This is to certify that the

thesis entitled

THE WHITAKER DATABASE OF DYSARTHRIC SPEECH:
Creation and Baseline Recognition Study

presented by

Ming-Shou Liu

has been accepted towards fulfillment
of the requirements for

Master's degree in Electrical Engineering

John R. Deller Jr.
Major professor

Date 8 August 1991.

THE WHITAKER DATABASE

OF

DYSARTHRIC SPEECH:

Creation and Baseline Recognition Study

By

Ming-Shou Liu

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Department of Electrical Engineering

1991

ABSTRACT

THE WHITAKER DATABASE

OF

DYSARTHRIC SPEECH:

Creation and Baseline Recognition Study

By

Ming-Shou Liu

This research represents the culmination of a three year project sponsored by the Whitaker foundation of which the primary purpose was to conduct research related to the development of a PC-based isolated word recognition (IWR) system for persons with severe motor and speech disabilities. This dissertation describes three aspects of the final stages of the work:

1. The creation of an isolated word database of dysarthric speech (Whitaker Database (WD)) which is publicly accessible over the internet computer network.

2. A baseline recognition study on the WD using a hidden Markov model approach.

3. Formulation of an IWR system concept and plans for its development and future enhancements.

To my Mother

A-Chu Yang

For her love, support and sacrifice

ACKNOWLEDGMENTS

First and most important, I would like to thank my advisor John R. Deller, Jr. for his patience and support in spite of his busy schedule. His direction was very important in helping me step into the Speech Processing world.

Secondly, I would like to thank all the members of my thesis committee: Dr. B. Ho, Dr. Roland Zapp, Dr. Bon K. Sy of Queens College of the City University of New York, and Dr. John R. Deller, Jr. I would also like to thank Dr. Linda Ferrier of Northeastern University for her permission to use her report in Section 2.2.2 on the various dysarthric speakers.

Many recognition procedures and utilities coded by Ross K. Snider were helpful in the development of this thesis. I would also thank my friend Pei-Chun Chen for her great encouragement and support.

I gratefully acknowledge the financial support provided by a grant from the Whitaker Foundation and the collaboration with the Speech and Language Pathology and Audiology department at Northeastern University, Boston.

Contents

List of Tables

List of Figures

1 Introduction and Background

1.1 The Purpose and Significance of this Research

Many significant advances have been achieved in both speaker-dependent and speaker-independent speech recognition in the past three decades (see, e.g. [1, 2, 3, 4, 5, 6, 14, 15, 23, 31]). Most research, however, has been concerned with the recognition of normal speech. The difficult problem of applying speech recognition technology to assisting persons with speech disabilities to communicate effectively is still an open issue for researchers, as indicated by the small amount of literature on the topic and the relatively small number of systems available to users (e.g. ANTIC [10], CDC [11]).

The inability to speak and write can be caused by a number of neuromuscular diseases, such as cerebral palsy (CP), aphasia, amyotrophic lateral sclerosis (ALS), multiple sclerosis (MS), Parkinson's disease, muscular dystrophy, laryngectomy, and others [33]. In this study we have focused upon the CP population which comprises a significant proportion of the total population of profoundly speech and motor disabled persons. CP is a prevalent condition, present in approximate one of every 330 live births [32]. Anyone working with these people has observed that many individuals persistently try to express their needs and feelings vocally, even though many attempts may fail. However, due to the difficulty of controlling their articulator movements and voicing in uttering messages, it is frequently impossible for them to produce intelligible and fluent continuous speech. The goal of this study in a general sense is to adapt existing electronic technology to devices which will assist such persons to express ideas and feelings, to have normal social lives and interpersonal interactions, and to function in the mainstream of society.

1.2 Previous Work and Relation to the Current Project

This work represents the culmination of a three year research effort sponsored by the Whitaker Foundation of which one of the subgoals was to conduct research related to the development of a PC-based isolated word recognition (IWR) system for severely dysarthric speech. Previous work on this project has been reported in the papers of Sy, Hsu, Deller *et al.* [4, 5, 12, 30, 31]. In particular, Hsu's thesis research was concerned with the development (on a mainframe system) of hidden Markov model (HMM) [25] based IWR software, and its testing using a 200 word database collected from an moderately dysarthric (cerebral palsied) individual, and a digit (10 words) study involving four other persons whose speech spans a spectrum of dysarthria [7, 12].

The subsequent work of Snider [6, 28] and this author has generally been concerned with scaling down Hsu's software to operate on a reasonably ordinary personal computer (PC) in real time, and with extensive testing of the resulting algorithms. In this process, we have made a point of carefully collecting and organizing a large database of isolated word dysarthric speech (Whitaker Database (WD)) with which to test the system. The WD has been made publicly accessible to other research centers over the internet computer network. Whereas Snider's work was principally concerned with scaling and programming the PC-based software, and with developing sampling and editing software for manipulating the new data, this author has been chiefly concerned with the creation of the database, and with the testing and enhancement of the recognition software. The result has been the completion of enhanced, flexible PC-based IWR software which can now be tested "in the field" in conjunction with a system concept to be described.

Accordingly, this thesis consists of three parts which describe the three research components noted above:

1. Creation and distribution of the WD,

2. Execution of a baseline recognition study using HMM-based software, and

3. Refinement of the PC-based HMM IWR software for dysarthric speech, and development of plans and strategies for its distribution and testing.

We note that two specific engineering developments from previous work will be used in this thesis. They are an algorithm due to Deller and Hsu [4] and Deller and Snider's diagonalization strategy [6, 28]. The first implementation of the recognition software was developed by Hsu in his doctoral work [12]. A fast and simple adaptive Weighted Recursive Least Square (WRLS) algorithm was derived for the purpose of feature extraction at the acoustic level. This algorithm enjoys a small improvement in computational complexity over the conventional WRLS algorithm. The adaptive method also provides several useful by-products in the context of the recursion which the conventional one usually does not have [5, 8]. In the word-level processing, an enhanced HMM based approach was developed to operate under the constraints of having highly variable speech as well as a lack of statistical information about the speech.

The second engineering development from previous work is as follows: Given several HMM's and the observation sequence $\mathbf{O}$, we need to choose the word model which has the highest likelihood $P(\mathbf{O}|\mathbf{M})$ [25]. A frequently used algorithm to evaluate the HMM for maximum likelihood criterion is the Baum-Welsh "Forward-Backward" procedure [25]. The Forward-Backward procedure generally requires $\mathcal{O}(N^2)$ operations per observation for an N state, fully connected, HMM. Deller and Snider [6, 28] found that the number of calculations can be reduced to $\mathcal{O}(N)$ by diagonalizing the matrix[1] $\mathbf{A}$ in the HMM. All the evaluation work in this thesis is based on this *diagonalized* matrix.

[1] $A = \{a_{ij}\}$ is called the state transition probability distribution,

2 Collection and Creation of the Whitaker Database

2.1 Introduction to the Database

2.1.1 Data Acquisition System

The utterances were spoken by 6 speakers and recorded on TDK type II tape cassettes. A TEAC W-450R stereo cassette deck with Dolby-C noise reduction was used. The recording took place in the Department of Speech and Language Pathology and Audiology at Northeastern University in Boston and was supervised by Dr. Linda J. Ferrier, Assistant Professor in that department. All data were recorded in an acoustically isolated booth.

The recordings were played back using a duplicate TEAC tape deck and then sampled in the Speech Processing Laboratory in the Department of Electrical Engineering at Michigan State University. The MetraByte "STREAMER" data acquisition system was used to facilitate the sampling. The equipment setup for sampling is shown in Fig. 1. The filter used is an active bandpass[2] fourth order Butterworth filter with a lowpass cutoff frequency of 4.7kHz (the sample rate is 10kHz). The frequency response of the filter is shown in Fig. 2. A MetraByte DAS16F 12-bit analog to digital (A/D) conversion board was set to accept a signal with dynamic range of ±10 volts. To make certain that the input to the A/D board did not exceed the dynamic range of the board, the input signal was monitored with an oscilloscope and the gain of the amplifier adjusted appropriately. Data are stored in 16 bit records, one per sample. Encoded in the 16 bit record are 12 bits of measurement data and 4 bits that specify the channel.

[2]This filter is effectively *lowpass* for speech which rarely contains significant energy below about 75 Hz.

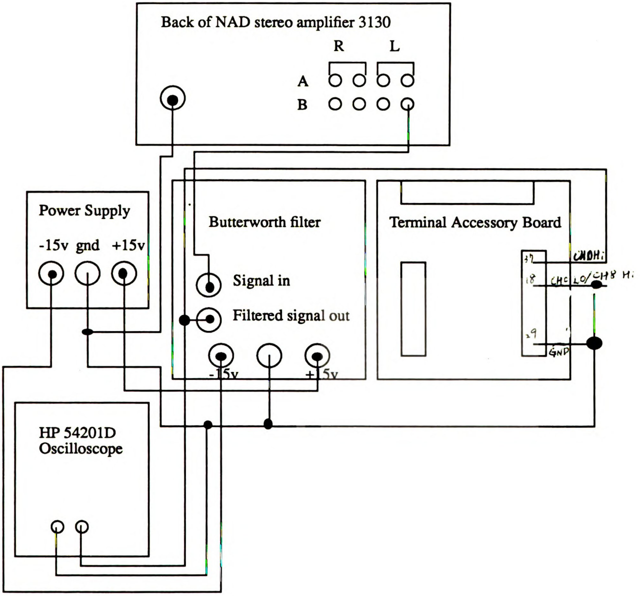

Figure 1: Equipment setup for sampling.

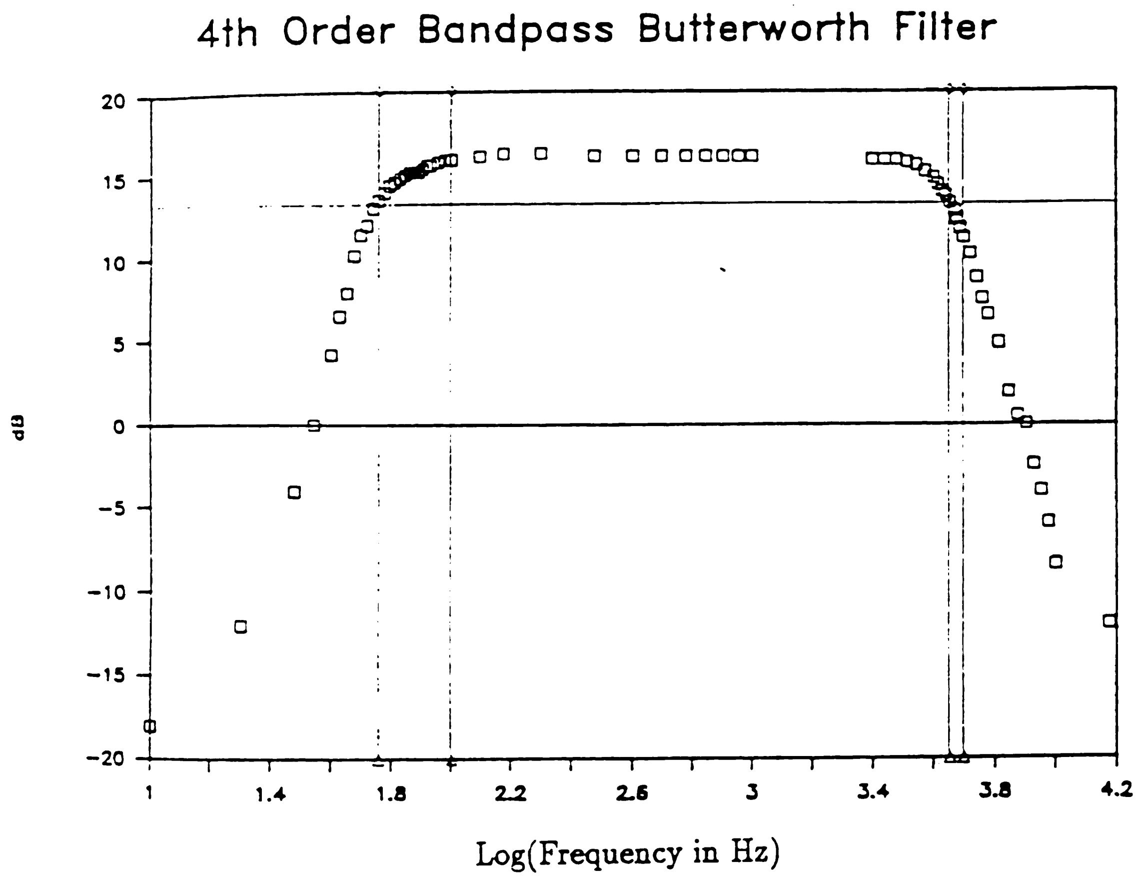

Figure 2: Frequency response of anti-aliasing filter.

At the beginning of the project, the sampling process was carried out word by word. That is, we located the word on the audio tape, made a file for it and then sampled the word. It took about 90 seconds per word to complete this task. This method is time-consuming and unrealistic because 17,895 words needed to be processed. In order to solve this problem, Snider wrote a program called "Wavemark". With this routine, utterances from an entire cassette tape can be sampled then stored in a large file (about 30 Mbytes). "Wavemark" can then also be used to extract the words from the large file with a screen editing facility [28]. This procedure reduces the per word processing time by a factor of about 10. The program also has a provision for playing back (audio) any selected portion of an utterance. Details are found in [29].

2.1.2 Composition of the Whitaker Database (WD)

The word sets in the WD are partitioned into the *TI-46* word list and *Grandfather* word list. These word list were selected for the WD to provide one partition of vocabulary which has become "standard" in speech recognition studies, and one which is significant for its speech science attributes.

There are 46 words in the TI-46 word list. They are utterances of the 26 letters of the alphabet, 10 digits (zero to nine) and 10 the words "start", "stop", "yes", "no", "go", "help", "erase", "rubout", "repeat" and "enter". This word list is suggested as a standard by Texas Instruments [9]. The Grandfather word list consists of 35 words which are shown in Table 1. The set is called "Grandfather" because it was taken from a passage commonly used by speech therapists which begins with the sentence "Let me tell you about my grandfather ...". These words are chosen by Dr. Ferrier due to their phonetic diversity [13].

There are 27 cassette tapes in the Speech Processing Laboratory. Each word in the TI-46 and Grandfather word list was uttered 30 to 45 times by one of the six speakers.

missing	several	to	well	thinks	long
my	old	you	ever	an	frock
coat	usually	still	he	dresses	about
years	is	wish	know	himself	buttons
all	grandfather	as	swiftly	black	beard
in	yet	nearly	clings	ninety-three	

Table 1: Grandfather word list.

Each utterance of each word ultimately became a distinct file. The collection of all the files sampled from these tapes comprise the *Whitaker Database* (WD). Each file consists of integer samples with dynamic range from -2048 to +2048. All the files are ASCII with <CR><LF> after each integer.

2.2 Summary

2.2.1 Characteristics of the Whitaker Database

- The vocabulary sets in the WD are the TI-46 and Grandfather word lists indicated in Table 1.
- There are 17,895 ASCII files in the database, each file represents an utterance of a single word. The end points of each word were detected by hand using the "wavemark" utility described above.
- Each sample point in each file is represented by an integer and is followed by <CR><LF>.
- The dynamic range of the integer samples is from -2048 to +2048.

2.2.2 Characteristics of the Speakers

The following clinical assessments of the speakers are taken from a report by Dr. Linda J. Ferrier, Assistant Professor of Speech and Language Pathology and Audiology, Northeastern University, Boston. Dr. Ferrier is the clinical consultant to this project. She also received Whitaker funding to support her interaction with the subject population, analysis of data from a clinical perspective, and writing clinical assessments of the subjects' speech and language disorders. The author appreciates Dr. Ferrier's permission to use the following descriptions[3]:

1. Speaker DC is a 48-year-old male with a diagnosis of spastic athetoid cerebral palsy (CP). His intelligibility is mildly impaired, and his voice has a typical strained-strangled quality and consonants are imprecise.

2. Speaker CJ is a 41-year-old male with a diagnosis of athetoid CP. His intelligibility is moderately impaired but decreases with fatigue. Speech characteristics include slow rate of speech, imprecise consonants, vowel distortions, and little variation in pitch or loudness. He is consistently over-loud. Vowel distortions appear to be caused by deviation of the mandible to the left.

3. Speaker LE[4] is a 40-year-old male with spastic CP with dysarthria. His intelligibility is in the moderate to severely impaired range, speech is slow with little variation in loudness or pitch. He has particular difficulty with transition from one consonant to the next in consonant clusters. and some difficulty initiating sounds and dysfluencies often occur at the beginning of the words.

4. Speaker BD is a 39-year-old male with spastic CP. His intelligibility is mildly to moderately impaired, and his voice shows occasional pitch breaks, inappropriate nasality, and he is occasionally dysfluent. He has poor breath support for speech. His amplitude is low and there is little variation in pitch or loudness.

---

[3]Our subjects all have some form of cerebral palsy, but there is nothing specific to this disorder in our work.

[4]LE is the main speaker in Hsu's previous work [12].

5. Speaker LL is a 47-year-old male with quadriplegic CP, mixed spastic/ataxic. His intelligibility is severely impaired, utterances are short, consonants are imprecise.

6. Speaker PW is a 28-year-old male with severe athetoid CP. His intelligibility is severely impaired, consonants and vowels are extremely distorted, and loudness is extremely variable.

2.3 How To Access the Database

The Whitaker database is accessible through the internet computer network[5]. The database can be obtained from a MSU file server through *anonymous* ftp. The database is in the subdirectory "speech" under the directory "pub". Six subdirectories "DC", "CJ", "LE", "BD", "LL", "PW" are in the database, the directory structure is shown in Fig. 3. The file naming convention is as follows: "coat.0502" means this word is the fifth utterance of the word "coat". The last two digits in the file name are used for internal grouping, and the user may ignore them. Files are compressed so that they will not require excessive space. All the files (utterances) uttered by a single speaker are "tarred" together so that only one instruction can be used to obtain all the files in a tape. The name of the "tarred" files are "t.tar" and "g.tar", where "t" means the TI-46 word list and "g" means the Grandfather word list.

In summary, the steps for accessing the WD are as follows:

1. **ftp archive.egr.msu.edu**, both the login name and password are *anonymous*

[5]The method of accessing the speech data is subject to change due to periodic changes in the computing facilities. Please contact the author by e-mail if there is any problem in accessing the data. Electronic mail addresses are: lium@frith.egr.msu.edu or deller@eecae.ee.msu.edu

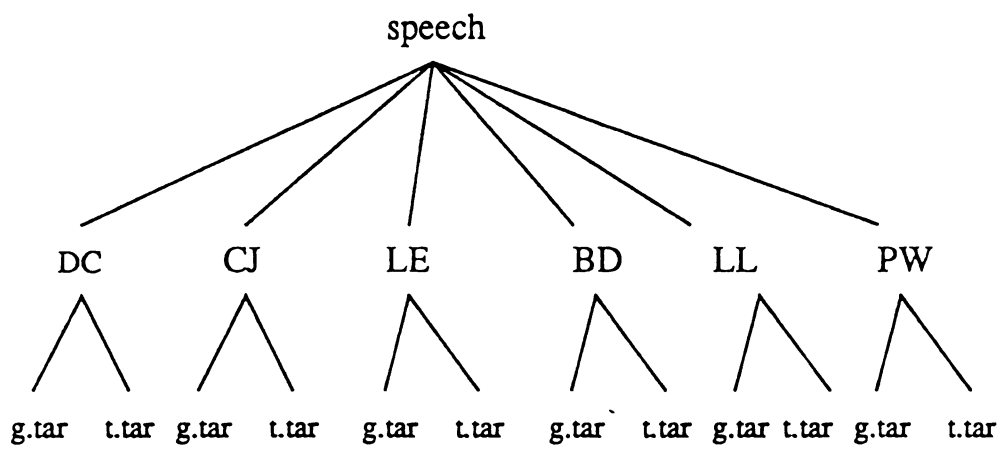

Figure 3: Directory structure of Whitaker Database in the computer network.

2. **cd pub**

3. **cd speech**

4. **cd DC** if you are interested in the speaker DC.

5. **get t.tar** if you are interested in TI-46 database.

6. **tar xfv t.tar**, this is to "untar" the file. "x" means *extract.*

3 Technical Description of the System

A wide variety of approaches to the recognition of human speech has been proposed in the past three decades. In this chapter, we briefly describe the techniques which were applied in this research. Details of the underlying technical methods can be found in many references (e.g. see [3]) and are not further addressed here.

3.1 Feature Extraction

To extract a feature one has to look at a small segment or frame of speech. We define a frame of speech to be the product of a shifted window with the speech sequence. For a sample rate of 10kHz, we use a Hamming window of length 256, which is shifted 50 samples for each feature computation.

Two types of vector features are employed in this work:

Linear prediction (LP) [16, 19] has been applied extensively in parameterizing speech samples. Here we use 14^{th} order LP parameters resulting from the autocorrelation method, where L-D recursion [3, 22, 26] is used to solve the autocorrelation equation. A computer program which computes the LP parameters is found in Appendix B.

The other method applied to parameterize the waveform is mel-cepstral analysis. [3, 21]. We use a 10^{th} order cepstrum produced using a 1024 point FFT [24]. A computer program for this approach is found in Appendix C.

3.2 Vector Quantization (VQ)

The recognition approach taken here is based on discrete symbol hidden Markov models (HMM) of isolated words. Accordingly the observations used are discrete symbols chosen from a finite set. A vector quantizer is required to map each continuous parameter vector into a finite integer index [17, 25].

Two distance measures are used to measure feature similarities in the VQ process. In the LP case, we use the Itakura's distance [26]:

$$d(\hat{\mathbf{a}}, \mathbf{a}) = \log \left[\frac{\mathbf{a}\mathbf{R}\mathbf{a}^t}{\hat{\mathbf{a}}\mathbf{R}\hat{\mathbf{a}}^t} \right] \tag{1}$$

Where $\mathbf{a}$ is a reference LP vector[6] and $\hat{\mathbf{a}}$ is an estimated LP vector[7].

Unlike LP coefficients, the cepstral parameters may be interpreted as coefficients of a Fourier series expansion of the periodic log spectrum. Accordingly, they are based on a set of orthonormal functions; thus we can simply choose the Euclidean distance between mel-cepstral vectors as the distance measure [3].

An 128 symbol codebook was developed using the κ-means algorithm. In our work, we employ the binary clustering approach, i.e., we let $\kappa = 2$. A computer program to geneate the 128 symbol codebook is found in Appendix D. The binary structure has the advantage that it reduces the number of searches from L to $\log_2 L$ [12, 17], where L is the number of symbols in the codebook.

3.3 The Hidden Markov Model (HMM)

The hidden Markov model (HMM) has been used in automatic speech recognition successfully in recent years for modeling speech waveforms [4, 7, 12, 14, 15, 23, 28] at various acoustic levels (word and subword) as well as for modeling languages. Some computationally efficient algorithms have been developed in the previous work by Snider to evaluate the likehood of the HMM. A major advantage of using an HMM in the problem of dysarthric speech recognition is that the HMM is a stochastic

[6]In the present case, an entry in the codebook.

[7]In the present case, derived from a frame of speech.

modeling approach which can automatically handle the "large" variability in speech for recognition purposes.

4 Speech Recognition Experiments

In this chapter, we focus on a baseline recognition study on the WD using a hidden Markov modeling approach in an effort to learn more about the characteristic features of dysarthric speech which affect recognition performance. If not specially stated, we use eight utterances for training, seven for testing, a 128 symbol codebook, and six state Bakis HMMs. The percentage given is the ratio of the number of correctly recognized words to the total number of testing words. An example comprehensive experimental result for speaker DC is found in Appendix A.

4.1 Size of the Codebook

Since the recognition system is based on discrete symbol HMMs of isolated words, a vector quantizer is required to map each continuous parameter vector into a finite integer index. The number of indices (code vectors) used should correspond to the number of meaningful clusters in the feature vectors in the population. Very roughly speaking, these codes (clusters) represent distinct acoustic tokens. If too few are used, many dissimilar features will be quantized into the same token. If too many are used, superfluous and ambiguous codes exist. Either situation potentially degrades performance. With normal speech typically 64–128 codes provide good performance for speaker independent recognition. The following experiments were implemented to determine whether fewer codes would improve recognition of dysarthric speech, under the hypothesis that fewer acoustic tokens may exist in some speakers' utterances.

Experimental results given in Table 2 for the TI-46 database and Table 3 for the Grandfather database show the effects of different size codebooks on the recognition rate. A quick glance at the recognition rate would seem to indicate that a larger codebook is better. Closer inspection reveals that this is not always the case. Large

	16 symbol			128 symbol		
	correct	top 2	top 4	correct	top 2	top 4
Speaker DC	72.05%	82.92%	90.68%	89.13%	95.96%	98.45%
Speaker CJ	81.06%	92.24%	97.52%	81.99%	92.24%	95.96%
Speaker LE	58.39%	70.50%	84.16%	68.94%	81.06%	90.37%
Speaker BD	76.09%	88.51%	95.65%	77.02%	90.68%	96.27%
Speaker LL	47.21%	65.53%	80.43%	56.83%	73.60%	83.85%

Table 2: Results of different codebook sizes using the TI-46 database.

	16 symbol			128 symbol		
	correct	top 2	top 4	correct	top 2	top 4
Speaker DC	90.61%	98.78%	98.78%	91.43%	97.55%	99.59%
Speaker CJ	61.63%	84.75%	91.02%	86.53%	95.51%	97.55%
Speaker LE	73.06%	84.49%	93.06%	74.29%	83.67%	93.06%
Speaker BD	68.98%	82.04%	93.06%	79.18%	90.61%	93.47%
Speaker LL	57.55%	76.33%	84.90%	60.00%	74.29%	85.31%

Table 3: Results of different codebook sizes using the Grandfather database.

codebooks do not work as well for the discriminating vowel sounds. For example, the recognition of the dipthong /ai/ (utterances of letter "a") for small codebooks (16 symbols) is better than that for a large codebook (128 symbols), but a larger codebook is necessary for the fricatives. For example, the utterance /pi/ (letter "p") is frequently recognized as /i/ (letter "e") if the 16 symbol codebook is used.

The reason why a large codebook does not work for the vowel case is generally explained as follows: dysarthric speakers have difficulty in controlling their articulators, and multiple symbols in the codebook which are close "acoustically" can accordingly represent the same vowel sound. Increasing the number of symbols will not increase the recognition rate. In fact, as noted above, too many symbols made degrade performance. However, fewer symbols do not provide the acoustic diversity necessary to represent frictives, for example.

	ergodic model			left-to-right model		
	correct	top 2	top 4	correct	top 2	top 4
Speaker DC	84.47%	92.86%	97.51%	89.13%	95.96%	98.45%
Speaker CJ	80.75%	90.68%	94.10%	81.99%	92.24%	95.96%
Speaker LE	64.60%	76.09%	89.13%	68.94%	81.06%	90.37%
Speaker BD	70.81%	83.54%	92.86%	77.02%	90.68%	96.27%
Speaker LL	55.90%	68.94%	81.37%	56.83%	73.60%	83.85%

Table 4: Recognition performance with different models using the TI-46 database.

4.2 HMM Structure

One of the important factors that was found to greatly affect the recognition rate is the HMM model structure. In this study, two types of model structure were considered, the "ergodic" and "left-to-right (Bakis)" model [25]. For IWR in which (at least) one HMM is designated for each word in the vocabulary, it should be clear that a left-to-right model is more appropriate than an ergodic model, since time and model states are associated in a natural manner [25]. In addition to the property that the state transitions always occur from left to right in the Bakis model, an additional constraint is placed on the state transition coefficients to make sure that "large" changes in state indices do not occur. That is, $a_{ij} = 0$ if $j > \Delta$. In our system, we take $\Delta = 2$. Experimental results in Table 4 and Table 5 also show that the Bakis model yields better performance than the ergodic model.

This result is contrary to Hsu's findings. Hsu found the ergodic structure to be slightly preferable to the Bakis structure [7, 12]. His results, however, were based on a digit database collected from speaker LE. In fact, if we examine only the recognition rate of digits for speaker LE, ergodic structure and Bakis structure produce the same recognition rate in this study as well. Thus, we could conclude that although the Bakis model is intuitively more appropriate for normal speech, the choice of Bakis vs. ergodic model in the dysarthric case may be vocabulary and speaker-dependent.

	ergodic model			left-to-right model		
	correct	top 2	top 4	correct	top 2	top 4
Speaker DC	88.57%	95.92%	97.55%	91.43%	97.55%	99.59%
Speaker CJ	82.04%	90.61%	95.51%	86.53%	95.51%	97.55%
Speaker LE	71.02%	81.22%	89.39%	74.29%	83.67%	93.06%
Speaker BD	68.57%	84.08%	93.06%	79.18%	90.61%	93.47%
Speaker LL	55.10%	70.61%	81.22%	60.00%	74.29%	85.31%

Table 5: Recognition performance with different models using the Grandfather database.

4.3 Acoustic Parameterization

The choice of parametric vector representation for the acoustic waveform is an important factor in automatic speech recognition. We have used weighted recursive least squares (WRLS) estimation (with weights chosen to implement a forgetting factor [4, 12]) and autocorrelation methods to compute LP parameters, and cepstral analysis. Results in Table 7 show that there is no significant difference between the WRLS and autocorrelation LP methods. In this experiment we use five utterances for training and five for testing. The words which were used to compare the WRLS and autocorrelation methods are shown in Table 6 which consists of 40 words spoken by speaker LE. These words are a subset of a 200 word database which is reported in the Ph.D. dissertation of Hsu[8] [12]. From Table 8 and Table 9, we see that the experiments show that the mel-based cepstrum significantly improves the performance with respect to the LP case. This result is consistent with a finding of Davis and Mermelstein [2] on IWR of normal speech[9].

[8]Except for the window employed, the WRLS and autocorrelation methods are nearly equivalent procedures. These experiments were performed ***prior to the creation of the WD*** as a quick check of the expected similarity of performance between the two methods.

[9]The result of Davis and Mermelstein was based on dynamic time warping

a	american	about	becomes
bicycle	calculus	child	doesnt
drink	enough	existed	from
father	gauge	go	has
home	in	just	knows
landmark	muscle	movies	notion
never	old	opinion	paycheck
problem	question	rattle	shaky
sounds	topic	today	tell
usually	vote	who	with

Table 6: Vocabulary for the comparison of WRLS and autocorrelation methods of LP parameter computation. These words are not in the WD for reasons explained in the text.

	ergodic model			Bakis model		
	correct	top 2	top 4	correct	top 2	top 4
WRLS	54.00%	65.00%	74.00%	59.50%	67.00%	77.00%
autocorrelation	56.00%	68.00%	74.00%	59.00%	71.00%	80.50%

Table 7: Recognition results comparing WRLS and autocorrelation methods of computing LP parameters.

	LP			Mel-Cepstrum		
	correct	top 2	top 4	correct	top 2	top 4
Speaker DC	84.78%	93.79%	98.14%	89.13%	95.96%	98.45%
Speaker CJ	78.88%	89.44%	94.10%	81.99%	92.24%	95.96%
Speaker LE	59.63%	72.05%	84.47%	68.94%	81.06%	90.37%
Speaker BD	74.53%	86.34%	97.20%	77.02%	90.68%	96.27%
Speaker LL	43.48%	56.52%	72.67%	56.83%	73.60%	83.85%

Table 8: Recognition results comparing LP and mel-cepstrum using the TI-46 database.

	LP			Mel-Cepstrum		
	correct	top 2	top 4	correct	top 2	top 4
Speaker DC	89.39%	95.92%	98.37%	91.43%	97.55%	99.59%
Speaker CJ	73.47%	85.71%	93.06%	86.53%	95.51%	97.55%
Speaker LE	67.35%	79.59%	89.39%	74.29%	83.67%	93.06%
Speaker BD	74.29%	84.90%	91.43%	79.61%	90.61%	93.47%
Speaker LL	57.96%	71.84%	86.12%	60.00%	74.29%	85.31%

Table 9: Recognition results comparing LP and mel-cepstrum using the Grandfather database.

4.4 Silent Portion Extraction

For many dysarthric speakers, "steady state" vowel-like phonemes are the easiest sounds to produce because they do not require dynamic movement of the vocal system. Conversely, phonetic transitions in speech are more difficult to produce for dysarthric individuals because they require fine muscle control to move the articulators. Many dysarthric individuals are not able to consistently and reliably make such transitions between two phonemes due to lack of muscle control. Consequently, it is reasonable to assume that acoustic transitions in dysarthric speech are of much larger variance than stationary regions. Hsu tested this hypothesis by pursuing a method to clip out the dynamic regions from the speech in order to decrease the variability. These experiments revealed significant performance improvement as a result of this procedure [7, 12].

Early experiments conducted during Snider's work [28] suggested that this clipping procedure might have been effective principally because it was removing short silent regions from the acoustics. To test this hypothesis, a silence detection strategy based on the zero-crossing and energy thresholds was employed to remove short silent regions. The thresholds were carefully selected so that the technique would extract only silence regions without removing the weak frictives and other low-amplitude portions of the speech. However, most experiments reported in Table 10 and Table 11 do not support Snider's hypothesis. These results suggest that silence portion extraction algorithm does not benefit the system performance and Hsu's improvement from the clipping procedure is apparently not due to silence extraction alone as Snider suspected.

	Silent portion removed			Silent portion kept		
	correct	top 2	top 4	correct	top 2	top 4
Speaker DC	86.65%	95.34%	98.14%	89.13%	95.96%	98.45%
Speaker CJ	82.92%	91.30%	97.20%	81.99%	92.24%	95.96%
Speaker LE	68.32%	81.06%	91.93%	68.94%	81.06%	90.37%
Speaker BD	74.22%	88.51%	95.96%	77.02%	90.68%	96.27%
Speaker LL	53.73%	64.91%	79.19%	56.83%	73.60%	83.85%

Table 10: Effect of silent portion extraction on recognition performance using the TI-46 database.

	Silent portion removed			Silent portion kept		
	correct	top 2	top 4	correct	top 2	top 4
Speaker DC	91.43%	96.33%	98.78%	91.43%	97.55%	99.59%
Speaker CJ	84.90%	93.88%	97.55%	86.53%	95.51%	97.55%
Speaker LE	72.65%	83.67%	90.61%	74.29%	83.67%	93.06%
Speaker BD	75.10%	88.16%	95.10%	79.18%	90.61%	93.47%
Speaker LL	60.00%	71.43%	83.27%	60.00%	74.29%	85.31%

Table 11: Effect of silent portion extraction on recognition performance using the Grandfather database.

4.5 Number of Training Utterances

Training of each HMM was based on the Baum-Welch reestimation procedure for multiple observation sequences [25]. The problem of having little training data with which to accurately characterize the statistical distributions in the HMM, which is common to most HMM training problems, is extraordinary in the dysarthric speech problem. The experimental results in Table 12 and Table 13 show that the number of training sequences has a significant effect on the recognition rate. However, the number of observation sequences used for training is limited, since any attempt to collect large bodies of speech data by lengthy recording sessions is impractical. Such sessions are mentally and physically fatiguing for many persons, a fact which only contributes to the variability one is trying to characterize by collecting more data. In order to get the best performance from the system, we suggest a *retraining* strategy.

	5 observation sequences			8 observation sequences		
	correct	top 2	top 4	correct	top 2	top 4
Speaker DC	81.99%	91.61%	95.03%	89.13%	95.96%	98.45%
Speaker CJ	79.81%	89.13%	94.41%	81.99%	92.24%	95.96%
Speaker LE	62.11%	75.16%	87.27%	68.94%	81.06%	90.37%
Speaker BD	69.25%	82.61%	91.61%	77.02%	90.68%	96.27%
Speaker LL	47.83%	61.49%	77.33%	56.83%	73.60%	83.85%

Table 12: Effect of number of training observation sequences on recognition using the TI-46 database.

	5 observation sequences			8 observation sequences		
	correct	top 2	top 4	correct	top 2	top 4
Speaker DC	87.76%	94.29%	97.55%	91.43%	97.55%	99.59%
Speaker CJ	74.69%	89.39%	95.92%	86.53%	95.51%	97.55%
Speaker LE	60.41%	73.47%	85.71%	74.29%	83.67%	93.06%
Speaker BD	68.57%	80.82%	89.80%	79.18%	90.61%	93.47%
Speaker LL	48.57%	64.49%	80.82%	60.00%	74.29%	85.31%

Table 13: Effect of number of training observation sequences on recognition using the Grandfather database.

Whenever the recognition is incorrect or correct but the likelihood of the recognized word is not sufficiently different from that of other candidates, we retrain the model by incorporating the new observations into the existing HMM.

4.6 Number of States in the HMM

It is clear that the Markov structure cannot correctly reflect the temporal speech waveform unless enough states are involved. One idea is to let the number of states correspond roughly to the number of phonemes within words, hence models with two to 10 states would be appropriate [25]. For computational efficiency, however, including fewer states is favorable.

The experimental results show that for short words, especially single syllable words, using fewer states results in better performance. This is consistent with the

	6 states	8 states	10 states
Speaker DC	89.13%	90.99%	88.51%
Speaker CJ	81.99%	83.85%	84.78%
Speaker LE	68.94%	70.50%	68.94%
Speaker BD	77.02%	77.95%	75.47%
Speaker LL	56.83%	57.14%	57.45%

Table 14: Effect of number of states in HMM on recognition performance using the TI-46 database.

	6 states	8 states	10 states
Speaker DC	91.43%	92.24%	92.65%
Speaker CJ	86.53%	84.49%	88.16%
Speaker LE	74.29%	72.65%	75.92%
Speaker BD	79.18%	76.73%	77.14%
Speaker LL	60.00%	60.41%	56.73%

Table 15: Effect of number of states in HMM on recognition performance using the Grandfather database.

assumption that the number of states roughly reflects the number of phonemes within words. Since the TI-46 word list contains 26 alphabetic characters and 10 digits (most of which are short words), the effect of increasing the number of states in using TI-46 is not as obvious as that in using the Grandfather word list. Note, however, that for Speaker LL who routinely produces short sounds, the use of fewer states results in better performance consistent with expectation.

5 The Prototype IWR System for Dysarthric Speech

The long term goal of this research is the development of an "artificially intelligent" communication aid to serve the needs of a person who is severely speech disabled and whose motor skills will only permit simple responses in answering "interrogations" by the device.

In research related to the speech recognition function of such a device, experiments with the dysarthric speech database yielded results which were highly sensitive to many analysis parameters, in particular, the settings of Hsu's transition clipping procedure [12]. Whereas Hsu's hypothesis was that the clipping procedure was effective because it removed transitional acoustics from the observation sequence [7, 12], preliminary experiments conducted during Snider's work [28] suggested that the clipping procedure might be beneficial principally because it was removing "gaps" or short silent regions from the acoustics. In most of the experiments in Section 4.4, however, discarding the silent regions is seen to cause a decrement in performance, though this is not generally true. These results significantly affected our thinking about the proper course of action in the development of the communication aid.

The conclusion from these mixed results is that building a "fixed box" for all the speech disabled individuals is not possible nor appropriate because the choice of parameters to improve the recognition rate is highly speaker-dependent. Our future plan is to cooperate with clinical centers in the development of *customized* systems for a few selected speech-disabled clients with "small" task requirements, for example, issuing a small set of verbal commands to an assistive device. "Customized" means that the inclusion of specific modules and parameter-choices in the system will be based on the needs and speech characteristics of the client. The clinical center will transmit digitized speech data over the electronic mail service (email) on the computer network to the Speech Processing Laboratory. These data along with knowledge of the needs of the clients will be used to create a customized recognition system (software) which will then be returned to the clinical center over the network. Periodic updates (adaptation) of the software can be accomplished by the same means, particularly if the system is designed to record information about recognition errors and representative confused utterances. Such adaptation can also be achieved "on-line" if the system is apropriately designed. In parallel, of course, an opportunity exists for further research and development as we gain experience from this endeavor.

In this research, we have developed a fundamental speech recognition software module which, in keeping with the basic philosophy expressed above, remains flexible for user-specific customization. In addition, the "front" and "back" ends of the device remain unspecified, to be customized for individual users. For example, the basic operation could be as follows: 1) The user hits one key first, utters the words vocally, and then hits the key again to indicate the end of the utterance. 2) The software will then process the incoming speech by coding the speech signal, quantizing, and computing the probability for each prestored model. 3) Finally, the software presents a list of probable words in the decreasing order of their likelihood measure for the user to affirm, to deny, or from which to make a selection (see [30, 31]).

One relatively straightforward technical problem remains in the development of a complete prototype recognition system. The system is running on a general-purpose PC, and thus, the speed of the recognition is limited. Even on a "high-end" PC based on an Intel 80486 microprocessor with math coprocessor support, it takes about 15 seconds, for example, to recognize a word from a 46 word vocabulary (TI-46) with the current system. In order to achieve the real-time speech recognition system, we can include Snider's *compression* approach [6, 28] to reduce the computational complexity, or employ a programmable signal processing board to maximize the speed, to achieve real-time operation.

The performance of this prototype system depends on numerous inter-related factors. Although our approach can easily be adjusted to adapt to different dysarthric cases and maximize the performance, further study of several approaches to enhance the recognition system is in progress and will be discussed in the next section.

6 Conclusion

6.1 Summary

Recognition of the speech of severely dysarthric individuals requires a technique which is robust to extraordinarily high variability and very little training data. Many experimental results show that the recognition of dysarthric speech is a distinctly different problem from that of normal speech, and new strategies and approaches will be needed. Because the personal needs and the degree of dysarthria of the speakers are different, this effort has suggested that a flexible system in which system parameters can be selected on an individual basis is preferable to a "fixed" system.

The principal contributions of the this research are:

1. The creation of the "Whitaker Database": The WD provides a well-organized speech data set which is accessible over the internet computer network. The words in the database were carefully selected for their phonetic richness and complexity. It is hoped that this database will serve as a standard for researchers around the world with which many systems can be compared and meaningfully evaluated.

2. Extensive experimental studies on the WD to determine effects of various recognition parameters and strategies on performance. These studies resulted in the conclusion that "customization" of the recognition system to individual speakers is the proper design philosophy. This, in turn led to the conception of an "on-line" development and testing paradigm to be employed in cooperation with clinical centers in future work.

6.2 Future Work

In view of the current research on IWR for dysarthric speech, several issues which should be addressed in future work have been identified.

First, collection of speech data by lengthy recording sessions is a stressful experience for many dysarthric speakers, and resulting mental and physical fatigue and frustration introduce more variability. Consequently, training data are severely limited. We have suggested a "retraining" strategy as an area of future work in Section 4.5. The recognizer must have a convenient way to let the speaker identify the correctness of the recognized word and decide if the retraining process is required. Of course, the system should have the ability to decide automatically whether or not the retraining process is required when the recognition is correct in order to make it more robust.

Second, it is common for some speakers to introduce unnatural and irregular pauses within words. We introduced the idea of building a silent model as in Section 4.4. In this case, the incoming speech is represent as an arbitrary sequence of phone and silent models:

$$\text{signal} = (\text{silent}) \text{ - phone - } (\text{silent}) \text{ - phone - } (\text{ silent})$$

where the silent part is optional and may appear in general between any two phones in the signal. A similar strategy was proposed by Levinson in an HMM-based level-building connected word recognition system as a means for accounting for inter-word silence [15]. The significant benefits from this algorithm are: 1) It can be used to automate the end-point detection process, and 2) It avoids removing the transition information of dysarthric speech which can occur if silence regions are removed using conventional silence detection approaches [27].

Third, recognition based on sub-word (e.g. phoneme) modeling would alleviate some of the problems encountered in collecting sufficient training data. In fact, such an approach might be a natural solution for some speakers who tend to use only a small number of phones. A natural extension of this idea would be to incorporate a grammar and begin recognition of "continuous" speech or at least isolated word sentence or phrase utterances. While the use of a grammar and these "higher-level" considerations were beyond the scope of the present work, significant benefits may result from their use in future research.

References

[1] Bahl, L.R., P.F. Brown, P.V. de Souza, and R.L. Mercer, "A new algorithm for the estimation of hidden Markov model parameters," *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing,* New York, Vol. 1, pp. 493-496, 1988.

[2] Davis, S.B. and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Transactions on Acoustics, Speech, and Signal Processing,* vol. 28, pp.357-366, 1980.

[3] Deller, J.R., J.G. Proakis and J.H.L. Hansen, *Discrete Time Processing of Speech Signals,* New York: Macmillian, 1993.

[4] Deller, J.R. and D. Hsu, "On the use of HMM's to recognize cerebral palsy speech: Isolated word case," *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing,* Glasgow, vol. 1, pp. 290-293, 1989.

[5] Deller, J.R. and D. Hsu, "An alternative adaptive sequential regression algorithm and its application to the recognition of cerebral palsy speech," *IEEE Trans. Circuits and Systems*, vol. 34, pp. 782-787, 1987.

[6] Deller, J.R. and R.K. Snider, " 'Quantized' hidden Markov models for efficient recognition of cerebral palsy speech," *Proceedings 1990 IEEE International Symposium on Circuits and Systems,* New Orleans, vol. 3, pp. 2041-2044, 1990.

[7] Deller, J.R., D. Hsu and L.J. Ferrier, "On the use of hidden Markov models for the recognition of dysarthric speech," *Computer Methods and Programs in Biomedicine*, in press.

[8] Deller, J.R. and G.P. Picaché "Advantages of a Givens rotation approach to temporally recursive linear predition analysis of speech," *IEEE Transactions on Acoustics, Speech, and Signal Processing,* vol. 37, no. 3, pp. 429-431, 1989.

[9] Doddington, G.R. and T.B. Schalk, "Speech recognition: Turning theory to practice," *IEEE Spectrum,* vol. 18, pp. 26-32, 1981.

[10] Foulds, R.A., G. Balesta, W.J. Crochetiere and C. Meyer, "The Tufts non-vocal communication program," *Proceedings 1976 Conference on Systems and Devices for Disabled*, pp. 14-17.

[11] Heckathorne, C.W. and D.S. Childress, "Applying anticipatory text selection in a writing aid for people with a severe motor impairment," *Micro* (IEEE Computer Society), vol. 3, pp. 17-23, 1983.

[12] Hsu, D., *Computer Recognition of Nonverbal Speech Using Hidden Markov Model* (Ph.D. Dissertation), Northeastern University, Boston, 1988.

[13] Johnson, W., F.L. Darley, and D.C. Spreisterbach, *Diagnostic Methods in Speech Pathology*, New York: Harper & Row, 1963.

[14] Lee, K.-F., *Automatic Speech Recognition, the Development of the SPHINX system,* (Ph. D Dissertation), Carnegie-Mellon University, 1989.

[15] Levinson, S.E., "Structural methods in automatic speech recognition," *Proceedings of the IEEE,* vol. 73, pp. 1625-1650, 1985.

[16] Makhoul, J., "Linear prediction: A tutorial review," *Proceedings of the IEEE,* vol. 63, pp. 561-580, 1975.

[17] Makhoul, J., S. Roucos and H. Gish, "Vector quantization in speech coding," *Proceedings of the IEEE,* vol. 73, pp. 1551-1587, 1987.

[18] Mann, H.B. and A. Wald, "On the statistical treatment of linear stochastic difference equation," *Econometrica*, vol. 11, pp. 173-220, 1943.

[19] Markel, J.D. and A.H. Gray, Jr., *Linear Prediction of Speech,* New York: Springer-Verlag, 1976.

[20] Niemann, H., M. Lang and G. Sagerer, *Recent Advances in Speech Understanding and Dialog Systems*, New York: Springer-Verlag, 1987.

[21] O'Shaughnessy, D., *Speech Communication: Human and Machine,* Reading, Massachusetts: Addison Wesley, pp. 420 - 424, 1987.

[22] Parsons, T.W., *Voice and Speech Processing*, New York: McGraw-Hill, 1986.

[23] Picone, J., "Continuous speech recognition using hidden Markov models," *IEEE ASSP Magazine*, vol. 62, pp. 29 - 41, 1990.

[24] Press, W.H., B.P. Flannery, S.A. Teukolsky and W.T. Vetterling, *Numerical Recipes in C*, New York: Cambridge University Press, 1988.

[25] Rabiner, L.R., "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE,* vol. 77, pp. 257-285, 1989.

[26] Rabiner, L.R. and R.W. Schafer, *Digital Processing of Speech Signals,* Englewood-Cliffs, New Jersey: Prentice-Hall, pp.120-135, 1978.

[27] Rosenthal, L.H., L.R. Rabiner, R.W. Schafer, P. Cummiskey, and J.L. Flanagan, "A multiline computer voice response system utilizing ADPCM coded speech," *IEEE Transactions on Acoustics, Speech, and Signal Processing,* vol. 22, no. 5, pp.339-352, 1974.

[28] Snider, R.K., *Efficient Discrete Symbol Hidden Markov Model Evalution Using Transformation and State Reduction* (M.S. Thesis), Michigan State University, 1990.

[29] Snider, R.K., *Laboratory Manual,* Speech Processing Laboratory, Michigan State University, 1991.

[30] Sy, B.K., *A Knowledge-Based Message Generation System for Nonverbal Severely Motor Disabled Persons: Design and Prototype Testing* (Ph.D. Dissertation), Northeastern University, Boston, 1988.

[31] Sy, B.K. and J.R. Deller, "An AI-based communication system for motor and speech disabled persons: Design and methodology and prototype testing," *IEEE Transactions on Biomedical Engineering,* vol. 36, no. 5, 1989.

[32] *UCP Prospectus*, United Cerebral Palsy of Chicago, 1983.

[33] Van Hattum, R.J., *Communication Disorers*, New York: MacMillan, 1980.

[34] Wilpon, J.G. and L.R. Rabiner, "Application of hidden Markov models to automatic speech endpoint detection," *Computer Speech and Language*, vol. 2, pp. 321 - 341, 1987.

APPENDICES

A Experimental Result for Speaker DC

Shown in this appendix is an example experimental result for speaker DC using the TI-46 word list. In this experiment, eight utterances were used for training, seven for testing, a 128 symbol codebook, and 6 state Bakis model. The first column represents correct words, the second column are recognized words, the third column represents the number of correct recognitions to that point in the table, the fourth column are the number of times the correct word appeared in the two words to which the recognizer assigned highest likelihood to that point in the table (column five and six are similar results for top four and eight candidates), and the last column represents the total number of testing words.

A	A	p=1	p2=1	p4=1	p8=1	tot=1
A	X	p=1	p2=1	p4=1	p8=2	tot=2
A	ENTER	p=1	p2=1	p4=1	p8=3	tot=3
A	A	p=2	p2=2	p4=2	p8=4	tot=4
A	A	p=3	p2=3	p4=3	p8=5	tot=5
A	A	p=4	p2=4	p4=4	p8=6	tot=6
A	A	p=5	p2=5	p4=5	p8=7	tot=7
B	B	p=6	p2=6	p4=6	p8=8	tot=8
B	V	p=6	p2=7	p4=7	p8=9	tot=9
B	V	p=6	p2=7	p4=8	p8=10	tot=10
B	REPEAT	p=6	p2=8	p4=9	p8=11	tot=11
B	B	p=7	p2=9	p4=10	p8=12	tot=12
B	B	p=8	p2=10	p4=11	p8=13	tot=13
B	B	p=9	p2=11	p4=12	p8=14	tot=14
C	C	p=10	p2=12	p4=13	p8=15	tot=15
C	C	p=11	p2=13	p4=14	p8=16	tot=16
C	C	p=12	p2=14	p4=15	p8=17	tot=17
C	C	p=13	p2=15	p4=16	p8=18	tot=18
C	C	p=14	p2=16	p4=17	p8=19	tot=19
C	C	p=15	p2=17	p4=18	p8=20	tot=20
C	C	p=16	p2=18	p4=19	p8=21	tot=21
D	D	p=17	p2=19	p4=20	p8=22	tot=22
D	D	p=18	p2=20	p4=21	p8=23	tot=23
D	G	p=18	p2=21	p4=22	p8=24	tot=24
D	D	p=19	p2=22	p4=23	p8=25	tot=25
D	D	p=20	p2=23	p4=24	p8=26	tot=26
D	D	p=21	p2=24	p4=25	p8=27	tot=27
D	D	p=22	p2=25	p4=26	p8=28	tot=28
E	P	p=22	p2=26	p4=27	p8=29	tot=29
E	E	p=23	p2=27	p4=28	p8=30	tot=30
E	E	p=24	p2=28	p4=29	p8=31	tot=31
E	G	p=24	p2=28	p4=30	p8=32	tot=32
E	D	p=24	p2=29	p4=31	p8=33	tot=33
E	E	p=25	p2=30	p4=32	p8=34	tot=34
E	E	p=26	p2=31	p4=33	p8=35	tot=35
F	STOP	p=26	p2=31	p4=33	p8=35	tot=36
F	STOP	p=26	p2=31	p4=34	p8=36	tot=37
F	F	p=27	p2=32	p4=35	p8=37	tot=38
F	F	p=28	p2=33	p4=36	p8=38	tot=39
F	F	p=29	p2=34	p4=37	p8=39	tot=40
F	L	p=29	p2=35	p4=38	p8=40	tot=41
F	F	p=30	p2=36	p4=39	p8=41	tot=42
G	G	p=31	p2=37	p4=40	p8=42	tot=43
G	G	p=32	p2=38	p4=41	p8=43	tot=44
G	G	p=33	p2=39	p4=42	p8=44	tot=45
G	G	p=34	p2=40	p4=43	p8=45	tot=46
G	G	p=35	p2=41	p4=44	p8=46	tot=47
G	G	p=36	p2=42	p4=45	p8=47	tot=48
G	G	p=37	p2=43	p4=46	p8=48	tot=49
H	H	p=38	p2=44	p4=47	p8=49	tot=50
H	H	p=39	p2=45	p4=48	p8=50	tot=51
H	H	p=40	p2=46	p4=49	p8=51	tot=52
H	H	p=41	p2=47	p4=50	p8=52	tot=53
H	H	p=42	p2=48	p4=51	p8=53	tot=54
H	H	p=43	p2=49	p4=52	p8=54	tot=55
H	H	p=44	p2=50	p4=53	p8=55	tot=56
I	I	p=45	p2=51	p4=54	p8=56	tot=57
I	I	p=46	p2=52	p4=55	p8=57	tot=58
I	I	p=47	p2=53	p4=56	p8=58	tot=59
I	I	p=48	p2=54	p4=57	p8=59	tot=60
I	I	p=49	p2=55	p4=58	p8=60	tot=61
I	I	p=50	p2=56	p4=59	p8=61	tot=62
I	I	p=51	p2=57	p4=60	p8=62	tot=63
J	J	p=52	p2=58	p4=61	p8=63	tot=64
J	J	p=53	p2=59	p4=62	p8=64	tot=65
J	YES	p=53	p2=59	p4=63	p8=65	tot=66

J	J	p=54	p2=60	p4=64	p8=66	tot=67
J	J	p=55	p2=61	p4=65	p8=67	tot=68
J	J	p=56	p2=62	p4=66	p8=68	tot=69
J	J	p=57	p2=63	p4=67	p8=69	tot=70
K	K	p=58	p2=64	p4=68	p8=70	tot=71
K	K	p=59	p2=65	p4=69	p8=71	tot=72
K	K	p=60	p2=66	p4=70	p8=72	tot=73
K	K	p=61	p2=67	p4=71	p8=73	tot=74
K	K	p=62	p2=68	p4=72	p8=74	tot=75
K	K	p=63	p2=69	p4=73	p8=75	tot=76
K	K	p=64	p2=70	p4=74	p8=76	tot=77
L	L	p=65	p2=71	p4=75	p8=77	tot=78
L	L	p=66	p2=72	p4=76	p8=78	tot=79
L	L	p=67	p2=73	p4=77	p8=79	tot=80
L	L	p=68	p2=74	p4=78	p8=80	tot=81
L	L	p=69	p2=75	p4=79	p8=81	tot=82
L	L	p=70	p2=76	p4=80	p8=82	tot=83
L	L	p=71	p2=77	p4=81	p8=83	tot=84
M	M	p=72	p2=78	p4=82	p8=84	tot=85
M	M	p=73	p2=79	p4=83	p8=85	tot=86
M	M	p=74	p2=80	p4=84	p8=86	tot=87
M	M	p=75	p2=81	p4=85	p8=87	tot=88
M	M	p=76	p2=82	p4=86	p8=88	tot=89
M	M	p=77	p2=83	p4=87	p8=89	tot=90
M	M	p=78	p2=84	p4=88	p8=90	tot=91
N	N	p=79	p2=85	p4=89	p8=91	tot=92
N	N	p=80	p2=86	p4=90	p8=92	tot=93
N	ENTER	p=80	p2=87	p4=91	p8=93	tot=94
N	ENTER	p=80	p2=87	p4=92	p8=94	tot=95
N	ENTER	p=80	p2=88	p4=93	p8=95	tot=96
N	N	p=81	p2=89	p4=94	p8=96	tot=97
N	N	p=82	p2=90	p4=95	p8=97	tot=98
O	O	p=83	p2=91	p4=96	p8=98	tot=99
O	O	p=84	p2=92	p4=97	p8=99	tot=100
O	O	p=85	p2=93	p4=98	p8=100	tot=101
O	O	p=86	p2=94	p4=99	p8=101	tot=102
O	O	p=87	p2=95	p4=100	p8=102	tot=103
O	O	p=88	p2=96	p4=101	p8=103	tot=104
O	O	p=89	p2=97	p4=102	p8=104	tot=105
P	P	p=90	p2=98	p4=103	p8=105	tot=106
P	D	p=90	p2=99	p4=104	p8=106	tot=107
P	P	p=91	p2=100	p4=105	p8=107	tot=108
P	P	p=92	p2=101	p4=106	p8=108	tot=109
P	P	p=93	p2=102	p4=107	p8=109	tot=110
P	P	p=94	p2=103	p4=108	p8=110	tot=111
P	P	p=95	p2=104	p4=109	p8=111	tot=112
Q	Q	p=96	p2=105	p4=110	p8=112	tot=113
Q	Q	p=97	p2=106	p4=111	p8=113	tot=114
Q	Q	p=98	p2=107	p4=112	p8=114	tot=115
Q	Q	p=99	p2=108	p4=113	p8=115	tot=116
Q	Q	p=100	p2=109	p4=114	p8=116	tot=117
Q	Q	p=101	p2=110	p4=115	p8=117	tot=118
Q	Q	p=102	p2=111	p4=116	p8=118	tot=119
R	R	p=103	p2=112	p4=117	p8=119	tot=120
R	R	p=104	p2=113	p4=118	p8=120	tot=121
R	STOP	p=104	p2=113	p4=119	p8=121	tot=122
R	R	p=105	p2=114	p4=120	p8=122	tot=123
R	L	p=105	p2=115	p4=121	p8=123	tot=124
R	R	p=106	p2=116	p4=122	p8=124	tot=125
R	R	p=107	p2=117	p4=123	p8=125	tot=126
S	S	p=108	p2=118	p4=124	p8=126	tot=127
S	S	p=109	p2=119	p4=125	p8=127	tot=128
S	X	p=109	p2=120	p4=126	p8=128	tot=129
S	S	p=110	p2=121	p4=127	p8=129	tot=130
S	YES	p=110	p2=122	p4=128	p8=130	tot=131
S	S	p=111	p2=123	p4=129	p8=131	tot=132

S	S	p=112	p2=124	p4=130	p8=132	tot=133
T	THREE	p=112	p2=125	p4=131	p8=133	tot=134
T	T	p=113	p2=126	p4=132	p8=134	tot=135
T	T	p=114	p2=127	p4=133	p8=135	tot=136
T	T	p=115	p2=128	p4=134	p8=136	tot=137
T	T	p=116	p2=129	p4=135	p8=137	tot=138
T	T	p=117	p2=130	p4=136	p8=138	tot=139
T	T	p=118	p2=131	p4=137	p8=139	tot=140
U	U	p=119	p2=132	p4=138	p8=140	tot=141
U	U	p=120	p2=133	p4=139	p8=141	tot=142
U	ZERO	p=120	p2=134	p4=140	p8=142	tot=143
U	U	p=121	p2=135	p4=141	p8=143	tot=144
U	U	p=122	p2=136	p4=142	p8=144	tot=145
U	U	p=123	p2=137	p4=143	p8=145	tot=146
U	U	p=124	p2=138	p4=144	p8=146	tot=147
V	V	p=125	p2=139	p4=145	p8=147	tot=148
V	V	p=126	p2=140	p4=146	p8=148	tot=149
V	V	p=127	p2=141	p4=147	p8=149	tot=150
V	V	p=128	p2=142	p4=148	p8=150	tot=151
V	V	p=129	p2=143	p4=149	p8=151	tot=152
V	V	p=130	p2=144	p4=150	p8=152	tot=153
V	V	p=131	p2=145	p4=151	p8=153	tot=154
W	W	p=132	p2=146	p4=152	p8=154	tot=155
W	W	p=133	p2=147	p4=153	p8=155	tot=156
W	W	p=134	p2=148	p4=154	p8=156	tot=157
W	W	p=135	p2=149	p4=155	p8=157	tot=158
W	W	p=136	p2=150	p4=156	p8=158	tot=159
W	W	p=137	p2=151	p4=157	p8=159	tot=160
W	W	p=138	p2=152	p4=158	p8=160	tot=161
X	X	p=139	p2=153	p4=159	p8=161	tot=162
X	X	p=140	p2=154	p4=160	p8=162	tot=163
X	X	p=141	p2=155	p4=161	p8=163	tot=164
X	X	p=142	p2=156	p4=162	p8=164	tot=165
X	X	p=143	p2=157	p4=163	p8=165	tot=166
X	X	p=144	p2=158	p4=164	p8=166	tot=167
X	X	p=145	p2=159	p4=165	p8=167	tot=168
Y	Y	p=146	p2=160	p4=166	p8=168	tot=169
Y	Y	p=147	p2=161	p4=167	p8=169	tot=170
Y	Y	p=148	p2=162	p4=168	p8=170	tot=171
Y	Y	p=149	p2=163	p4=169	p8=171	tot=172
Y	Y	p=150	p2=164	p4=170	p8=172	tot=173
Y	Y	p=151	p2=165	p4=171	p8=173	tot=174
Y	Y	p=152	p2=166	p4=172	p8=174	tot=175
Z	REPEAT	p=152	p2=167	p4=173	p8=175	tot=176
Z	Z	p=153	p2=168	p4=174	p8=176	tot=177
Z	Z	p=154	p2=169	p4=175	p8=177	tot=178
Z	Z	p=155	p2=170	p4=176	p8=178	tot=179
Z	Z	p=156	p2=171	p4=177	p8=179	tot=180
Z	Z	p=157	p2=172	p4=178	p8=180	tot=181
Z	Z	p=158	p2=173	p4=179	p8=181	tot=182
ONE	ONE	p=159	p2=174	p4=180	p8=182	tot=183
ONE	ONE	p=160	p2=175	p4=181	p8=183	tot=184
ONE	ONE	p=161	p2=176	p4=182	p8=184	tot=185
ONE	ONE	p=162	p2=177	p4=183	p8=185	tot=186
ONE	ONE	p=163	p2=178	p4=184	p8=186	tot=187
ONE	ONE	p=164	p2=179	p4=185	p8=187	tot=188
ONE	ONE	p=165	p2=180	p4=186	p8=188	tot=189
TWO	TWO	p=166	p2=181	p4=187	p8=189	tot=190
TWO	TWO	p=167	p2=182	p4=188	p8=190	tot=191
TWO	TWO	p=168	p2=183	p4=189	p8=191	tot=192
TWO	TWO	p=169	p2=184	p4=190	p8=192	tot=193
TWO	TWO	p=170	p2=185	p4=191	p8=193	tot=194
TWO	TWO	p=171	p2=186	p4=192	p8=194	tot=195
TWO	TWO	p=172	p2=187	p4=193	p8=195	tot=196
THREE	THREE	p=173	p2=188	p4=194	p8=196	tot=197
THREE	THREE	p=174	p2=189	p4=195	p8=197	tot=198

THREE	THREE	p-175	p2-190	p4-196	p8-198	tot-199
THREE	THREE	p-176	p2-191	p4-197	p8-199	tot-200
THREE	REPEAT	p-176	p2-192	p4-198	p8-200	tot-201
THREE	THREE	p-177	p2-193	p4-199	p8-201	tot-202
THREE	THREE	p-178	p2-194	p4-200	p8-202	tot-203
FOUR	FOUR	p-179	p2-195	p4-201	p8-203	tot-204
FOUR	FOUR	p-180	p2-196	p4-202	p8-204	tot-205
FOUR	FOUR	p-181	p2-197	p4-203	p8-205	tot-206
FOUR	FOUR	p-182	p2-198	p4-204	p8-206	tot-207
FOUR	FOUR	p-183	p2-199	p4-205	p8-207	tot-208
FOUR	FOUR	p-184	p2-200	p4-206	p8-208	tot-209
FOUR	FOUR	p-185	p2-201	p4-207	p8-209	tot-210
FIVE	FIVE	p-186	p2-202	p4-208	p8-210	tot-211
FIVE	FIVE	p-187	p2-203	p4-209	p8-211	tot-212
FIVE	FIVE	p-188	p2-204	p4-210	p8-212	tot-213
FIVE	FIVE	p-189	p2-205	p4-211	p8-213	tot-214
FIVE	FIVE	p-190	p2-206	p4-212	p8-214	tot-215
FIVE	FIVE	p-191	p2-207	p4-213	p8-215	tot-216
FIVE	FIVE	p-192	p2-208	p4-214	p8-216	tot-217
SIX	SIX	p-193	p2-209	p4-215	p8-217	tot-218
SIX	SIX	p-194	p2-210	p4-216	p8-218	tot-219
SIX	SIX	p-195	p2-211	p4-217	p8-219	tot-220
SIX	SIX	p-196	p2-212	p4-218	p8-220	tot-221
SIX	SIX	p-197	p2-213	p4-219	p8-221	tot-222
SIX	SIX	p-198	p2-214	p4-220	p8-222	tot-223
SIX	SIX	p-199	p2-215	p4-221	p8-223	tot-224
SEVEN	SEVEN	p-200	p2-216	p4-222	p8-224	tot-225
SEVEN	SEVEN	p-201	p2-217	p4-223	p8-225	tot-226
SEVEN	SEVEN	p-202	p2-218	p4-224	p8-226	tot-227
SEVEN	SEVEN	p-203	p2-219	p4-225	p8-227	tot-228
SEVEN	SEVEN	p-204	p2-220	p4-226	p8-228	tot-229
SEVEN	SEVEN	p-205	p2-221	p4-227	p8-229	tot-230
SEVEN	SEVEN	p-206	p2-222	p4-228	p8-230	tot-231
EIGHT	EIGHT	p-207	p2-223	p4-229	p8-231	tot-232
EIGHT	H	p-207	p2-224	p4-230	p8-232	tot-233
EIGHT	H	p-207	p2-225	p4-231	p8-233	tot-234
EIGHT	H	p-207	p2-226	p4-232	p8-234	tot-235
EIGHT	EIGHT	p-208	p2-227	p4-233	p8-235	tot-236
EIGHT	EIGHT	p-209	p2-228	p4-234	p8-236	tot-237
EIGHT	H	p-209	p2-228	p4-235	p8-237	tot-238
NINE	NINE	p-210	p2-229	p4-236	p8-238	tot-239
NINE	NO	p-210	p2-229	p4-236	p8-239	tot-240
NINE	M	p-210	p2-230	p4-237	p8-240	tot-241
NINE	NINE	p-211	p2-231	p4-238	p8-241	tot-242
NINE	NINE	p-212	p2-232	p4-239	p8-242	tot-243
NINE	NINE	p-213	p2-233	p4-240	p8-243	tot-244
NINE	NINE	p-214	p2-234	p4-241	p8-244	tot-245
ZERO	ZERO	p-215	p2-235	p4-242	p8-245	tot-246
ZERO	ZERO	p-216	p2-236	p4-243	p8-246	tot-247
ZERO	ZERO	p-217	p2-237	p4-244	p8-247	tot-248
ZERO	ZERO	p-218	p2-238	p4-245	p8-248	tot-249
ZERO	ZERO	p-219	p2-239	p4-246	p8-249	tot-250
ZERO	ZERO	p-220	p2-240	p4-247	p8-250	tot-251
ZERO	ZERO	p-221	p2-241	p4-248	p8-251	tot-252
START	START	p-222	p2-242	p4-249	p8-252	tot-253
START	START	p-223	p2-243	p4-250	p8-253	tot-254
START	START	p-224	p2-244	p4-251	p8-254	tot-255
START	START	p-225	p2-245	p4-252	p8-255	tot-256
START	START	p-226	p2-246	p4-253	p8-256	tot-257
START	START	p-227	p2-247	p4-254	p8-257	tot-258
START	START	p-228	p2-248	p4-255	p8-258	tot-259
STOP	STOP	p-229	p2-249	p4-256	p8-259	tot-260
STOP	STOP	p-230	p2-250	p4-257	p8-260	tot-261
STOP	STOP	p-231	p2-251	p4-258	p8-261	tot-262
STOP	STOP	p-232	p2-252	p4-259	p8-262	tot-263
STOP	STOP	p-233	p2-253	p4-260	p8-263	tot-264

STOP	STOP	p=234	p2=254	p4=261	p8=264	tot=265
STOP	STOP	p=235	p2=255	p4=262	p8=265	tot=266
YES	YES	p=236	p2=256	p4=263	p8=266	tot=267
YES	YES	p=237	p2=257	p4=264	p8=267	tot=268
YES	YES	p=238	p2=258	p4=265	p8=268	tot=269
YES	YES	p=239	p2=259	p4=266	p8=269	tot=270
YES	YES	p=240	p2=260	p4=267	p8=270	tot=271
YES	YES	p=241	p2=261	p4=268	p8=271	tot=272
YES	YES	p=242	p2=262	p4=269	p8=272	tot=273
NO	NO	p=243	p2=263	p4=270	p8=273	tot=274
NO	NO	p=244	p2=264	p4=271	p8=274	tot=275
NO	NO	p=245	p2=265	p4=272	p8=275	tot=276
NO	NO	p=246	p2=266	p4=273	p8=276	tot=277
NO	NO	p=247	p2=267	p4=274	p8=277	tot=278
NO	NO	p=248	p2=268	p4=275	p8=278	tot=279
NO	NO	p=249	p2=269	p4=276	p8=279	tot=280
GO	FOUR	p=249	p2=269	p4=276	p8=279	tot=281
GO	GO	p=250	p2=270	p4=277	p8=280	tot=282
GO	GO	p=251	p2=271	p4=278	p8=281	tot=283
GO	NO	p=251	p2=272	p4=279	p8=282	tot=284
GO	GO	p=252	p2=273	p4=280	p8=283	tot=285
GO	GO	p=253	p2=274	p4=281	p8=284	tot=286
GO	GO	p=254	p2=275	p4=282	p8=285	tot=287
HELP	HELP	p=255	p2=276	p4=283	p8=286	tot=288
HELP	HELP	p=256	p2=277	p4=284	p8=287	tot=289
HELP	HELP	p=257	p2=278	p4=285	p8=288	tot=290
HELP	HELP	p=258	p2=279	p4=286	p8=289	tot=291
HELP	HELP	p=259	p2=280	p4=287	p8=290	tot=292
HELP	HELP	p=260	p2=281	p4=288	p8=291	tot=293
HELP	HELP	p=261	p2=282	p4=289	p8=292	tot=294
ERASE	ERASE	p=262	p2=283	p4=290	p8=293	tot=295
ERASE	ERASE	p=263	p2=284	p4=291	p8=294	tot=296
ERASE	ERASE	p=264	p2=285	p4=292	p8=295	tot=297
ERASE	ERASE	p=265	p2=286	p4=293	p8=296	tot=298
ERASE	ERASE	p=266	p2=287	p4=294	p8=297	tot=299
ERASE	ERASE	p=267	p2=288	p4=295	p8=298	tot=300
ERASE	ERASE	p=268	p2=289	p4=296	p8=299	tot=301
RUBOUT	RUBOUT	p=269	p2=290	p4=297	p8=300	tot=302
RUBOUT	RUBOUT	p=270	p2=291	p4=298	p8=301	tot=303
RUBOUT	RUBOUT	p=271	p2=292	p4=299	p8=302	tot=304
RUBOUT	Y	p=271	p2=292	p4=300	p8=303	tot=305
RUBOUT	RUBOUT	p=272	p2=293	p4=301	p8=304	tot=306
RUBOUT	STOP	p=272	p2=294	p4=302	p8=305	tot=307
RUBOUT	RUBOUT	p=273	p2=295	p4=303	p8=306	tot=308
REPEAT	REPEAT	p=274	p2=296	p4=304	p8=307	tot=309
REPEAT	REPEAT	p=275	p2=297	p4=305	p8=308	tot=310
REPEAT	REPEAT	p=276	p2=298	p4=306	p8=309	tot=311
REPEAT	REPEAT	p=277	p2=299	p4=307	p8=310	tot=312
REPEAT	REPEAT	p=278	p2=300	p4=308	p8=311	tot=313
REPEAT	REPEAT	p=279	p2=301	p4=309	p8=312	tot=314
REPEAT	REPEAT	p=280	p2=302	p4=310	p8=313	tot=315
ENTER	ENTER	p=281	p2=303	p4=311	p8=314	tot=316
ENTER	ENTER	p=282	p2=304	p4=312	p8=315	tot=317
ENTER	ENTER	p=283	p2=305	p4=313	p8=316	tot=318
ENTER	ENTER	p=284	p2=306	p4=314	p8=317	tot=319
ENTER	ENTER	p=285	p2=307	p4=315	p8=318	tot=320
ENTER	ENTER	p=286	p2=308	p4=316	p8=319	tot=321
ENTER	ENTER	p=287	p2=309	p4=317	p8=320	tot=322

recognition rate = 89.1304 percent

B Program Listing: LP Parameter Generating Program

This program computes the LP parameters of an input speech file using the autocorrelation method, and then quantizes the LP parameters.

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include <time.h>
#include <math.h>
#define N                       256 /* Hamming window size */
#define NUM                   50000 /* maximum allowed for speech data */
#define MO                       14 /* Model Order of LPC */
#define NLEVELS                   7 /* number of levels in binary codebook */
#define LEVELINDX               126 /* 2^NLEVELS-2 */
#define TOTVECT                 254 /* number of vectors in codebook */
#define CODEFILE     "codedc36.dat" /* output file directive */

int count;
int index2[100][2];
double speech_data[NUM];
double window[N];
double a[MO+1];                 /* estimated LP parameters */
double r[MO+1];                 /* short term autocorrelation */
double codebook[TOTVECT][MO];
char codefile[80];
FILE *outfile;

/**************************************************************************
 * Program name: lpcqnt_a.c                                               *
 * Command     : run386 lpcqnt_a cdbkfile.dat                             *
 * Description : Computing the LP parameters using autocorrelation        *
 *               method with 256 points                                   *
 *               Hamming window as a frame.                               *
 * Date        : July 19, 1990                                            *
 **************************************************************************/

main(argc,argv)
int argc;
char *argv[];
{
int i,j,k,h,t;
int numread;
int M,p,sum,m;
char infiles[80],infilename1[80],infilename[80],filename[100];
char instring[40],numstring[5],input1[30];
short buffer[64];
FILE *infile1,*infile2,*infile,*file;
int total_data;
void lpc_computation();
void codebook_entry();

if( argc < 1 )
   {
   printf("*****  After program name enter two file names  *****\n");
   printf("1.   The first file name is the name of the codebook.\n");
   printf("2.   The second is the name of the file that contains the paths and\n");
   printf("     names of all the data files to be qauntized.\n\n");
   printf("Example:   ");
   printf("lpc2 codebook.dat allfiles.dat\n",argv[0]);
   exit(0);
   }
strcpy(codefile,argv[1]);

codebook_entry();

count=0;
strcpy(infilename,"tstti.dat");
printf("Start data input -  infilename = %s\n",infilename);
if ( (infile = fopen(infilename, "r")) == NULL)
   {
```

```
    printf("fopen failed for infilename %s.\n",infilename);
    exit(0);
    }
while( fscanf(infile,"%s\n",instring) != EOF)
    {
    for(h=1; h<8; h++)
        {
        strcpy(filename,"c:\\dc36\\test\\bin\\");
        strcat(filename,instring);
        strcat(filename,".36");
        switch(h)
            {
            case 1  : strcpy(numstring,"9"); break;
            case 2  : strcpy(numstring,"a"); break;
            case 3  : strcpy(numstring,"b"); break;
            case 4  : strcpy(numstring,"c"); break;
            case 5  : strcpy(numstring,"d"); break;
            case 6  : strcpy(numstring,"e"); break;
            case 7  : strcpy(numstring,"f"); break;
            default : break;
            }
        strcat(filename,numstring);
        _pmode = 0x8000;
        if ( (file = fopen(filename, "r")) == NULL)
            {
            printf("fopen failed for filename %s\n",filename);
            exit(0);
            }
        printf("reading in data from file %s .....\n",filename);
        k=t=0;
        do
            {
            numread = fread((void *)buffer, sizeof(short),64,file);
            if(numread == 0)
                break;
            for(i=0; i<64; i++)
                {
                speech_data[t] = (int)buffer[i];
                t++;
                }
            }while( feof(infile) == 0 || numread == 64 );  /** for with.cp5 file **/
        fclose(file);
        strcpy(filename,"\\dc36\\test\\qnt\\");
        strcat(filename,instring);
        strcat(filename,".vq");
        strcat(filename,numstring);
        _pmode = 0x4000;
        outfile = fopen(filename,"w");
        printf("          Quantizing data...\n");
        count=0;
        lpc_computation(t);
        printf("%u Quantized lpc vectors written to file  %s\n",count,filename);
        fclose(outfile);
        }
    }
}

/***************************************************************************
 *   This function computes the lpc vectors from the speech data and      *
 *   then vector quantizes the lpc vectors                                *
 ***************************************************************************/
void lpc_computation(total_data)
int total_data;
{
  int n,i;
  void LD_recursion();
```

```
  void shift_window();
  void vector_quantize();

  a[0]=1;
  for(n=0; (n+N)<total_data; n=n+50)
  {
    shift_window(n);
    LD_recursion();
    vector_quantize();
  }
}

/*****************************************************************************
 * This function takes 256 points from sampled data using Hamming window    *
 * to implement short term LP analysis                                       *
 *****************************************************************************/
void shift_window(n)
int n;
{
  int i;
  for (i=0; i<N; i++)
  {
    window[i]=speech_data[n]*(0.54-0.46*cos(2*PI*i/N));
    n++;
  }
}

/*****************************************************************************
 * This routine use Levinson-Durbin Recursion to get the LP parameters      *
 *****************************************************************************/
void LD_recursion()
{
  int i,l;
  double ai,aj,temp;
  double e;           /* xi, the average energy in the predition residual */
  double k;           /* kappa, reflection coefficient */
  void comp_corr();   /* compute short term autocorrelation */

  comp_corr();
  /* initialization */
  e=r[0];
  for (l=1; l<=MO; l++)
  {
    /* step 1 */
    temp=0.0;
    for (i=1; i<l; i++)
      temp=temp+a[i]*r[l-i];
    k=(r[l]-temp)/e;
    /* step 2 */
    a[l]=k;
    for (i=1; i<=l/2; i++)
    {
      ai=a[i];
      aj=a[l-i];
      a[i]=ai-k*aj;
      a[l-i]=aj-k*ai;
    }
    /* step 3 */
    e=e*(1.-k*k);
  }
}

/*****************************************************************************
 * This procedure computes short-term autocorrelation                       *
 *****************************************************************************/
void comp_corr()
```

```
{
  int i,j;

  for (i=0; i<=MO; i++)
  {
    r[i]=0;
    for (j=0; j+i<N; j++)
      r[i]=r[i]+window[j]*window[j+i];
    r[i]=r[i]/N;
  }
}

/***************************************************************************
 * This routine vector quantizes the computed lpc vector with respect to   *
 * the given codebook.                                                      *
 ***************************************************************************/
void vector_quantize()
{
int i,index1,index2,vq;
double idm1,idm2;
double itakura_dist_meas();
int level_index();

index1 = 0;
index2 = 1;
idm1 = itakura_dist_meas(codebook[index1]);
idm2 = itakura_dist_meas(codebook[index2]);
if( idm1 > idm2 )
   index1 = index2;

for(i=1; i<NLEVELS; i++)
   {
   index1 = (index1 - level_index(i))*2 + level_index(i+1);
   index2 = index1 + 1;
   idm1 = itakura_dist_meas(codebook[index1]);
   idm2 = itakura_dist_meas(codebook[index2]);
   if( idm1 > idm2 )
      index1 = index2;
   }
vq = index1 - LEVELINDX;
fprintf(outfile,"%d\n",vq);
count=count+1;
}

/***************************************************************************
 * This routine calculates which vector to compare next in the codebook    *
 * once a vector index in the previous level is given                       *
 ***************************************************************************/
int level_index(k)
int k;
{
int num;
num = (int)pow((double)2,(double)k) - 2;
return num;
}

/***************************************************************************
 * This routine calculates the Itakura Distance Measure between the         *
 * computed lpc vector and a vector from the codebook                       *
 ***************************************************************************/
double itakura_dist_meas(array)
double array[];
{
  int i,j;
  double temp1[MO+1],temp2[MO+1];
  double a1[MO+1],entry[MO+1];
```

```
   double idml,idm2,idm;

   entry[0]=1.0;
   al[0]=1.0;
   for (i=1; i<=MO; i++)
   {
     entry[i]=0.-array[i-1];
     al[i]=0.-a[i];
   }
   for (i=0; i<=MO; i++)
   {
     temp1[i]=0;
     temp2[i]=0;
     for (j=0; j<=MO; j++)
       if (i<j)
       {
         temp1[i]=temp1[i]+al[j]*r[j-i];
         temp2[i]=temp2[i]+entry[j]*r[j-i];
       }
       else
       {
         temp1[i]=temp1[i]+al[j]*r[i-j];
         temp2[i]=temp2[i]+entry[j]*r[i-j];
       }
   }
   idml=0;
   idm2=0;
   for (i=0; i<=MO; i++)
   {
     idml=idml+temp1[i]*al[i];
     idm2=idm2+temp2[i]*entry[i];
   }
   idm=log(idm2)-log(idml);
   return idm;
}

/*****************************************************************************/
void codebook_entry()
{
FILE *infile3;
int i,j,k,m;
char input11[80],input12[80],input13[80];
void extractword();

infile3 = fopen(CODEFILE,"r");

printf("\nReading %s\n",CODEFILE);
m=0;
for(i=1; i<=NLEVELS; i++)
   {
   fgets(input11,80,infile3);
   printf("%s",input11);
   for(j=0; j<(int)pow((double)2,(double)i); j++)
      {
      fgets(input11,67,infile3);
      fgets(input12,67,infile3);
      fgets(input13,80,infile3);
      extractword(input11,input12,input13,m);
      for (k=0; k<14; k++) printf("%f ",codebook[m][k]);
      printf("\n");
      m++;
      }
   }
fclose(infile3);
}
```

```
/******************************************************************************/
void extractword(in1,in2,in3,m)
char in1[80],in2[80],in3[80];
int m;
{
   int i,j,k;
   char temp1[30],temp2[30],temp4[30];
   for( j=0; j<30; j++)
   {
      temp1[j] = '\0';
      temp2[j] = '\0';
      temp4[j] = '\0';
   }
   for(i=0; i<6; i++)
   {
      k=0;
      for(j=i*11; j<(i+1)*11; j++)
      {
         temp1[k] = in1[j];
         temp2[k] = in2[j];
         k++;
      }
      codebook[m][i] = atof(temp1);
      codebook[m][i+6] = atof(temp2);
   }
   for(i=0; i<2; i++)
   {
      k=0;
      for(j=i*11; j<(i+1)*11; j++)
      {
         temp4[k] = in3[j];
         k++;
      }
      codebook[m][i+12] = atof(temp4);
   }
}
```

C Program Listing: Cepstral Parameter Generating Program

This program computes the mel-cepstral parameters of an input speech file using a 1024 point FFT, and then quantizes the cepstral parameters.

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include <time.h>
#include <math.h>
#define N                    256 /* Hamming window size */
#define NUM                50000 /* maximum allowed for speech data */
#define MO                    10 /* model order of cepstrum */
#define NLEVELS                7 /* number of levels in binary codebook */
#define LEVELINDX            126 /* 2^NLEVELS-2 */
#define TOTVECT              254 /* number of vectors in codebook */
#define CODEFILE  "codele27.dat" /* codebook file */
#define FFT                 1024 /* number of point for FFT */

int count;
int freq[22];                    /* mel_frequency */
int speech_data[NUM];
double window[FFT];              /* 256 samples plus zero padding */
double c[MO+1];                  /* cepstrum parameters */
double codebook[TOTVECT][MO];
FILE *outfile;                   /* pointer to quantized file */

/****************************************************************************
 * Program name: ceps_qnt.c                                                 *
 * Command     : run386 cepstrum                                            *
 * Description : cepstral analysis training data with 1024 points FFT and   *
 *               silent portion kept and then quantize these cepstrum       *
 *               parameters                                                 *
 * Date        : August 1, 1990                                             *
 ****************************************************************************/

main()
{
   int i,j,k,h,t;
   int numread;
   int M,p,sum,m,mt;
   int indata[NUM],index2[100][2];
   char infiles[80],infilenamel[80],infilename[80],filename[100];
   char instring[40],numstring[5],input1[30];
   short buffer[64];
   FILE *infile2,*infile,*file;
   int total_data;
   void cepstrum_comp();
   void codebook_entry();
   void mel_freq();

   codebook_entry();
   mel_freq();

   count=0;
   for (i=0; i<FFT; i++)
      window[i]=0.0;
   strcpy(infilename,"tstti.dat");
   printf("Start data input -  infilename = %s\n",infilename);
   if ( (infile = fopen(infilename, "r")) == NULL)
   {
      printf("fopen failed for infilename %s.\n",infilename);
      exit(0);
   }
   while( fscanf(infile,"%s\n",instring) != EOF)
   {
      for(h=1; h<9; h++)
      {
         strcpy(filename,"c:\\le27\\train\\bin\\");
         strcat(filename,instring);
         strcat(filename,".27");
```

```
         switch(h)
         {
            case 1  : strcpy(numstring,"1");break;
            case 2  : strcpy(numstring,"2");break;
            case 3  : strcpy(numstring,"3");break;
            case 4  : strcpy(numstring,"4");break;
            case 5  : strcpy(numstring,"5");break;
            case 6  : strcpy(numstring,"6");break;
            case 7  : strcpy(numstring,"7");break;
            case 8  : strcpy(numstring,"8");break;
            default : break;
         }
      strcat(filename,numstring);
      _pmode = 0x8000;
      if ( (file = fopen(filename, "r")) == NULL)
         {
         printf("fopen failed for filename %s\n",filename);
         exit(0);
         }
      printf("reading in data from file %s .....\n",filename);
      k=t=0;
      do
         {
         numread = fread((void *)buffer, sizeof(short),64,file);
         if(numread == 0)
            break;
         for(i=0; i<64; i++)
            {
            speech_data[t] = (int)buffer[i];
            t++;
            }
         }while( feof(infile) == 0 || numread == 64 );  /* for with.cp5 file */
      fclose(file);
      strcpy(filename,"\\le27\\train\\qnt\\");
      strcat(filename,instring);
      strcat(filename,".vq");
      strcat(filename,numstring);
      _pmode = 0x4000;
      outfile = fopen(filename,"w");
      printf("          Quantizing data...\n");
      count=0;
      cepstrum_comp(t);
      printf("%u Quantized lpc vectors written to file  %s\n",count,filename);
      fclose(outfile);
      }
   }
}

/*****************************************************************************
 * This function computes the cepstrum parameters from the speech data and   *
 * then vector quantizes the cepstrum parameters                             *
 *****************************************************************************/
void cepstrum_comp(total_data)
int total_data;
{
   int n,i;
   float f[2*FFT+1];
   double mel[21],rf[FFT/2+1];
   void shift_window();
   void stdft();
   void mel_energy();
   void mel_cepstrum();
   void vector_quantize();

   for(n=0; (n+N)<total_data; n=n+50)
   {
```

```
      shift_window(n);
      for (i=1; i<=FFT; i++)
      {
         f[2*i]=0.0;
         f[2*i-1]=(float)window[i-1];
      }
      stdft(f,FFT,1);
      for (i=1; i<=(FFT/2); i++)
         rf[i]=sqrt((double)f[2*i-1]*(double)f[2*i-1]+(double)f[2*i]*(double)f[2*i]
      mel_energy(mel,rf);
      mel_cepstrum(mel);
      vector_quantize();
   }
}

/******************************************************************************
 * This function takes 256 points from sampled data using Hamming window and *
 * put zero in remaining position to implement 2048 points FFT               *
 ******************************************************************************/
void shift_window(n)
int n;
{
   int i;

   for (i=0; i<N; i++)
   {
      window[i]=speech_data[n]*(0.54-0.46*cos(2*PI*i/N));
      n++;
   }
}

/******************************************************************************
 * This routine use radix-2, 2048 points FFT to implement short term DFT     *
 ******************************************************************************/
#define SWAP(a,b) tempr=(a);(a)=(b);(b)=tempr

void stdft(data,nn,isign)
float data[];
int nn,isign;
{
        int n,mmax,m,j,istep,i;
        double wtemp,wr,wpr,wpi,wi,theta;
        float tempr,tempi;

        n=nn << 1;
        j=1;
        for (i=1;i<n;i+=2) {
                if (j > i) {
                        SWAP(data[j],data[i]);
                        SWAP(data[j+1],data[i+1]);
                }
                m=n >> 1;
                while (m >= 2 && j > m) {
                        j -= m;
                        m >>= 1;
                }
                j += m;
        }
        mmax=2;
        while (n > mmax) {
                istep=2*mmax;
                theta=6.28318530717959/(isign*mmax);
                wtemp=sin(0.5*theta);
                wpr = -2.0*wtemp*wtemp;
                wpi=sin(theta);
                wr=1.0;
```

```
                wi=0.0;
                for (m=1;m<mmax;m+=2) {
                        for (i=m;i<=n;i+=istep) {
                                j=i+mmax;
                                tempr=wr*data[j]-wi*data[j+1];
                                tempi=wr*data[j+1]+wi*data[j];
                                data[j]=data[i]-tempr;
                                data[j+1]=data[i+1]-tempi;
                                data[i] += tempr;
                                data[i+1] += tempi;
                        }
                        wr=(wtemp=wr)*wpr-wi*wpi+wr;
                        wi=wi*wpr+wtemp*wpi+wi;
                }
                mmax=istep;
        }
}

#undef SWAP

/******************************************************************************
 * This routine computes the MEL-frequencies, then computes the critical     *
 * band energy and put these values in the same array.                       *
 ******************************************************************************/
void mel_energy(mel,f)
double mel[];
double f[];
{
  double ratio,r;
  int i,j;

  for (i=1; i<=20; i++)
  {
     ratio=1.0/(freq[i]-freq[i-1]);
     mel[i]=0.0;
     for (j=1; j<(freq[i]-freq[i-1]); j++)
     {
        r=ratio*j;
        mel[i]=mel[i]+r*r*f[freq[i-1]+j]*f[freq[i-1]+j];
     }
     ratio=1.0/(freq[i+1]-freq[i]);
     for (j=0; j<(freq[i+1]-freq[i]); j++)
     {
        r=1-ratio*j;
        mel[i]=mel[i]+r*r*f[freq[i]+j]*f[freq[i]+j];
     }
     mel[i]=log10(mel[i]);
  }
}
/******************************************************************************
 * This routine computes MEL-based cepstral coefficients with critical band *
 * filtering.                                                               *
 ******************************************************************************/
void mel_cepstrum(mel)
double mel[];
{
   int n,k;
   double a;

   for (n=1; n<=MO; n++)
   {
      c[n]=0.0;
      for (k=1; k<=20; k++)
      {
         a=n*(k-0.5)*PI/20.0;
         c[n]=c[n]+mel[k]*cos(a);
```

```
      }
   }
}

/***************************************************************************
 * This routine vector quantizes the cepstrum parameters with respect to the *
 * given codebook.                                                           *
 ***************************************************************************/
void vector_quantize()
{
int i,index1,index2,vq;
double idm1,idm2;
double euclidean_dist_meas();
int level_index();

index1 = 0;
index2 = 1;
idm1 = euclidean_dist_meas(codebook[index1]);
idm2 = euclidean_dist_meas(codebook[index2]);
if( idm1 > idm2 )
   index1 = index2;

for(i=1; i<NLEVELS; i++)
   {
   index1 = (index1 - level_index(i))*2 + level_index(i+1);
   index2 = index1 + 1;
   idm1 = euclidean_dist_meas(codebook[index1]);
   idm2 = euclidean_dist_meas(codebook[index2]);
   if( idm1 > idm2 )
      index1 = index2;
   }
vq = index1 - LEVELINDX;
fprintf(outfile,"%d\n",vq);
count=count+1;
}

/***************************************************************************
 * This routine calculates which vector to compare next in the codebook once *
 * a vector index in the previous level is given                            *
 ***************************************************************************/
int level_index(k)
int k;
{
int num;
num = (int)pow((double)2,(double)k) - 2;
return num;
}

/***************************************************************************
 * This routine calculates the Euclidean Distance between the cepstrum      *
 * parameters and the one in the codebook.                                  *
 ***************************************************************************/
double euclidean_dist_meas(array)
double array[];
{
  int i;
  double idm;

  idm=0.0;
  for (i=0; i<MO; i++)
     idm=idm+(c[i+1]-array[i])*(c[i+1]-array[i]);
  return idm;
}

/***************************************************************************/
void codebook_entry()
```

```
{
FILE *infile3;
int i,j,k,m;
char input11[80],input12[80];
void extractword();

infile3 = fopen(CODEFILE,"r");

printf("\nReading %s\n",CODEFILE);
m=0;
for(i=1; i<=NLEVELS; i++)
    {
    fgets(input11,80,infile3);
    printf("%s",input11);
    for(j=0; j<(int)pow((double)2,(double)i); j++)
        {
        fgets(input11,67,infile3);
        fgets(input12,80,infile3);
        extractword(input11,input12,m);
        m++;
        }
    }
fclose(infile3);
}

/*************************************************************/
void extractword(in1,in2,m)
char in1[80],in2[80];
int m;
{
    int i,j,k;
    char temp1[30],temp2[30],temp3[10],temp4[30];
    for( j=0; j<30; j++)
    {
       temp1[j] = '\0';
       temp2[j] = '\0';
       temp4[j] = '\0';
    }
    for( j=0; j<10; j++)
    {
       temp3[j] = '\0';
    }
for(i=0; i<6; i++)
    {
    k=0;
    for(j=i*11; j<(i+1)*11; j++)
        {
        temp1[k] = in1[j];
        k++;
        }
    codebook[m][i] = atof(temp1);
    }
for(i=0; i<4; i++)
    {
    k=0;
    for( j=i*11; j<(i+1)*11+1; j++)
        {
        temp4[k] = in2[j];
        k++;
        }
    codebook[m][i+6] = atof(temp4);
    }
}
/******************************************************************************
 * This routine computes the MEL-frequencies.                                 *
 ******************************************************************************/
```

```
void mel_freq()
{
  double interval,scale,mel_f;
  int i;

  interval=10000.0/FFT;
  scale=(log10(5000.0)-3)/11.0;
  freq[0]=0;
  for (i=1; i<11; i++)
  {
     mel_f=100.0*(i);
     if ( fmod(mel_f,interval) < interval/2 )
        freq[i]=floor(mel_f/interval);
     else
        freq[i]=floor(mel_f/interval)+1;
     mel_f=3.0+scale*i;
     mel_f=pow(10.0,mel_f);
     if ( fmod(mel_f,interval) < interval/2 )
        freq[i+10]=floor(mel_f/interval);
     else
        freq[i+10]=floor(mel_f/interval)+1;
  }
  mel_f=3.0+scale*11;
  mel_f=pow(10.0,mel_f);
  if ( fmod(mel_f,interval) < interval/2 )
     freq[21]=floor(mel_f/interval);
  else
     freq[21]=floor(mel_f/interval)+1;
}
```

D Program Listing: Codebook Generating Program

This program produces a seven-level, binary tree codebook for cepstral parameters. The input of this program is a large file which consists of cepstral parameters of all the words spoken by one speaker in the TI-46 or Grandfather word list.

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include <math.h>

#define MO              10    /* Model Order of Cepstrum */
#define NLEVELS          7   /* number of levels in binary codebook */
#define SYMBOL         128   /* 2^NLEVELS */
#define CEP          75000   /* number of cepstrum parameters */

int    group[CEP][2],change;
long   total_count,counta,countb;
double table[CEP][MO];
double centroid[NLEVELS+1][SYMBOL][MO];
FILE   *outfile;}

/*****************************************************************************
 * Program name: cdbkgen.c                                                   *
 * Command     : cdbkgen                                                     *
 * Description : generate a N-level codebook by using the cepstral analysis  *
 * Date        : August 14, 1990                                             *
 *****************************************************************************/
main()
{
   int    i,j,k,level,nt;
   int    aa,bb,cc;
   long   m;
   long   readcode();
   double distance();
   void   separate();
   void   compute_centroid();
   void   perturb();

   outfile = fopen("codebd26.dat","w");
   total_count = readcode();
   printf("total_count=%ld\n",total_count);
   for (m=0; m<total_count; m++)
      group[m][0]=group[m][1]=0;
   compute_centroid(0,0,0);
   printf("first centroid is %10.6f\n",centroid[0][0][0]);
   for(level=0; level<NLEVELS; level++)
   {
      for(i=0; i<(int)pow((double)2,(double)level); i++)
      {
         perturb(level,i,centroid[level][i]);
         nt=0;
         do
         {
            change=0;
            separate(level,i);
            nt++;
            compute_centroid(level+1,2*i,i);
            compute_centroid(level+1,2*i+1,i);
            printf("level=%d,symbol=%d,iteration=%d,counta=%ld,countb=%ld\n",level,
         } while (change==1);
      }
      for (m=0; m<total_count; m++)
         group[m][0]=group[m][1];
   }
   for(i=1;i<=NLEVELS;i++)
   {
      fprintf(outfile,"level  %u .......\n",i);
      for(j=0; j<(int)pow((double)2,(double)i); j++)
      {
         for(k=0;k<MO;k++)
            fprintf(outfile,"% 10.6f ",centroid[i][j][k]);
```

```
        fprintf(outfile,"\n");
      }
   }
   fclose(outfile);
}

/****************************************************************************
 * This procedure reads the cepstral parameters file and put these vectors *
 * into an array.                                                          *
 ****************************************************************************/
long readcode()
{
   FILE *infile;
   int j,k;
   long i;
   char code[20];

   infile = fopen("bd26.dat","r");
   i=0;
   while (feof(infile)==0)
   {
      for (j=0; j<MO; j++)
      {
         fscanf(infile,"%s",code);
         table[i][j]=atof(code);
      }
      i++;
      fscanf(infile,"\n");
   }
   fclose(infile);
   return i;
}

/*****************************************************************************
 * This procedure computes the centroid in a cluster                         *
 *****************************************************************************/
void compute_centroid(level,symbol,now)
int level,symbol,now;
{
   int i;
   long j,k;

   k=0;
   for (i=0; i<MO; i++)
   {
      centroid[level][symbol][i]=0.0;
      for (j=0; j<total_count; j++)
         if ((group[j][1] == symbol) && (group[j][0]==now))
         {
            if (i==0)
               k=k+1;
            centroid[level][symbol][i]=centroid[level][symbol][i]+table[j][i];
         }
      centroid[level][symbol][i]=centroid[level][symbol][i]/k;
   }
}

/*****************************************************************************
 * This procedure splits the centroid into 2 vectors                         *
 *****************************************************************************/
void perturb(level,symbol,vectora)
int level;
int symbol;
double vectora[MO];
{
   register int j;
```

```
   for (j=0; j<MO; j++)
   {
      centroid[level+1][symbol*2][j]=vectora[j]*1.01;
      centroid[level+1][symbol*2+1][j]=vectora[j]*0.99;
   }
}

/*****************************************************************************
 * This procedure separates one group into 2 clusters                        *
 *****************************************************************************/
void separate(level,symbol)
int level,symbol;
{
   long   i;
   double dist1,dist2;
   double distance();

   counta=countb=0;
   for (i=0; i<total_count; i++)
   {
      if (group[i][0]==symbol)
      {
         dist1=distance(table[i],centroid[level+1][2*symbol]);
         dist2=distance(table[i],centroid[level+1][2*symbol+1]);
         if ( dist1 < dist2 )
         {
            if (group[i][1] != 2*symbol)
               change=1;
            group[i][1]=2*symbol;
            counta=counta+1;
         }
         else
         {
            if (group[i][1] != 2*symbol+1)
               change=1;
            group[i][1]=2*symbol+1;
            countb=countb+1;
         }
      }
   }
}

/*****************************************************************************
 * This procedure computes the Euclidean distance                            *
 *****************************************************************************/
double distance(vector1,vector2)
double vector1[],vector2[];
{
   int i;
   double dist;

   dist=0.0;
   for (i=0; i<MO; i++)
      dist=dist+(vector1[i]-vector2[i])*(vector1[i]-vector2[i]);
   return dist;
}
```