Bolt Beranek and Newman Inc.

Report No. 4843

Research on Narrowband Communications

Quarterly Progress Report No. 5
18 August—17 November 1981

Prepared for:
Defense Advanced Research Projects Agency

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER Report 4843	2. GOVT ACCESSION NO. AD-A157846	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) RESEARCH ON NARROWBAND COMMUNICATIONS		5. TYPE OF REPORT & PERIOD COVERED Quarterly Progress Rep. No.5 18 August - 17 November 1981
		6. PERFORMING ORG. REPORT NUMBER BBN Report No. 4843
7. AUTHOR(s) John Makhoul Richard Schwartz Salim Roukos		8. CONTRACT OR GRANT NUMBER(s) F19628-80-C-0165
9. PERFORMING ORGANIZATION NAME AND ADDRESS Bolt Beranek and Newman Inc. 19 Moulton Street Cambridge, MA 02238		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE November 1981
		13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Deputy for Electronic Technology (RADC/EEV Hanscom AFB, MA 01731 Mr. Anton Segota, Contract Monitor		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this document is unlimited. It may be released to the Clearinghouse, Dept. of Commerce, for sale to the general public.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

20

18. SUPPLEMENTARY NOTES

This research was supported by the Defense Advanced Research Projects Agency under ARPA Order No. 3515, AMD. 4.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Speech compression, linear prediction, clustering, spectral template, vocoder, unsupervised learning, diphone, phonetic vocoder, phoneme recognition, time warping, segmentation, segment vocoder, segment quantization and space sampling.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This quarter we concentrated all our efforts on the unsupervised learning approach to the very-low-rate vocoder. In particular, we have detailed the design of a segment vocoder. The segment vocoder represents speech in terms of a sequence of segment templates, which have been automatically extracted from running speech.

In addition to designing the segment vocoder, we implemented three programs: (1) the segmentation program needed to extract segment templates,

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(2) an initial version of the segment vocoder, and (3) a display and playout program for comparing two files. All three programs were written to allow interactive use.

We have performed some experiments with the segment vocoder. The results of these experiments are discussed in this QPR.

keywords include:-

(19)

Report No. 4843

RESEARCH ON NARROWBAND COMMUNICATIONS

Quarterly Progress Report No. 5
18 August - 17 November 1981

Prepared by:

Bolt Beranek and Newman Inc.
10 Moulton Street
Cambridge, Massachusetts 02238

Prepared for:

Advanced Research Projects Agency

TABLE OF CONTENTS

		Page
1.	OVERVIEW	1
1.1	Outline	1
1.2	Segment Vocoder	2
1.3	Segmenter	2
1.4	Segment Vocoder Program	3
2.	SEGMENT VOCODER	4
2.1	Inventory of Segment Templates	5
2.1.1	Average Template Duration	5
2.1.2	Type of Template	7
2.1.3	Clustering of Templates	9
2.1.4	Amount of Training Speech Needed	10
2.2	Segment Matcher	11
2.3	Segment Template Network	12
2.4	Distance Metric	14
2.4.1	Dynamic Programming vs. Fixed Warping	14
2.4.2	Time-Weighting	15
2.4.3	Time/Gain/Voice Differences	15
2.5	Transmission of Other Information	16
2.5.1	Voicing Flag	16
2.5.2	Gain	17
2.5.3	Segment Duration	19
2.5.4	Detailed Time Warping	19
2.5.5	Pitch	20
2.5.6	Bit-Rate Estimates	21
2.6	Speaker Normalization	22
3.	SEGMENTATION PROGRAM	24
3.1	Variable Length Segments	24
3.2	Complexity of Segmenter Algorithm	25
3.3	Segmentation Algorithm	25

4. SOFTWARE DEVELOPMENT AND EXPERIMENTAL RESULTS 29

4.1 Software Description 29
4.2 Segment Quantization 30

1. OVERVIEW

This Quarterly Progress Report covers work performed during the period 18 August 1981 through 17 November 1981.

In this quarter we concentrated our efforts on the unsupervised learning approach to the design of a very-low-rate (VLR) vocoder. In particular, we considered several alternative designs for a "segment vocoder," where whole speech segments on the order of phonemes are coded. In support of this effort we implemented an automatic segmenter, an initial version of the segment vocoder, and a diagnostic display/real-time playout program for research on the segment vocoder. We report on the results of a few experiments with different segment quantization schemes.

1.1 Outline

Section 2 gives a detailed description of our plans for implementing the segment vocoder. Section 3 describes our implementation of the segmentation program used to presegment the training and input data. Section 4 describes the current implementation of the segment vocoder and some experimental results. Below is a summary of Sections 2-4.

1.4 Segment Vocoder Program

A program was written to serve as a test bed for all of our proposed algorithms for the segment vocoder. The program was written with many user switchable flags to control which options are to be executed. At the present time, the program has algorithms for "space sampling" segments of speech, finding the nearest template for each segment of input speech, and reconstructing spectral parameter tracks to be resynthesized. Each option is being tested incrementally by listening to the reconstructed speech.

The modules implemented, experiments tried, and conclusions drawn are enumerated in detail in Section 4.

2. SEGMENT VOCODER

In QPR-3 we discussed our implementation of a variable order Markov chain to reduce the conditional entropy of each spectral transmission, given the previous few transmitted spectra. Though there is a decrease in the entropy, it is slow (a decrease from 6 bits to 4.5 bits with 100 states), and the amount of training required grows exponentially with the order of the Markov chain. The amount of training needed can be reduced by decreasing the number of templates, but at around 6 bits the quality becomes poor, with intelligibility starting to suffer.

During the last quarter we proposed a new approach, called "segment quantization." In analogy to "vector quantization" (clustering) of single frame spectral vectors (of Log Area Ratios), segment quantization quantizes a whole sequence of input frames (a segment) minimizing the average error over the segment. We can then represent speech as a sequence of segment templates which have been extracted automatically from running speech. The segment vocoder takes advantage of the correlation between successive frames of speech, thereby reducing the bit rate below that of a single frame vector quantization vocoder.

We list below a proposed set of algorithms for a very low rate vocoder between 100-200 b/s. For many of the algorithms we also suggest alternate algorithms and their expected impact.

2.1 Inventory of Segment Templates

There are several considerations in determining the segment template inventory. Among these are:

1. Average template duration,
2. Type of template (e.g., "phoneme-like", "diphone-like"),
3. Clustering of templates,
4. Amount of training data needed.

2.1.1 Average Template Duration

We have shown that it is possible to design a vocoder with good quality and intelligibility using single frame vector quantization and variable frame rate transmission. However, the minimum bit rate for the spectral parameters alone in this vocoder is 200 b/s (8 b/frame * 25 frames/s).

We can now consider quantizing each pair of spectra jointly. If all 2^8 possible spectra could occur independently in each of the two positions, we would have 2^{16} possible pairs. We would only have to send half as many codes per second of speech, so the bit rate would remain the same. However, we know that not all 2^{16} possible pairs will occur, and those that do occur are not equally likely - hence some reduction in bit rate is possible. Still, as with the Markov chain system, the amount of training and number of possible sequences quickly becomes unmanageable as the length of the sequence grows.

Now we appeal to the benefits of segment quantization and to the nature of speech perception. By quantizing each segment to the nearest segment template, we can reduce the required number of sequences (segment templates), since some of the frames in any segment match are now allowed to have a larger error, as long as the average error is low enough.

Probably more important is the perceptual effect of quantization. We know that small changes in the spectra of speech should not change the meaning. After all, there are only 40-50 possible phonemes in English. However, if we quantize the spectrum of each frame of speech independently, we introduce a random noise component into the spectral parameters. This random noise is perceived as "roughness" or "wobble" and can be very annoying and distracting. While VFR tends to smooth it out, it also results in some slurring. If, on the other hand, we replace each segment (say 10 frames) of input speech with another segment template that has been extracted from real speech, without quantization, then, within each segment, the quality will be identical to that of unquantized LPC vocoded speech. We know from our phonetic synthesis work, which is based on the concatenation of diphone templates, that the overall speech quality resulting from template synthesis is good. We have also found in our recent experiments, that if the spectral templates being used to resynthesize the speech are a close match to the input speech the quality of the synthesized speech is acceptable. Of course, if the segment templates is very different from the

original segment, the perceived meaning may be changed or lost. This requires only that we have enough templates to account for a few times the possible phonetic differences.

We have chosen to use segments of roughly phoneme duration (i.e., 10-12/s or 8-10 frames per segment), since we know that this low transmission rate will make a total of 100-200 b/s possible. At the same time, we know that if we were to allow 8000 segment templates (13 bits) we would be allowing several different segment templates for each likely pair of phonemes. Our segmentation program is therefore tuned for this average rate.

2.1.2 Type of Template

Given that we have decided that our segments should be roughly 8-10 frames long on the average, we still need to specify the particular criteria for a segment. Two types of segments that we have considered are "phoneme-like" and "diphone-like" segments. Phoneme-like segments correspond to a steady state with a transition on each end, and a diphone-like segment as being two steady states on either side of a transition. Although the unsupervised learning approach is based on not using "phonemes" or "diphones" explicitly however, it is useful to model speech as a sequence of steady states and transitions.

We have found that transitions can usually be detected more consistently than steady states in speech. However, given a set

of transitions, we can determine the "diphone-like" segments by simple rules. One effect, yet to be determined, is the difference in quality when we abut and smooth within a steady state or a transition.

I thought this was the basis for using a diphone model in the first place

One argument in favor of diphone-like segments is that they should be less sensitive to small shifts in segment boundaries if we use the "space sampling" distance metric discussed below, since space sampling tends to ignore steady states.

Another issue is the relative number of templates and amount of training speech necessary for each type of segment. If we assume that phonemes are largely independent of context, then we may need only a few (say 10-20) templates for each of 40-50 phonemes. On the other hand, even one template for each diphone requires 2500-3000 templates. Since some diphones occur much more frequently than others in natural speech, we need much more speech before we could hope to see even half the possible diphones.

Our initial experiments with a small data base of 4000 segments (described in Section 4) were inconclusive. The phoneme-like segments resulted in a slightly smaller computed error for the best match than did the diphone-like segments; probably due to the small training issues discussed above. However, the quality with diphone-like segments was slightly better than with the phoneme-like segments, probably due to issues related to smoothing between templates.

2.1.3 Clustering of Templates

In order to keep the bit rate at a minimum, we must at least eliminate any segment templates from the inventory that are sufficiently similar to another template. With 4000 templates, we believe at least half could be eliminated (a decrease of 1 bit) with no loss in quality. As the number of templates increases (say to 32000) we expect that the percentage that can be eliminated will increase. A clustering algorithm could be used to eliminate duplicates. A simple algorithm is to make one pass through the data base, finding for each segment the nearest of the accepted templates, discarding the new template if there is another template that is sufficiently close.

Using the same distance metric that we use to compare segment templates with input segments, we could use a more complex clustering algorithm to determine the "optimal" set of templates in a mean square sense. However, we know from our experience with spectral clustering that we need to have at least 50 training segments for each of the final templates for the template quantization to be robust. If we end up with 8000 templates, this would require at least 400,000 initial templates, or a 12-hour data base. We may consider such a data base at some time in the future, since we need only digitize the speech and let the programs run. We may need to run the program on the VAX in stand-alone mode to have the necessary paging space. But, for the time being, we are only considering 1-2 hours of training data.

If we use a binary clustering algorithm analogous to that used for spectra (see QPR-1) we can greatly reduce the computation needed to find the nearest segment template. clustering algorithm. The binary lookup for one of 8000 templates takes only 13 comparisons instead of 8000 - a factor of 600 in speed. For this reason primarily, we will implement a binary segment clustering algorithm. What parameters are you using to perform the search?

2.1.4 Amount of Training Speech Needed

As mentioned in the preceding section, we would probably like about 8000 templates in the final inventory. This means 32000-64000 initial templates or 1-2 hours of speech. We will probably digitize about 1/2 hour of speech from each of 2-3 speakers, to experiment with several single talker systems (with about 2000-4000 templates). We will also digitize 5-15 minutes from each of several speakers for experiments with multispeaker systems, and for further research into "speaker normalized" systems.

We have implemented a program on the VAX that allows us to continuously digitize speech, as it is spoken, into fixed length files for later batch processing. Thus, digitizing 15 minutes of speech takes little more than 15 minutes, once begun.

2.2 Segment Matcher

Given an inventory of segment templates, we need to be able to determine the sequence of templates that best matches an input utterance. We consider two methods. The first method, "presegmentation", is very simple and requires much less computation, but is also suboptimal. The second method, dynamic programming segmentation, requires roughly 200 times more computation, but will allow us to decrease the bit rate for the same spectral error.

The presegmentation method uses the same segmentation program (discussed in Section 3) used on the training data to segment the input. With the segmentation of the input thus fixed, each segment can then be compared exhaustively (or using the binary search) to the inventory of templates. Since the same segmenter is used, the type of segments is the same for the templates as for the input, and a reasonable match can usually be found. However, the match is not optimal.

A simple dynamic programming approach can simultaneously find the best segmentation and the best sequence of templates for that segmentation. The procedure is outlined below. First, let's assume that no segment can be shorter than 5 frames or longer than 25 frames.

1. For j = frame i-25 to i-5
2. Find the nearest template to match the input between frames j and i

3. Add this score to the best match ending at frame j

4. Keep track of the best match over all j

5. Record at frame i the best j, and the best template

6. Increment i and go to (1)

The above procedure requires roughly 200 as many segment distance computations as the presegmentation approach. It could be made 4 times faster by only considering every other frame as a possible ending point.

A reasonable compromise between computational requirements and optimality would be achieved by tuning the segmentation program to produce more segment boundaries than needed, and then for the dynamic programming to only consider these frames as possible ending points. If only 20 "plausible boundaries" were created per second, the dynamic programming would be 25 times faster.

2.3 Segment Template Network

One way to significantly reduce the bit rate is to constrain the sequence of segment templates. In the phonetic vocoder, we allowed any given diphone to be followed only by a diphone that started with the consistent phoneme. This cut the bit rate by half from the case where any diphone can follow any other diphone. However, it also decreased the intelligibility somewhat. In the segment vocoder we have considered two basic

alternatives for the application of sequential constraints, using three coding options: Determine for each segment template,

1. those segment templates which: have been observed to follow the first segment template
2. determine those segment templates that start with a spectrum that is spectrally close to the last spectrum of the first template.

Either method above determines, for each template, a subset of the other templates, and also a ranking among that subset (by probability or distance). Given this subset, we can either

(a) Disallow all templates that don't fit the sequential model, or

(b) Code the subset with a short code, and reserve an escape sequence to allow indication of one of the other templates, or

(c) Use a Huffman code based on relative probability or distance, with some assumed low probability for the unobserved templates.

Assuming 800 possible templates, estimating probabilities of pairs of template would be impossible with only a few hours data. Therefore, the heuristic based on distance is favored.

Coding options (b) and (c) result in a variable length code, which is sensitive to channel errors. However, the hard constraint in option (a) would probably result in a loss in quality. Therefore, we feel that option (b) might result in the best compromise.

2.4 Distance Metric

A critical issue in the segment vocoder is the choice of a distance metric and a segment comparison algorithm. We have discussed such aspects as a warped frequency axis to reflect auditory perception, and a component of the distance measure to reflect differences between segments in spectral derivative. Here we discuss some additional issues relevant to the segment vocoder, all assuming the same basic spectral distance measure.

2.4.1 Dynamic Programming vs. Fixed Warping

In our phonetic vocoder, there is a network dynamic programming (DP) algorithm for comparing a path through the diphone network to input speech. Thus, implicitly, the matcher allows several different time warpings between any particular template and a region of the input hypothesized to correspond to that input. The DP then uses the best of the scores for that correspondence.

A second approach is to assume some predetermined warping and compute only one score. A warping that we have experimented with is called "space sampling". In space sampling, a segment is sampled in equal increments along its multi-dimensional trajectory, rather than in equal units of time. If two segments to be compared are first "space-sampled", then, essentially, differences in time within each segment are ignored.

The dynamic programming approach will always get a smaller

distance between the two segments, at the cost of one or two orders of magnitude in computations. The question is, can it allow some time-warping that is not reasonable? Let us consider this further. If we were going to transmit the particular time-warping used for the match along with the template identity, the DP approach would be clearly better. However, since, at 100-200 b/s, we cannot send time-warping information, we need to say whether we can hear the difference between speech and a time-warped version of itself. This question will be answered only by experiment.

2.4.2 Time-Weighting

For the "space-sampling" metric, there is a possible problem. Most of the samples are taken from the short spectral transition, rather than the longer steady state. Therefore, the error tends to give less weight to the steady state region. A solution to this is to multiply each difference between two space samples by the duration of speech that it accounts for. Thus, the error will be more uniformly distributed in time.

2.4.3 Time/Gain/Voice Differences

As discussed in the following section, we try to predict voicing, gain, timing within a segment, and even the total duration of the segment either partially or completely from the transmitted segment itself. Therefore, differences between segments in these features should be added to the distance

between the segments, so that when the "nearest" template identity is transmitted, the receiver will also have a good estimate for the other (time, gain, voicing) values. This modification must also be used in the distance metric for clustering.

Including these additional dimensions we will require more templates, but the total bit rate should decrease, due to the statistical dependence among the spectrum and all the other features.

2.5 Transmission of Other Information

The discussion so far has been limited to transmission of a spectral sequence. The receiver also needs, for each frame, a voicing flag, a pitch value, and a gain value. It needs to know how many frames are to be synthesized with each template, and, given that the matcher used some time warping, it needs to know or assume the time warping. If we were to use even one bit for every 20 ms for a gain value and a pitch value, that alone would require 100 b/s. Below we propose algorithms for transmitting and/or predicting each of these values. Some of the proposals would require too many bits for the 100-200 b/s system, but are included for completeness.

2.5.1 Voicing Flag

In our work on spectral clustering, we found that there was

a high correlation between the spectrum and the voicing flag. In particular, roughly 85% of the clusters were either voiced at least 90% of the time, or unvoiced at least 90% of the time. Allowing for 10% voicing error (which would sound bad) we would have to transmit a voicing flag for those 15% of the spectra that can be both voiced or unvoiced. If we want a fixed transmission rate, we can instead define additional clusters that differ only by voicing. This would increase the number of clusters by 15%.

The voicing flag is much better predicted in segments than in individual frames. In addition, isolated voiced or unvoiced frames can be eliminated from the templates before using them. Therefore, we probably can have the receiver always use the voicing bit that is in the templates. Again, if we ever see any significant differences in voicing between two segment templates that are otherwise similar, we can always add them to the inventory.

2.5.2 Gain

We know that the conditional variance of gain conditioned on the nearest spectral cluster is almost as high as the unconditioned variance. Again, we might hope for something better with segments. Even though the absolute gain values might not be predicted, we would certainly expect that the shape of the gain contour might be predicted by the spectral sequence.

The first step that is necessary is to account for overall

speaking differences between the training data and the input data. This can be done by continuously normalizing the gain of both templates and the input to the speaking level. The program can basically keep running estimates of speaking level and make all measurements relative to that.

The second step is for the synthesizer to compare the received gain track of each segment with those of the two adjacent segments. If there are obvious problems (e.g., both ends of the gain track are higher than the ends of the adjacent segments) the gain tracks can be adjusted by rule.

Third, given the first two steps, the transmitter measures the difference in the average gain of the template and the input. This difference is then quantized (to 1 or 2 bits) and transmitted. This adjustment to the gain can be thought of as an indication of stress level which is somewhat independent of spectrum (phonemes). Since the gain adjustment is mostly due to stress, it might be able to be predicted from the relative duration of the segment (see below). That is, if the input segment is longer than the template, it is probably also louder.

Fourth, we must smooth the gain track with those of the adjacent segments to eliminate discontinuities.

Taken together, these four steps should reproduce a gain track that sounds as good as the original. If more is needed (for a higher bit rate system), we could also transmit another 2-

bit code to indicate an adjustment to the average slope of the gain track, (i.e., a ramp to add to the gain track).

2.5.3 Segment Duration

We could, without additional error, transmit a 4-bit code to indicate the duration of each segment. However, we probably would not mind a 15% error in the total duration of a segment. This assertion suggests the use of a logarithmic quantization for segment duration. We should also be able to predict the segment duration from the distributions of durations for the initial training templates that matched the template being used. Based on the original VLR feasibility study, we feel that 2 bits for total segment duration will be sufficient.

If we change the segment durations, then, in a real-time vocoder, we need a feedback mechanism to ensure that the error in time doesn't build up. We propose the method used in the VLR study. That is, rather than quantizing to the nearest available duration, we would quantize to whichever of the two closest levels was best in terms of reducing any accumulated real-time error.

2.5.4 Detailed Time Warping

In a VLR vocoder, we could not afford bits to specify what time warping was used. This means that the synthesizer would have to use a predetermined algorithm for stretching or shrinking the template to the desired duration. The middle of the template

could either stretch proportionately or we could use our diphone synthesis rules for nonuniform stretching. In a higher rate vocoder, if necessary, we could transmit 2-bit code that says which of 4 time-warping adjustments to use. We believe that, given a spectral sequence, the time evolution of that sequence is relatively invariant in natural speech. Therefore, we believe that transmission of time-warping adjustments would be unnecessary.

2.5.5 Pitch

For pitch transmission we will first use the analogue of the piecewise linear fit within each phoneme-like region. That is, the vocoder will transmit a pitch difference between the pitch at the transition in the middle of the preceding diphone-like segment and the transition in the middle of the current segment. We have shown that these pitch differences can be coded on a nonuniform scale without significant loss in quality.

The transmitter uses its knowledge of what quantized values it has transmitted in the past to determine the new value that will most closely approximate the actual input pitch value. The pitch is reconstructed at the receiver by linear interpolation between transmitted pitch values. If an entire diphone-like segment template that has been transmitted is unvoiced, then no pitch value need be sent.

2.5.6 Bit-Rate Estimates

Below we estimate the bit rate for three different VLR vocoders.

Network - Variable Length Codes:

	Bits/Segment
Segment Template 1/8000, allowing 256 choices on the avg.	8
Total segment duration	2
Pitch (for 2/3 of segments)	1 1/2
Gain	1 1/2
	13 x 11 = 143 b/s

Assumes an average of 11 segments/second.

Network - Fixed Length Codes:

	Bits/Segment
Hard decision - 1/1000 templates	10
Templates	
Duration	2
Pitch	3
Gain	2
	17 x 11 = 187 b/s.

No Network - Fixed Length Codes:

	Bits/Segment
Spectral Template 1/8000	13
Duration	2
Pitch	3
Gain	2
	20 x 11 = 220 b/s.

The three systems described above would, we assume, all probably

have about the same quality. The third system transmits 13 bits for each template, even though only 1000 templates could occur knowing the previous template. However, there is much less computation and complexity required than in the first and second system. The second system takes advantage of the sequential constraints, but is restricted to a fixed length code for each value, and always sends pitch, even if unvoiced. The first system uses a variable length code. It normally sends a 7-bit code for the template, but once every 10 segments it sends an escape code plus another 10 bits to say which of the 1000 templates to use. It entropy codes the pitch differences, and only sends pitch if voiced. It also entropy codes the 4 gain adjustment levels.

2.6 Speaker Normalization

We would like the final vocoder to be usable by many people, without each person having to record an hour of speech. The most straightforward way to accomplish this is to train the system on enough speakers and speech so it has seen most of the sequences that are likely to occur. However, this might increase the bit rate considerably due to having multiple templates for "related" sounds from different speakers.

Our previous work on multiple speaker phonetic synthesis showed us that we could use templates from one speaker to synthesize the speech of another by a frequency scaling and

filtering. This same normalization could be used on analysis. However, this normalization would not account for different pronunciations between speakers.

A compromise position would be to train the vocoder (the segment templates) on the speech of many speakers. However, the speech of each training speaker and the input would each have their own speaker characteristics removed. In this way, the template inventory would have variations in pronunciation, but would not have multiple entries that differ only by vocal tract length and overall spectrum.

We may also try a speaker normalization based on each speaker's distribution of the LAR's directly, rather than frequency scaling and spectral filtering, since our experience with speaker identification has shown us that different speakers have different distributions of LAR parameters.

3. SEGMENTATION PROGRAM

The segmentation program is used to segment the training data into templates that can be used to represent speech. It can also be used to presegment the input speech in one mode of operation of the segment vocoder, though this function may be performed jointly with segment classification using dynamic programming.

There are many different criteria that could be used for determining the set of templates. We have already argued that segments need to be of a duration comparable to phoneme duration in order to achieve the low bit rate required. Below we discuss some of the relevant issues.

3.1 Variable Length Segments

The simplest type of segmentation would be to use a fixed size block of frames, say 10 frames. Both the templates and the input speech could be divided into fixed blocks, which would be easily matched. The problem with this approach is that it would require a much higher data rate than a variable size block or segment for the same spectral error. We know that speech consists of units that vary in length. Therefore, if we used a fixed size block, we would have to have a shifted version of each block for each possible lag between the block and the natural segment. There would also be extra templates needed to represent different combinations of different phonemes, diphones, and

partial phonemes. Therefore, we have decided to extract only those units that are unique, in that they can be used, with appropriate time warping, to represent all of speech.

3.2 Complexity of Segmenter Algorithm

Given our decision to use phoneme-like or diphone-like units for our segment templates, the next question is whether the segmenter should be designed to do the best possible job of finding those units that we call phonemes, or instead, find units that are acoustically distinct according to a simple distance criterion. For the present, we have chosen the latter approach. The segmenter finds relative steady state regions and transitions between those regions, using simple weighted spectral derivative measures and peak picking algorithms.

3.3 Segmentation Algorithm

Here we present the current segmentation algorithm. We expect that there will be only minor changes, in light of the fact that we have decided to keep the algorithm simple.

The first step is to compute a spectral derivative with respect to time. The first component of this derivative is the Euclidean distance between the log-area-ratio (LAR) vectors of two successive frames. This distance is multiplied by two weighting factors that are intended to make the segmentation be somewhat closer to a true phonetic segmentation.

We weight the distance by the energy in the immediate region. The formula for this weight is:

$$W1 = \frac{\max(e1, e0)}{40} \quad (1)$$

where e1 and e0 are the log energy in the two frames being compared. These log energy values are typically around 45-50 dB for vowels and 10-15 dB for silence.

We also multiply the distance by a function of the spectral tilt, favoring voiced regions:

$$W2 = \frac{(25-G1)}{30} \quad (2)$$

where G1 is the first log area ratio of the second frame being compared. G1 is typically -15 for sonorants and +5 for fricatives.

In addition to this 1-frame spectral derivative, we compute a 3-frame derivative that compares the LAR vectors with a lag of 3 frames. The same weighting factors are used. This 3-frame derivative measure is used to detect slow transitions, such as between successive sonorants and vowels.

Once the derivative measures are computed, the program looks for sharp transitions, which are defined as regions with high 1-frame derivatives. For each derivative there are two thresholds. The program searches for a frame with a 1-frame derivative greater than the higher threshold. Then the program looks for

the next frame with a derivative below the lower threshold. It then looks backward from the second point for the last frame above the higher threshold. Then, it computes the time between the first and last frame with derivatives above the higher threshold. If this duration is less than 5 frames (50 ms) it creates one boundary at the frame with maximum derivative. If the region is longer, it creates two boundaries, one at each end of the region.

When the program has found two boundaries, it then uses the same algorithm, between these two boundaries, looking at the 3-frame derivative. If large slow transitions are found, the program inserts boundaries between the two 1-frame derivative boundaries.

Once the program has found all the transition boundaries, it must create segments for the segment quantizer. If the segment quantizer is to use phoneme-like segments, then segments are defined as the regions between successive boundaries. Each segment starts one frame after the boundary and ends on the next boundary.

If the segment quantizer is using diphone-like segments, the program looks for steady points between the boundaries. The steady point is defined as the frame with the lowest 1-frame derivative in the middle half of the region between two boundaries. The segments go from the frame after the steady point to the next steady point.

If the segmenter detects long silences while creating segments for templates, it does not create a segment. A silence is defined as a region that is at least 25 frames long with maximum energy below 20 dB, or 35 frames long with maximum energy below 25 dB.

With the current thresholds and data base, the segmenter creates about 11 segments per second of running speech. It finds about 80-90% of the hand labelled phoneme boundaries, while creating 10-20% extra boundaries. As discussed in the next section, there are only minor differences in the quality of the segment vocoder output when using the automatic segmentation instead of the manual segmentation.

4. SOFTWARE DEVELOPMENT AND EXPERIMENTAL RESULTS

4.1 Software Description

During this quarter, we implemented and tested several features of the segment vocoder. We developed two interactive programs to evaluate the different techniques for vocoding speech based on segment quantization. The first program (SEGMQUANT) simulates the vocoder, while the second program (DISPLAY) is a diagnostic tool.

SEGMQUANT is a highly interactive and flexible program that allows the simulation of the different aspects of segment vocoding. It is structured to allow an incremental testing of the different features of the vocoder. The spectrum was quantized (using segment quantization) in several manners. We use this incremental approach to isolate the degradation due to the different parameters of the vocoder.

The second program DISPLAY is a diagnostic tool to compare two signals (e.g., the input and output of the segment vocoder, or two different versions of vocoder output). It provides graphics support and listening capability. Interactively, one can generate parameter plots to compare the trajectories of the two signals, look at their time synchrony and listen to the corresponding segments of the two signals. This program has been useful in locating, analyzing and understanding several problems in the vocoded speech. We describe several experiments and results obtained with the above two programs.

4.2 Segment Quantization

During this quarter, we compared several methods for segmenting speech. The methods were evaluated by listening to the quality of the output speech of a vocoder based on segment quantization. The vocoder consists of the following steps:

1. The input speech is segmented into segments whose duration is comparable to the duration of a phoneme. Each segment is space-sampled as described in QPR4. We used 7 space-samples per segment to make linear interpolation error due to space-sampling negligible.
2. Each segment is quantized to the nearest segment template using a Euclidean distance on LARs. The spectra from the chosen templates are used for the vocoder output.
3. The input timing, gain and pitch are used without quantization in synthesizing the output speech.

In this system, the degradation in the quality of vocoded speech is due to spectral quantization. The mean-square-error (on LARs) per space-sample (in analogy to per frame) is computed. The same segmentation method was consistently used both in getting the segment templates, and in segmenting the vocoder input.

The segment templates are obtained by segmenting a database of continuous speech. We used the same data base used earlier for diphone recognition. It contains 55 Harvard sentences and 200 short phrases uttered by a single male speaker. The resulting segments are all used as segment templates. Since, we had only 4000 templates (12 bits), clustering was not necessary.

Three segmentation methods were used. The first was supervised and based on representing speech as a sequence of phonetic units. The other two methods are unsupervised and model speech as a sequence of steady states in the spectral domain. These methods are:

1. Diphone Segmentation: Speech was segmented into diphones by hand. We reported on this effort earlier. This supervised approach isolates well established phonetic units of speech production. However, it requires a considerable manual effort.

2. Diphone-Like Segmentation: Speech was automatically segmented by the segmentation algorithm described in Section 3. The segments represented units of speech that correspond to a spectral trajectory from the middle of a steady state, through a transition, to the middle of the following steady state.

3. Phoneme-Like Segmentation: The same automatic segmentation algorithm is used. However, the segments are shifted; a segment begins in the middle of a transition, continues through a steady-state and ends in the middle of the following transition.

Segmentation Method	Number of Templates	Input Segmentation Rate (seg/s)	MSE per space-sample
Diphone	4200	12	37.25
Diphone-like	3800	11.5	40.61
Phoneme-like	3800	11.5	37.60

Table 4.1 Comparison of three segmentation methods.

The advantage of automatic segmentation is the minimal effort required to obtain a large inventory of templates. However, it is not clear that segmentation based on a crude measure (spectral derivative) will be as efficient for vocoding

as diphone segmentation carefully determined by hand, particularly since diphones are "well-established" units of speech.

The purpose of comparing the two automatic segmentation methods is to determine which segments are better concatenated: diphone-like segments or phoneme-like segments.

A set of 5 sentences spoken by the same speaker, but not used for the templates, were vocoded. Table 4.1 summarizes the experimental conditions and the corresponding mean-square error. The average bit rate for the spectral information was around 138 b/s. For the sake of comparison, the quantization error of the segment vocoders corresponds to a frame-by-frame vector quantizer of 4.5 bits per frame.

The intelligibility of the vocoded speech using diphone or diphone-like segmentation was comparable as judged in an informal listening test. While the output speech of the vocoder using diphone-like segmentation was generally crisper than with diphone segmentation, it sometimes resulted in missing or altered phonemes. A particular occurrence of vowel modification was determined to be due to input segmentation error. Two diphones in the input were considered as one segment by the diphone-like automatic segmenter. This problem may be reduced by improving the segmenter.

The overall output speech of the two vocoders is just

intelligible. However, the intelligibility is expected to improve by adding more segment templates: a relatively easy task with automatic segmentation. We expect the intelligibility to increase without increasing the bit rate since clustering will be used to eliminate templates that are very similar. The current data base of 4000 templates is expected to have several redundant templates. Furthermore, there are several techniques to improve the intelligibility without increasing the bit rate, discussed in Section 2, that we plan to investigate.

Though the quantization error of phoneme-like segmentation is slightly smaller than diphone-like segmentation, the quality of vocoded speech was quite inferior for this method. This is a preliminary result that indicates that diphone-like segmentation is more effective.

Duration Weighting

We investigated a different distance metric for segment quantization. The mean-square error of each space-sample is weighted in the calculation of the distance between two segments. The weights are proportional to the duration of each input space-sample.

Since transitions usually represent large spectral changes, most space-samples of an equi-distant space-sampling representation are in the transition regions instead of the steady-states. To improve the spectral matching of the steady-

state part of an input segment, we used duration weighting. The duration of a space-sample is defined as the average of the two durations: The time interval from the previous space-sample and the time interval to the following space-sample. Hence, space-samples that correspond to steady-states will have a better spectral match to the templates than samples from transitions.

The output speech of the phoneme-like segment vocoder with duration weighting improved significantly with duration weighting. The diphone-like segment vocoder with duration weighting was just slightly better than without weighting. The diphone-like vocoders are still better than the phoneme-like vocoders. We plan to investigate other nonlinear duration weights. For the diphone-like vocoder, only 20% of the classifications of the input segments were changed by duration weighting, perhaps explaining the small effect of duration weighting for the diphone-like vocoder.