# REAL-TIME PROBABILISTIC SEGMENTATION FOR SEGMENT-BASED SPEECH RECOGNITION[1]

*Steven C. Lee and James R. Glass*

Spoken Language Systems Group
Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139, USA

## ABSTRACT

In this work, we investigate modifications to a probabilistic segmentation algorithm to achieve a real-time, and pipelined capability for our segment-based speech recognizer [4]. The existing algorithm used a Viterbi and backwards $A^*$ search to hypothesize phonetic segments [2]. We were able to reduce the computational requirements of this algorithm by reducing the effective search space to acoustic landmarks, and were able to achieve pipelined capability by executing the $A^*$ search in blocks defined by reliably detected phonetic boundaries. The new algorithm produces 30% fewer segments, and improves TIMIT phonetic recognition performance by 2.4% over an acoustic segmentation baseline. We were also able to produce 30% fewer segments on a word recognition task in a weather information domain [11].

## 1. INTRODUCTION

The SUMMIT segment-based speech recognizer developed by our group searches an acoustic-phonetic graph during the decoding phase [4]. Although this graph can conceivably contain all possible segmentations of the speech signal, we have traditionally chosen to incorporate an explicit segmentation phase into the recognizer in order to reduce the size of the search space. The segmentation has typically consisted of restricting the *locations* of phonetic transitions, by identifying a set of allowable boundaries or landmarks, and also restricting the set of *connections* between landmarks (i.e., hypothetical phonetic segments).

The use of a segmental framework for recognition allows us to consider a richer set of acoustic-phonetic features than can be incorporated into conventional frame-based representations [6]. Currently, for example, feature vectors are extracted for phonetic analysis both over hypothesized phonetic segments and at their corresponding boundaries. We have always realized, however, that the use of an explicit segmentation stage can be a source of possible error if necessary phonetic segments are not hypothesized. Unfortunately we cannot search the entire segment space in near real-time; thus the segmentation stage is an important component in our recognizer.

Our early *acoustic* segmentation methods used spectral information to identify landmarks and segments [3]. More recently, we have developed a segmentation procedure which uses a frame-based Viterbi and backward $A^*$ to produce a phonetic graph [2]. Since this method uses probabilistic acoustic-phonetic models, the resulting graphs appear to be a better match to the subsequent segment-based search. Thus they can generally be smaller than our acoustic segmentations, and have improved phonetic and word recognition accuracies as well [6]. Although this probabilistic segmentation algorithm is effective, it is computationally intensive, and cannot run strictly in a left-to-right fashion. In this work [10], we describe modifications to the algorithm which enable us to achieve real-time recognition performance while maintaining the improved quality of the graphs.

## 2. EXPERIMENTAL FRAMEWORK

Experiments for this work are conducted in phonetic recognition and word recognition. For phonetic recognition, the TIMIT acoustic-phonetic corpus is used [8]. As is frequently done by others to report TIMIT recognition results, the set of 61 TIMIT labels are collapsed into a set of 39 labels [9]. For word recognition, the JUPITER corpus is used [5]. The corpus consists of spontaneous speech data from a live telephone-based weather information system. While complete experimental results are presented here for TIMIT, only final results are presented for JUPITER due to space limitations.

Utterances are represented by 14 MFCCs computed at 5 ms intervals. Both boundary-based diphone models and segment-based models are used. The context-dependent diphone models are mixtures of diagonal Gaussians based on MFCC averages extending out to 75 ms on both sides of the boundary [5]. The segment models are also mixtures of diagonal Gaussians, based on measurements taken over segment thirds; delta energy and delta MFCCs at segment boundaries; segment duration; and the number of boundaries within a segment [4]. Language constraints in all recognition experiments are provided by a bigram. Error rate is computed as the sum of substitutions, insertions, and deletions. To measure computation, a real-time factor is used. It is defined as total recognition processing time on a 200MHz Pentium Pro, divided by the total time of the speech utterances being processed. A number greater than one translates to processing slower than real-time.

## 3. LANDMARK-BASED REPRESENTATION

In our original probabilistic segmentation procedure the first pass phonetic recognizer is frame-based [2]. In this work, we sought to reduce the computational requirements of the algorithm by shrinking the search space of the first pass recognizer. Instead of scoring at regularly spaced 10 ms frames, we first investigated using lower frame-rates. In addition, we experimented with

[1]This research was supported by DARPA under contract N66001-96-C-8526, monitored through Naval Command, Control and Ocean Surveillance Center.

| Frame-interval (ms) | Error Rate (%) | Real-Time Factor |
|---|---|---|
| 10 (constant) | 28.9 | 3.01 |
| 20 (constant) | 28.2 | 1.52 |
| 30 (constant) | 29.4 | 1.01 |
| 33 (variable) | 28.5 | 0.92 |

**Table 1:** TIMIT *dev* set results for various frame rates.

*landmarks* that have been detected by a spectral change algorithm. These variable frame-rate landmarks have been successfully applied previously to an acoustic segmentation algorithm, and eliminate large amounts of computation spent considering sections of speech unlikely to be segment boundaries [4].

To study the viability of decreasing computation by lowering the frame rate of the first pass recognizer, we evaluated the phonetic recognition performance and computation requirements using different frame rates. Although the segmentation algorithm produces a graph of segmentations rather than just a single choice, we felt that overall top-choice performance would be correlated to the overall quality of the corresponding graph. Frame-based diphone models were used for these experiments.

As shown in Table 1, the results are divided into two sections. The top section presents results for regularly spaced frames, and the bottom section presents results for variable spaced landmarks. The table shows that as the frame-interval increases (decreasing frame rate), computation expectedly decreases. For regularly spaced frames, error rate improves initially as the frame rate decreases but worsens substantially at very low frame rates. For landmarks, error rate is competitive even when compared to the best error rate from regularly spaced frames. Overall, the table shows that switching from a constant frame-interval of 10 ms to landmarks does not significantly degrade error rate, but significantly reduces computation. Based on these results, all subsequent experiments in this paper use a landmark-based search.

## 4. BLOCK PROCESSING

In addition to minimal computation requirements, a real-time algorithm must also be able to run in a pipeline. The original probabilistic segmentation algorithm could not run in a pipeline because it relied on the backward $A^*$ search to produce the $N$-best paths. This required the completion of the forward Viterbi search before the backward $A^*$ search could begin. This section addresses the pipelining problem and describes a block processing algorithm in which the Viterbi and $A^*$ searches run in blocks defined by reliably detected phonetic boundaries. In addition, this section introduces the concept of soft boundaries to allow the $A^*$ search to recover from mistakes by the boundary detection algorithm.

### 4.1. Mechanics

Figure 1 illustrates the block probabilistic segmentation algorithm. As the speech signal is being processed, probable segment boundaries are located. As soon as one is detected, the algorithm runs the forward Viterbi and backward $A^*$ searches in the block defined by the two most recently detected boundaries. The $A^*$ search outputs the $N$-best paths for the interval

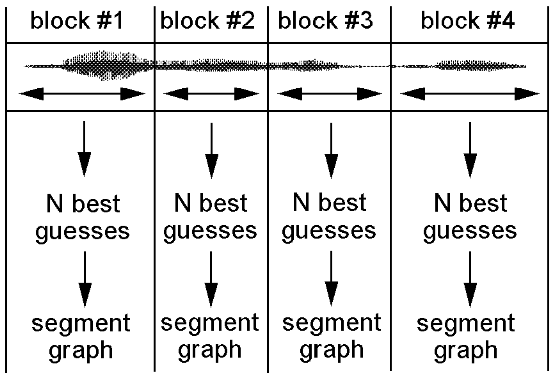

**Figure 1:** Illustration of block processing using hard boundaries.

of speech spanned by the block, and the segment-graph for that section is subsequently constructed. The algorithm continues by processing the next detected block. The end result is that the segment-graph is produced in a pipelined left-to-right manner as the input is being streamed into the algorithm.

### 4.2. Boundary Detection Algorithms

The boundary detection algorithm used to detect the block boundaries must have two properties. First, the boundaries detected must be very reliable, as the $N$-best algorithm running in each block cannot possibly produce a segment that crosses a block. A missed boundary by the boundary detection algorithm is much preferred to one that is inserted because the probabilistic segmentation algorithm running within each block can hypothesize segment boundaries inside the block. Second, the boundary detection algorithm must produce boundaries at a reasonable frequency so that the latency for the segmentation algorithm is not too long. In this work, two different boundary detection algorithms were examined. They are described separately below.

**Acoustic boundaries** The acoustic boundary detection algorithm detects probable segment boundaries based on acoustic change. Boundaries are placed at major peaks of spectral change in the speech signal. These boundaries are a subset of the landmarks used to save computation. A threshold on the height of the peaks controls the frequency of the boundaries. In this work, the threshold is set such that a boundary is detected on average every 200 ms. Using this threshold, approximately 85% of the detected boundaries in TIMIT were within 10 ms of an actual segment boundary in the phonetic transcription. Since even humans frequently disagree about the precise placement of segment boundaries, we believed this was a reasonable result.

**Viterbi boundaries** The Viterbi boundary detection algorithm is based on statistics in the Viterbi search. Boundaries are placed at frames where all active nodes above a threshold are transition nodes. The threshold controls the frequency of the boundaries. In this work, it was set to produce a boundary on average every 200 ms. The performance of this algorithm is similar to that of the acoustic boundary detection algorithm.

**Experiments** In this experiment, the difference in performance between acoustic and Viterbi boundaries in the block segmentation algorithm was examined. The boundaries were evaluated on segment-based recognition performance and computational requirements. The TIMIT *dev* set results are shown in Figure 2. Recognition performance in terms of number of segments per second versus error rate, is plotted on the left, and computation performance, shown as the number of segments per second versus the real-time factor, is plotted on the right. The number of segments per second is controlled by a variable $N$ that determines the number of $N$-best paths to include in the segment-graph. The acoustic boundaries are represented by the broken lines, and the Viterbi boundaries are represented by the solid lines. The computation plots on the right show that they both require about the same amount of computation. However, the recognition plot on the left shows that the Viterbi boundaries clearly outperform the acoustic boundaries in terms of recognition error rate. Therefore, all subsequent TIMIT experiments use Viterbi boundaries. For JUPITER, acoustic boundaries outperform Viterbi boundaries. All subsequent JUPITER experiments use acoustic boundaries.

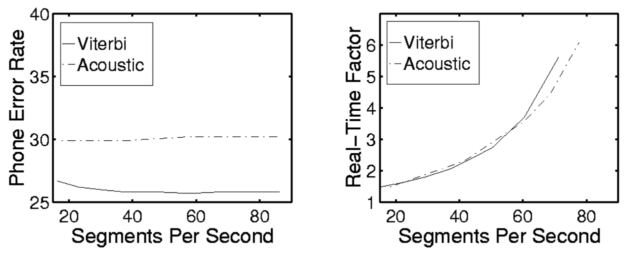

**Figure 2:** Plots showing recognition and computation performance of acoustic versus Viterbi boundaries.

## 4.3. Recovery From Errors

The statistics presented for each of the boundary detection algorithms show that they are generally reliable. However, they are not perfect. In particular, they do occasionally insert a boundary where a boundary does not exist. When this occurs, the $N$-best algorithm running between the boundaries cannot hypothesize actual segments that cross the boundary.

To counter this problem, soft boundaries were introduced. Figure 3 illustrates this concept. In contrast to Figure 1, where the $N$-best algorithm runs between every neighboring hard boundary, the $N$-best algorithm runs between *every other* soft boundary. This allows the $N$-best algorithm to recover from mistakes in the boundary detection algorithm by hypothesizing segments that span parts of two blocks. Unfortunately, this benefit comes at a cost. An algorithm using soft boundaries requires more computation than one using hard boundaries because some sections of the speech signal are processed twice. In addition, an algorithm based on soft boundaries has a higher latency because the output lags the latest input data by at least one block.

**Experiments** In this experiment, the performance difference between soft and hard boundaries was examined. Again, the boundaries were evaluated on segment-based recognition performance and computational requirements. The TIMIT *dev* set results are shown in Figure 4. Recognition performance in terms of number of segments per second versus error rate, is plotted on the left, and computation performance, shown as the number of

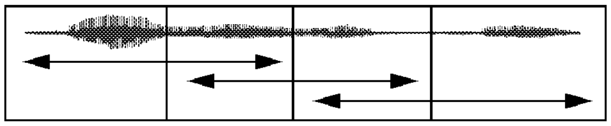

**Figure 3:** Illustration of block processing using soft boundaries.

segments per second versus the real-time factor, is plotted on the right. The soft boundaries are represented by the broken lines, and the hard boundaries are represented by the solid lines. The left recognition plot shows that the soft boundaries outperform the hard boundaries (especially at low segment rates) in terms of error rate, but the right computation plot shows that this performance comes at a cost of greater computation, as expected. Similar results were obtained for JUPITER. This is one tradeoff to be taken into account when looking for an optimal operating point for the segmentation algorithm.

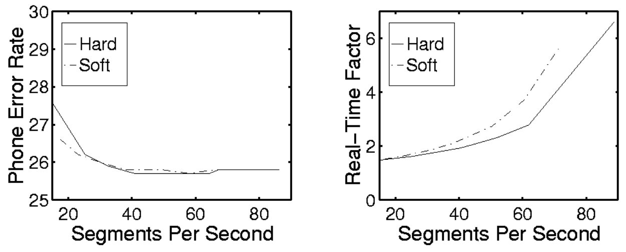

**Figure 4:** Plots showing recognition and computation performance of soft versus hard boundaries.

## 5. FINAL EXPERIMENTS

Based on the results from the development experiments, final TIMIT experiments, done on the core *test* set, used Viterbi boundaries. For TIMIT, an improvement over the baseline acoustic segmentation in terms of error rate, number of segments, and computation was attained using soft Viterbi boundaries. In addition, when the recognizer was allowed to run without any computational constraints, a further error rate reduction was achieved by simply increasing the size of the segment-graph. This result is shown in Table 2.

| | Error Rate (%) | Segments/Second |
|---|---|---|
| Baseline | 29.1 | 87.2 |
| Real-time | 28.4 | 56.6 |
| Slower than real-time | 28.1 | 61.3 |

**Table 2:** Final TIMIT recognition results on the core *test* set.

Final JUPITER experiments, done on the *test* set, used acoustic boundaries. For JUPITER, the new segmentation algorithm achieved an improvement in terms of error rate and number of segments using acoustic soft boundaries. However, the algorithm at that operating point required significantly more computation than the baseline. To further reduce computation, the full set of phonetic labels used in the segmentation algorithm were collapsed into a set of broad-classes. A broad class size of 20 was able to achieve an improvement in word error rate and number of segments at a much more reasonable level of computation.

Table 3 summarizes the *test* set results for JUPITER. In the table, the real-time result used the set of broad-class models, and the slower than real-time result used the full set of models.

| | Error Rate (%) | Segments/Second |
|---|---|---|
| Baseline | 10.6 | 99.7 |
| Real-time | 10.5 | 65.2 |
| Slower than real-time | 10.0 | 76.3 |

**Table 3:** Final JUPITER recognition results on the *test* set.

## 6. DISCUSSION

In this paper, various modifications to the probabilistic segmentation algorithm presented in [2] were explored, with the goal of creating an algorithm that is fast, runs in a pipeline, and results in competitive recognition error rate. Computational savings were attained by using acoustic landmarks located at irregular intervals rather than regularly spaced frames. A left-to-right pipeline capability was achieved by running the probabilistic segmentation algorithm in blocks defined by probable segment boundaries.

The algorithm developed in this paper has several attractive attributes. First, the algorithm allows for a tradeoff between accuracy and computation as determined by the number of segments produced. If computation is an important factor for an application, the algorithm can be tuned to run faster than the baseline while still producing a competitive error rate. If error rate is more important than computation, the algorithm can be tuned to produce an error rate significantly better than the baseline. More importantly, the algorithm outputs a smaller segment-graph containing more relevant segments than that produced by the baseline acoustic segmentation. This allows more sophisticated segment-based modeling techniques to be explored. For example, Chang has developed a novel segment-based acoustic modeling technique, termed near-miss modeling, that relies on a quality segment-graph [1].

Finally, the algorithm produces information in the first pass recognizer that can be reused to guide acoustic modeling in the subsequent segment-based search. For example, if the first-pass recognizer identifies a segment to be a fricative, then the segment-based search can use features and models tailored for distinguishing between phones within the fricative class. Heterogeneous measurements that improve within-class classification performance have been developed by Halberstadt [6], and can easily be applied to this framework.

There are several possible extensions to this work. Currently the number of segments in the segment-graph is controlled by $N$, the number of paths used to produce the segment-graph. This $N$ is a constant, regardless of the size of the block being processed, or the confidence that the segments in the block are correct. Allocating a larger $N$ to bigger blocks or blocks with low confidence should help to distribute segments to areas of the speech signal with more uncertainty.

In this work, the relative improvement achieved for word recognition is less than for phonetic recognition. We believe that the algorithm's weaker performance on words can be attributed to a pronunciation network mismatch. For phonetic recognition, the pronunciation network used in probabilistic segmentation and in the subsequent segment-based search is the same. As is typical in phonetic recognition, this network allows any phone to follow any other phone. For JUPITER, the pronunciation network used in probabilistic segmentation allows any phone to follow any other phone, but the network used in the subsequent segment-based search contains tight word-level phonetic constraints.

This paper concentrated on the tradeoff between recognition performance and computation, without regard to memory requirements. However, memory can affect the speed of execution as well if the memory requirements are so enormous that time spent swapping memory dominates over time spent computing. This phenomenon is seen at very large $N$ in this paper.

Finally, a word graph search which directly computes a graph could replace the $N$-best computation. This should eliminate redundant computation used to expand previously seen segmentations in the $N$-best search [7].

## 7. REFERENCES

1. J. Chang, *Near-Miss Modeling: A Segment-Based Approach to Speech Recognition.* Ph.D. thesis, MIT, 1998.
2. J. Chang and J. Glass, "Segmentation and modeling in segment-based recognition," in *Proc. Eurospeech*, Rhodes, Greece, pp. 1199–1202, 1997.
3. J. Glass, *Finding Acoustic Regularities in Speech: Applications to Phonetic Recognition.* Ph.D. thesis, MIT, 1988.
4. J. Glass, J. Chang, and M. McCandless, "A probabilistic framework for feature-based speech recognition," in *Proc. ICSLP*, Philadelphia, PA, pp. 2277–2280, 1996.
5. J. R. Glass and T. J. Hazen, "Telephone-based conversational speech recognition in the JUPITER domain," in *these Proceedings*, Sydney, Australia, 1998.
6. A. Halberstadt and J. Glass, "Heterogeneous measurements and multiple classifiers for speech recognition," in *these Proceedings*, Sydney, Australia, 1998.
7. I. Hetherington, M. Phillips, J. Glass, and V. Zue, "$A^*$ word network search for continuous speech recognition," in *Proc. Eurospeech*, Berlin, Germany, pp. 1533–1536, 1993.
8. L. Lamel, R. Kassel, and S. Seneff, "Speech database development: Design and analysis of the acoustic-phonetic corpus," in *Proc. DARPA Speech Recognition Workshop*, Palo Alto, CA, pp. 100–109, 1986.
9. K. Lee and H. Hon, "Speaker-independent phone recognition using hidden Markov models," *IEEE Trans. ASSP*, vol. 37, no. 11, pp. 1641–1648, 1989.
10. S. Lee, *Real-Time Probabilistic Segmentation for Segment-Based Speech Recognition.* M.Eng. thesis, MIT, 1998.
11. V. Zue, S. Seneff, J. Glass, L. Hetherington, E. Hurley, H. Meng, C. Pao, J. Polifroni, R. Schloming, and P. Schmid, "From interface to content: Translingual access and delivery of on-line information," in *Proc. Eurospeech*, Rhodes, Greece, pp. 2047–2050, 1997.