Performance of the LEMS HMM speech recognizer with PLP features and with multiple-window log-spaced DFT spectra

by

Athanassios Protopapas

B.Sc., University of Patras, 1991

M.Sc., Brown University, 1993

Thesis

Submitted in partial fulfillment of the requirements for the

Degree of Master of Science in the Division of

Engineering at Brown University

May 1995

This thesis by Athanassios Protopapas

is accepted in its present form by the

Division of Engineering as satisfying the

thesis requirement for the degree of Master of Science

Date ....

Harvey F. Silverman

Approved by the Graduate Council

Date ....

Acknowledgments

I would like to thank the Dean of Engineering, Professor Harvey F. Silverman, for introducing me to the speech group and agreeing to devote some of his valuable time to supervise my work, even though I am a member of another department. I am grateful for his support and constructive criticism, always on practical and down to earth engineering grounds. I am also indebted to the graduate students in the speech group, in particular Daniel Mashao, Yoshi Gotoh, and John Adcock for their constant help and willingness to share their concerns and their code with an outsider. This work would not have been possible without their directions and programs. Speaking of programs, Dr. Hynek Hermansky kindly provided the C code for PLP linear predictive processing saving me a lot of time and trouble. Finally, I would like to thank the rest of the Engineering graduate students who tolerated my impositions on computer time and disk space in a place where these are goods in exceptionally high demand, and the Cognitive Science graduate students who kindly volunteered to serve as subjects in the listening experiment.

Contents

List of Figures

List of Tables

Introduction

Interest in human speech perception has led to the investigation of some options for machine speech recognition using the development tools of the Laboratory for Engineering Man/Machine Systems (LEMS) at the Division of Engineering at Brown University. The results of using two kinds of cepstral features that can be extracted from the speech signal are described below. The Perceptual Linear Prediction (PLP) method has recently become popular in the speech recognition field, but had not been tested at LEMS before; original processing used LPC-based cepstra. Features based on Discrete Fourier Transform (DFT) spectra have improved recognition rates, due mainly to recent improvements in nonlinear sampling of spectral values; exploration of these is still under way. In the following chapters the LEMS system and speech database are first described briefly and then the system's performance is tested with these two kinds of features. The PLP-based features and the resulting recognition performance are described in Chapter 2. In Chapter 3 a spectral representation is proposed that is based on DFTs computed on overlapping speech segments of various lengths in order to preserve short-lived phenomena in the higher frequencies and in Chapter 4 conclusions and directions for future research are given.

The primary motivation for this research comes from the realization that word recognition is such a difficult task for computers and so effortless for humans and that significant advances in the machines' performance can only be expected to arise out of systematic investigation of the human auditory system and of the charac-

teristics of the speech waveform. Since speech has evolved along with the human auditory system, one expects that the two are matched to a very high degree and that research in both will provide complementary pieces of information and directions for engineering artificial speech-recognition systems. The current state of the art of machine speech recognition, impressive as it may be by engineering standards, is substantially inferior to human speech perception performance, and it may be argued that a paradigm shift in engineering research is necessary. However, all options have not been examined yet, and the limitations of current techniques should serve as indications of where subsequent investigations should be directed.

The two methods described below, although within the realm of current engineering developments, claim to approximate human auditory properties or match properties of speech a little better than those traditional methods that have been developed primarily with mathematical considerations in mind. The results of their application to a hidden Markov model (HMM) speech recognizer will show whether such improvements in the speech processing front-end to a statistical recognition system may confer a significant advantage for the system's performance. Lack of substantial improvements, however, should not be taken as a clear indication that more auditory-like speech parameterizations are fruitless for machine speech recognition; that may reflect the fundamental inability of current statistical systems to capture some essential properties of human speech perception processes, or that such systems are incompatible with speech representations that retain particular kinds of linguistically relevant information.

The task of unconstrained word recognition in naturally occurring speech is at the moment beyond the capabilities of current computational procedures and available hardware. Whether that reflects basic misconceptions about human speech and its perception by humans or processing and storage limitations is not clear at this point. However, in order to work on the problem at hand, it is not merely convenient, but

imperative, to constrain it so that it becomes manageable and within the scope of our equipment. Restricted formulations of the problem of speech recognition may involve constraints on the number of talkers whose speech can be handled by the system, the number of words that can be recognized, the rate at which the words are spoken, the acoustic environment and recording equipment used, or any combination thereof. Thus it is possible to attain a very high level of performance with a suitably restricted vocabulary consisting of words spoken one at a time by a single talker only. Relaxation of these constraints leads to deterioration of performance and, in consequence, to very important and interesting engineering problems. These problems change when different variables are studied. For example, in a large-vocabulary system, storage of all possible words and practical algorithms for determining which one is most likely present in the speech stream are primary concerns. On the other hand, in a small-vocabulary talker-independent system, the system's ability to adapt to different talkers, or to bypass their differences becomes most important. In the following, discussion is restricted to the characteristics of the LEMS systems; a review of the problems of other configurations and the techniques being employed to solve them can be found in Rabiner & Juang (1993) [10].

Chapter 1

The LEMS recognizer and speech database

Before describing the particular methods that were tried in order to improve performance, it was necessary to obtain some measures of the system's previous level of performance, with an LPC-based feature analysis method, and of the best possible performance, that of human listeners. In this chapter, the characteristics of the LEMS recognizer are discussed first and the speech database is described in order to evaluate the problem at hand. Then a reference performance level is presented, obtained with a human-listening experiment, and a baseline performance level is obtained with an HMM trained on LPC features using existing LEMS signal processing code. Other signal processing techniques currently being investigated by LEMS students are still in development and results from them will not be reported here.

1.1 The recognizer

The speech research group in the Laboratory for Engineering Man-machine Systems (LEMS) has developed a testing platform for various signal processing and statistical learning algorithms based on a hidden Markov model. The word recognition problem

has been constrained to recognition of "alpha-digit" words, i.e., a vocabulary consisting of the 26 letters of the English alphabet ("a" to "z"), the ten digits ("zero" to "nine"), and the words "period" and "space." For practical purposes this is a very important word set, as it allows (indirectly) for recognition of any kind of text as long as it can be spelled out. For purposes of engineering research it is also a very interesting set, because the similarities between several words in the vocabulary lead to the realization of important problems in speech recognition, solutions to which may be extended to more general word-recognition systems.

Besides dealing with a difficult to distinguish word set, the LEMS system has also relaxed the constraint of words being spoken in isolation, thus being a true connected-speech recognition system. This makes the problems of word identification much harder because there are no clearly discernible boundaries between individual words. In fact, word edges are often merged with those of adjacent words making the task of word segmentation impossible on an acoustic basis. In order for a word to be segmented out of the speech stream it must first be identified. Relying on silence segments is unreliable not only because of the words' being spoken in a connected manner; the gaps in the acoustic energy that exist in the speech signal often correspond to word-internal silence, such as that of unvoiced stop consonants (i.e., [p], [t], [k], and [č]). The approach taken in the LEMS system is to evaluate entire "utterances," i.e., word strings, against the spoken segment, taking into account the probability of the whole string being present on the basis of combined word probabilities.

Each word-model is defined as having a specific number of "states," each of which is associated, during training, with sets of acoustic features based on processing of the speech stream. The hidden Markov model (HMM) is trained on the transition probabilities between such states as well as on their duration (explicit duration modeling, see [7]). The feature vectors used by the system may specify up to 28 different

feature values, usually occupied by 12 cepstral coefficients, 12 first order differences, and up to four energy and energy difference features. Both discrete observations and continuous observations may be modeled. In the discrete system each set of features is quantized into a vector codebook by k-means clustering and the training is only taking the centroids of the clusters into account (via the codebooks). In the continuous system the actual feature values are used, which usually results in a performance improvement by a few percent at the cost of significantly higher computational requirements.

1.2 The speech database

The speech group in LEMS has created a speech database containing a large number of continuously spoken alpha-digit utterances. Each of 120 male and female talkers (all native American English speakers) spoke up to 48 strings of 10–20 words (letters and digits) each, for a total of 8 hours of speech. The recordings were subsequently divided into 3 files per talker of up to 16 utterances each, with appropriate header information about the talker and the validity and content of each utterance. In the complete set, 80 talkers (about 5 hours of speech) are used for training and 20 talkers are used for intermediate testing (to adjust system parameters, etc.) The speech from the remaining 20 talkers makes up the final testing set. The speech has been sampled at 16 KHz and quantized to 16 bits with PCM encoding for ease of application of a variety of processing algorithms.

In the work reported here, a subset of the database has been used, comprising 1942 utterances from 60 talkers, including 36 male and 24 female talkers. 1276 utterances from 45 talkers (27 male and 18 female) make up the training set and 666 utterances from the remaining 15 talkers (9 male and 6 female) form the testing set. There are, on average, 12 words in each utterance. The reasons a smaller set was used were limitations on CPU time and disk space. Since the subset used was

randomly sampled from the entire database, including many repetitions of all words and several talkers from each sex, it was not expected that this restriction would result in performance significantly different from using the entire set.

1.3 Human word recognition performance

Before running the LEMS recognizer on the testing set, ceiling performance level had to be established, as defined by human performance. One may argue that strings of random letters and digits, being devoid of content, is not the kind of people are used to perceiving and they should not be expected to excel at it. However, humans are the best speech recognition machines around. Therefore one should not expect any machine recognition performance to exceed human performance. The results of a listening test should indicate the recognition performance for which one should strive. Since there is no sentential context of any sort to aid human perception, and it is known in advance that only letters, digits, "space," and "period" may be spoken, the only advantage for human listeners is their lowest level speech perception system, which our recognizer is built to emulate. This would not be true if, for example, meaningful grammatical sentences were the utterances to be recognized. On the other hand, since most letter strings in the LEMS database form real English words, humans can use this information to anticipate upcoming letters. For this reason, the present listening test is not a strictly appropriate control test and the validity of the results cannot be precisely assessed.

1.3.1 Experimental method

In order to conduct this experiment the time data files were converted to raw audio format, downloaded to a Silicon Graphics workstation in the Department of Cognitive & Linguistic Sciences, and the files saved as single-utterances in Audio Interchange

File Format (AIFF) so that they could be recognized and played out by the native audio tools. A computer program was written to control the experiment, taking into account the following issues:

- Because the digits and letters are spoken rapidly and continuously, this is a very hard task, even for humans, and requires the listeners' undivided attention. Even so, it is frequently impossible to hear and write down all the words in an utterance, particularly for the utterances of the fastest talkers. Thus, each utterance was presented twice in order for the subjects to fill in whatever they missed the first time. This did not confer any unfair advantage for the human listeners because subjects were instructed to complete the utterance on the second pass but not to correct misheard words. Furthermore, the computer had no restriction on the amount of time an utterance was examined in order for the most probable interpretation to be identified; human listeners' auditory memory, however, is very limited both in capacity and in duration, so the tasks would not be comparable if humans were only allowed a "passing" interpretation and not a second presentation.

- Because humans adjust to the particular characteristics of individual talkers, and since it was such a hard task to begin with, trials were blocked by talkers, i.e., all utterances from one talker were presented before proceeding to the next talker. Otherwise subjects would not be able to cope with the speed of speech and talker variation at the same time, resulting in artificially high error rates. Again, this was not an unfair advantage because the computer is also tested talker by talker, although this has absolutely no effect on its performance.

The subjects were three graduate students in the Dept. of Cognitive & Linguistic Sciences who volunteered their participation. They were instructed to write down the words they heard in the order they heard them. They were informed that only the letters "a" to "z," the digits "0" to "9," and the words "space" and "period" were

possible words in this set. They had to press a button on the computer keyboard to initiate a trial, and then press it again to repeat the same utterance in order to complete their response. They were asked not to make corrections of misperceived words on the second pass. After presentation of each block of 14 to 16 utterances (all from one talker) the subjects had the option of continuing the experiment with the next talker block or quitting.

1.3.2 Results and Discussion

The three subjects that participated in this experiment identified the words in 343 utterances, for a total of 4697 responses. The total number of errors was 45, or 0.96%, including 16 substitutions, 16 omissions, 9 insertions, and 4 reversals of order. The substitutions including misperceptions of "t" as "d" or "g," misperceptions of "8" as "a," and a few others which occurred only once, including an instance of an "m" that was identified as "n" and once vice versa. Two substitutions were caused by recording or file creation errors that resulted in the initial word being only partially present (thus a supposed "b" was heard as "e").

Clearly, humans are quite good at this, although a one percent error rate is rather high and reflects the difficulty of the task. Two things need to be noted here: First, people do not normally utter letters and digits at this rate. When no sentential context is present to aid word perception, people speak slowly and try to articulate more clearly. The characteristics of the LEMS speech database made this a hard task. Second, humans had the advantage of realizing that most of the letter word strings formed real words. This way, subjects may have anticipated the following word (i.e, letter name) to continue or complete a word. Although it is impossible to precisely quantify the extent of that with such a small overall error rate, it should be noted that the majority of the errors were made in utterances forming real words, therefore the extent of anticipation was limited. On the other hand, if the words were spoken

at a slightly slower rate, there probably would have been almost zero omissions and insertions, bringing the total percentage of errors down. So, overall, the advantages of being human may outweigh the disadvantages in this task, but it is not possible to perform a better controlled experiment given the nature of the speech database. It is concluded that the ceiling performance sought for our computer recognizer should approach 99% in order to be comparable to human performance.

1.4 Recognizer training and testing standards

The procedure and the speech file sets used for this test have been kept constant for all tests using different methods and parameters. In particular, the same training and testing sets already described (the `small_train` and `small_test` set, respectively) were always used, and the same small subset of the training set was always used to create the vector quantization codebooks. The number of coefficients varied depending on the order of the model tested, but this was only important when generating the features and when using the continuous HMM recognizer. The discrete model always used three vector codebooks. The recognizer was trained for 30 iterations using gamma-duration modeling in the discrete model tests and Poisson-duration modeling in the continuous model tests.

Testing was done with the `batchrec` program, which takes into account the number of words correctly recognized (C), the number of substitutions (s), the number of deletions (d), and the number of insertions (i), in relation to the total number of words (T) in the testing set, and calculates the percent correct (c) by the formula

$$c = \frac{C - i - d - s}{T} \times 100\%.$$

Since the words are not presented to the recognizer pre-segmented, but on a whole utterance basis, and the observation probabilities are computed with respect to the entire observation string, some form of alignment is necessary for the appropriate

calculation of insertions and deletions.

A final test was conducted to examine sensitivity of the methods to the recording conditions, using a small set of utterances (90 utterances total, from 3 talkers) that were recorded both with the usual head-mounted microphone, that was also used for the standard training and testing sets, and with beamforming from a microphone array, at the same time. Thus the exact same utterances were available in two very different spectral environments, and a robust recognizer should be able to handle them without substantial performance deterioration, since there was no difference in speech content. Naturally, the speech from the head-mounted microphone sounds like the speech in the training and testing sets. The speech from the microphone array sounds distant and is much softer. Although there are no quantitative assessments of the differences between the beamformed recordings and the head-mounted microphone recordings, the beamformed data were subjectively judged to be very intelligible. The file lists and the shell scripts that were written to run the LEMS programs can be found in the Appendix.

1.5 Recognition performance using LPC features

Traditionally, linear predictive coding (LPC) has been the signal processing method of choice for speech recognition, as it permits significant compression of the relevant spectral information through a stable and simple algorithm. Since the procedure for "efficient encoding of speech" was first introduced by Atal in the early 1970s [1, 2], LPC has become a standard speech processing tool with impressive performance in several fields. Although it can be a general waveform modeling technique (with some constraints on the spectral properties of waveforms), much of its success in speech processing is probably due more to its algorithmic simplicity and stability than to its dubious vocal-tract-modeling capabilities that are derived using wildly simplified equations and solutions.

It is known that vowel production may be characterized to a good approximation by an all-pole model that is driven by a quasi-periodic pulse train. To a lesser extent, some consonants may also be approximated by a similar all-pole system that is excited by some random noise (with a flat spectrum). Because the production of most consonants involves source excitation generated not at the end of the vocal tract but at some intermediate point, the air flow in the point of constriction is not laminar and the LPC approximation to the vocal tract area function is often not very satisfactory. In such cases, higher orders of LPC may be required to capture essential characteristics of the waveform, but then the LPC coefficients lose their physical meaning. Nasal sounds, which are produced by two coupled resonators (oral and nasal cavities), are among the most difficult to handle with LPC, because the zeroes in the correct vocal tract transfer function can never be perfectly accommodated by any low-order all-pole model.

Using methods based on Linear Predictive Coding (LPC) coefficients, previously reported recognition performance of the LEMS recognizer have ranged between 80% and 90%. In particular, Michael M. Hochberg (1992), using the discrete HMM, achieved overall performance rates of 81.4% with gamma duration modeling, 81.9% with Gaussian duration modeling, and 84.3% with constrained Gaussian duration modeling [7]. The highest overall performance achieved with the continuous HMM and LPC-based features has been 89.4% [8]. Although these results give an idea of the levels of performance any subsequent methods should be judged against, a more appropriate comparison can be made if all methods are measured under identical conditions. In particular, it is important to compare models that have been trained and tested on the same speech data. For this reason, the LEMS recognizer was trained with a discrete model based on the LPC features of `type6`, as they are referred to in the processing selection setting of the feature generating program. These features are zero-mean cepstral coefficients derived from LPC coefficients that

A-set Confusion Matrix:

	a	h	j	k	8
a	83	3	6	13	8
h	3	90	2	1	3
j	.	.	86	3	.
k	0	.	1	80	.
8	0	6	.	.	77
~	14	1	5	4	11

E-set Confusion Matrix:

	b	c	d	e	g	p	t	v	z	3
b	50	1	5	0	1	8	.	8	1	1
c	.	62	.	1	.	.	1	1	6	.
d	14	3	41	1	1	2	0	6	1	.
e	7	9	6	89	15	10	1	5	2	.
g	.	1	2	.	54	2	.	.	1	.
p	11	.	3	0	.	65	0	.	.	.
t	.	3	30	1	18	7	94	6	1	1
v	11	1	1	1	.	6	.	70	1	1
z	.	15	.	.	.	.	.	2	88	1
3	.	1	.	.	.	.	0	.	.	95
~	6	3	13	7	10	1	3	3	2	1

Nasal/Glide Confusion Matrix:

	l	m	n	7	9
l	84	11	2	.	.
m	3	43	13	1	2
n	1	32	69	2	1
7	.	.	.	96	.
9	.	.	.	.	92
~	11	14	16	1	4

Table 1.1: Confusion matrices for the most difficult word subsets, with LPC `type6` features used to train the discrete recognizer.

are calculated from speech frames 40 ms long (not preemphasized) using the Durbin algorithm with mel-type frequency warping (Constantinides frequency warping technique, warp coefficient=0.4).

The overall performance of the discrete model was 73.13%. Table 1.1 shows the confusion matrices for the most difficult sets. Recognition of individual words ranged between 40.6% and 99.4%, with "d," "m," "b," and "g" being the most poorly recognized (40.6%, 43.2%, 50.0%, and 53.7%, respectively). The best recognized words were "four," "six," "space," and "seven" (99.4%, 99.3%, 97.9%, and 96.2%, respectively). Talker accuracy also varied widely, from 54.6% (male talker `eew`, average of all three test files) to 87.7% (male talker `gms`, average of all three test files). Overall, the utterances of the female talkers were recognized, on average, as well as those of the male talkers (73.9% vs. 73.1%).

The discrepancy between these results and previously reported performance lev-

els using LPC processing should be mainly attributed to the selection of the speech data that are used for k-means vector clustering (to create the vector quantization codebooks) and, to a lesser extent, to performance fluctuations due to different initial random weights of the HMM. To show the large effect of the vector clustering training set, the recognizer was trained on `type6` features using codebooks available in the LEMS archives from previous trainings. The bottom-line recognition performance rose to 80.67%, with individual word recognition ranging from 35.8% to 99.5% and talker recognition ranging between 64.4% and 91.2%. To verify the dependence on initial training weights, the discrete model was trained for a second time with the same standard codebooks that were used throughout this thesis and bottom-line recognition performance was 72.21%, i.e., almost 1% less than after the previous training. It is concluded that the recognition performance statistics reported herein are accurate to at most 1% and this conclusion is valid only when identical codebooks are used. This weakens the validity of comparisons with previous work and it is recommended that only the results reported here are taken into account when comparing processing techniques.

1.5.1 Performance of the continuous model

Using the best-performing discrete model obtained with the standard procedure as a starting point, a tied-mixture full-covariance HMM was trained for 30 iterations. Table 1.2 shows the confusion matrices for the most difficult word sets. Note the substantial improvement in almost all cases. The bottom-line recognition performance of the continuous model was 77.27%, with recognition performance of individual words ranging from 58.4% to 100.0%. The most poorly recognized words were "d," "b," and "m," (recognized at 58.4%, 60.2%, and 62.5%, respectively) and the best recognized ones were "six," "x," "seven," and "four" (recognized at 100.0%, 99.4%, 99.4%, and 99.4%). Performance by talker ranged from 62.2% for the most difficult talker (male

A-set Confusion Matrix:

	a	h	j	k	8
a	86	7	2	4	9
h	1	90	1	.	2
j	1	.	95	4	.
k	0	.	1	92	.
8	0	1	1	.	84
~	12	1	2	.	5

E-set Confusion Matrix:

	b	c	d	e	g	p	t	v	z	3
b	60	3	2	1	1	7	.	7	1	1
c	.	73	1	1	3	.	1	.	7	.
d	9	2	58	2	.	.	0	7	.	.
e	.	6	4	82	6	3	0	2	1	.
g	1	3	8	1	73	7	4	4	.	.
p	15	.	4	0	.	77	0	1	.	.
t	1	.	17	0	7	2	88	1	.	.
v	7	1	.	1	.	2	.	69	1	.
z	.	11	1	0	.	.	.	3	91	.
3	1	.	.	.	.	.	1	1	.	99
~	5	2	5	11	9	2	5	5	.	1

Nasal/Glide Confusion Matrix:

	l	m	n	7	9
l	90	14	1	.	1
m	3	62	15	.	.
n	.	19	77	.	.
7	.	.	.	99	.
9	.	.	.	.	99
~	7	5	8	1	1

Table 1.2: Confusion matrices for the most difficult word subsets, with LPC `type6` features used to train the continuous model.

talker `eew`, average of all three test files) to 90.8% for the talker easiest to recognize (male talker `gms`, average of all three test files). Overall, speech from female talkers was recognized on average somewhat better than speech from male talkers (79.8% vs. 75.8%). In conclusion, it should be noted that the continuous model performed better than the discrete model, as expected, but the bottom-line recognition performance still lagged behind results previously reported by the LEMS group. The reason may be that the initial tied-mixture model was based on non-optimal vector quantization that hindered further training.

1.5.2 Sensitivity to the recording conditions

As outlined in Section 1.4, beamformed data and simultaneous recordings with the head mounted microphone were used to assess the sensitivity of the LPC technique

A-set Confusion Matrix:

	a	h	j	k	8
a	45	7	.	.	20
h	2	64	8	.	.
j	.	7	75	23	.
k	.	.	.	23	.
8	.	.	.	.	20
~	53	21	17	54	60

E-set Confusion Matrix:

	b	c	d	e	g	p	t	v	z	3
b	67	4	7	5	.	.	.	47	5	.
c	.	65	.	.	.	.	3	7	62	.
d	.	.	21	.	.	.	6	7	.	.
e	17	22	21	68	14	10	17	.	24	6
g	.	4	29	18	50	10	26	.	.	.
p	8	.	7	5	7	70	.	13	.	.
t	.	.	7	.	.	.	31	.	.	.
v	.	4	.	.	.	10	.	20	.	.
z	.	.	.	.	.	.	.	.	10	.
3	.	.	.	1	.	.	.	.	.	85
~	8	.	7	3	29	.	17	7	.	9

Nasal/Glide Confusion Matrix:

	l	m	n	7	9
l	71	15	4	4	3
m	.	38	12	.	11
n	6	23	79	.	16
7	.	.	.	96	.
9	.	.	.	.	38
~	23	23	4	.	32

Table 1.3: Confusion matrices for the most difficult word subsets, with LPC `type6` features used to train the continuous model, when testing on the beamformed speech data.

to spectral distortions not related to the speech content. Bottom-line recognition performance of the discrete model (trained in the standard manner) on the high-quality set was 83.5% and on the beamformed data 31.1%. This result is in agreement with earlier reports by the LEMS group [12], where the discrete model trained on high-quality speech was found to perform at 32.8% on the beamformed data. The continuous model achieved a bottom-line recognition performance of 85.8% on the high-quality set and 39.0% on the beamformed set, a small improvement over the discrete model, but still substantially worse than performance when tested with speech similar to that it was trained on. Table 1.3 shows the confusion matrices for the continuous model when tested with the beamformed speech. The deterioration of performance is evident in all words.

In conclusion, regarding recognition performance using LPC processing to derive

the cepstral features, it may be noted that overall the results achieved were worse than other results previously reported, and this discrepancy may be attributed to the choice of vector sets for creating the quantization codebooks. The models' performance was comparable to that previously reported when testing with the smaller sets of simultaneously recorded beamformed and high-quality speech. These results will be used for the evaluation of the new techniques introduced in the next chapters.

Chapter 2

Using Perceptual Linear Prediction

2.1 Advantages and drawbacks of PLP

PLP is a technique developed by Hynek Hermansky that approximates auditory spectra by using "concepts from the psychophysics of hearing" [5]. In other words, PLP is a way to get rid of a lot of spectral information that is unlikely to be useful for speech recognition because it is not retained after human auditory processing, as psychophysical experiments have shown. Discarding such spectral information results in higher recognition rates because there are fewer distracting features left that can prevent statistical generalizations of the linguistically relevant sort. Because of this, PLP is becoming popular in the speech recognition field, and there are even some freeware speech processing software products offering PLP processing capabilities. In comparative studies, PLP has been designated "the method of choice" [3, 4, 6].

PLP is an all-pole model, like LPC, so its advantages and disadvantages are similar to those of LPC. In fact, PLP is nothing more than the LPC of a sound waveform whose spectral characteristics have been transformed to match what is

believed to reflect the characteristics of the human auditory system. As such, PLP addresses the issues that concern the applicability of a linear frequency axis and of uniform amplitude and frequency resolutions in speech perception, but confers no advantage with respect to the inability of all-pole models to correctly approximate speech sounds other than vowels and glides. Therefore, the performance of PLP cannot be expected to be free from the aforementioned shortcomings of LPC that result from its being an all-pole model.

An additional observation regarding PLP is the effect of its short-term nature on the characterization of transient phenomena. Apart from sustained vowels, which are in fact rare in regular speech, linguistically relevant speech events involve short-lived acoustic phenomena, which result from the movements of the various articulators of the vocal tract. Such events include the bursts that occur at the onsets of stop consonants, the formant frequency transitions between phonated segments, etc. Not only are transient phenomena, in principle, outside the scope of all-pole modeling (which is formulated for quasi-stationary signals), but also the time scale of the various interesting speech features is not uniform. For example, the burst of a [b] may precede phonation of the following vowel by a few milliseconds, typically between zero and twenty, whereas forty milliseconds or more signify the presence of a [p] in the context of identical formant transitions. Clearly, a temporal window of 40ms or more will inherently fail to capture the essence of such phenomena, although it may be appropriate for the formant transitions. Since integration over some time period is necessary for the extraction of spectral data, there is also a lower limit to the temporal resolution of a reasonable characterization process.

2.2 Description of the PLP technique

Following the original publication by Hynek Hermansky [5], PLP can be broken down into the following steps: First one weights the speech segment with the usual

Hamming window,

$$W(n) = 0.54 + 0.46 \cos \frac{2\pi n}{N-1},$$

for a time window N samples long. A window length of about 20 ms was suggested in the original publication, but the LPC processing of the LEMS recognizer used 40 ms, so lengths between 10 and 35 ms were tried. The windowed samples are then transformed into the frequency domain (the discrete Fourier transform is computed via the Fast Fourier Transform algorithm) and the short-term power spectrum is computed from the magnitude of the DFT:

$$P(\omega) = \Re[S(\omega)]^2 + \Im[S(\omega)]^2.$$

The frequency axis of the power spectrum is warped into the Bark scale by Schroeder's Bark-Hertz transformation equation

$$\Omega(\omega) = 6 \ln \left\{ \frac{\omega}{1200\pi} + \sqrt{\left(\frac{\omega}{1200\pi}\right)^2 + 1} \right\}$$

and the resulting warped power spectrum is convolved with the power spectrum of a simulated critical band masking curve $\Psi(\Omega)$ (an approximation of physiological auditory filters which are supposed to be of approximately constant shape on the Bark scale). This discrete convolution

$$\Theta(\Omega_i) = \sum_{\Omega=-1.3}^{2.5} P(\Omega - \Omega_i)\Psi(\Omega)$$

reduces the spectral resolution of $\Theta(\Omega)$, which is then downsampled to 1-Bark and preemphasized by a simulated equal-loudness curve

$$\Xi\left(\Omega(\omega)\right) = E(\omega)\Theta\left(\Omega(\omega)\right)$$

to approximate the human sensitivity curve via the function $E(\omega)$.

The resulting spectrum undergoes a dynamic range compression by a cubic root

$$\Phi(\Omega) = \Xi(\Omega)^{0.33},$$

approximating the power law of hearing. Hermansky's claim was that since the aforementioned operations greatly reduce variation in spectral amplitude, the resulting spectrum can be modeled with a relatively low order all-pole model. Such a model can be computed from the first few autocorrelation values of the inverse DFT of the transformed spectrum with a fast recursive algorithm. Finally, the autoregressive coefficients of the all-pole model can be transformed into cepstral coefficients which are known to give stable results in speech recognition, because they are not sensitive to frequency shifts of the spectrum. The number of the parameters in the all-pole model is called *order* of the PLP model, and is the same as the number of cepstral coefficients that are derived from them.

2.3 Recognition performance with the PLP features

The C code for the PLP implementation was kindly provided by Dr. Hermansky himself. Some code had to be written to interface it to the LEMS programs, and now it can be run as part of the LEMS recognizer by a shell script which takes the desired order of the PLP model and the temporal size of the processing window as command line parameters and creates the feature files. The creation of codebooks, quantization of the feature vectors, training, and testing were done in the same manner described above for the LPC processing and with the same files.

2.3.1 Tuning of the processing parameters

In order to determine the optimal PLP model order and processing window length for the LEMS recognizer and speech database, tests were run with many different settings of these parameters. Hermansky noted in his original paper that a 5^{th}-order model gave best results for across-talker recognition. However, he was using a sampling

rate of 8 KHz, therefore those results may not apply to the LEMS setup because it uses 16 KHZ. Having a wider spectral region to model, more poles may be necessary to capture the essentials of the shape over the entire 8 KHz range. Because of the effective reduction in spectral resolution with PLP analysis, the additional spectral structure above 4 KHz may be small. Few speech segments contain information in that region (mostly dental and alveolar fricatives, whose spectra have little energy at the lower frequencies), so not many more that five cepstral coefficients were expected to be necessary. However, because of the presence of consonants that are not well approximated by all-pole models (notably nasals), higher orders may be necessary for good overall recognition performance.

Figure 2.1 shows the overall recognition performance of the LEMS recognizer for different parameter settings of the speech processing module. The performance metric is the "bottom line recognition performance" that is calculated by the `batchrec` program (see Section 1.4). Note that recognition performance appears to be relatively insensitive to the two parameters over a large range of variation. A few repeated trials showed that the accuracy of any given point is about 1.5% because of some dependency of the HMM model on the initial values (random initial transition probabilities). Therefore, the observed pattern should be interpreted with some tolerance. It should also be noted that the overall performance of the LEMS recognizer is very much higher than that reported by Hermansky of his rudimentary recognizer (between 50% and 60% for PLP orders between 5 and 11); this difference is clearly attributable to the sophistication and the statistical power of the HMM. There is also a substantial difference between recognition performance with PLP processing and recognition performance with LPC processing. No PLP parameter combinations produce results worse than those with LPC, and the best overall performance (84.1%) is 11% better than that of `type6`.

From these results, we may conclude that lower orders of PLP analysis are in-

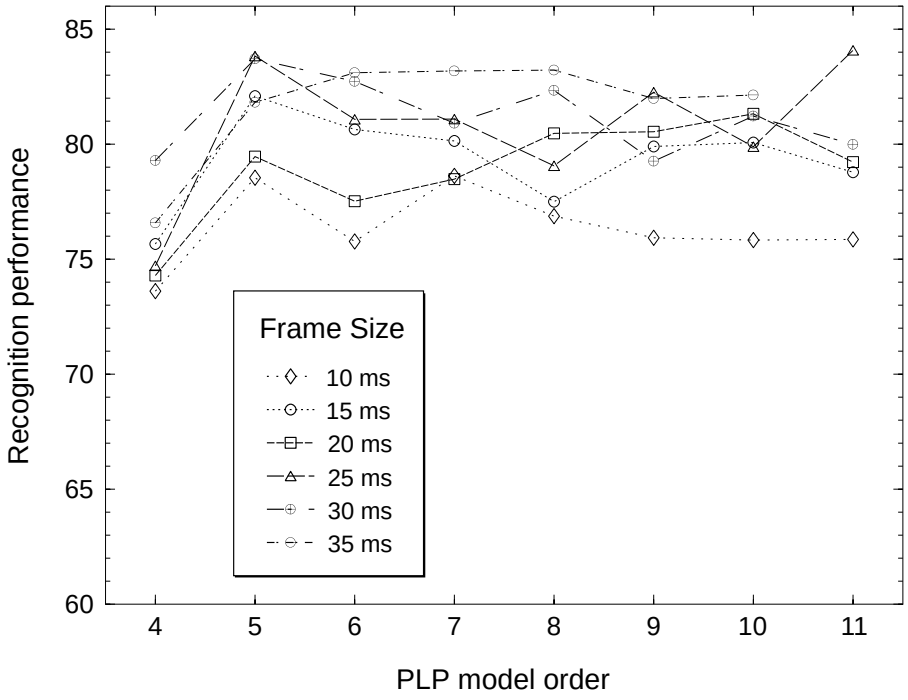

Figure 2.1: Bottom line recognition performance for the discrete LEMS recognizer with PLP processing of the speech signal, for several different model orders and processing window lengths.

deed a little better than higher orders, perhaps because they do not confuse the recognizer with irrelevant variation but retain most of the (linguistically) important spectral information. Although this difference is not very large, it is very important that similar or better recognition performance can be attained with fewer than half the parameters. LPC models previously tested with the LEMS recognizer used 12 cepstral coefficients derived from 14^{th}-order LPC coefficients. While it is known that higher-order LPC coefficients add little to the description of the speech waveform, it is unclear whether significantly fewer cepstral coefficients from LPC parameters could have produced equally high percentages of correct recognition. That is, 26 LPC

parameters (12 cepstral coefficients, their frame-to-frame differences, plus energy and energy difference) are outperformed by 12 PLP parameters (5 cepstral coefficients and their frame-to-frame differences, plus energy and its difference). This is not exactly rigorous for the discrete model, where three codebooks are used in each case (PLP and LPC), but it applies to the continuous models, and it is also important in computational considerations, because the training and the testing files need to be coded into some set of parameters, either for the calculation of the codebooks, or for the training of the model.

With respect to the size of the processing window, it is clear from these results that longer windows give rise to higher performance, probably because the spectrum needs some time for good definition of the salient features. In particular, windows with lengths of the order of a pitch period lead to poor results because of the amplitude variation within each pitch period. For example, for a voice with F_0 around 100 Hz (normal for an adult male talker), processing should be performed on speech segments much longer than 10 ms. Too long processing windows, on the other hand, will wash out critical spectral transitions. Thus it was not considered useful to test with processing windows longer than 35 ms, because the advantage of the 35 ms window over the shorter windows already seemed to be marginal, and longer windows were likely to miss altogether the shortest phenomena. Furthermore, the shorter processing windows confer computational advantages, since fewer points need to be taken into account for the computations.

2.3.2 Performance differences between words

It is of particular interest to examine the recognition performance for each word in order to identify the most problematic cases and attempt to deal with them directly, if performance on those turns out to be discrepant from overall performance. Indeed, observation of the performance broken down by individual words reveals a

common underlying pattern of most of the PLP models used. In particular, the words involving voiced stops (i.e., "b," "d," and "g") or nasals ("m" and "n") are always among the most poorly recognized, often by a wide margin. Also, "p" and "l" are often among the most poorly recognized words, whereas the longer words (all the numbers and "w") are those that are recognized best. Figure 2.2 shows the recognition performance of the six most poorly recognized words, by PLP model order and processing window length.

Overall, these graphs are not very easy to interpret because of the variability between adjacent points: What is, for example, the drawback of 6^{th}-order PLP on 20 ms windows that leads to such a poor performance for the word "d," but not for any other word? And what is wrong with 5^{th}-order processing with an (overall superior) 35 ms processing window that diminishes performance on "g" only? Most likely, these extreme fluctuations are the result of small probability differences in the recognizer that lead to different biases in different models, either because of the initial (random) weights or for other processing reasons. These small biases may lead to large recognition differences when difficult words are tested, although overall performance will not change much. An example is the tradeoff between "m" and "n," which are difficult to discriminate.

Although the individual word graphs are not smooth at all, there is a general pattern indicating that words beginning with stop consonants are favored by lower order models, whereas words ending in nasal consonants are better recognized by higher-order models. It is not at all surprising that higher order processing improves recognition of the nasal sounds—these are the sounds whose spectrum is the most difficult to match with an all-pole model spectrum, so higher order is imperative in order to improve the approximation. It is also not surprising that non-nasal sounds, as well as the rest of the words, are better recognized when low-order models are used, because then only the most basic characteristics of the speech sounds are retained,

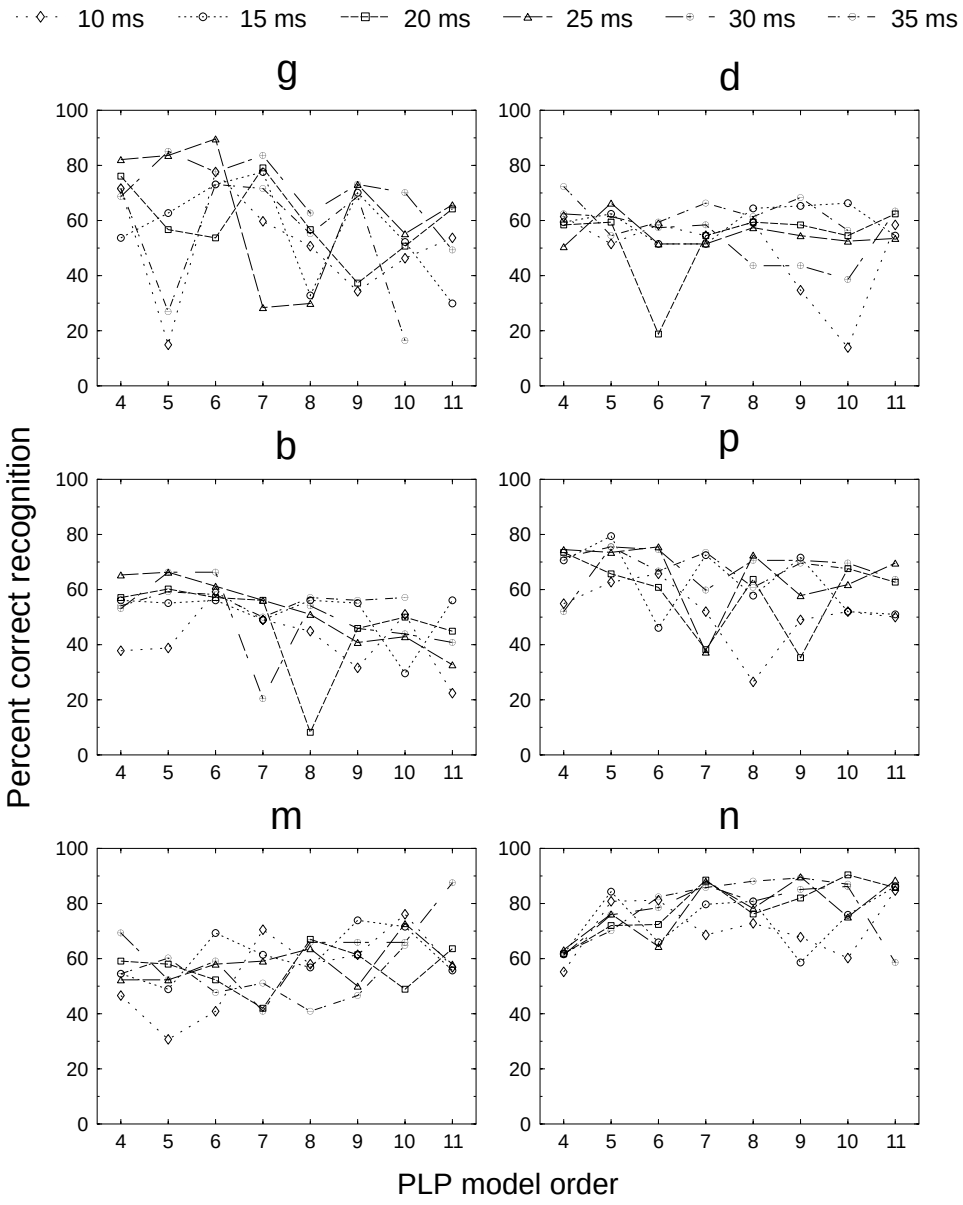

Figure 2.2: Recognition performance of the most poorly recognized words, using the LEMS recognizer (discrete model) with PLP processing, by PLP model order and length of processing window (in ms, legend at the top of the page).

A-set Confusion Matrix:

	a	h	j	k	8
a	93	1	2	6	8
h	0	90	1	.	1
j	.	.	97	2	.
k	0	.	.	90	.
8	1	4	.	1	90
~	5	4	1	1	1

E-set Confusion Matrix:

	b	c	d	e	g	p	t	v	z	3
b	57	2	2	0	1	.	.	3	.	.
c	.	94	5	0	4	2	3	.	11	.
d	8	.	61	0	1	1	0	1	.	.
e	13	2	7	93	15	18	3	4	.	1
g	.	.	2	.	55	.	0	.	.	.
p	8	.	6	0	.	61	1	1	.	.
t	1	.	15	.	13	14	91	3	1	.
v	9	.	1	3	1	.	.	88	1	1
z	.	2	.	0	.	.	.	1	87	.
3	.	.	.	.	.	1	.	.	.	99
~	3	.	1	4	7	4	2	.	1	.

Nasal/Glide Confusion Matrix:

	l	m	n	7	9
l	84	10	0	.	.
m	2	41	6	.	.
n	2	48	88	1	.
7	.	.	.	99	.
9	.	.	.	.	98
~	13	1	5	.	2

Table 2.1: Confusion matrices for the most difficult word subsets, when the recognizer is trained on 8^{th}-order PLP features, calculated from speech frames 35 ms long.

and the details which tend to vary a lot between talkers are discarded.

Somewhat surprisingly, longer processing windows favor not only the relatively more stationary nasal sounds, but also the words containing abrupt transitions and short-lived phenomena, such as the stop consonants. Window sizes of 25 or 35 ms generally outperform shorter sizes for all the problematic words; this may be attributed to variability within pitch periods, as explained above. Interestingly, parameter combinations that lead to good recognition of "m" usually result in poor performance on "n," and vice versa. Examination of the confusion matrices (for example, Table 2.1) indicates that the relative discrimination of "m" and "n" remains low for all models and, depending on the particular combination of signal processing parameters, one will be confused for the other more often. It seems that higher PLP order generally improves this situation, but not very much.

Examination of the confusion matrices reveals another interesting aspect of the

difficulties associated with the E-set. In particular, a very common confusion is for words beginning with a consonant followed by "e" to be recognized as "e," i.e., the consonant transitions are not given enough importance. Since the duration of the whole word is much longer than that of the initial transient phenomena that characterize the consonant, and most of the word length is in fact identical to an "e," this should not come as a surprise. It seems that a different weighting scheme may be more appropriate for these words, so that the initial information is more highly valued in these words where the final information is rather useless (since it is shared by many words). Perhaps the number of states and their relative importance need to be reevaluated for some words.

2.3.3 Performance of the continuous model

It was not possible, because of constraints on computer time and disk space, to train and test continuous models with all these combinations of parameters. A continuous model was chosen to be trained based on features from 8^{th}-order PLP computed on 35 ms speech frames, because the discrete model trained with these features achieved one of the best recognition scores. The curve for 35 ms processing window (see Figure 2.1) shows that for this length of speech frame, performance does not vary much with model order, for orders between 6 and 8. The confusion matrices for this model (Table 2.1) are similar to those of any other parameter combination that achieved high performance. The stops "b" and "g" and the nasal "m" were poorly recognized, as usual (see Figure 2.2). The utterances from male talkers were recognized on average 84.7% correct, whereas those from female talkers were recognized on average slightly less, 81.3% correct. The female talker's `jls` speech was recognized most poorly, at 67.6% on average, and the male talker's `gms` speech was recognized best, 92.8% on average. Moreover, a model of 8^{th} order was a reasonable choice given previous results with 5^{th}-order PLP on speech sampled at

A-set Confusion Matrix:

	a	h	j	k	8
a	95	4	3	1	4
h	0	96	.	.	1
j	0	.	97	1	.
k	.	.	.	97	.
8	1	.	.	1	93
~	4	.	.	1	2

E-set Confusion Matrix:

	b	c	d	e	g	p	t	v	z	3
b	63	.	4	1	.	4	0	5	.	.
c	1	97	1	0	.	.	1	1	14	.
d	12	.	70	0	1	1	.	3	.	.
e	5	.	.	93	1	4	0	1	1	.
g	.	1	6	0	93	.	1	.	.	.
p	7	.	4	0	.	81	2	.	.	.
t	.	.	10	0	1	7	92	1	.	.
v	7	1	.	1	.	1	.	89	.	.
z	.	2	.	0	.	.	.	.	85	.
3	.	.	1	.	.	1	.	.	.	100
~	4	.	4	4	3	1	3	.	.	.

Nasal/Glide Confusion Matrix:

	l	m	n	7	9
l	89	7	0	.	.
m	4	66	11	.	.
n	0	23	82	.	.
7	.	.	.	100	.
9	.	.	.	.	97
~	6	5	7	.	3

Table 2.2: Confusion matrices for the most difficult word subsets, when the continuous model is trained on 8^{th}-order PLP features, calculated from speech frames 35 ms long.

8 Khz and the present findings.

After twenty additional iterations of training, the continuous-model bottom-line performance was 89.7%. Table 2.2 shows the confusion matrices for the most difficult word sets. As expected, recognition performance improved for all words. The accuracy by word ranged between 63.3% and 100.0%, with "b," "m," and "d" being most poorly recognized (63.6%, 65.9%, and 70.3%, respectively) and the numbers from "three" to "seven" all recognized 100.0% of the time. Performance by talker ranged from 79.4% (for the female talker `jls`) to 96.9% (for the female talker `crw`). Overall, speech from the male talkers was recognized about as well as speech from female talkers; in fact, speech from the females was recognized only slightly better (90.3% vs. 89.3%).

A-set Confusion Matrix

	a	h	j	k	8
a	59	43	8	23	55
h	2	43	.	.	.
j	6	.	75	8	.
k	.	.	.	.	.
8	.	.	.	20	
~	33	14	17	69	25

E-set Confusion Matrix

	b	c	d	e	g	p	t	v	z	3
b	.	.	.	.	.	.	.	7	.	.
c	8	74	.	.	.	.	9	7	33	3
d	.	.	21	3	.	.	3	.	.	.
e	92	17	79	86	50	90	74	73	57	21
g	.	.	.	.	7	.	.	.	.	.
p	.	.	.	.	.	.	.	.	.	.
t	.	.	.	.	.	.	3	.	.	.
v	.	.	.	.	7	.	.	7	.	.
z	.	.	.	.	.	.	.	.	5	.
3	.	.	.	.	.	.	.	.	.	59
~	.	9	.	11	36	10	11	7	5	18

Nasal/Glide Confusion Matrix

	l	m	n	7	9
l	71	23	21	4	8
m	23	23	21	.	8
n	.	23	33	4	16
7	.	.	.	83	.
9	.	.	.	.	5
~	6	31	25	8	62

Table 2.3: Confusion matrix for the difficult words in the beamformed data from the microphone array using 8^{th}-order PLP features computed on 35 ms speech segments.

2.3.4 Sensitivity to the recording conditions

In order to investigate the sensitivity of the PLP coefficients to the recording conditions, the files described in Section 1.4 were used to test the models as they were already trained with the usual training set. Performance of the discrete model was 88.6% on the speech recorded with the head-mounted microphone and 22.2% on the beamformed speech from the microphone array. Table 2.3 shows the confusion matrices for the difficult words in the beamformed data. Clearly, because of the lower amplitude of the speech signal in these utterances, most of the words in the E-set were recognized as "e," totally ignoring the initial transition states.

The continuous model achieved 91.5% on the speech from the head-mounted microphone and 43.1% on the beamformed data, a significant improvement over both the discrete model trained with PLP-based cepstral features and the continuous

A-set Confusion Matrix:

	a	h	j	k	8
a	53	7	.	15	15
h	2	79	.	.	.
j	2	.	83	15	.
k	.	.	.	15	.
8	4	14	.	8	65
~	39	.	17	46	20

E-set Confusion Matrix:

	b	c	d	e	g	p	t	v	z	3
b	8	.	.	.	.	.	.	40	.	.
c	8	74	.	.	7	.	3	.	48	.
d	8	.	64	.	7	.	11	.	.	.
e	50	13	29	65	14	70	40	40	29	6
g	.	.	.	5	14	10	11	.	5	.
p	.	.	.	.	.	.	.	.	.	.
t	.	.	.	.	.	.	20	.	.	.
v	.	.	.	.	.	.	.	.	.	.
z	.	.	.	.	.	.	.	.	5	.
3	.	.	.	.	.	.	.	.	.	68
~	25	13	7	30	57	20	14	20	14	26

Nasal/Glide Confusion Matrix:

	l	m	n	7	9
l	80	54	29	4	27
m	.	31	21	.	11
n	.	.	12	.	3
7	.	.	.	83	.
9	.	.	.	.	22
~	20	15	38	12	38

Table 2.4: Confusion matrix for the difficult words in the beamformed data from the microphone array using 8^{th}-order PLP features computed on 35 ms speech segments.

model trained with LPC-based cepstral features. The confusion matrices for this test are shown in Table 2.4.

2.3.5 Combining parameters

Because of the different advantages of high and low model orders, in particular with respect to the most difficult words, it is interesting to investigate the effects of combining more than one PLP model order in the same feature vector. This way one may be able to combine the advantages of both low and high PLP order processing in a single HMM recognizer. Since window length seems to have a uniform effect on all words, i.e., longer windows result in better recognition performance, combinations involving shorter temporal windows were not investigated, although that might be justifiable on acoustic-phonetic grounds only. Examination of the recognition results

for the various models showed that processing of 25 ms windows with a 5^{th}-order PLP analysis was a particularly successful combination, because it achieved one of the highest overall recognition performances (83.8%) as well as recognition rates among the highest for the generally problematic stop consonants (see Figure 2.2). This combination was not, however, as good a choice for the words involving nasal sounds, for which a 10^{th}-order PLP analysis with a 30 ms window gave much better results.

The two processing parameter pairs were combined using five cepstral coefficients and their differences from a 25 ms window along with ten cepstral coefficients (without the differences) from the 30 ms window. It was not possible to include the ten differenced cepstra from the 30 ms processing because the feature vectors have a maximum size of 28 and four were already reserved for the energy features. Recognition rate was expected to be at least equal to that from the 5^{th}-order PLP alone, since all 5^{th}-order parameters were included, and probably somewhat better, either at the nasals only, because their recognition was improved by higher order PLP processing, or overall, because of the extra information given to the model. However, it was also possible that the extraneous information given to the model would obscure the statistical generalization and that the final recognition rate could be lower than that of the 5^{th}-order PLP processing alone.

The usual set of utterances was used to train an HMM model with the combined PLP features. Because only three codebooks are allowed in the vector quantization program (`fastvq`), both the 5^{th}-order cepstral coefficients and their differences had to be included in the first codebook, and the 10^{th}-order cepstral coefficients in the second codebook (so that there would still be a codebook left for the energy and its difference). This may have prevented the formation of good vector clusters and may have caused a lower-than-normal performance of the resulting model. Unfortunately, it was not possible to get around this problem by using only the continuous model

A-set Confusion Matrix:

	a	h	j	k	8
a	94	4	5	3	5
h	0	90	.	.	1
j	.	.	92	1	.
k	1	.	2	95	.
8	1	3	.	.	89
~	3	3	1	2	5

E-set Confusion Matrix:

	b	c	d	e	g	p	t	v	z	3
b	59	.	2	1	.	.	.	4	.	.
c	.	91	.	0	1	.	3	.	7	.
d	9	.	72	1	.	.	0	6	.	.
e	11	3	8	93	12	11	0	3	1	.
g	.	.	2	0	58	.	.	.	.	.
p	7	.	3	1	.	74	0	1	.	.
t	2	.	10	0	21	8	95	3	.	1
v	6	1	1	1	1	2	.	83	1	.
z	.	3	.	.	.	.	.	.	91	.
3	.	.	.	.	.	1	.	.	.	98
~	5	1	2	4	6	5	1	1	1	1

Nasal/Glide Confusion Matrix:

	l	m	n	7	9
l	84	6	1	.	2
m	0	67	19	.	.
n	1	24	77	1	.
7	.	.	.	99	.
9	.	.	.	.	97
~	14	3	3	.	1

Table 2.5: Recognition performance of the HMM recognizer (discrete model) using PLP cepstral coefficients of 5^{th} order and their differences (in the first codebook), cepstral coefficients of 10^{th} order (in the second codebook), and energy and differenced energy (in the third codebook), after 50 iterations of training with Poisson duration modeling.

because it is necessary to begin with a discrete model. The results of the recognition testing were 82.8% after 30 iterations and 84.9% after 50 iterations of the training program. Overall, there does not seem to be a substantial improvement in recognition rate. The achieved bottom-line performance of the combined processing model is about 1% above the best single-processing model, and that is only after 20 additional training iterations.

There was not a big improvement in the recognition rates of the most difficult words (see Table 2.5), which suggests that combining information from different processing orders does not provide the HMM recognizer with additional information, at least when the discretization of the vectors prevents full utilization of this extra

information. The recognition rates for the problematic cases were 58.2% for the "g," 72.3% for the "d," 59.2% for the "b," 73.5% for the "p," 67.0% for the "m," and 77.4% for the "n." It is very likely that the additional parameters that might fine-tune recognition by refining the category boundaries were averaged together with the basic ones by the k-means clustering procedure, thus decreasing the advantages of the combination. This might be resolved with the continuous observation HMM model, which takes into account all the parameters independently of each other.

2.3.6 Combined parameters with the continuous model

In order to test the possibility that the additional information from the combination of different PLP model orders improves recognition performance, the output model of the discrete training after 30 iterations was converted to a continuous tied-mixture model, and the train program was run for 20 iterations to fine-tune the distinctions between the different observation sequences. The overall recognition performance of the trained continuous model was 85.6%, i.e., 0.7% better than the discrete model trained for the same number of iterations. The confusion matrices for the most poorly recognized words (Table 2.6) show a modest improvement in most cases. Since there was no improvement compared to the results of the single-window, single-order PLP processing, the additional computational cost of the combined parameters disqualifies it from further consideration. For this reason, no sensitivity tests were performed.

A-set Confusion Matrix:

	a	h	j	k	8
a	91	.	7	1	4
h	2	97	.	.	1
j	0	.	91	1	.
k	1	.	.	97	.
8	3	1	.	1	92
~	3	1	2	1	3

E-set Confusion Matrix:

	b	c	d	e	g	p	t	v	z	3
b	68	.	2	1	.	2	.	8	.	.
c	.	87	.	0	.	.	2	.	5	.
d	4	.	51	0	.	.	.	3	.	.
e	5	3	5	91	7	5	1	3	2	.
g	.	1	12	0	84	2	3	1	.	.
p	7	.	9	1	.	89	4	1	.	.
t	2	1	18	.	6	1	86	1	.	.
v	10	2	.	2	.	.	0	81	.	.
z	.	5	.	.	.	.	.	1	93	.
3	.	.	.	.	.	1	1	.	.	100
~	3	1	3	4	3	.	2	2	.	.

Nasal/Glide Confusion Matrix:

	l	m	n	7	9
l	92	9	0	.	.
m	2	69	18	.	.
n	0	18	74	.	2
7	.	.	.	100	.
9	.	.	.	.	97
~	5	3	7	.	1

Table 2.6: Recognition performance of the HMM recognizer (continuous model) using PLP cepstral coefficients of 5^{th} order and their differences (in the first codebook), cepstral coefficients of 10^{th} order (in the second codebook), and energy and differenced energy (in the third codebook), after 20 iterations of additional training.

Chapter 3

Using log-spaced Fourier spectra

The second method tried with the LEMS recognizer was based on nonlinear sampling of DFT spectra, with an important variation: Each of four frequency regions was taken from a DFT computed from a different-sized segment of the speech signal. In this section the relative advantages and drawbacks of speech recognition attempts based on DFT spectra are outlined, the reasons for nonlinear sampling of the spectral values are cited, and the intuitions behind the choice of multiple window sizes are clarified. The results obtained with the recognizer are then discussed with respect to predictions and to previous results of the LEMS speech group.

3.1 Advantages and disadvantages of log-spaced spectra

Because of the disadvantages of all-pole modeling that were mentioned in the previous chapter, an alternative option is now often taken, that of using cepstral coefficients obtained from raw DFT spectra. D. J. Mashao, Y. Gotoh, and H. F. Silverman from the LEMS speech group have been developing some feature extraction techniques which use nonlinear sampling of smoothed DFT spectra and convert the resulting

warped spectral representation into the cepstral domain [9]. Their formulation of the nonlinear sampling defines α regions of the linear frequency axis, the ith one being of width $A\beta^{i-1}$ where i goes from 1 to α. If the total width of the frequency region is N, the following formula must be satisfied:

$$\sum_{i=1}^{\alpha} A\beta^{i-1} = N.$$

Each region is linearly sampled to contribute the same number of points, so the higher frequency regions end up being sampled more sparsely than the low frequency regions. The results so far indicate that higher values of β, i.e., less uniform and more "log-like" sampling of the spectrum, lead to better recognition results.

Human audition is known to function on an approximately logarithmically-spaced frequency axis, although at the cochlear level the lowest frequencies are linearly spaced and only after about 2 KHz is the spacing logarithmic. In the cortical auditory maps, it is claimed that at some level the frequency axis may be logarithmic throughout its range. This can have very important implications for speech perception, in particular with respect to talker normalization, since identical utterances produced by talkers with different vocal tract lengths would differ only by a shift and not by a scaling factor in log-spaced spectra. Therefore, in the cepstral representation (a form of which is perhaps computed in human auditory cortex) such utterances would have identical cepstral magnitude and differ in cepstral phase, whereas in cepstra from linearly-spaced spectra there would be no clear invariance between the two. In other words, a cepstral transformation of a log-spaced spectrum should lead to higher talker invariance in the representation of speech, and so perhaps to higher overall identification performance, if the relevant speech features are indeed preserved and the irrelevant ones are not too disturbing.

It is in this last respect, i.e., preservation of irrelevant features, that speech recognition from DFT spectra may suffer, because DFT spectra tend to capture a lot of acoustic information which is not only unnecessary for speech, but in fact detrimen-

tal to the statistical generalizations of the recognizer. Such information includes the resonances of the recording chamber, the transfer characteristics of the recording equipment, the orientation of the talker with respect to the microphone, etc. Although the use of a head-mounted microphone minimizes variation due to most of these sources (except for the recording equipment characteristics), one becomes bound to using the head-mounted microphone all the time in order to maintain recognition performance. If this is not a problem then signal processing methods based on DFT spectra may outperform techniques that smooth or normalize extraneous variation.

3.1.1 Window size and transient phenomena

As explained before (see Section 2.1), there should be an advantage for higher temporal resolution in speech processing because of the many transient phenomena that help distinguish between speech sounds. On the other hand, too short processing windows have very low frequency resolution, and can only represent frequencies down to the inverse of their length. In addition, they are subject to amplitude variation within pitch periods. In order to preserve low-frequency information, which is known to be very important for speech perception, and to highlight short-lived events in the high frequencies at the same time, it is possible to take several Fourier transforms, each on a speech frame of different size, and to use the values from each one only for the frequencies for which its size is optimal. In the human ear there is no fixed temporal analysis window; each frequency is identified as soon as it is possible to detect it. We can approximate this behavior without too much computational overhad with a small number of DFTs calculated on overlapping windows of different sizes.

Similar considerations have led to the development of wavelet functions for use in speech processing systems. The method proposed here bears strong resemblances to wavelet processing, but it was not within the scope of this study to introduce

completely novel techniques. Rather, some refinements and modifications of well-known and studied existing procedures are proposed. For this reason, the relatively crude method of combining DFTs was preferred over an elaborate wavelet study. If this approach proves fruitful, true wavelet processing will be an obvious extension to the system as the subject of a future study.

3.2 Description of the method

A DFT-based method has been developed, where higher-frequency spectral information is obtained from shorter temporal windows (speech segments). Since the sampling frequency used in the LEMS setup is 16 KHz, the maximum frequency that can be represented in the digitized speech is 8 KHz. The region between 0 and 8 KHz was divided into four logarithmically spaced subintervals, i.e., from 0 Hz to 1 KHz, from 1 Hz to 2 KHz, from 2 KHz to 4 KHz, and from 4 KHz to 8 KHz. The acoustic energy present in each of these intervals was calculated from a different DFT, so four DFTs were computed for each processing frame: A 1024-point DFT (64 ms window), a 512-point DFT (32 ms window), a 256-point DFT (16 ms window), and a 128-point DFT (8 ms window). All four windows were aligned at their initial points, and processing advanced by 8 ms (frame step), so that each speech segment would contribute to exactly one DFT of the shortest window length.

The resulting transforms were converted to magnitude and phase values, and a common 512-point magnitude vector was constructed by using some components from each DFT magnitude in its designated frequency region. Thus the 1024-point DFT (which produced 512 complex spectral values) contributed the 64 points that correspond to frequencies from 15.6 Hz (the first bin equals Nyquist frequency divided over the number of complex points) to 1000.0 Hz, the 512-point DFT contributed 32 points, from 1031.25 Hz to 2000.0 Khz, the 256-point DFT contributed 32 points, from 2062.5 Hz to 4000.0 Hz, and the 128-point DFT contributed 32 points, from

4125.0 Hz to 8000.0 Hz. The intermediate points in the combined spectral magnitude vector (whose values were still linearly spaced) were filled with the values of the closest preceding point that was filled.

The reason that the spectral magnitudes were first combined linearly was that it was necessary to filter the resulting vector with an FIR cepstral filter in order to remove the ripple of the harmonic structure. Because of the different bin sizes of the four spectra, this meant either to use a separate cepstral filter for each spectrum, or to "stretch out" the spectra with the fewest points in order to bring them all to the same frequency scale and then use a single filter for the combined magnitude vector. The latter option was taken for computational convenience; in an optimized version the former option would be preferable. An optimal 41-tap cepstral smoothing filter (`type_10_fir`, in Appendix A) was kindly provided by Daniel J. Mashao of the LEMS speech group. Subsequent to application of the FIR filter (via direct convolution) the combined magnitude vector was interpolated logarithmically yielding a 100-point "log-DFT" magnitude vector. Finally, the first 12 cepstral coefficients of the resulting vector were computed using the formula

$$c_n = \sum_{i=0}^{N-1} M(i) \cos \frac{n(i+0.5)\pi}{M},$$

where M is the size of the log-DFT vector, i.e., $M = 100$, and n ranges from 1 to 12. The resulting coefficients were then passed to the LEMS programs for feature vector generation. The C code for this procedure is listed in Appendix A.

3.3 Recognition performance with the DFT features

The recognition performance of the LEMS recognizer with the multiple window spectral features reached 83.1% after 30 iterations of the discrete model, and 89.9 after

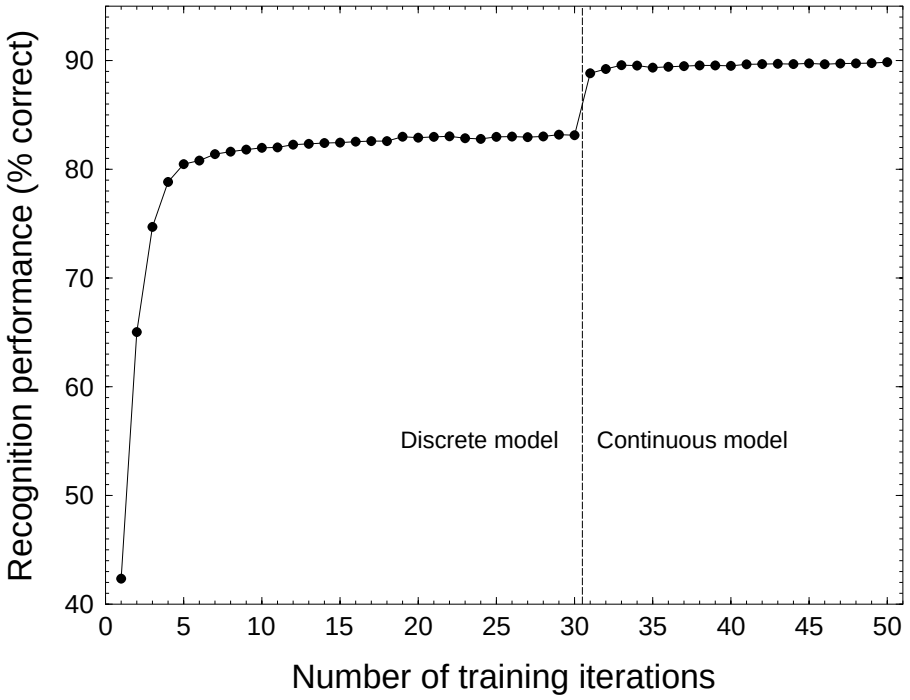

Figure 3.1: Bottom line recognition performance for the LEMS recognizer with cepstral features computed from multiple-window log-spaced DFT spectra, as a function of the number of training iterations.

20 additional iterations with the continuous model. Figure 3.1 shows the bottom-line recognition performance reported by `batchrec` as a function of the number of iterations. The performance measure does not always increase monotonically, but fluctuates a little after a few iterations. Table 3.1 shows the discrete model's performance on the most difficult sets. Note that performance of the discrete model is not better than when using PLP-based features, but is 10% better than with the LPC-based features. The same words that were most poorly recognized with PLP and LPC features were again the most poorly recognized here; notably, performance on "d" was an unacceptable 24.6%. The most poorly recognized words, besides "d,"

```
 A-set Confusion Matrix:
    a  h  j  k  8
a  92  8  1  4 10
h   1 89  .  .  3
j   0  1 94  9  1
k   .  .  4 84  .
8   1  .  .  . 83
~   5  1  1  3  3

 E-set Confusion Matrix:
    b  c  d  e  g  p  t  v  z  3
b  67  .  9  1  . 12  1  4  .  1
c   . 85  .  0  .  .  3  . 10  1
d   3  . 25  .  .  .  0  .  .  .
e   5  2 11 94 12 11  3  6  1  .
g   .  1  .  . 42  .  .  .  .  .
p  10  . 10  1  3 55  1  1  .  .
t   3  2 39  0 28 20 90  3  2  .
v  10  1  3  1  .  1  . 83  2  .
z   .  9  .  0  3  .  0  2 85  .
3   .  .  .  .  .  .  .  1  . 95
~   1  1  4  3 12  2  1  2  1  3

 Nasal/Glide Confusion Matrix:
    l  m  n  7  9
l  88  5  .  1  .
m   1 43 12  .  1
n   1 40 81  2  1
7   .  .  . 97  .
9   .  .  .  . 90
~  10 12  7  .  8
```

Table 3.1: Confusion matrices for the most difficult word subsets, from the performance test of the discrete model trained on cepstral features from multiple-window log-spaced DFT spectra.

were "g," "m," and "p" (with scores 41.8%, 43.2%, and 54.9%, respectively) and the best recognized words were "space," "six," "seven," and "x" (with scores 99.7%, 98.7%, 97.5%, and 97.0%, respectively). The speech of the female talkers was recognized on average worse than speech from male talkers: 79.5% overall for the females vs. 85.7% overall for the males. This may reflect the predominance of male talkers in the training set or perhaps there is more variability between females than between males. The utterances of female talker `jls` were the most poorly recognized (average of 3 files was 62.5%) and those of male talker `gms` were recognized best (average of 3 files was 91.9%).

Table 3.2 shows the confusion matrices for the most difficult words using the continuous model. There was a significant improvement in all words, which brought performance on the most poorly recognized word, "d," up to 54.5% (from 24.6% with

A-set Confusion Matrix:

	a	h	j	k	8
a	95	4	1	1	3
h	1	96	.	.	2
j	0	.	98	5	.
k	.	.	1	92	.
8	1	.	.	.	94
~	3	.	1	2	1

E-set Confusion Matrix:

	b	c	d	e	g	p	t	v	z	3
b	78	1	11	1	.	5	1	8	.	1
c	.	93	.	0	1	.	2	.	9	.
d	7	.	54	.	.	.	.	.	.	.
e	4	1	2	94	3	5	3	2	1	.
g	.	2	13	.	93	2	1	1	1	.
p	6	.	11	0	.	81	4	.	.	.
t	1	1	8	.	3	6	88	3	1	.
v	2	1	1	1	.	.	.	84	1	1
z	.	3	.	0	.	.	.	1	88	.
3	.	.	.	.	.	.	.	.	.	99
~	2	.	.	2	.	1	2	1	.	.

Nasal/Glide Confusion Matrix:

	l	m	n	7	9
l	94	8	2	.	1
m	2	59	14	.	.
n	0	30	81	1	.
7	.	.	.	99	.
9	.	.	.	.	97
~	4	3	3	.	3

Table 3.2: Confusion matrices for the most difficult word subsets, from the performance test of the continuous model trained on cepstral features from multiple-window log-spaced DFT spectra.

the discrete model) and performance on "g" to an impressive 92.5% (from 43.2%). The most poorly recognized words, besides "d," were "m," and "b", recognized at 59.1% and 77.6%. The best recognized words were "zero" and "six," with a perfect (100.0%) recognition score. Recognition performance by talker ranged from 78.1% for the female talker `jls` to 95.9% for the female talker `crw`. The speech of the male talkers was recognized about as well as speech from the female talkers (90.0% vs. 89.6%, respectively), which is expected because of the independece of cepstral magnitude from frequency scaling that results from differences in vocal tract length, as discussed in Section 3.1.

A-set Confusion Matrix

	a	h	j	k	8
a	41	36	8	8	20
h	.	7	.	.	.
j	.	.	25	8	.
k	4	7	17	31	5
8	2	.	.	.	10
~	53	50	50	54	65

E-set Confusion Matrix

	b	c	d	e	g	p	t	v	z	3
b	17	.	14	9	14	10	6	7	14	9
c	.	9	7	1	.	10	.	.	.	.
d	8	.	.	1	.	.	3	.	.	.
e	17	17	14	34	7	.	11	7	.	.
g	.	.	.	.	.	.	.	.	.	.
p	.	.	.	3	7	.	.	7	.	.
t	.	9	7	5	14	40	34	20	.	3
v	8	.	.	3	.	.	3	7	.	.
z	.	9	.	.	.	.	6	.	19	.
3	.	9	7	.	.	.	.	.	.	18
~	50	48	50	44	57	40	37	53	67	71

Nasal/Glide Confusion Matrix

	l	m	n	7	9
l	23	15	.	4	8
m	17	31	12	21	27
n	37	8	54	17	11
7	.	.	.	29	.
9	.	.	.	.	19
~	23	46	33	29	35

Table 3.3: Confusion matrices for the most difficult word subsets in the beamformed data from the microphone array, using the discrete model with features from multiple-window log-spaced DFT spectra.

3.3.1 Sensitivity to the recording conditions

The sensitivity of this method to the recording conditions was tested in the same manner that PLP was tested in Section 2.3.4. The performance of the discrete model that was already trained on the training set for 30 iterations was 85.0% on the speech from the head-mounted microphone (conditions identical to those of the training set) and negative on the beamformed data. The negative value has no meaning of percentage, but indicates that the substitutions, omissions and insertions brought the total number of errors to be higher than the total number of correctly identified words. Table 3.3 shows the confusion matrices for the difficult word sets. Note that many of these words, and in fact many of the rest as well, were very rarely recognized, often resulting in recognition of "silence." The low amplitude of the

A-set Confusion Matrix:

	a	h	j	k	8
a	18	14	.	8	.
h	.	14	.	.	.
j	2	7	17	8	.
k	.	.	.	15	.
8	2	.	.	.	5
~	78	64	83	69	95

E-set Confusion Matrix:

	b	c	d	e	g	p	t	v	z	3
b	.	.	7	.	.	.	.	.	.	.
c	.	17	.	3	.	.	.	.	.	.
d	.	.	.	.	7	.	.	.	5	.
e	17	.	.	19	.	10	.	.	.	.
g	.	.	.	.	7	10	6	7	.	.
p	.	4	7	5	7	.	3	7	5	.
t	.	.	.	.	.	.	14	.	.	.
v	.	4	7	1	.	.	.	13	.	.
z	.	9	.	.	.	.	9	7	14	.
3	.	.	.	.	.	.	.	.	.	15
~	83	65	79	72	79	80	69	67	76	85

Nasal/Glide Confusion Matrix:

	l	m	n	7	9
l	51	8	12	12	16
m	6	.	.	.	.
n	.	15	25	4	.
7	.	.	.	17	.
9	.	.	.	.	14
~	43	77	62	67	70

Table 3.4: Confusion matrices for the most difficult word subsets in the beamformed data from the microphone array, using the continuous model with features from multiple-window log-spaced DFT spectra.

waveforms proved to be beyond the generalization ability of the spectral technique, which is perhaps based too much on the particulars of the spectral shape and not the important peaks and valleys. In the cases where the final segment of the words in the E-set attained some minimum value, an "e" was recognized, since the initial burst and transition was probably too low in amplitude to be taken into account.

Performance of the continuous model was only slightly better, overall 14.9% (compared to 90.8% on the simultaneous recording with the head-mounted microphone). The confusion matrices for the most difficult word sets are shown in Table 3.4. The same pattern, with most segments labeled silent, is evident. It is concluded that this DFT-based method is very sensitive to the recording conditions, presumably because it retains spectral information not relevant for speech, which makes it unsuitable for applications that use telephone or other transmission lines with unpredictable variation.

Chapter 4

Conclusion

In this thesis two new signal processing methods have been applied to the LEMS HMM speech recognizer and the performance of the model has been compared to that obtained with the previously used LPC-based features. Parameters computed with the method of Perceptual Linear Prediction (PLP), as well as cepstral features from DFT spectra with logarithmic frequency warping computed on multiple overlapping speech segments, have improved recognition performance by 10–13% percent, when identical training and testing conditions are used. Table 4.1 shows the bottom-line recognition performance for the discrete model (after 30 training iterations) and the continuous model (after 20 additional iterations). Using the continuous model greatly improves recognition rates, allowing them to near 90%, but this comes at a great computational cost. For practical applications, it is questionable whether a continuous model can be used with real-time speech, as for example would be necessary for automatic data entry via telephone. Furthermore, the best achieved performance rate would mean that one in ten words is not identified correctly. This can be improved by using a dictionary, if the spoken utterances form real words. Then a distance metric based on the confusion matrices of a given model can be used to correct misidentified words (i.e., letters) in order to complete a (spelled) word with maximum probability. Such a system cannot be used with numbers, but

Processing type	Recognition performance		
	Testing set (`small_test`)	Speech from same microphone	Speech from microphone array
LPC `type6`			
Discrete	73.1[a]	83.5	31.1
Continuous	77.3[b]	85.8	39.0
PLP (order 8, 35 ms)			
Discrete	83.2	88.6	22.2
Continuous	89.7	91.5	43.1
Log-spaced DFT			
Discrete	83.1	85.0	−16.5
Continuous	89.9	90.8	14.9

[a]With different codebooks performance was 80.7%. Previous LPC results have reached 84.3%.

[b]Best continuous-model performance with LPC reported so far from LEMS is 89.4%.

Table 4.1: Bottom-line recognition performance for the final models of each of the three methods that are compared.

digits are much better recognized already, and a second repetition by the talker would diminish the probability of error. However, if a speech recognition system is used for clients to say their names, or for other strings that cannot be searched in a dictionary, such solutions become obviously inadequate. Any name with ten letters or more (not an unreasonable assumption for first-and-last-name utterances) will probably be misspelled. The probability of recognition for some letters is so low, even when using the best techniques, that an altogether different set of words should be used. Perhaps the well known distinct words used for the letters of the alphabet

in noisy transmission lines (e.g., by DX radio amateurs), and sometimes even for telephone transmission, would be preferable.

Table 4.1 shows the overall recognition performance for the independent set of utterances that have been recorded with the same microphone as the utterances in the training set and with beamformed data from simultaneous recording with the microphone array at LEMS. These data are very important for the evaluation of the methods if the recognizer is aimed at talker-independent and microphone-independent speech recognition. In other words, what is the practical value of a system that can recognize forty words with 20%-40% success? It is clear that systems based on DFT spectra will not outperform systems based on more speech-specific reduction of acoustic information. Environmental factors, such as room shape and dimensions, and recording and transmission equipment, result in grossly distorted speech spectra that present no difficulty to human listeners but incapacitate artificial systems that have retained spectral details in their training.

A surprising result of this work was the apparent sensitivity of the recognizer to the vector quantization codebooks. Using the same signal processing method and the same HMM training and testing sets and procedures, overall performance lagged 7–10% behind performance results previously reported, when the standard (for this thesis) feature vector sets were used to create the codebooks as opposed to using some pre-existing codebooks created from an unknown set of files. It has been assumed that any sufficiently large random sample of feature vectors would lead approximately to the same vector clusters and, consequently, to very similar and functionally equivalent codebooks. Because of this, and of the prohibitive computational cost of running the k-means clustering program on the entire training set, only a small subset of feature vectors has been used (10–15 out of 90 feature files) in this and other work in LEMS. Future studies should address the issue of performance dependence on the selection of feature vectors for k-means clustering and indicate a procedure for determining

the optimal training set, perhaps using a few test trials of several iterations each.

Because of the dependence on codebook training, it is recommended that comparisons be made between processing methods only using identical sets and procedures, as was done in this thesis. A potential problem with this approach is that there is no guarantee that using the same speech files for vector clustering will result in comparable distortion measures when using different signal processing techniques. Therefore, using the same vectors to create the codebooks may result in "good," representative codebooks with one processing method and "bad," distorted codebooks in another. For this reason, recognition performance obtained previously using different codebooks was also taken into account when comparing techniques. In order to make a valid and conclusive comparison, each signal processing method should be separately optimized (but under comparable circumstances and similar procedures) and the final best results should then be brought into comparison. Clearly, the work presented here has not yielded a final answer to the question of which method is best, and comparing with results of previous work is particularly problematic because most previous results have been obtained using substantially larger training and testing sets, which was not possible for this study because of computer time and space constraints. Future research should be conducted on optimizing PLP for the LEMS recognizer and on implementing wavelet processing for the reasons mentioned in Chapter 3. The end results can then be tested against those from the currently used DFT-based techniques.

It is well known in the engineering field, as well as in the disciplines concerned with research in human speech perception, that transformation of the acoustic signal of speech to words in the brain is a very complicated problem. It appears that superficial statistical generalizations based on the spectral properties of speech signals may not be adequate for a unique characterization of the speech message contained in the signal. The search for acoustic invariants in speech continues, but has not

been particularly fruitful. In this thesis, the performance of a powerful statistical recognizer that is based on state-of-the-art mathematical formulations was tested with algorithms that make slight modifications to the speech spectra according to known facts about human audition. From one point of view this attempt has been very successful, because it improved recognition rates and showed that paying some attention to auditory characteristics may yield significant gains. From another point of view, the results reported herein point at the inability of current formulations to capture the essentials of the speech signals and indicate that we should perhaps concern ourselves more with the only known solution to the problem, i.e., the human brain. Depending on one's needs and application requirements, either of these conclusions may be drawn.

Bibliography

[1] B. S. Atal. Speech analysis and synthesis by linear prediction of the speech wave. *The Journal of the Acoustical Society of America,* **47**(1):65, 1970.

[2] B. S. Atal & S. L. Hanauer. Speech analysis and synthesis by linear prediction of the speech wave. *The Journal of the Acoustical Society of America,* **50**(2):637–655, 1971.

[3] J. W. Creekmore, M. Fanty, & R. A. Cole. A comparative study of five spectral representations for speaker-independent phonetic recognition. In *Proceedings of the Twenty-Fifth Asilomar Conference on Signals, Systems, and Computers*, pp. 330–334. IEEE Computer Society Press, Los Alamitos, CA, 1991.

[4] M. Fanty, R. Cole, & M. Slaney. A comparison of DFT, PLP, and cochleagram for alphabet recognition. In *Proceedings of the Twenty-Fifth Asilomar Conference on Signals, Systems, and Computers*, pp. 326–329. IEEE Computer Society Press, Los Alamitos, CA, 1991.

[5] H. Hermansky. Perceptual linear predictive (PLP) analysis of speech. *The Journal of the Acoustical Society of America,* **87**(4):1738–1752, April 1990.

[6] H. Hermansky, N. Morgan, A. Bayya, & P. Kohn. Rasta-PLP speech analysis technique. In *Proc. 1992 ICASSP*, pp. I-121–I-124.

[7] M. M. Hochberg. A comparison of state-duration modeling techniques for connected speech recognition. Technical Report LEMS-112, LEMS, Division of Engineering, Brown University, October 1992.

[8] D. J. Mashao, Y. Gotoh, & H. F. Silverman. Analysis of LPC/DFT features for an HMM-based alphadigit recognizer. *IEEE Signal Processing Letters*, in press.

[9] D. J. Mashao, Y. Gotoh, & H. F. Silverman. A DFT-based parameterized signal processing for HMM-based alpha-digit recognizer. LEMS Tech Report, in preparation.

[10] L. Rabiner & B.-H. Juang. *Fundamentals of speech recognition.* Prentice Hall Signal Processing Series, Englewood Cliffs, NJ, 1993.

[11] H. F. Silverman & Y. Gotoh. On the implementation and computation of training and HMM recognizer having explicit state durations and multiple-feature-set tied-mixture observation probabilities. LEMS Monograph Series: 1-1, LEMS, Division of Engineering, Brown University, 1994.

[12] H. F. Silverman, S. E. Kirtman, J. E. Adcock, & P. C. Meuse. *Experimental results for baseline speech recognition performance using input acquired from a linear microphone array.* In Proceedings DARPA Speech and Natural Language Workshop, Morgan Kaufmann, San Mateo, Ca, February 1992.

Appendix

A. C code for computing cepstral coefficients from multiple-window log-spaced DFT spectra

B. Shell script to generate PLP features and vector codebooks, and train and test the recognizer

C. Training and testing file sets

A. C code for computing cepstral coefficients from multiple-window log-spaced DFT spectra

```
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>
#include <math.h>

#include "fft.h"
#include "model_dat.h"
#include "dsp_globs.h"

#define MU 0.98
#define NYQUIST 8000
#define FFT_SIZE 512
#define LOG_FFT_SIZE 100
#define LIN_FFT_PART 20
#define FILTER_SIZE 41
#define CEP_SIZE 12
#define FRAME_RATE (FFT_SIZE/4)

typedef struct {
  double coeff[FILTER_SIZE];   /* coefficients */
} filter;

void get_log_axis ( double *points ) {

  int i ;
  double first, last, inter, lin_res ;

  lin_res = (double)NYQUIST/(double)FFT_SIZE ;
  first = log ( lin_res * LIN_FFT_PART ) ;
  last  = log ( (double)NYQUIST ) ;
  inter = (last - first)/(LOG_FFT_SIZE-LIN_FFT_PART-1) ;
  for ( i=0 ; i<LIN_FFT_PART ; i++ )
    points[i] = lin_res * (double)i ;
  for ( i=LIN_FFT_PART ; i<LOG_FFT_SIZE ; i++ )
    points[i] = exp(first + (double)(i-LIN_FFT_PART) * inter) ;
}

filter type_1010_fir = {{  /* f = [ 0 0.1 0.2 1] */
  -4.0335e-03,  3.8544e-03,  5.3562e-03,  7.1796e-03,  8.1562e-03,
   7.2626e-03,  3.8528e-03, -2.0721e-03, -9.7257e-03, -1.7526e-02,
  -2.3285e-02, -2.4590e-02, -1.9339e-02, -6.2860e-03,  1.4537e-02,
   4.1617e-02,  7.2067e-02,  1.0203e-01,  1.2733e-01,  1.4423e-01,
   1.5018e-01,  1.4423e-01,  1.2733e-01,  1.0203e-01,  7.2067e-02,
   4.1617e-02,  1.4537e-02, -6.2860e-03, -1.9339e-02, -2.4590e-02,
  -2.3285e-02, -1.7526e-02, -9.7257e-03, -2.0721e-03,  3.8528e-03,
   7.2626e-03,  8.1562e-03,  7.1796e-03,  5.3562e-03,  3.8544e-03,
  -4.0335e-03 }};

void filter_fft ( double *data, double *smooth ) {

  int n, i, r, len=FFT_SIZE ;
  filter *fir=&type_1010_fir ;
  double y[FFT_SIZE+FILTER_SIZE] ;

  for ( n = 0; n < len ; n++)
    y[FILTER_SIZE/2+n] = data[n];

  for ( n = 0; n < FILTER_SIZE/2; n++)
    y[n] = data[FILTER_SIZE/2-n-1];

  for ( n = 0; n < FILTER_SIZE/2; n++)
    y[len+FILTER_SIZE/2+n] = data[len-n-1];

  n = 0;
  for ( i = FILTER_SIZE-1; i < len+2*FILTER_SIZE; i++ ) {
    smooth[n] = 0.0;
    for ( r = 0; r < FILTER_SIZE; r++ ) {
```

```
      if ( i >= r )
smooth[n] += y[i-r]*fir->coeff[r];
    }
    n++;
  }
}


void window ( double *s_in, int n, double *s_out ) {

  int i ;
  double nn ;

  nn = (double)n ;
  for ( i = 0 ; i < n ; i++ )    /* Hamming window */
    s_out[i] = s_in[i] * (0.54-0.46*cos(2.0*M_PI*(double)i/nn));

}

void tp_cepstra ( short *si_data, double *s_data, int pts,
                  double *cepstrum, sp_def *sp ) {

  int n, n_data, offset, r ;
  double mu=MU, max, f_val[LOG_FFT_SIZE] ;
  double fft_data[2*FFT_SIZE], *partial_spectra[4], comp_fft[FFT_SIZE] ;
  double smooth_fft[FFT_SIZE], log_fft[LOG_FFT_SIZE];
  double left, right, bin, offs, fstep, cep_max ;
  double ip, fcn ;

  if ( (sp->frame_size != (2*FFT_SIZE)) || (sp->sp_type  != 1010)   ||
      (sp->frame_step != FRAME_RATE)   || (sp->num_feat != CEP_SIZE)||
      (sp->feat_base  != 1) || (sp->pre_emp != MU) ) {
    fprintf ( stderr, "Error in sp definition! tp_cepstra aborting.\n" );
    exit ( 10 ) ;
  }

  n_data = sp->frame_size ;
  s_data[0]=(double)si_data[0] ;
  for ( n=1 ; n<pts ; n++ ) s_data[n] = (double)si_data[n]
                                      - mu * (double)si_data[n-1] ;
  if ( n_data > pts ) {
    s_data[pts] = - mu * (double)si_data[pts-1] ;
    for ( n=pts+1 ; n<n_data ; n++ ) s_data[n] = 0.0 ;
  }

  for ( n=0 ; n<4 ; n++ ) partial_spectra[n] =
                             (double *)malloc(FFT_SIZE*sizeof(double)) ;
  offset = 0 ;

  window ( s_data+offset, (2*FFT_SIZE), fft_data ) ;
  realft ( fft_data-1, FFT_SIZE ) ;
  periodogram ( fft_data, FFT_SIZE, partial_spectra[0] ) ;
  for ( n=0 ; n<(FFT_SIZE/8) ; n++ ) {
    comp_fft[n] = partial_spectra[0][n] ;
  }

  window ( s_data+offset, FFT_SIZE, fft_data ) ;
  realft ( fft_data-1, (FFT_SIZE/2) ) ;
  periodogram ( fft_data, (FFT_SIZE/2), partial_spectra[1] ) ;
  for ( n=0 ; n<32 ; n++ ) {
    comp_fft[(FFT_SIZE/8)+2*n]   = partial_spectra[1][32+n] ;
    comp_fft[(FFT_SIZE/8)+2*n+1] = partial_spectra[1][32+n] ;
  }
  window ( s_data+offset, (FFT_SIZE/2), fft_data ) ;
  realft ( fft_data-1, (FFT_SIZE/4) ) ;
  periodogram ( fft_data, (FFT_SIZE/4), partial_spectra[2] ) ;
  for ( n=0 ; n<32 ; n++ ) {
    comp_fft[(FFT_SIZE/4)+4*n]   = partial_spectra[2][32+n] ;
    comp_fft[(FFT_SIZE/4)+4*n+1] = partial_spectra[2][32+n] ;
    comp_fft[(FFT_SIZE/4)+4*n+2] = partial_spectra[2][32+n] ;
```

```
    comp_fft[(FFT_SIZE/4)+4*n+3] = partial_spectra[2][32+n] ;
  }
  window ( s_data+offset, (FFT_SIZE/4), fft_data ) ;
  realft ( fft_data-1,  (FFT_SIZE/8) ) ;
  periodogram ( fft_data,  (FFT_SIZE/8), partial_spectra[3] ) ;
  for ( n=0 ; n<32 ; n++ ) {
    comp_fft[(FFT_SIZE/2)+8*n]   = partial_spectra[3][32+n] ;
    comp_fft[(FFT_SIZE/2)+8*n+1] = partial_spectra[3][32+n] ;
    comp_fft[(FFT_SIZE/2)+8*n+2] = partial_spectra[3][32+n] ;
    comp_fft[(FFT_SIZE/2)+8*n+3] = partial_spectra[3][32+n] ;
    comp_fft[(FFT_SIZE/2)+8*n+4] = partial_spectra[3][32+n] ;
    comp_fft[(FFT_SIZE/2)+8*n+5] = partial_spectra[3][32+n] ;
    comp_fft[(FFT_SIZE/2)+8*n+6] = partial_spectra[3][32+n] ;
    comp_fft[(FFT_SIZE/2)+8*n+7] = partial_spectra[3][32+n] ;
  }

  max = 0.0 ;
  for ( n=0 ; n<FFT_SIZE ; n++ ) if ( comp_fft[n] > max )
                                                  max = comp_fft[n] ;
  for ( n=0 ; n<FFT_SIZE ; n++ ) comp_fft[n] -= max ;

  filter_fft ( comp_fft, smooth_fft ) ;

  get_log_axis ( f_val ) ;
  fstep = (double)NYQUIST / (double)FFT_SIZE ;
  for ( n=0 ; n<(LOG_FFT_SIZE-1) ; n++ ) {
    bin   = f_val[n] / fstep ;
    offs  = modf ( bin, &ip ) ;
    left  = smooth_fft[(int)floor(bin)] ;
    right = smooth_fft[(int) ceil(bin)] ;
    log_fft[n] = left + offs*(right-left) ;
  }
  log_fft[LOG_FFT_SIZE-1] = smooth_fft[FFT_SIZE-1] ;

  for ( n=0 ; n<CEP_SIZE ; n++ ) {
    cepstrum[n] = 0.0 ;
    for ( r=0 ; r<LOG_FFT_SIZE ; r++ )
      cepstrum[n] += log_fft[r] * cos( (double)(n+1)*((double)r+0.5)
                                       * M_PI / (double)LOG_FFT_SIZE ) ;
  }
  cepstrum[0] /= 2.0 ;
  cepstrum[1] /= 1.5 ;
  cep_max = 0.0 ;
  for ( n=0 ; n<CEP_SIZE ; n++ ) {
    fcn =  fabs(cepstrum[n]) ;
    if ( fcn > cep_max ) cep_max = fcn ;
  }
  if ( cep_max > 1000.0 ) {
    fprintf ( stderr, "tp_cepstra: Cepstrum overflow, rescaled (%7.0f)\n",
                                                                  cep_max );
    cep_max /= 1000.0 ;
    for ( n=0 ; n<CEP_SIZE ; n++ )
      cepstrum[n] /= cep_max ;
  }
  for ( n=0 ; n<4 ; n++ ) {
    free(partial_spectra[n]) ;
    partial_spectra[n] = NULL ;
  }
}
```

B. Shell script to generate PLP features and vector codebooks, and train and test the recognizer

```
#!/usr/bin/csh
# Call this with first parameter the PLP order and second parameter
# the processing frame (in ms), e.g., plptest 10 15
clear
cd /big309/speech/tp
echo "Starting batch for PLP-$1, with $2 ms processing window"
echo -n "$HOST   "
date
echo -n "$HOST   " >& /big309/speech/tp/logs/$1_$2.log
date >>& /big309/speech/tp/logs/$1_$2.log
mkdir plp$1_$2
cd plp$1_$2
mkdir train
mkdir test
cd train
echo "Generating features for training set..."
nice +19 /big309/speech/tp/commands/plpfeat_train_wo $1 $2 >& /dev/null
echo "Generating codebooks:"
echo "K-means clustering for cepstral coefficients..."
nice +19 /pro/rec/v3/vq/kmeans/kmeans -n 256 -s $1 -iter 100 -o 0         \
 cepstra$1_$2 all3.feat avk3.feat bwg3.feat dah3.feat dmw3.feat grg3.feat \
 hfs3.feat jrd3.feat mas3.feat mit3.feat mpr3.feat pas3.feat sek3.feat    \
 tdc3.feat wlw3.feat >& /dev/null
echo "K-means clustering for differenced cepstral coefficients..."
nice +19 /pro/rec/v3/vq/kmeans/kmeans -n 256 -s $1 -iter 100 -o 12        \
 dcepstra$1_$2 all3.feat avk3.feat bwg3.feat dah3.feat dmw3.feat grg3.feat\
 hfs3.feat jrd3.feat mas3.feat mit3.feat mpr3.feat pas3.feat sek3.feat    \
 tdc3.feat wlw3.feat >& /dev/null
echo "K-means clustering for energy coefficients..."
nice +19 /pro/rec/v3/vq/kmeans/kmeans -n 256 -s 4 -iter 100 -o 24         \
 energy$1_$2 all3.feat avk3.feat bwg3.feat dah3.feat dmw3.feat grg3.feat  \
 hfs3.feat jrd3.feat mas3.feat mit3.feat mpr3.feat pas3.feat sek3.feat    \
 tdc3.feat wlw3.feat >& /dev/null
echo "File conversion and codebook generation"
mv cepstra$1_$2.RVS.100 cepstra$1_$2.rvs.100
mv dcepstra$1_$2.RVS.100 dcepstra$1_$2.rvs.100
mv energy$1_$2.RVS.100 energy$1_$2.rvs.100
/bin/rm *RVS*
nice +19 /pro/rec/v3/vq/kmeans/converts cepstra$1_$2.rvs.100  \
 cepstra_float$1_$2.rvs.100 >>& /big309/speech/tp/logs/$1_$2.log
nice +19 /pro/rec/v3/vq/kmeans/converts dcepstra$1_$2.rvs.100 \
 dcepstra_float$1_$2.rvs.100 >>& /big309/speech/tp/logs/$1_$2.log
nice +19 /pro/rec/v3/vq/kmeans/converts energy$1_$2.rvs.100   \
 energy_float$1_$2.rvs.100 >>& /big309/speech/tp/logs/$1_$2.log
echo "Cepstral coefficients codebook..."
nice +19 /pro/rec/v3/vq/kmeans/make_codebook cepstra_float$1_$2.rvs.100  \
 codebook_cepstra$1_$2 >>& /big309/speech/tp/logs/$1_$2.log
echo "Differenced cepstral coefficients codebook..."
nice +19 /pro/rec/v3/vq/kmeans/make_codebook dcepstra_float$1_$2.rvs.100 \
 codebook_dcepstra$1_$2 >>& /big309/speech/tp/logs/$1_$2.log
echo "Energy coefficients codebook..."
nice +19 /pro/rec/v3/vq/kmeans/make_codebook energy_float$1_$2.rvs.100   \
 codebook_energy$1_$2 >>& /big309/speech/tp/logs/$1_$2.log
echo "Quantizing feature vectors of training files..."
foreach f ( *.feat )
 nice +19 /pro/rec/bin/fastvq $f codebook_cepstra$1_$2 \
          codebook_dcepstra$1_$2 codebook_energy$1_$2  \
          >>& /big309/speech/tp/logs/$1_$2.log
 end
echo "Training the HMM recognizer..."
```

```
nice +19 /pro/rec/bin/train *.feat -default_model -linear -codebooks 3 \
 -iter 30 -dur_type gamma >& /dev/null
echo -n "Probability after last iteration of training: "
tail -1 prob.dat
cp models.dat ../test
mv prob.dat prob$1_$2.dat
echo "Generating feature files for testing set..."
cd ../test
nice +19 /big309/speech/tp/commands/plpfeat_test_wo $1 $2 \
 >>& /big309/speech/tp/logs/$1_$2.log
cp ../train/codebook_* .
echo "Quantizing feature vectors of testing files..."
foreach f ( *.feat )
 nice +19 /pro/rec/bin/fastvq $f codebook_cepstra$1_$2 \
          codebook_dcepstra$1_$2 codebook_energy$1_$2  \
          >>& /big309/speech/tp/logs/$1_$2.log
 end
echo "Running recognizer..."
nice +19 /pro/rec/bin/batchrec *.feat -model models.dat >& /dev/null
mv models.dat models$1_$2.dat
compress models$1_$2.dat
mv stats.out stats$1_$2.out
tail -177 stats$1_$2.out > stats$1_$2.summary
compress stats$1_$2.out
echo "Batch terminated."
if ( -f stats$1_$2.summary ) then
  /bin/rm /big309/speech/tp/logs/$1_$2.log
  echo "Overall statistics for PLP-$1, $2 ms processing window follow:"
  tail -1 stats$1_$2.summary
endif
```

C. Training and testing file sets

Training set (small_train), from /speechbig/time/all/

```
abb2.time   abb3.time   aks2.time   aks3.time
all2.time   all3.time   amm2.time   amm3.time
avf2.time   avf3.time   avk2.time   avk3.time
bad2.time   bad3.time   boa2.time   boa3.time
bwg2.time   bwg3.time   cdt2.time   cdt3.time
cvc2.time   cvc3.time   dah2.time   dah3.time
dhd2.time   dhd3.time   dlc2.time   dlc3.time
dmw2.time   dmw3.time   dsr2.time   dsr3.time
ejh2.time   ejh3.time   grg2.time   grg3.time
gws2.time   gws3.time   haf2.time   haf3.time
hfs2.time   hfs3.time   jca2.time   jca3.time
jeo2.time   jeo3.time   jrd2.time   jrd3.time
kkm2.time   kkm3.time   lac2.time   lac3.time
mas2.time   mas3.time   mbl2.time   mbl3.time
mck2.time   mck3.time   mit2.time   mit3.time
mla2.time   mla3.time   mmc2.time   mmc3.time
mpr2.time   mpr3.time   mrh2.time   mrh3.time
 nk2.time    nk3.time   pas2.time   pas3.time
pja2.time   pja3.time   rmv2.time   rmv3.time
sek2.time   sek3.time   sjw2.time   sjw3.time
srm2.time   srm3.time   tdc2.time   tdc3.time
tsj2.time   tsj3.time   vlf2.time   vlf3.time
wlw2.time   wlw3.time
```

Testing set (small_test), from /speechbig/time/all/

```
adg1.time   adg2.time   adg3.time
ats1.time   ats2.time   ats3.time
bah1.time   bah2.time   bah3.time
bcb1.time   bcb2.time   bcb3.time
bsk1.time   bsk2.time   bsk3.time
crw1.time   crw2.time   crw3.time
dsc1.time   dsc2.time   dsc3.time
eew1.time   eew2.time   eew3.time
gac1.time   gac2.time   gac3.time
gms1.time   gms2.time   gms3.time
jls1.time   jls2.time   jls3.time
jtf1.time   jtf2.time   jtf3.time
kak1.time   kak2.time   kak3.time
lag1.time   lag2.time   lag3.time
mdh1.time   mdh2.time   mdh3.time
```

Beamformed data files, from /array3/dual_data/bf/time/

```
jea1.time   hfs1.time   msb1.time
jea2.time   hfs2.time   msb2.time
```

Simultaneous recordings with head-mounted microphone, from /array3/dual_data/dat/time/

```
jea1.time   hfs1.time   msb1.time
jea2.time   hfs2.time   msb2.time
```