, New York, 1987.

[53]   Hunt, M. J., Bateman, D. C. and Richardson, S., "An Investigation of PLP and IMELDA Acoustic Representation and of Their Potential Combination," *Proc. ICASSP 91*, pp. 881-884, Toronto, Canada, May 1991.

[54]   Hunt, M. J., and Lefebvre, C., "A Comparison of Several Acoustic Representations for Speech Recognition with Degraded and Undegraded Speech," *Proc. ICASSP 89*, pp. 262-265, Glasgow, UK, May 1989.

[55]   Haeb-Umbach, R. and Ney, H., "Linear Discriminant Analysis for Improved Large Vocabulary Continuous Speech Recognition," *Proc. ICASSP 92*, pp. 13-16, San Francisco, CA, March 1992.

[56]   Aubert, X., Haeb-Umbach, R. and Ney, H., "Continuous Mixture densities and Linear Discriminant Analysis for Improved Context-Dependent Acoustic Models," *Proc. ICASSP 93*, pp. 648-651, Minneapolis, MN, 1993.

[57]   Eisele, T., Haeb-Umbach, R. and Langmann, D., "A Comparative Study of Linear Feature Transformation techniques for Automatic Speech Recognition," *Proc. ICSLP 96*, pp. 252-258, PA, October 1996

[58]   Haeb-Umbach, R., Geller, D. and Ney, H., "Improvements in Connected Digit Recognition Using Linear Discriminant Analysis and Mixture Densities," *Proc. ICASSP 93*, pp. 239-242, Minneapolis, MN, April 1993.

[59]   Zahorian, S., Qian, D. and Jagharghi, A., "Acoustic-phonetic Transformations for Improved Speaker-independent Isolated Word Recognition," *Proc. ICASSP 91*, pp. 561-564, Toronto, Canada, May 1991.

[60]   Sakoe, H. and Chiba, S., "Dynamic Programming Algorithm Optimization for Spoken Word Recognition," *IEEE Trans. ASSP*, vol. 26, no. 1, February 1978.

[61]   Lippmann, R., "An Introduction to Computing with Neural Nets," *IEEE ASSP magazine*, pp. 4-22, April 1987.

[62]   Baum, L., "An Inequality and Associated Maximization Technique in Statistical Estimation of Probabilistic Functions of a Markov Process," *Inequalities*, vol.3, pp. 1-8, 1972.

[63]   Jelinek, F., *Statistical Methods for Speech Recognition*, The MIT Press, Cambridge, Massachusetts, 1999.

[64]   Young, S. J., Odell, J., Ollason, D., Valtchev, V., and Woodland, P., "Hidden Markov Model Toolkit V2.1 reference manual," *Technical report*, Speech group, Cambridge University Engineering Department, March 1997.

[65]   Young, S. and Woodland, P., "State Clustering in HMM-Based Continuous Speech Recognition," *Compute Speech and Language*, vol. 8, pp. 369-383, October 1994.

[66]   Cole, R., Muthusamy, Y., and Fanty, M., "The ISOLET spoken letter database," *Tect. Rep.* 90-004, Oregon Graduate Inst., 1990.

[67]   Cole, R., Fanty, M., and Muthusamy, Y., "Speaker-Independent Recognition of Spoken English Letters," *Proc. Int. Joint Conf. Neural Networks*, vol.2, pp. 45-51, June 1990.

[68]   Loizou, P. C., and Spanias, A. S., "High-performance Alphabet Recognition," *IEEE Trans. Speech and Audio Processing*, vol. 4, pp. 430-445, August 1996.

[69]   Dermatas, E. S., Fakotakis, N. D., and Kokkinakis, G. K., "Fast Endpoint Detection Algorithm for Isolated Word Recognition In Office Environment," *Proc. ICASSP 91*, pp. 733-736, Toronto, Canada, 1991.

[70]   Lamel, L., Kasel, R. and Seneff, S., "Speech Database Development: Design and Analysis of the Acoustic-Phonetic Corpus," *Proc. of the DARPA Speech Recognition Workshop*, pp. 26-32, March 1987.

[71]   Jankowski, C., kalyanswamy, Basson, S. and Spitz, J., "NTIMIT: A Phonetically Balanced, Continuous Speech, Telephone Bandwidth Speech Database," *Proc. ICASSP 90*, pp. 109-112, Albuquerque, NM, April 1990.

[72]   Furui, S., "On the role of spectral transition for speech perception," *J. Acoust. Soc. Am. 80*, pp. 1016-1025, October 1986.

[73]   Duda, R. and Hart, P., *Pattern classification and scene analysis*, John Wiley and Sons, New York, NY, 1973.

[74]   Ming, J. and Smith, F., "Improved Phone Recognition Using Baysian Triphone Models," *Proc. ICASSP 98,* pp. 409-412, Seattle, WA, May 1998.

[75]   Lamel, L. and Gauvain, J., "High Performance Speaker-Independent phone Recognition Using CDHMM," *Proc. Eurospeech 93,* pp. 23-30, Berlin, Germany, September 1993.

[76]   Robinson, A., "An Application of Recurrent Nets to Phone Probability Estimation," *IRRR Trans. Neural Networks,* vol. 5, pp. 298-305, March 1994.

[77]   Deng, L. and Sameti, H., "Transitional Speech Units and their Representation by Regress Markov States: Application to Speech Recognition," *IEEE Trans. SAP,* vol. 4, pp. 301-306, 1996.

[78]   Mari, J., Fohr, D. and Junqua, J., "A Second-Order HMM for High Performance Word and Phoneme-Based Continuous Speech Recognition," *Proc. ICASSP 96,* pp.435-438, 1996.

[79]   Chang, J. W. and glass, J., "Segmentation and Modeling in Segment-based Recognition," *Proc. Eurospeech 97,* pp. 1199-1202, Rhodes, Greece, September 1997.

APPENDIX A

PHONE RECOGNITION EXPERIMENTS WITH HTK TOOLKIT

A.1 Monophones

A.1.1 Data Preparation

The following are things needed before training an HMM recognizer.

(1)   Prototype of each model (phone), generated using MakeProtoHMMSet perl script.

(2)   List of feature files for training

(3)   List of feature files for testing

(4)   MLF file of training set

(5)   MLF file of test set

(6)   List of all HMM models

(7)   Network lattice with bigram

(8)   Dictionary

A.1.2 Initialization

Initialize each model with **HInit** for 20 iterations using (1)(2)(4) and other options. Then initialize each model with **HRest** for 20 iterations using resulting model from **HInit** and (2) and (4). See HTK manual Section 8.2 and Section 8.4.

A.1.3 **Embedded Training**

Train all models in embedded mode with **HErest** for 5 iterations using resulting models from **HRest** and (2)(4) and (6).). See HTK manual Section 8.5.

A.1.4 **Recognition**

Use **HStats** to compute bigram statistics using (2) and (6) and use **HBuild** to create network lattice. See HTK manual Chapter 12.

Invoke **HVite** to find hypothesis phone transcription of each test sentence using results from **HErest** and from **HBuild** and also using (3)(6) and (7).). See HTK manual Chapter 13.

A.1.5 **Result Evaluation**

Use **HResults** to compute recognition accuracy using output transcription obtained with **HVite** and (5). Use -e option to ignore confusions between a pair of phones. See HTK manual Section 13.4.

A.1.6 **Sample Script**

The following is a sample script to train a monophone system.

--- Model initialization ---

HInit -A -T 1 -i 20 -I trnMLF -l ae -o ae -C hinit.conf -D -M hmms\hmm.0 proto\ae -S trnlist

HRest -A -T 1 -u tmvw -w 3 -v 0.05 -i 20 -I trnMLF -l ae -C hrest.conf -D -M hmms\hmm.1 hmms\hmm.0\ae -S trnlist

--- Compute bigram statistics ---

HLStats -A -D -b outmonbigrm monlist trnMLF

HBuild -A -D -m outmonbigrm monlistbigrm monnetworkbigrm

--- Recognition ---

HVite.exe -A -D -C hvite.conf -T 1 -H hmms\hmm.3\hmmdefs -S tstlist -l * -i tstrecbigrm -w monnetworkbigrm -p 0.0 -s 5.0 monvocabbigrm monlist

--- Evaluation ---

HResults -A -e si cl -e si vcl -e si epi -e el l -e en n -e sh zh -e ao aa -e ih ix -e ah ax -I tstMLF monlist tstrecbigrm

A.2 Right-Context Biphones

A.2.1 Data Preparation

In addition to monophone data preparation, there is some more information needed.

(9)   List of biphones

(10)   Biphone MLF file of training set

(11)   Biphone MLF file of test set

(12)   Biphone Network lattice with bigram

(13)   Biphone Dictionary

A.2.2 Cloning

Clone each biphone model using **HHed** tools with (6). See HTK manual Section 3.3.

A.2.3 **Embedded Training**

Train all models in embedded mode with **HErest** for 5 iterations using resulting models from **HHED** and (9) and (10). See HTK manual Section 8.5.

A.2.4 **Recognition**

Use **HStats** to compute bigram statistics using (2) and (9) and Use **HBuild** to create network lattice. . See HTK manual Chapter 12.

Invoke **HVite** to find hypothesis phone transcription of each test sentence using results from **HErest** and from **HBuild** and also using (3)(12) and (13). See HTK manual Chapter 13.

A.2.5 **Result Evaluation**

Use **HResults** to compute recognition accuracy using output transcription obtained with **HVite**. Use -e option to ignore confusions between a pair of phones. See HTK manual Section 13.4.

A.2.6 **Sample Script**

The following is a sample script to train a biphone system.

--- Model Cloning ---

HHed -A -B -D -C hhed.conf -T 1 -H hmm.16\hmmdefs -M hmm.17 makerc.hed monlist

--- Compute bigram statistics ---

HLStats -A -D -b outrcbigrm rclist trnMLF

HBuild -A -D -m outrcbigrm rclistbigrm rcnetworkbigrm

--- Recognition ---

HVite.exe -A -D -C hvite.conf -T 1 -H hmms\hmm.22\hmmdefs -S tstlist -l * -i tstrecbigrm -w rcnetworkbigrm -p 0.0 -s 5.0 rcvocabbigrm rclist

--- Evaluation ---

HResults -A -e si cl -e si vcl -e si epi -e el l -e en n -e sh zh -e ao aa -e ih ix -e ah ax -I tstMLF rclist tstrecbigrm

CURRICULUM VITA
For
MONTRI KARNJANADECHA

NAME: Montri Karnjanadecha

DATE OF BIRTH: April 28, 1968

DEGREES:

Master of Engineering (Electrical Engineering),
Prince of Songkla University, Songkla, Thailand,
August 1995

Bachelor of Engineering (Electrical Engineering),
Prince of Songkla University, Songkla, Thailand,
April 1990

PROFESSIONAL CHRONOLOGY:

Department of Computer Engineering, Prince of Songkla
University, Songkla, Thailand

Lecturer, April 1990 – Present