ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ Η/Υ

Δημιουργία εξειδικευμένου Front-end σε αναγνώριση φωνής για την υποβοήθηση της αξιολόγησης της διαδικασίας του Alignment και Segmentation ηχογραφήσεων για γνωστό κείμενο

Αθανασία Κολοβού

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

20-09-2004

ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Δημιουργία εξειδικευμένου Front-end σε αναγνώριση φωνής για την υποβοήθηση της αξιολόγησης της διαδικασίας του Alignment και Segmentation ηχογραφήσεων για γνωστό κείμενο

Αθανασία Κολοβού

Επιβλέπων καθηγητής:: Αλέξανδρος Ποταμιάνος

Περίληψη

Από τις διάφορες στρατηγικές που έχουν χρησιμοποιηθεί τα τελευταία χρόνια για τη δημιουργία συνθετικής φωνής υψηλής ποιότητας, αυτή που παρουσίασε τα καλύτερα αποτελέσματα είναι η στρατηγική του Variable Unit Selection Synthesis. Με την τεχνική αυτή για μια δεδομένη πρόταση, τη στιγμή της σύνθεσης το σύστημα επιλέγει τον καλύτερο συνδυασμό φωνητικών μονάδων, διαφορετικών ίσως μεγεθών και τύπου (phones, diphones, συλλαβές, λέξεις η ακόμα και τμήματα προτάσεων), μέσα από το Speech Corpus, το ειδικά επεξεργασμένο σύνολο τον ηχογραφήσεων από έναν ομιλητή, το οποίο και βασίστηκε σε ένα ειδικά διαμορφωμένο σύνολο κειμένων. Οι μονάδες αυτές συνδυάζονται μεταξύ τους, και μετά την εφαρμογή κάποιας επεξεργασίας συρράπτονται για να δημιουργήσουν τη συντεθειμένη πρόταση.

Το Speech Corpus αποτελείται από δύο στοιχεία :

- τις κωδικοποιημένες ηχογραφήσεις που βασίστηκαν σε ένα προεπιλεγμένο σύνολο κειμένων

- πληροφορίες σχετικά με τα φωνήματα που αποτελούν την κάθε πρόταση που περιλαμβάνεται σε αυτό.

Όλες αυτές οι πληροφορίες εξάγονται αυτόματα με τη χρήση ενός παραμετροποιήσιμου αναγνωριστή φωνής. Δεδομένου ότι γνωρίζουμε το κείμενο (και τη φωνητική αναπαράσταση του), το οποίο διάβασε ο ομιλητής, η διαδικασία του ταιριάσματος αυτή ονομάζεται forced alignment. Κατά το forced alignment επιδιώκουμε να βρούμε τα όρια των φωνημάτων που απαρτίζουν κάθε πρόταση. Για τους σκοπούς όμως της δημιουργίας του Speech Corpus χρειαζόμαστε και όλα τα επιπλέον δεδομένα που αναφέραμε παραπάνω. Τα στοιχεία αυτά όμως μπορούν να ανακτηθούν μέσα από τον αναγνωριστή.

Σκοπός της διπλωματικής εργασίας είναι η δημιουργία ενός εξειδικευμένου αλγορίθμου στο σύστημα αναγνώρισης HTK, που παρέχεται από το Speech,Vision and Robotics Group του πανεπιστήμιου του Cambridge. Ο αλγόριθμος αυτό θα παρέχει τη δυνατότητα παραμετροποίησης της διαδικασίας του forced alignment, καθώς και εργαλεία για τον έλεγχο και τη διασύνδεση των αποτελεσμάτων με τα αντίστοιχα σήματα φωνής.

TABLE OF CONTENTS

ACKNOWLEDGMENTS

Four years in Technical University of Crete are now history! After finishing my thesis, I began to reflect upon how I entered this area of research. It was the newly arrived course of Natural Language Processing that made me pick up the book of Jurafsky & Martin and start exploring the various fields in speech recognition.

I would like to thank my advisor Mr. Alexandros Potamianos for motivating me to enter this research area. His guidance, encouragement, and understanding throughout the course of this work, from topic selection to the final experiments, were invaluable. I am also very grateful to Chris Vosnidis and Constantinos Harizakis, for all the interesting talks we had in many matters involving this work and the help I received from them in the early stages of my work.

I would like to especially thank my family for their love and support over the years. Also I would like to thank my friends for their entertaining conversations over lunch or coffee and to wish them all the best for their future.

More importantly, I would like to thank the most important person in my life, my husband Harry. His support was the constant driving force behind this work. I would like to thank him for making my life at TUC much happier than I thought it would be when we first arrived here. His understanding and encouragement throughout the good and bad times were essential to concluding this work.

Chapter 1

Introduction

Motivation

A fundamental assumption in much of speech processing is that the basic unit of speech is the phoneme. Most speech recognizers identify words based on their phonetic representation, and nearly all speech synthesizers concatenate or synthesize waveform segments according to phonetic pronunciations. As a result, one requirement of researching and building speech synthesis systems is the availability of speech data that have been labeled and time aligned at the phonetic level.

Time aligned phonetic labels can be created by a trained human labeler or by an automatic method. Although precise evaluation of the phonetic labeling is difficult, there is a general consensus that manual labeling is more accurate than automatic labeling. To give further weight to the claim that manual alignments are more accurate than automatic alignments, systems that depend on alignment information can be developed using both methods and the performance of the two systems can be compared. Our case study took the manual aligned sentences from TIMIT corpus and compared them with the ones we created from our own segmentation procedure.

Although manual alignment is considered more accurate than automatic methods, it is too time consuming and expensive to be commonly used for aligning large corpora. In addition to the greater time required to generate manual alignments there is variability in manual generated alignments due to the subjective judgment of the human labeler. Because of these disadvantages of manual alignment, there is a need for a fast, inexpensive and accurate means of obtaining time aligned phonetic labeling of speech waveforms.

The topic of this thesis then is the development of a method of performing speech segmentation that is more accurate than current automatic methods and significantly faster than manual alignment. This principles used to develop such a method may then be applied to other aspects of speech processing such as speech recognition and speech synthesis.

General Overview of the current methods

As noted above, the most accurate method of creating time aligned phonetic labels is to employ a trained human labeler. This person typically generates phonetic alignments using a software tool that displays the waveform, spectrogram, label and possibly other information. The labeler assigns the phonetic labels with the speech sentences, by listening the segments of the waveform and by using knowledge of the relationship between the waveform, its spectrogram and its phonetic content. As a result training in phonetics and spectrogram reading is required to produce acceptable label alignments, and manual alignment is a resource intensive method.

The most common automatic method for aligning speech is called "forced alignment". In this method, recognition of the speech signal is performed with the search results constrained to the known sequence of phonemes. Such systems obtain the alignment, by forcing the recognition result to be the proposed phonetic sequence and this phonetic sequence is determined in advance by a pronunciation dictionary. In general, there is a strong link between automatic speech recognition and forced alignment techniques, in that the same general process can often be used for both tasks.

One reason for the greater accuracy of human labeling over automatic methods is that humans are better able to detect distinct events in the speech signal that correspond to the specific phonetic characteristics, such as the sudden increase in energy that signals the beginning of a plosive. The development of our method was motivated by the belief that if an automatic alignment system could use such acoustic phonetic information, its accuracy will become closer to that of human performance, while still maintaining the internal consistency of current automatic methods. The model for the proposed method uses forced alignment as foundation.

This model then incorporates specific acoustic phonetic features and performs a refinement procedure based on this feature extraction procedure.

Chapter 2

Previous Work in phonetic alignment

Of a review of 32 automatic alignment systems, 44%(14 systems use HMM recognition to obtain the alignments, and 25% (8 systems) use dynamic time warping(DTW).The remaining systems (10 systems) employ a wide variety of approaches including methods that use estimates of voicing, measures of spectral variation, diphone detection, rules that encode acoustic-phonetic knowledge, etc. In this chapter we will mostly discuss the HMM systems and we will describe the current state of the art in automatic phonetic alignment based on the systems reviewed here

Automatic alignment agreement with manual labels is most often reported in terms of what percentage of the automatic alignment boundaries are within a given threshold of the manually-aligned boundaries. For example Brugnara reported that for their system 88.95% of their automatic boundaries are within 20ms of the manual boundaries. This type of result will be reported as a percent "agreement" within the given threshold. Relative differences in the agreement between two systems will be reported using the terminology "reduction in error" even if the alternate terminology such as "increase in agreement" may be technically more correct.

HMM systems

As mentioned before, HMM speech recognizers can be used to obtain phonetic alignments using a process called forced alignment. In this technique, the sequence of speech segments is matched with the sequence of phonetic labels. The results of the Viterbi search, contains the phonetic alignment (as well as the position of each phoneme within each sentence).

In cases where the words are known but the phonetic sequence is not, a dictionary can be used in combination with pronunciation rules to generate a phoneme sequence for each word; these sequences can then be concatenated together, with optional pauses between words, to arrive at a phoneme sequence for the entire utterance. Because the task of phoneme alignment can be considered as simplified speech recognition, it is natural to adopt a successful paradigm of automatic speech recognition, namely HMMs, for alignment [ref. num].

Wightman and Talkin [1] developed an HMM-based system called "The Aligner", with the acoustic model training and Viterbi search implemented using the HTK toolkit. The aligner uses a 10ms frame rate and five mixture components per Gaussian to estimate the state occupation likehoods. Non-speech sounds, such as breath noise and lip marks are collapsed into a single "silence" model. The system was trained on unvoiced and voiced stop closures, whereas most HMM systems train the stop closure and the stop burst as one unit. The system was trained using the TIMIT labels as an initial segmentation. In evaluation of their system they did not use the TIMIT phonetic sequence directly, but they first mapped the words to canonical dictionary pronunciations, that performed forced alignment , and then mapped the forced-alignment phonemes to the TIMIT phoneme sequence; this allowed them to compare the phonetic boundary alignments while still performing forced alignment from word-level information. Performance on the TIMIT test set was approximately 80% agreement within 20 ms.

Pellom [3] developed an HMM for forced alignment with a variety of speech enhancement algorithms. This system uses a 5msec frame rate, 5-state monophone HMMs, gender dependent models, 16 Gaussian mixtures components per state, and Gamma distribution transition probabilities. When phoneme level transcription probabilities are not available, the system generates pronunciations using the CMU dictionary and word-juncture modeling. The system was trained and evaluated on TIMIT data that had been down-sampled to 8KHz, had agreement was 86.2% within 20msec. Pellom evaluated the same system on the NTIMIT corpus of telephone band speech and the CTIMIT corpus of cellular band speech, using various noise-reduction techniques. For NTIMIT the system with the best combination of speech enhancement algorithms has 76.8% agreement within

20msec; for CTIMIT the best performing system had 66.7% agreement within 20msec.

Ljolje and Riley [4] built a 3-state HMM system that has different types of phonetic models, depending on the availability of training data. If enough data are available for each phoneme in its left and right contexts, then a complete triphone model is used, although the left and right contexts are clusters of similar phonemes instead of individual phonemes. If sufficient data are not available for a full triphone model, then a "quasi-triphone" model is attempted; this quasi-triphone model has the left state dependent on the left context, the middle state context independent and the right state dependent on the right context. If sufficient data are not available for the quasi-triphone model, then left context dependent and right context dependent models are attempted. If sufficient data are still not available then context independent phoneme models are used. The HMM uses full covariance Gaussian probability density functions to estimate the state occupation probabilities, a Gamma distribution duration model and a 10msec frame rate. The models were trained and evaluated on TIMIT database. Two types of models were trained, those based on manual alignments in the TIMIT database, and those based on a mixture of manual alignments and Viterbi re-estimation of the alignments. In either case they found 80% agreement within 15msec.

Ljolje, Hirschbergh and Van Santen [6] trained a monophone (context-independent) three state HMM system with Gaussian estimation of the state occupation likehoods. Gamma distributions were used to model the phoneme durations, and the frame size was 2.5 msecs. The system was trained using and initial uniform-duration segmentation of the states instead of manual alignments. Training and evaluation was done on Italian utterances in carrier phrases. When mean biases were removed from the results, performance was 78.1% agreement within 20msecs.

Other methods for automatic Phonetic Alignment

The HMM approaches are certainly not the only ones. In this section we will discuss an alternative system relevant to this thesis.

An alignment system developed by Santen and Sproat applies edge detectors to spectral-domain representations and energy information in different frequency bands. This information is combined with a set of phonetic sequences for each word to arrive at the aligned phonetic sequence. This approach focuses on detecting phonetic boundaries (referred to as diphones) rather than the conventional HMM approach for estimating the likehood of each phonetic category at every frame of speech. They note that the spectral cues to different types of phonetic transitions are contained in different frequency bands; for example a boundary between an /f/ and an /s/ has a decrease in energy bellow 2000Hz and an increase in energy above 4000Hz; a boundary between an /f/ and a vowel however, an increase in energy in the 800 Hz to 2500 Hz frequency range. The authors group the set of phoneme into two classes, "broad" and "narrow". To account for these two type of diphones, van Santen and Sproat use two representations of the speech signal; the first representation is energy in five different bands (for classifying the broad diphones) and the second is a mel-frequency FFT representation (for classifying the narrow diphones). Then they perform edge detection on the frequency bands detecting both quick changes and less localized changes. The frequency band information is combined in such a way that exact synchronicity in time is not required. This information is combined using Baye's rule to estimate the "overall acoustic cost" for each diphone at each time frame. For the narrow diphones the mel-FFT representation is used with vectors of weights that characterize each diphone to locate the time point of greatest change between the two phonemes. Van Santen and Sproat reported a 50% agreement within 2 msecs and 95% agreement within 20 msecs when evaluated on a single speaker. Although the use of a single speaker in the test corpus does not guarantee that the results will generalize to multi-speaker corpora, the extremely high agreement argues for the merit of this method.

State of the art in phonetic alignment

For the systems reviewed above that were evaluated on microphone quality speech, performance ranged for 77% agreement to 90% agreement [16,89] within 20 msec; variables that may affect performance include the method of training the system, the number of speakers in the training and test corpora, the type of corpus (isolated word or continuous speech) , and the language used in training and testing. Average performance is about 84% agreement within 20 msec, and HMM systems tend to outperform other systems.

Our system (HMM-based system) in the baseline segmentation process (performed with HTK Toolkit) had 84.53% agreement within 20ms.

C h a p t e r 3

Baseline Segmentation System with HTK

As mentioned earlier, our research focuses on segmentation of speech given a phonetic transcription. We will use HMM based ASR system for the first stage. The second stage, which we will discuss in chapter 7, will refine the time marks.

- Get the transcriptions (these are built by hand and we call them the CORRECT transcriptions) from TIMIT prompts (create a MLF with them).

- Create our own transcriptions (machine built transcriptions) using forced Viterbi alignment, where the features and correct words are given - best states - produces labels for each input.

- Compare the machine segments with the hand labeled segments; get statistics for each phoneme (compute mean error, variance, bias error, histogram, context error rate).

The TIMIT Corpus

The TIMIT acoustic-phonetic corpus [11] is widely used in the research community to benchmark phonetic recognition experiments. It contains 6300 utterances from 630 American speakers. The speakers were selected from 8 predefined dialect regions of the United States, and the male to female ratio of the speakers is approximately two to one. Each utterance in the corpus was hand-transcribed by acoustic-phonetic experts, both at the phonetic level and at the word level. At the phonetic level, the corpus uses a set of 61 phones, but we mapped the phoneme set to contain only into a simpler 48 phoneme symbol set. The aim of this mapping is to delete all glottal stops, replace all closures preceding a voiced stop by a generic voiced closure (vcl), all closures preceding an unvoiced stop by a generic unvoiced closure (cl) and the different types of silence to a single generic silence (sil). The set of phonemes used is described in following figure.

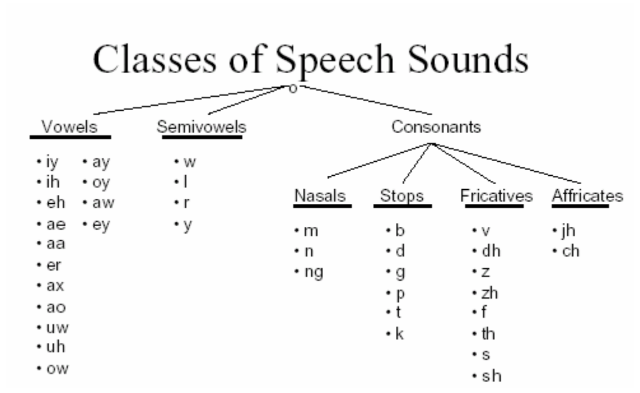


Figure 3.1 : The set of English phonemes

HTK Introduction

The Hidden Markov Model Toolkit (HTK) is a portable toolkit for building and manipulating hidden Markov models. HTK is primarily used for speech recognition research although it has been used for numerous other applications. HTK consists of a set of library modules and tools available in C source form. The tools provide sophisticated facilities for speech analysis, HMM training, testing and results analysis. The software supports HMMs using both continuous density mixture Gaussians and discrete distributions and can be used to build complex HMM systems.

Before describing the procedure we used to get the initial segmentation we will explain how and why we used HTK for it. HTK can be used to build a simple speech recognizer by manipulating hidden Markov models, being the core of most state-of-the-art speech recognition systems. The concept in speech recognition is that the acoustic information is sampled as a signal suitable for processing by computers and fed into a recognition process. The output of the system is a hypothesis for a transcription of the speech signal. Our automatic segmentation process is like having the following recognition system.

Input → given utterances

Lexical Model → phoneme sequence (phoneme or phoneme following another phoneme)

Acoustic Model → one HMM for each phoneme

Output → phonetic label for each utterance and the corresponding time boundaries

HTK Processing Stages

- **1st step – Data Preparation**

In TIMIT we find the information which gives us each sentence and its corresponding phoneme sequence. This is called a phone level transcription. Phone level transcriptions in TIMIT are build by hand so we call them «hand built» segmentation or the correct segmentation of the sentences. We need the correct segmentation to compare it with the one we will automatically generate for the same sentences (we call this the «automatic» segmentation).

Firstly we created an MLF file that contains all phone level transcriptions found in TIMIT. This was done with HTK, and we used the HLEd tool . (see Appendix for HLEd). Here the MLF file is created directly using the TIMIT phone list and by applying a script (HLEdscript), which maps the phonemes used form 61 to 48.The aim of this mapping is to delete all glottal stops, replace all closures preceding a voiced stop by a generic voiced closure (vcl), all closures preceding an unvoiced stop by a generic unvoiced closure (cl) and the different types of silence to a single generic silence (sil).

And in the following example we can see how HLEd affects this TIMIT label file

```
0 8920 h#
8920 9530 hh
9530 10694 ih
10694 12281 z
12281 12930 kcl
12930 13710 k
13710 15707 ae
```

```
15707 17540 pcl
17540 17830 t
```

The result will be :

```
0 5575000 sil
5575000 5956250 hh
5956250 6683750 ih
6683750 7675625 z
7675625 8568750 k
8568750 9816875 ae
9816875 10962500 p
10962500 11143750 t
```

Notice that label boundaries in TIMIT format are given in terms of sample numbers (16kHz sample rate), whereas the edited output file is in HTK format in which all times are in absolute 100ns units. The MLF that arises contains the phoneme sequence for each sentence and for each speaker but also contains start and end time marks for each phoneme within the sentence.

- **2nd step –Training Process**

Here we will have to get all TIMIT's train and test files (these are actually mfc files that are a result of the procedure described earlier, wav files > signal processing > feature vector) and separate each sentence in its corresponding phone sequence, by invoking forced segmentation technique. An HMM for each phoneme must be generated. In HTK this is done by writing a proto file first (this is a prototype HMM written in hmm definition language, just like it is described in the HTK book). The parameters of this prototype are not of a great importance, its main purpose is to give us the HMM topology (here we will use a 3-state left to right HMM).To create the HMM's for the phonemes we used HInit(exact implementation if HInit can be found in the Appendix). HInit is called for each phoneme in the phonelist, to initialize the HMM for that phoneme.

Next we used HHed for tying all individual HMM's together. This is how our acoustic model is generated. The models are then re-estimated with HRest. Here the parameters for each phoneme are re-estimated using the HMM definitions we got from HInit and a new hmm definition file will be created.
Now that the acoustic properties of the different phones have been encoded in statistical models and the sentence models are generated by concatenating all relevant phoneme models, the speech data can be assigned to the acoustic model of the complete phoneme sequence. This is done with the Viterbi algorithm and appropriate tool is HVite.

- **3nd step - Viterbi Alignment**

In speech recognition and several other pattern recognition applications, it is useful to associate an optimal sequence of states to a sequence of observations, given the parameters of a model. For instance, knowing which frames of features belong to which state allows location of the word boundaries across time. This is called alignment of acoustic feature sequences. A reasonable optimality criterion consists of choosing the state sequence (or path) that has the maximum likelihood with respect to a given model. This sequence can be determined recursively via the Viterbi algorithm.

In forced alignment the HMM is used to recognize the input speech with the Viterbi search constrained to only the correct sequence of phonemes. The result of the Viterbi search contains the phonetic alignment. In cases were the words are known but the phoneme sequence is not a dictionary can be used in combination with pronunciation rules to generate a phoneme sequence for each word, the sequences can be concatenated with optional pauses between words, to arrive at a phoneme sequence for the entire utterance.

HVite computes a new network for each input utterance using the word level transcriptions and a dictionary. By default, the output transcription will just contain the words and their boundaries. One of the main uses of forced alignment, however, is to determine the actual pronunciations used in the utterances used to train the

HMM system in this case, the -m option can be used to generate model level output transcriptions. This type of forced alignment is usually part of a *bootstrap* process; initially models are trained on the basis of one fixed pronunciation per word. Then HVite is used in forced alignment mode to select the best matching pronunciations. The new phone level transcriptions can then be used to retrain the HMMs. Since training data may have leading and trailing silence, it is usually necessary to insert a silence model at the start and end of the recognition network. The -b option can be used to do this.

As an illustration, executing
HVite -a -b sil -m -o SWT -I words.mlf -H hmmset dict hmmlist file.mfc

would result in the following sequence of events. The input file name file.mfc would have its extension replaced by lab and then a label file of this name would be searched for. In this case, the MLF file words.mlf has been loaded. Assuming that this file contains a word level transcription called file.lab, this transcription along with the dictionary dict will be used to construct a network equivalent to file.lab but with alternative pronunciations included in parallel. Since -b option has been set, the specified sil model will be inserted at the start and end of the network. The decoder then finds the best matching path through the network and constructs a lattice which includes model alignment information. Finally, the lattice is converted to a transcription and the output transcriptions would be written to an MLF using the -i option.

- **4th step - Evaluation**

The method of measuring the performance of an automatic alignment system is to assume that the manually generated labels are correct and compute the automatic alignment error relative to these values.

In this step we have to calculate mean error, variance and bias error, on start end time marks and find total error for each phoneme. We finally analyzed context behavior in our results and calculated error rates. Phones are divided into classes such as vowels, fricatives, nasals etc and we take phoneme pairs to calculate the requested error rates.

Results and discussion

In our approach for automatically determining phoneme boundaries, we compared the manually segmented transcriptions with the ones that were created automatically. We examined each phoneme individually, counted its mean, its standard deviation and its bias both in start and end time marks and we also grouped phonemes into their associated categories (see figure 3.1) and took phoneme pairs to see how a phoneme boundary is influenced by the preceding or the following phoneme's broad category.

Based on our results, comparing the two approaches, it is clear that the automatic alignments are worst when the phoneme and its adjacent one, are of the same category (or we can in other words say that they are acoustic similar). Most of the cases where the pair of phonemes belongs to the same category are actually encountered very few times (less than 50 as we can see in Table 3.4 were the appearance counts of each context are presented) but there are also cases where we have a count larger that 1000. For example ah+vowel has a mean error of 41.5ms and is found only 35 times and iy+vowel has a mean error of 21.8ms and is encountered 1443 times. Boundaries are also not well detected when we have silence following a phoneme especially when the phoneme belongs to the stop class (nasal and semivowel classes give a large mean error also), or when we have silence before a class (silence-stop is also the largest here as we can see in Tables 3.1 and 3.2).

On the other context combinations, affricates-fricatives transitions have a mean error of 24ms but this is not so often encountered in our transcriptions and a fricative-affricate transition has a large error of 37.3ms. The largest mean error on similar contexts is found when we have a stop phoneme and the adjacent phoneme is also a stop and this case is counted 2026 times (see context results of Tables 3.2 and 3.4 below). Vowel to semivowel transitions, are also very difficult to detect.

BIAS(ms)	AF	FR	NAS	SEM/V	SIL	STOP	VOW	*
AF	-13.0	-20.8	-3.4	-0.36	-12.3	-5.27	3.88	-7.33
FR	-37.3	-4.9	-3.0	-4.6	-10.4	-4.7	-0.7	-9.40
NAS	-1.5	-5.8	-5.8	-5.0	7.9	1.6	-2.8	-1.65
SEM/V	-2.9	-5.5	-1.4	-1.1	7.4	-1.7	3.3	-0.30
SIL	-10.1	-9.7	-12.4	-10.3	0	-37.7	1.9	-11.30
STOP	-0.4	-3.9	-9.1	-9.7	-27.2	-36.1	-4.9	-13.05
VOW	9.2	-6.2	-3.4	-1.3	14.3	-1.1	4.1	2.22
*	-8.02	-8.16	-5.55	-4.71	-2.89	-12.17	0.70	-5.83

Table 3.1 Context-context results for Bias

MEAN(ms)	AF	FR	NAS	SEM/V	SIL	STOP	VOW	*
AF	13.00	24.69	7.25	8.02	18.67	9.78	7.35	12.68
FR	37.31	15.94	9.66	9.42	19.08	10.34	7.45	15.60
NAS	7.50	9.96	16.57	15.90	24.88	11.59	9.29	13.67
SEM/V	4.19	8.61	10.41	16.55	23.46	7.22	15.94	12.34
SIL	10.92	15.58	15.13	15.15	0.0	39.55	10.58	15.28
STOP	6.86	11.48	13.76	11.47	33.83	40.82	8.47	18.10
VOW	15.12	8.76	9.40	19.98	23.20	6.56	19.65	14.67
*	13.56	13.57	11.74	13.78	20.45	17.98	11.25	14.62

Table 3.2 Context-context results for mean error

STD(ms)	AF	FR	NAS	SEM/V	SIL	STOP	VOW	*
AF	0.0	21.8	5.6	7.1	19.8	10.5	8.2	10.46
FR	20.1	15.0	9.8	8.7	20.7	10.9	9.7	13.58
NAS	6.0	8.4	16.9	16.8	22.9	12.4	12.5	13.72
SEM/V	2.8	6.7	18.4	19.7	42.8	8.1	16.5	16.44
SIL	6.7	14.1	11.7	21.8	0	18.3	12.5	12.77
STOP	6.2	11.7	12.4	8.8	32.9	27.3	9.0	15.5
VOW	25.9	6.9	11.5	20.7	21.4	7.6	23.6	16.84
*	9.72	12.14	12.37	14.81	23.50	13.60	13.16	14.19

Table 3.3 Context-context results for Standard Deviation (abs)

COUNT(ms)	AF	FR	NAS	SEM/V	SIL	STOP	VOW	*
AF	1	105	48	119	198	301	1665	2387
FR	3	1165	766	2137	2854	4669	13777	25371
NAS	56	2490	308	1361	1200	4063	6343	15821
SEM/V	5	1191	525	766	432	1679	16219	20817
SIL	119	2031	925	1772	0	1235	1453	12575
STOP	2182	3438	795	5529	1443	2026	14458	29871
VOW	21	14951	12454	9133	1408	15898	8286	62151
*	2387	25371	15821	20817	12575	29871	62151	168993

Table 3.4 Context-context appearance counts

The same tasks (HTK baseline process and evaluation) where performed with 8 and 16 mixtures per Gaussian and we the results are presented in the Appendix.

We then subtracted from all examples the value of global bias (we calculated all bias values for each phoneme and took a mean value of those) so that the error on each example will be reduced now. For a transition to a vowel for example, the average difference between automatic and manual segmentation can be more than halved when compensating for these biases (see Table 3.5).

On another experiment, we subtracted for each phoneme, the value of its bias (as we can see in Figire 3.2) and we presented those results on the column named "Error after sub. Individual Bias" of Table 3.5. The same experiments were done also by taking into consideration the context for each phoneme. So we present an experiment were we subtract the bias on every example, but this time the bias is a context-dependent value (for example if we had fricative—aa transition, we will subtract the bias value of the fricative—vowel context).

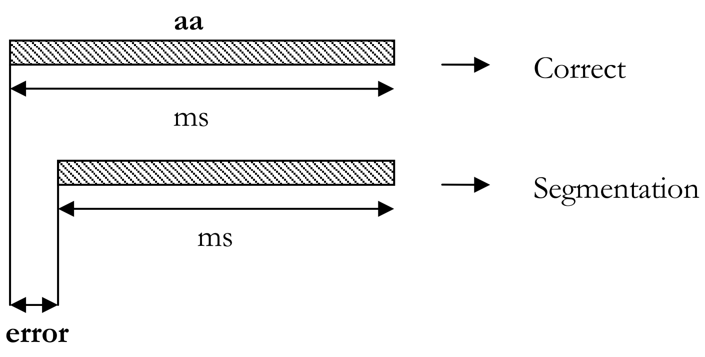


Figure 3.2: Bias definition

(ms)	All examples count %	After Sub. Global Bias %	After Sub. Phon. Bias %	After Sub. Context Bias %
[-4 , 4]	30.12	31.59	32.81	35.18
[-8 , 8]	53.85	58.97	57.03	58.28
[-10 , 10]	59.85	65.23	65.15	66.71
[-15 , 15]	76.00	78.62	78.87	80.07
[-16 , 16]	78.27	80.07	80.67	81.67
[-20 , 20]	84.53	85.42	85.77	86.73
[-30 , 30]	91.67	91.92	92.18	93.04
[-40 , 40]	95.03	95.26	95.42	96.04
[-50 , 50]	96.95	97.14	97.23	97.59
[-60 , 60]	98.13	98.25	98.29	98.52

Table 3.5 Overall “agreement” within a given threshold

We can see in Table 3.5 above, that in [-10, 10] area the count of examples is increasing after subtracting bias from all examples , which denotes that there was an improvement in the error and more phone examples were added into the [-10, 10] area. On the contrary the number of examples that are outside the [-10, 10] area is decreased, which is of course the expected outcome since the examples were “pushed” to a lower error category. The largest improvement in our results is encountered when we subtract context-context bias.

All this procedure was not determined to accomplish its purpose by its first implementation. We have a basic idea of what went wrong in this procedure but finding errors is rather difficult especially when there is no tool to graphically represent the results and to correlate them with the original waveforms. So, we created a graphical tool for visualization which will be presented in Chapter 5.

Histogram Results

Lastly, we constructed various histograms from our results and took some statistical measurements, all of which are summarized in the following tables. We took all of the differences we had calculated (between correct and segmentation transcriptions) and constructed a histogram to see the exact error distribution of all the examples.

To get a better insight on the boundary errors on the improvement after subtracting the bias histogram distributions are shown in the following figures.

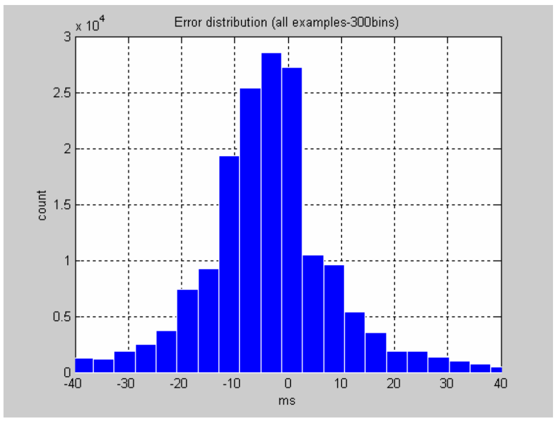


Figure 3.3

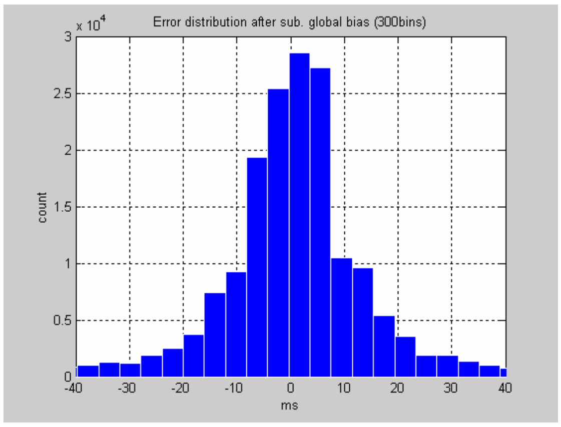


Figure 3.4

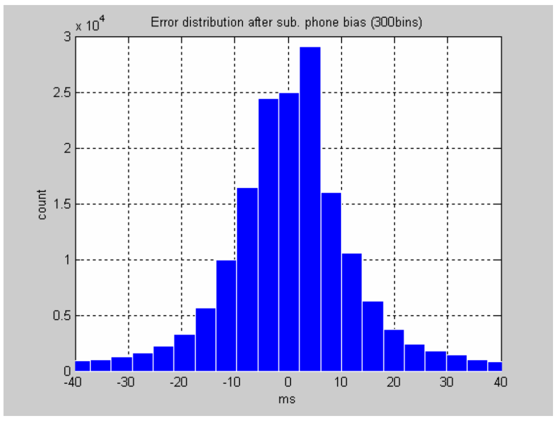


Figure 3.5

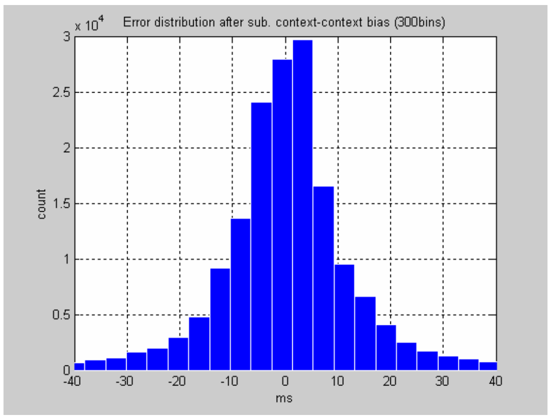


Figure 3.6

Figure 3.3 shows the initial error distribution of all examples and Figures 3.4, 3.5 and 3.6 the resulted distributions after the bias correction technique. All cases denote the improvement on the boundary placement as we can see from the figures mentioned, but the bias correction based purely on the basis of context gives the best results. As we can see in this case the histogram is more "smooth" left and right of the zero in the x-axis and it is also sifted in the center of x-axis .

Chapter 4

Basic Feature Extraction theory

The Fourier Transform (Interpretation in terms of time and frequency)

When the function *f* is a function of time and represents a physical signal, the transform has a standard interpretation as the spectrum of the signal. The real parts of the resulting complex-valued function *F* represent the amplitudes of their respective frequencies *(s)*, while the imaginary parts represent the phase shifts.

Linear Prediction Coefficients

Linear Prediction Coefficients (LPC) can parameterize the speech spectrum quite well. Vocal tract is modelled as an all-pole transfer function which is formed by a cascade of a small number of two-pole resonators representing the formants. Glottal model is represented as a two-pole low-pass filter.

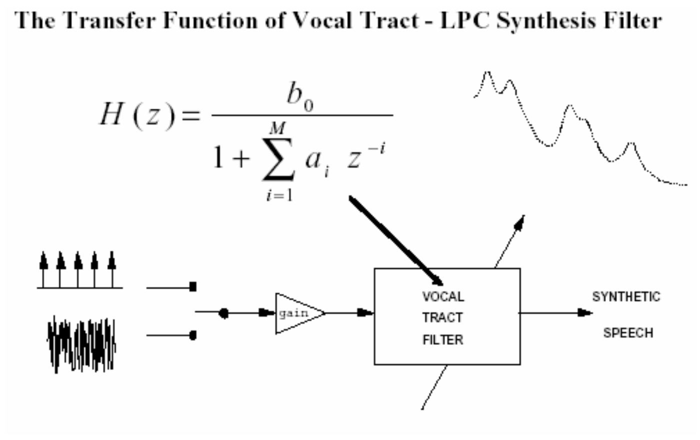

The Transfer Function of Vocal Tract - LPC Synthesis Filter
$H(z) = \frac{b_0}{1 + \sum_{i=1}^{M} a_i z^{-i}}$
gain
VOCAL TRACT FILTER
SYNTHETIC SPEECH

Figure 4.1

The LPC are the coefficients {a1, a2, ap}, estimated from the current frame of data, given the speech production model. An LPC sequence can be computed directly from the corresponding correlation sequence of equal length. A length twelve LPC vector can be derived from a length twelve autocorrelation, which in turn is interpreted as a smoothed Fourier spectrum. We used the Matlab function lpc() to get the LPS spectrum , we used Fs+2 coefficients, and plotted the frequency response of the result.

Cepstrum

A cepstrum (pronounced "kepstrum") is a Fourier analysis of the logarithmic amplitude spectrum of the signal. If the log amplitude spectrum contains many regularly spaced harmonics, then the Fourier analysis of the spectrum will show a peak corresponding to the spacing between the harmonics: i.e. the fundamental frequency. Effectively we are treating the signal spectrum as another signal, and then looking for periodicity in the spectrum itself. The cepstrum is so-called because it turns the spectrum inside-out. The x-axis of the cepstrum has units of quefrency, and peaks in the cepstrum (which relate to periodicities in the spectrum) are called rahmonics.The cepstrum may be defined verbally:

The cepstrum is the FFT of the log of the FFT

Mathematically:

X= FFT(the signal)

cepstrum of signal = FFT(log(X))

The real cepstrum uses the logarithm function defined for real values, while the complex cepstrum uses the complex logarithm function defined for complex values also. The complex cepstrum holds information about magnitude and phase of the initial spectrum, allowing the reconstruction of the signal. The real cepstrum only uses the information of the magnitude of the spectrum.

There are many ways to calculate the cepstrum, some of them need a phase-warping algorithm, and others do not. The cepstrum can be seen as information about rate of change in the different spectrum bands. Usually the spectrum is first transformed using the *Mel Frequency bands*. The result is called the *Mel Frequency*

Cepstral Coefficients or *MFCCs*. It is used for voice identification, pitch detection and much more. The cepstrum separates the energy resulting from vocal cord vibration from the "distorted" signal formed by the rest of the vocal tract.

Mel scale

The **mel scale**, proposed by Stevens, Volkman and Newman in 1937 is a scale of pitcheses judged by listeners to be equal in distance one from another. The reference point between this scale and normal frequency measurement is defined by equating a 1000 Hz tone, 40 dB above the listener's threshold, with a pitch of 1000 mels. Below about 500 Hz the mel and hertz scales coincide; above that, larger and larger intervalss are judged by listeners to produce equal pitch increments. As a result, four octaves on the hertz scale above 500 Hz are judged to comprise about two octaves on the mel scale. Mel scale is used to translate regular frequencies to a scale that is more appropriate for speech, since the human ear perceives sound in a nonlinear manner. In the case of speech recognition, a filter bank is applied of which the centre frequency of each bank is scaled according to the Mel scale. This scale takes into account the frequency resolution properties of the human ear. The inverse fourier transform of the log output of this filter bank yields the MelFrequency Cepstrum Coefficients.

The MFCC parameterization follows common requirements imposed on a speech parameterization for speech recognition purposes. Its main features are aimed above all at:

- to capture an important information presented in a speech signal for recognition purposes
- to handle as little data as necessary and
- to use any quick evaluation algorithm.

Moreover the benefit of MFCCs is also in their perceptually scaled frequency axis. The mel-scale offers higher frequency resolution on the lower frequencies in the same way as a sound is percept by the human auditory organ. In addition, the MFCCs offer through their cepstral nature abilities to model both poles and zeros.

Calculating Distances

The distance between two points is the length of the path connecting them. In the plane, the distance between points (x1,y1) and (x2,y2) is given by the Pythagorean theorem,

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}\,.$$

In Euclidean three-space, the distance between 3 points is

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}\,.$$

In general, the distance between points **x** and **y** in a Euclidean space is

$$d = |\mathbf{x} - \mathbf{y}| = \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2}\,.$$

However in statistics we prefer a distance that for each of the components (the variables) takes the variability of that variable into account when determining its distance from the centre. Components with high variability should receive less weight than components with low variability. This can be obtained by rescaling the components. By treating all values equally when calculating the distance from the mean point, it weights the differences by the range of variability in the direction of the sample point. This is done in the case of the Mahalanobis distance estimate. However we conducted some experiments with the Mahalanobis distance but there was no substantial improvement in our results, comparing them to the ones done with the Euclidean distance. Although the Euclidean metric is computationally more expensive than some metrics, it does give more weight to large differences in a single feature.

Chapter 5

A Matlab Software Tool for Speech Segmentation Refinement

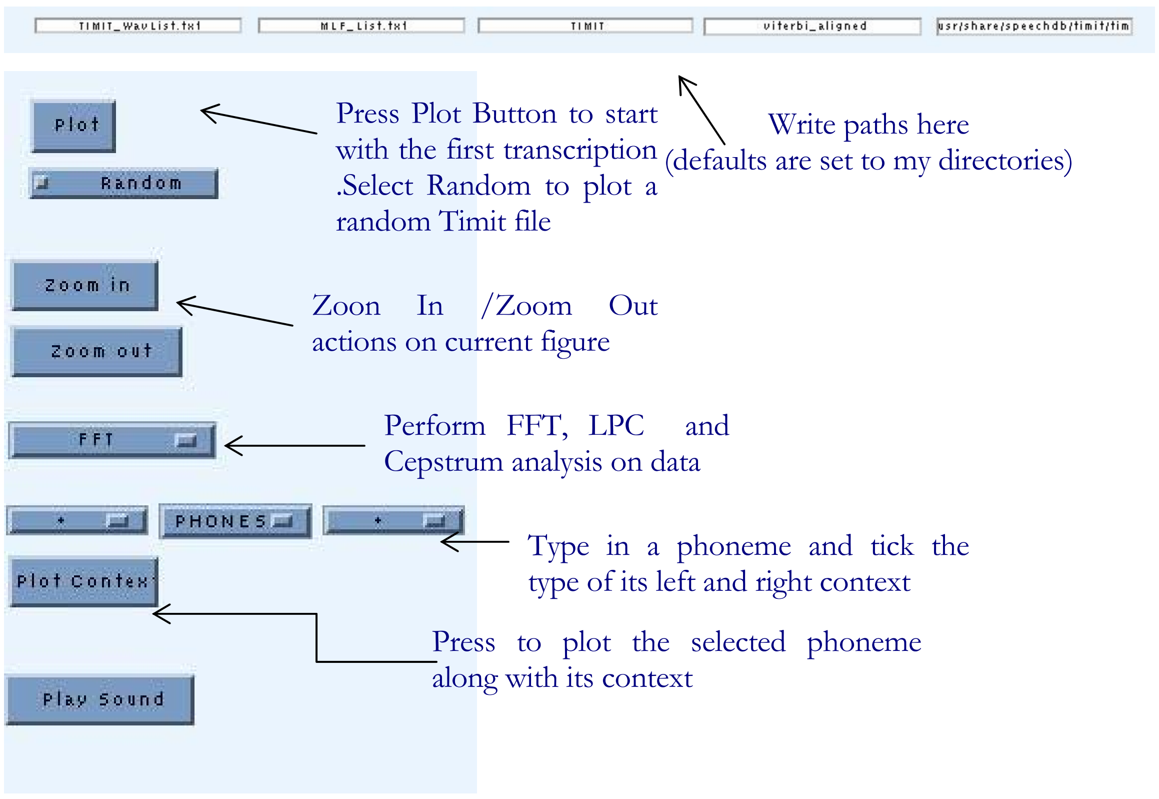
Tool description

Compatibility

Matlab 6.x and Matlab Signal Processing Toolbox

Introductory screen

When we run demo we get the following screen



Press Plot Button to start with the first transcription .Select Random to plot a random Timit file

Write paths here (defaults are set to my directories)

Zoon In /Zoom Out actions on current figure

Perform FFT, LPC and Cepstrum analysis on data

Type in a phoneme and tick the type of its left and right context

Press to plot the selected phoneme along with its context

Figure 5.1: The main Tool Area

First we need to make sure that the following files are in the default tool's directory

- TIMIT_WavList

This is a text file I created myself that goes like this…

/test/dr1/faks0/si1573

/test/dr1/faks0/si2203

It contains the paths to all wav files of TIMIT, and does not include the ".wav" file extension

- MLF_List

This is a also a text file that contains lines that go like this…

test_dr1_faks0_si1573

test_dr1_faks0_si2203

The purpose of this file is to help us "move" around the MLF files when we are processing them with Matlab, because each transcription starts with the file name extension given in the above specified format.

- TIMIT MLF

  We must specify the path for the MLF we have from TIMIT

- Segmentation MLF

  We must also specify the path of the MLF we have from the HTK segmentation procedure

  Then we must specify the following path :

- TIMIT Path

  Specify TIMIT database path were the wav files are selected.

Plot a waveform

Now that the paths are all completed, we press the **Plot** button and we get the following figure.

(When we press the plot button for the first time it may take a little time for the plots to appear, please be patient!)

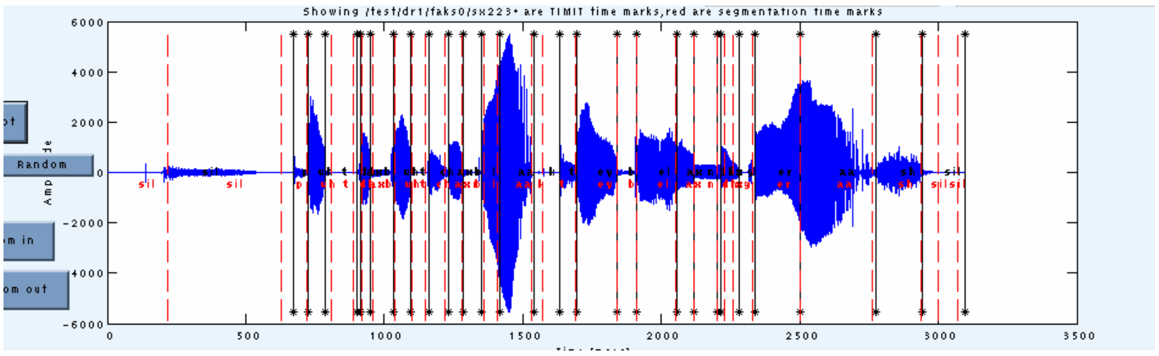

Showing /test/dr1/faks0/sx223* are TIMIT time marks,red are segmentation time marks
Random
m in
om out
Amplitude
6000
4000
2000
0
-2000
-4000
-6000
0
500
1000
1500
2000
2500
3000
3500

Figure 5.2: This is a plot of a transcription

What we see here is waveform along with its associated transcription. Red time marks and red phonemes plotted are from TIMIT labeling whereas the green ones are from the results of our segmentation procedure.

Pressing **Plot** button again generates a new plot of another TIMIT waveform file. Files are processed in the order they appear TIMIT_WavList.txt file. If we want a random file selected we press the **Random** checkbox and a random file from the 5071 files will be plotted now…this action will take a little longer to be completed.

Zoom In/Out Tool

By pressing the Zoom In button you can select a rectangle area with the mouse for zooming in.For ex. look at the following figure

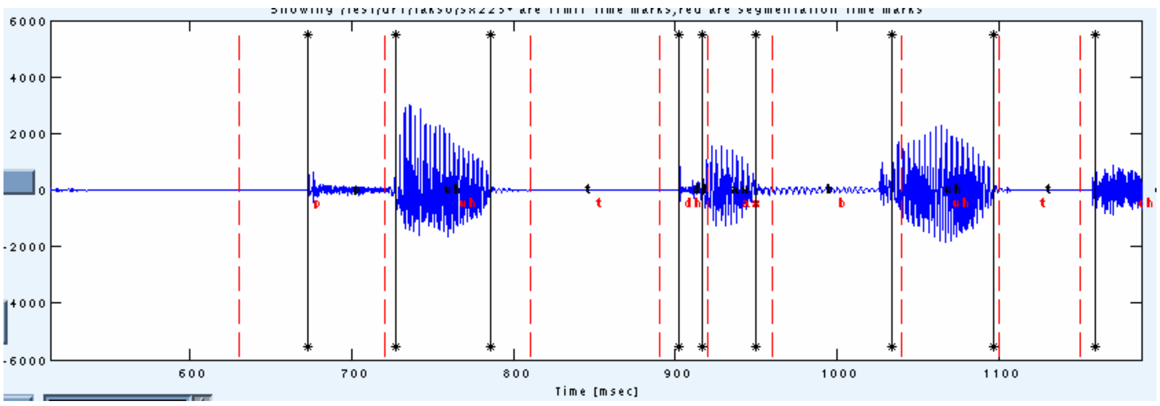

Showing /test/dr1/faks0/sx223* are TIMIT time marks,red are segmentation time marks
6000
4000
2000
0
-2000
-4000
-6000
600
700
800
900
1000
1100
Time [msec]

Figure 5.3: After zoom, a new plotted area

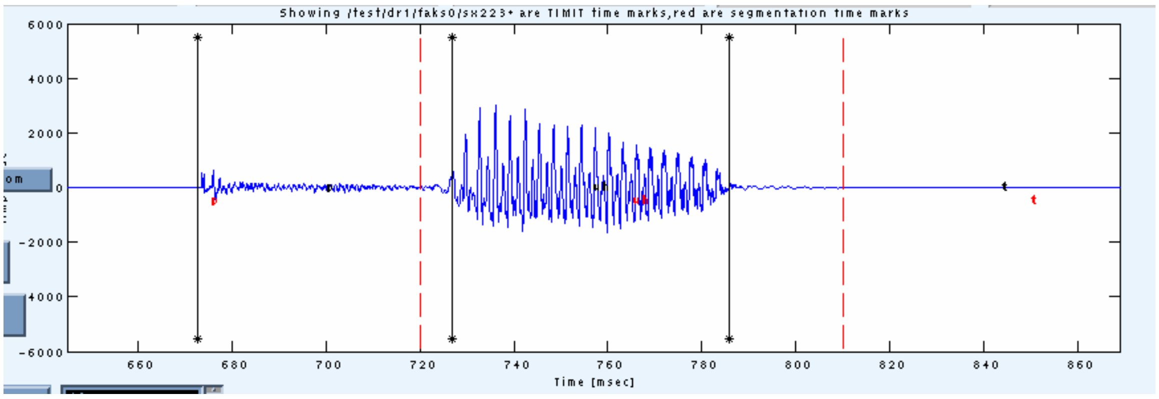


Figure 5.4: After another zoom…

You can zoom in as many times you want, by selecting new rectangles every time in the plot area. Zoom Out button takes you one step back in the plot area.

FFT/LPC/Spectogram/Cepstrum Tool

We can choose to do the Signal Analysis on the whole waveform or on the portion of the waveform selected with the zoom tool. Signal Analysis gives us the Power Spectrum, the LPC spectrum, the FFT Amplitude and the Cepstrum. FFT size is selected to be 1024 and LPC coefficients are selected FS+2. If we select **Spectrogram** from the pull down menu we get the spectrogram plot of the current transcription (if zoom or context are selected we get the plot for the whole transcription data, not the zoomed or context data)

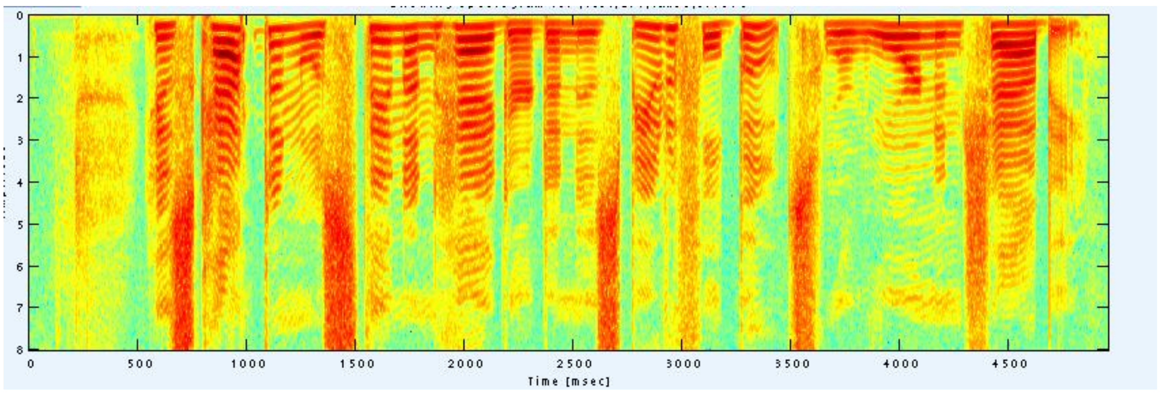


Figure 5.5: Spectogram

If we select **FFT** from the pull down menu we get the FFT results for the data that are shown in the current top figure.

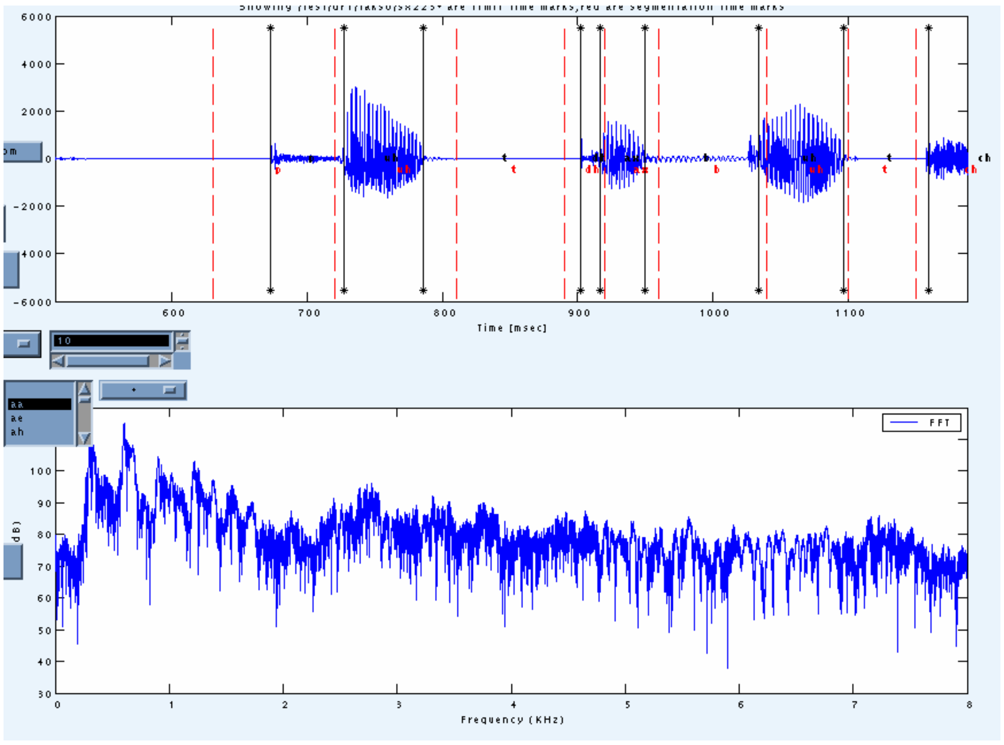

Showing /test/dr1/faks0/sx223* are TIMIT time marks,red are segmentation time marks
Time [msec]
FFT
(dB)
Frequency (KHz)
aa
ae
ah

Figure 5.6: After zoomFFT Analysis on zoomed data

If we select **LPC** from the pull down menu we get the LPC results for the data that are shown in the current upper figure.

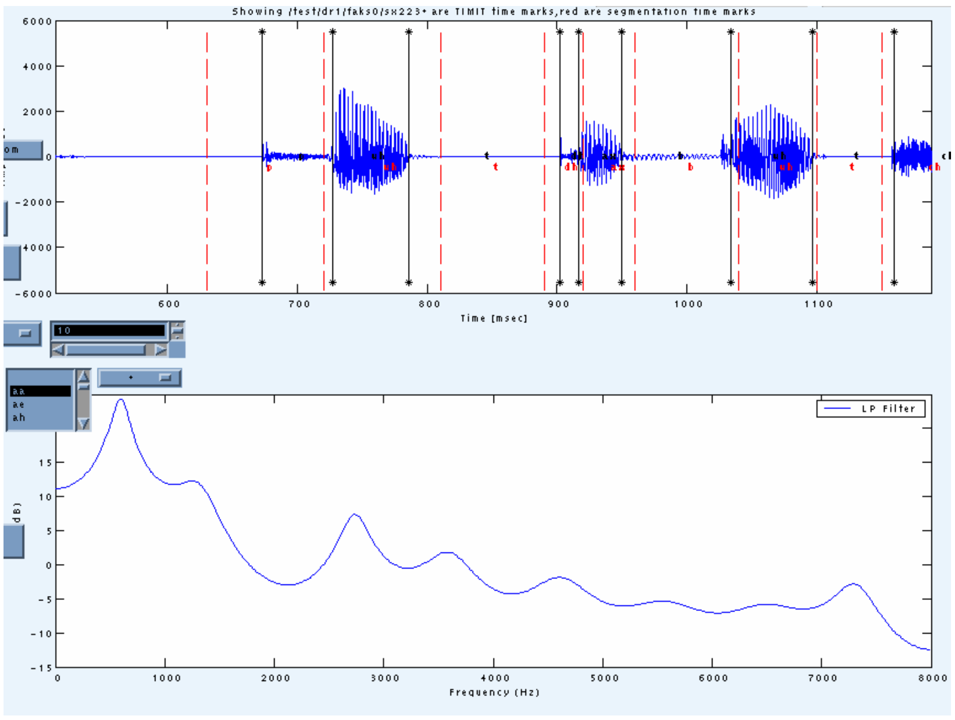

Showing /test/dr1/faks0/sx223* are TIMIT time marks,red are segmentation time marks
Time [msec]
LP Filter
(dB)
Frequency (Hz)
aa
ae
ah

Figure 5.7: After zoom, LPC Analysis on zoomed data

Context

Here we can select a specific phoneme and plot the part of the waveform were the phoneme appears along with its right or left context. We can select the type of the left and the right context by the pull down menus (ex. Affricate, fricative,nasal etc). By pressing the **Plot Context** button we get the resulted waveform. The result that appears on screen is the first one encountered in the MLF file. Pressing Plot Context button again will show us the next example found. If a context appears many times in a label file then the Plot Context action will give us the result within the label file before processing to the next label file in the MLF.

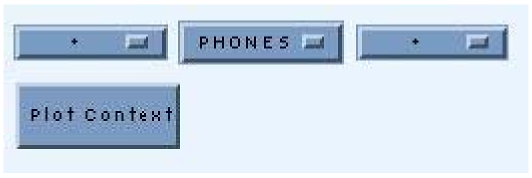


Figure 5.8:Context buttons

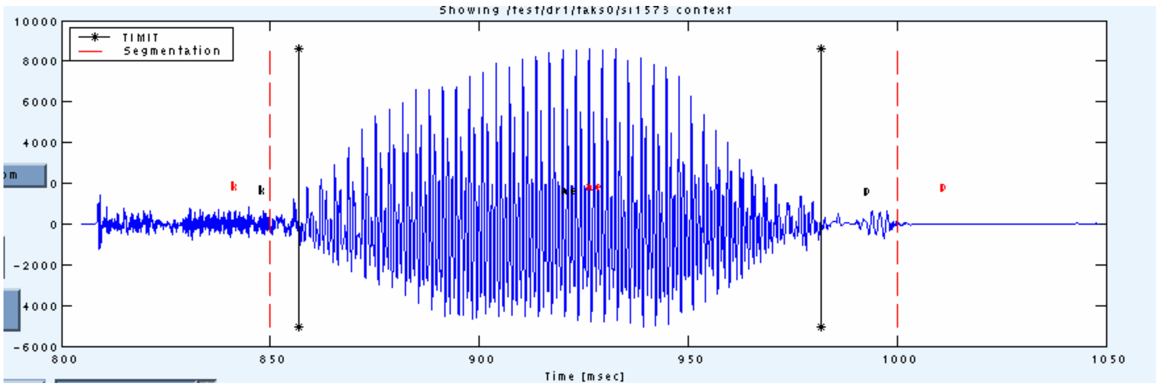


Figure 5.9: Context results of phoneme 'w'

After a result of the specified context is displayed we can use the zoom in tool to get a better look on the time marks for the selected phoneme's context

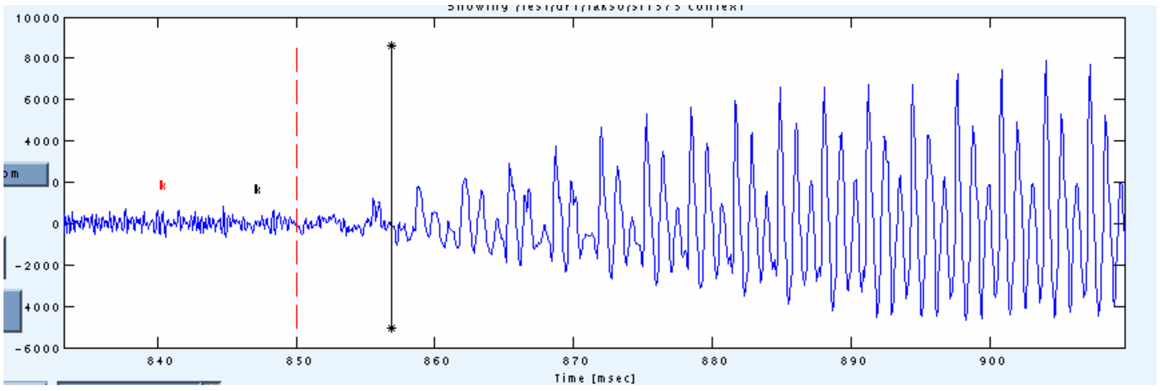


Figure 5.10: A zoom in on the previous figure

From this point we can perform signal analysis by selecting an option from the FFT pull down menu. The following is a LPC analysis on the above zoomed region.

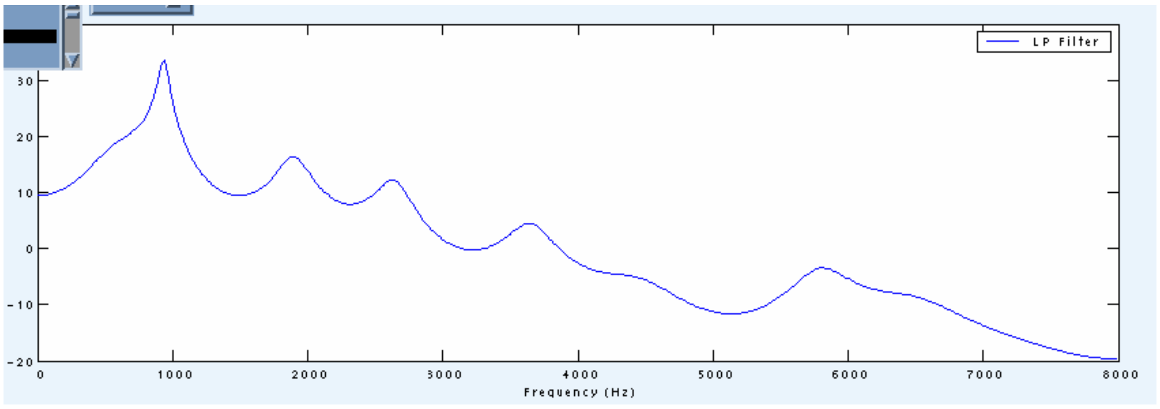


Figure 5.11: LPC analysis on the zoomed context data

Distances

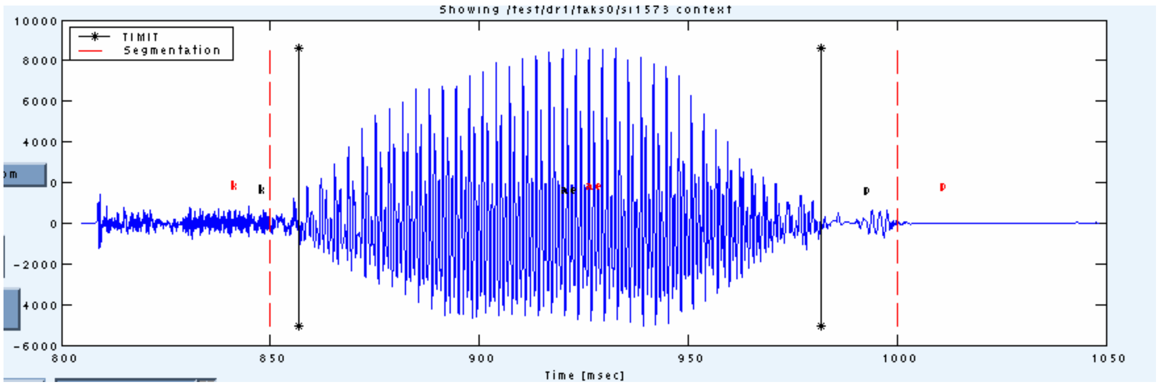


Figure 5.12: First we select a waveform from Context button

Then we Select the desired frame shift (in ms) and press the **Differences** button.

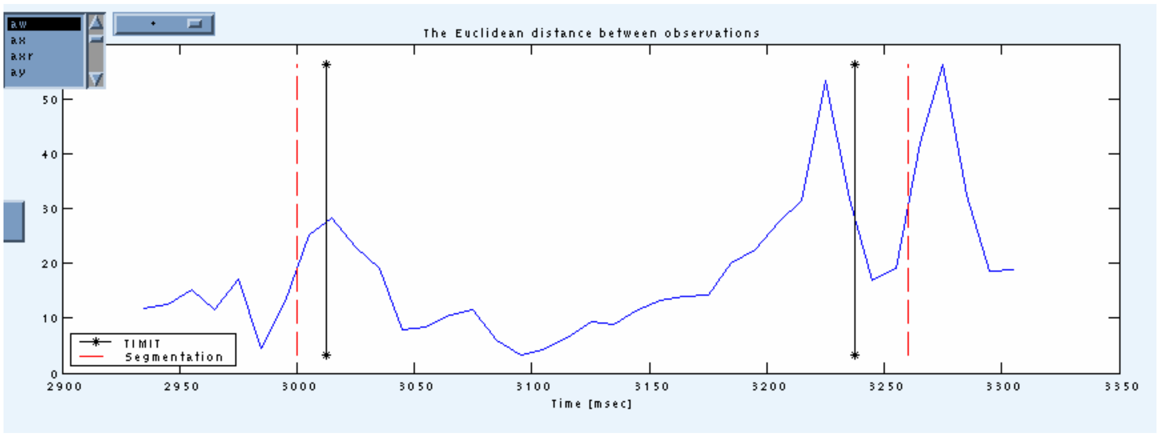


Figure 5.13: Euclidean distance (frame shift is 10ms)

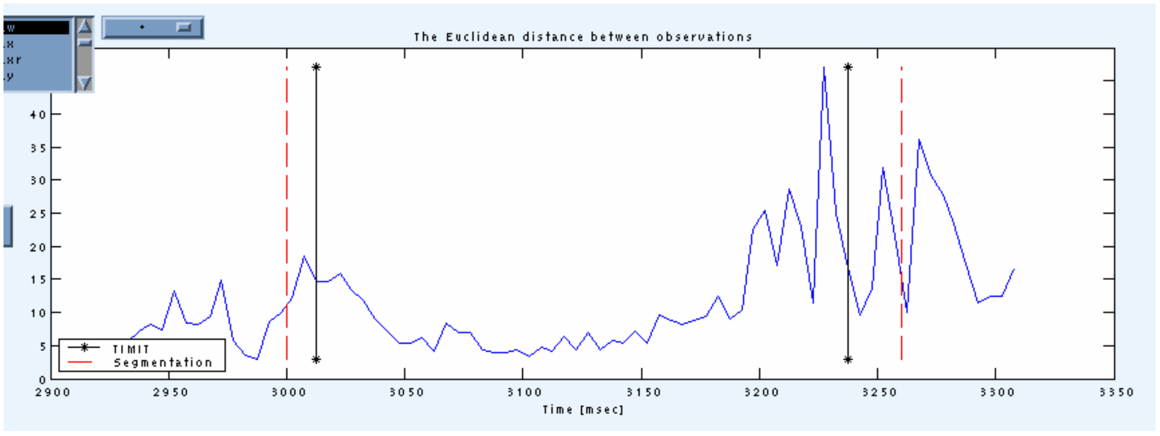


Figure 5.14: Euclidean distance (frame shift is 5ms)

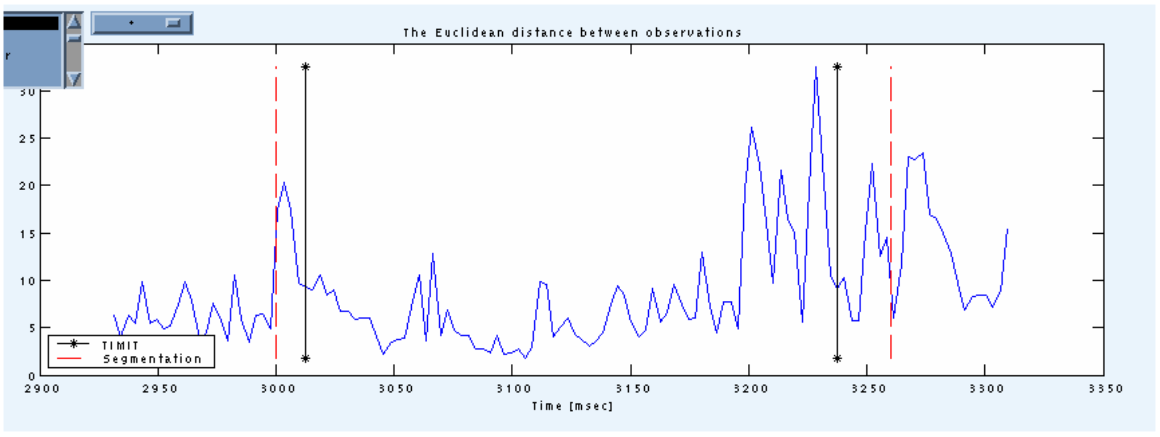


Figure 5.15: Euclidean distance (frame shift is 3ms)

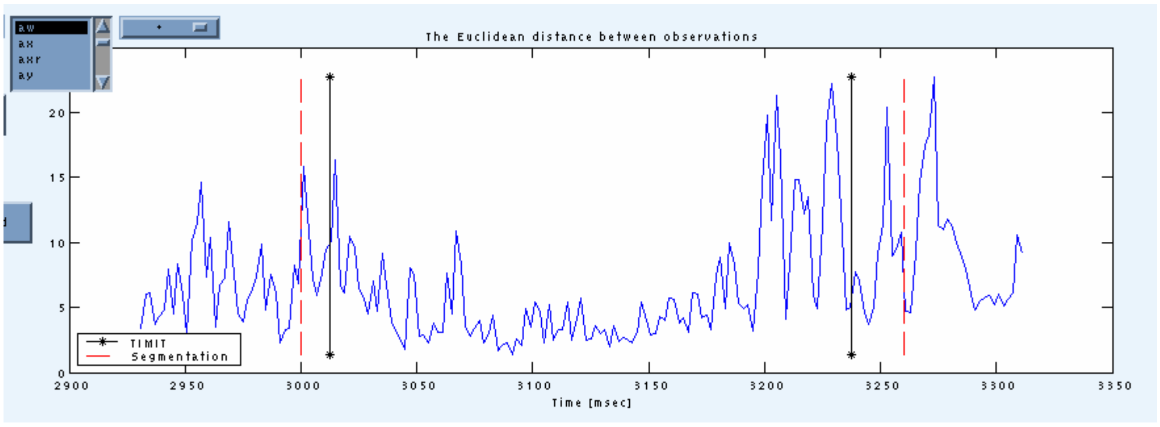


Figure 5.16: Euclidean distance (frame shift is 2ms)

Ceptsrum

With this option the Mel-front end is executed and we get the Mel and Ceptsrum coefficients on screen.

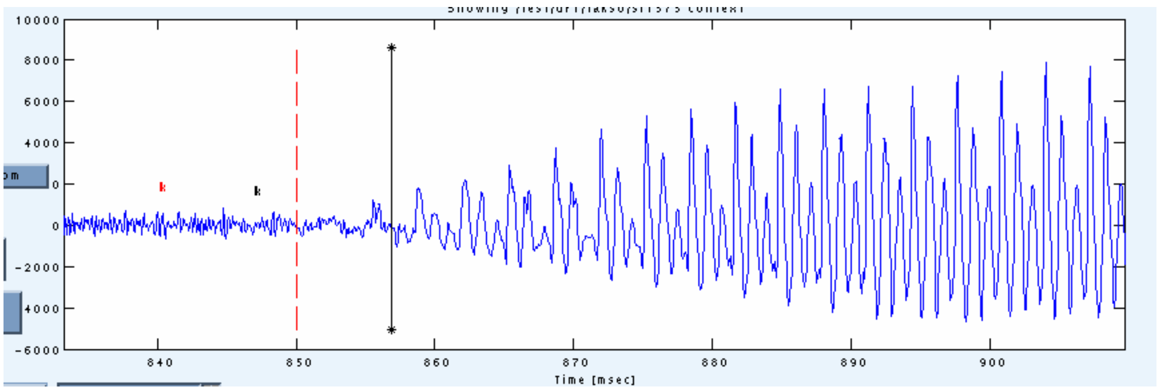


Figure 5.17:A waveform from Context button zoomed one time

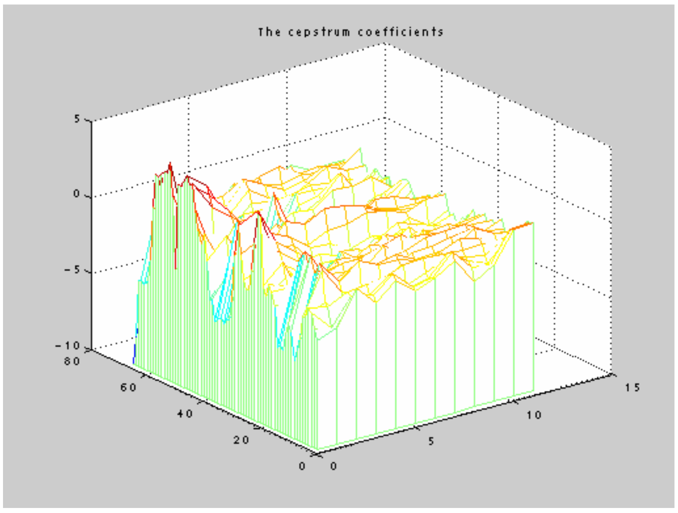


Figure 5.18: The Cepstrum coefficients of the above waveform

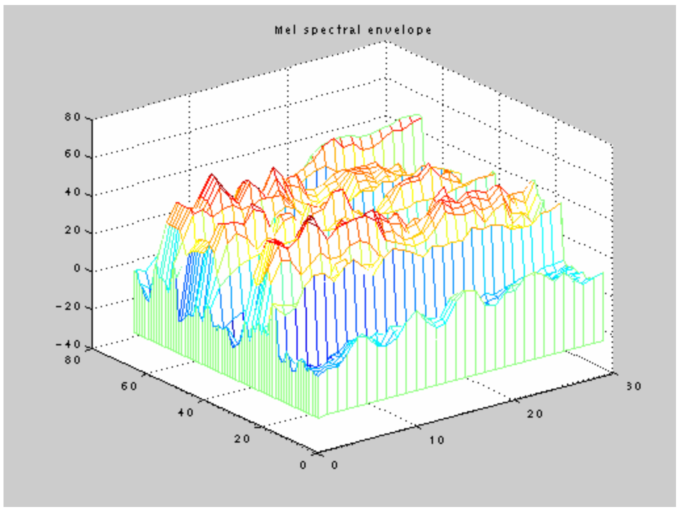


Figure 5.19: Mel Frequency coefficients (MFCC) of the above waveform

Front-End Implementation

Our front end implements the following procedure

A pre-emphasis filter whitens the speech signal and overlapping frames are multiplied by a hamming window. Next the magnitude squared of the DFT is computed. The magnitude squared is processed by a set of mel-filter banks to produce an estimate of the mel spectrum. The mel filter banks (their purpose is to emphasize the lower frequencies) are implemented as a series of overlapping triangle filters that are centered in the mel scale. The result is an estimate of the total energy in the ith critical band. Finally the logarithm of the mel-spectrum is taken to produce a weighted log-energy.

This tool was mainly used to visualize waveforms for the TIMIT corpus along with the associated phoneme boundaries from manual and automatic alignment. The results we got from the ASR technique consist of many different cases and the analysis and evaluation of all those results is rather difficult to be performed just by analyzing mean error or bias values.

With this tool, we can see each automatic boundary position inside each sentence's waveform and this way we can conduct a much more detailed analysis on the boundaries, see specific phoneme placement, see the context segments that are not well aligned according to the mean error analysis results of the previous section.

Also, this tool helped us in the refinement procedure described in the next chapter, were we are trying to shift the boundaries based on specific signal characteristics that best describe the transition between two phonemes. Those signal characteristics are plotted in the tool to help us monitor these transitions, especially by the Euclidean distance between signal features, as we will describe in the next section.

Chapter 6

Refinement Process

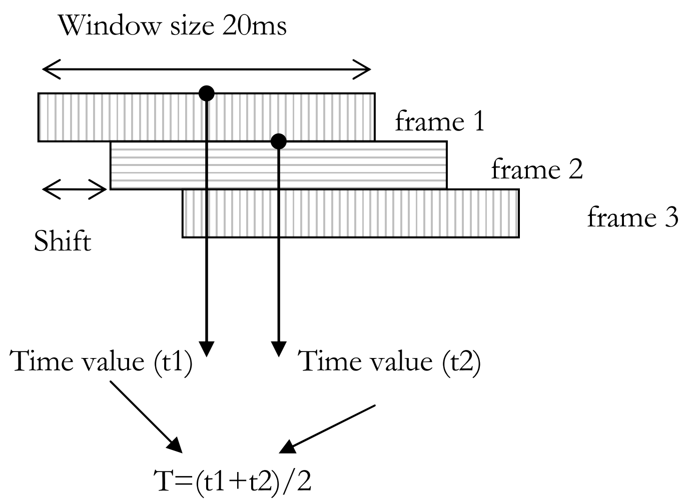
Introduction to the refinement process

The evolution of the speech waveform near a phoneme boundary is determined by the context. The same phone boundary might have different signal characteristics for different contexts. Thus for the boundary between any two phones a and b, a simple a-end b-start kind of detector will not work very accurately. Also the signal features that change sharply near the boundary depend on both the phonemes. For example, an unvoiced fricative-fricative boundary will not be accompanied with a sharp change in voicing nature unlike a boundary from a fricative to a vowel. Thus it seems logical to use separate models for each boundary based on both the phonemes defining the boundary. We model the boundary by taking signal features for equal number of small frames of speech data on both sides of the boundary location.

Since the region near the boundary is rapidly evolving the frame size varies from 10 to 2 ms (we tested our refinement algorithm for different values of frame size and evaluated our results to see which frame size gave us better results). Signal features are then calculated for these frames. We use MFCC's measures to obtain the feature space representation for these frames. The boundary refinement is carried out by using these models to search near the approximate time marks generated by the initial segmentation process for the actual boundary point. For every initial boundary estimate we position candidate boundary points at equal spacing on either side of the boundary and calculate the signal features.

Examples of time mark refinement

The information in each signal has to be represented in some manner. In the front-end we perform a filterbank analysis then our feature vector consists of the energies in each band averaged over 20ms. The feature vector type we use are Mel Frequency Cepstral Coefficients (MFCCs). Since the feature vectors could possibly have multiple elements, a means of calculating the local distance is required. Our measure is local because it involves a computational difference between a feature of one frame and a feature of the other. The distance measure between two feature vectors is calculated using the *Euclidean* distance metric. The process is to take the euclidean distance between two continuous frames i and j. The result is a one row vector with one value per frame pair. Although the Euclidean metric is computationally more expensive than some metrics, it does give more weight to large differences in a single feature. We then plot the resulted vector, after mapping the frame index values it represents to absolute time values. The next figure explains how we get from frame to time. Each window in the front-end is 20ms (320 frames). The value in frame 1 point is 10ms (the middle point) and frame 2 is 10ms plus the value of the shift size.



So the corresponding time value between frame 1 and 2 is calculated by taking the middle of each frame and the sum of the result.

Next we will present some examples of how the refinement process will be carried out. For each phoneme the front-end is executed and the Euclidean distance measures are calculated. We are looking for peaks in this measure that shows us the probable transition from one phoneme to another. In the following figures the discontinued line represents segmentation boundary and the straight line represents the correct boundary.

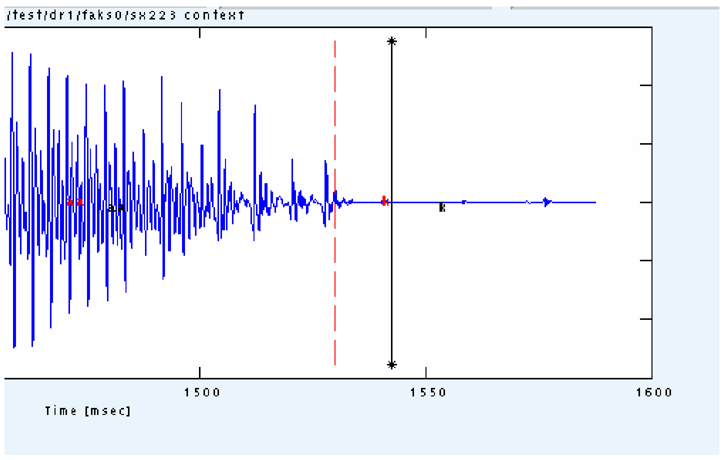


Figure 6.1:A portion of a phoneme (straight line with "*" is the correct boundary)

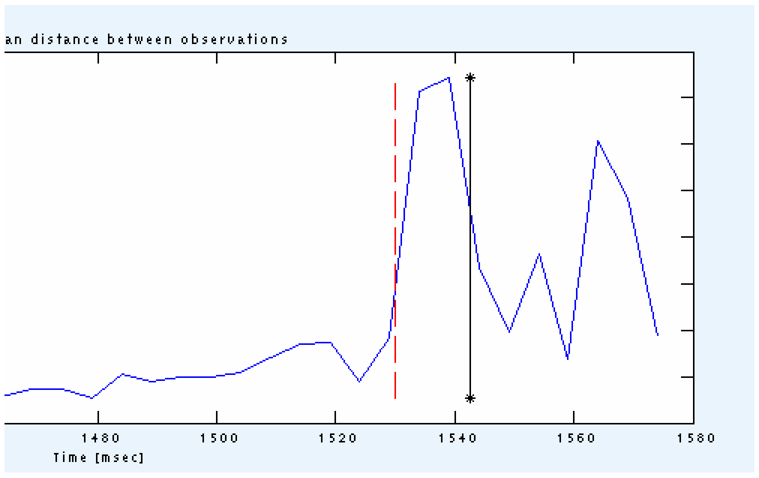


Figure 6.2: Euclidean distance with frame shift 10ms

A peak at around 1530ms indicates that there is a change in the waveform so this most likely a transition to the next phoneme. The discontinued boundary line (initial segmentation) will be moved to 1530ms now, which is closer to the correct boundary as we can see from the figure 6.2.

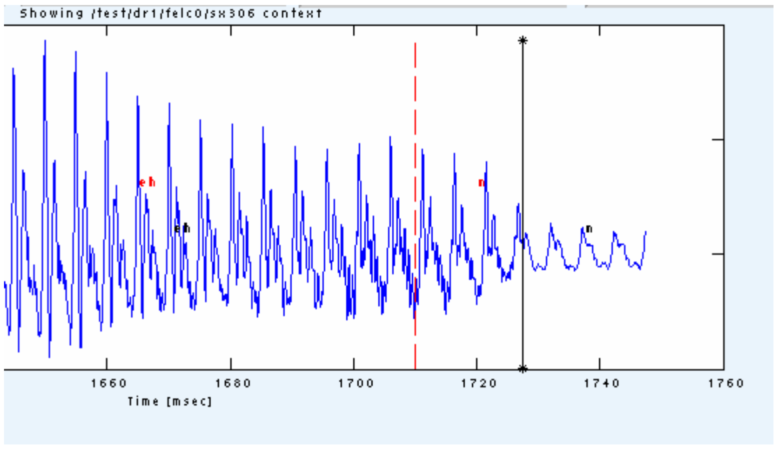


Figure 6.3: Another phoneme boundary placement(straight line with "*" is the correct boundary)

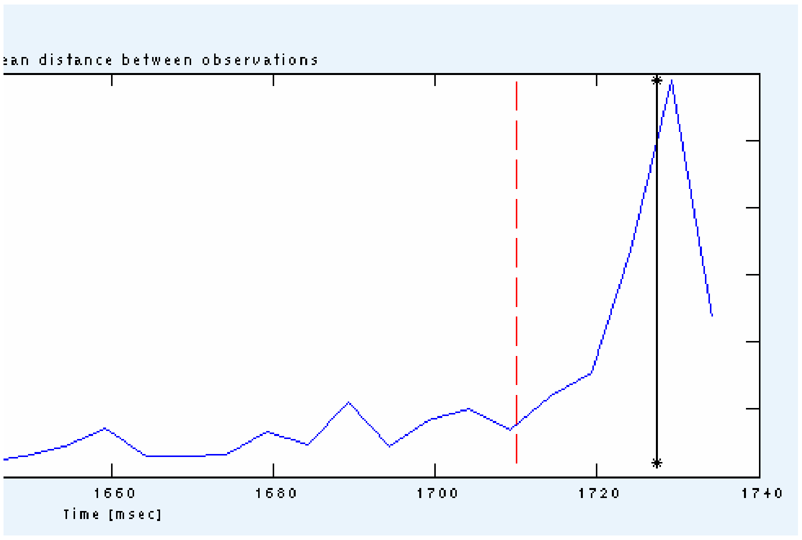


Figure 6.4: Boundary is moved to the peak position (straight line with "*" is the correct boundary)

But not all boundaries can be detected so easy. Bellow we see waveforms and their features that have no specific peak in the phoneme boundary (spurious peaks), so these boundaries can not be refined. Another problem in finding a candidate peak is when the error between the automatic and the hand segmentation is large. Searching near those boundaries may result in a move to a peak that does not represent the actual transition from one phoneme to another or in a peak that involves another phoneme that the one we trying to shift.

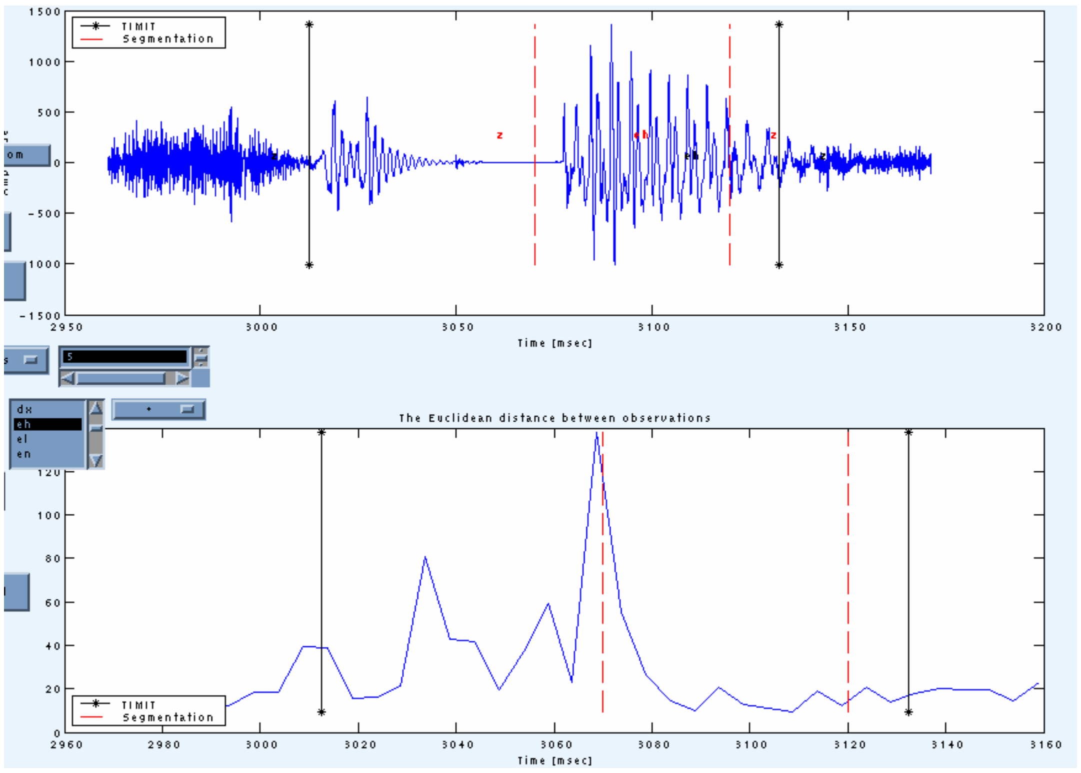


Figure 6.5: A difficult situation

In this figure we can see that for the phoneme ‘z’ there is a high peak near the segmentation time mark (dashed line) so the peak near the correct boundary will be ignored.

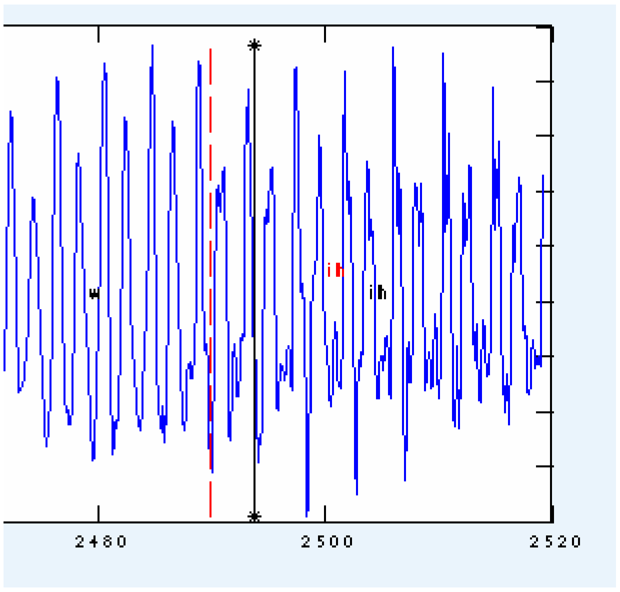


Figure 6.6: A difficult boundary(straight line with “*” is the correct boundary)

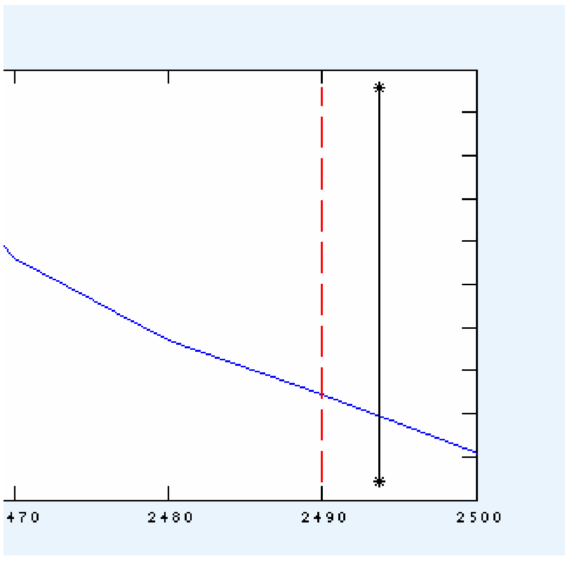


Figure 6.7:There is no actual peak (straight line with "*" is the correct boundary)

With frame shift selected at 10ms the Euclidean distance measure results in non candidate peak as we can see in the above figures. We reduce the frame shift to 5ms and we get the next plot.

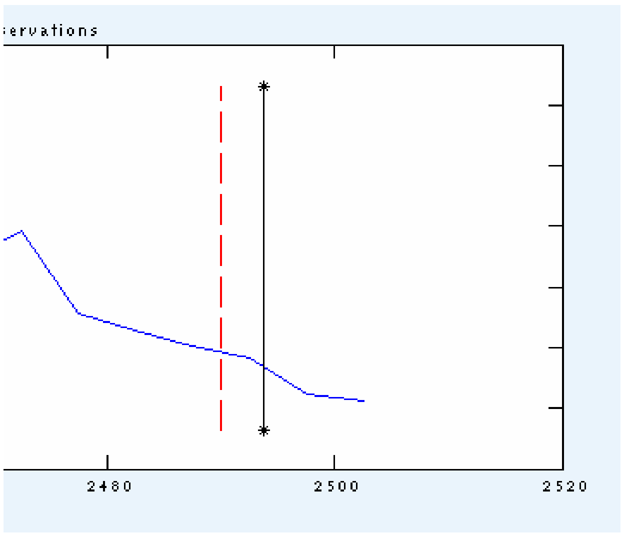


Figure 6.8: A small peak (frame shift is 5ms) (straight line with "*" is the correct boundary)

Here a small peak (nearly showing in the figure) will be calculated in the algorithm and the boundary will be positioned to that peak. But if we change the frame shift to 2ms the peak is now more clearly shown and better placed near the correct boundary.

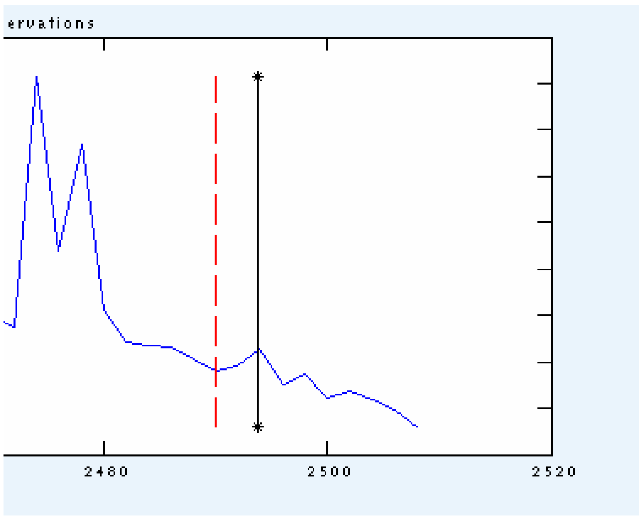


Figure 6.9: The peak is clear now(straight line with "*" is the correct boundary)

Our refinement algorithm is basically dependent on two parameters. First there is Window_Peak, which is chosen 30,20,15 and 10 ms and it is the window in which we search for a peak, selected left and right of each segmentation boundary. Secondly , Frame Shift which is chosen 10,5,3 and 2 ms and it is the frame shift used in the front-end to parameterize each waveform and calculate the MFCC feature vector. Experiments were conducted for each combination of Window_Peak and Frame shift and results are presented in the following pages. By the letter "f" we mean Frame Shift and with the letter "p" we refer to the Window_Peak parameter.

The segmentation algorithm goes like this:

```
Load Timit and Segmentation MLF's
For each phone segment
Segmentation_Time_Mark=End_Time_Mark of the phoneme
        Refine(Window_Peak,Frame_shift)
                Window_Start=(Segmentation_Time_Mark - Window_Peak)
                Window_End=(Segmentation_Time_Mark + Window_Peak)
                ML=Do front-end on selected data
                ED=Euclidean Distance Measures(ML )
                P=Find Peaks(ED)
                For each Peak in P
                        If (Peak>=Window_Start) OR (Peak <=Window_End)
```

```
                                        Segmentation_Time_Mark=Peak
                                   Else
                                        Segmentation_Time_Mark= Segmentation_Time_Mark
                                   End
                         End
                    End
                    Print to refined_mlf resulted time marks
          End
```

In the next table we present the error distributions for each of these methods after refinement. From this table it can be seen that the refined time marks have a much higher percentage of boundaries corresponding to the smaller tolerance limits of 4 and 8 ms. For the 16 ms tolerance limit, there is a significant improvement in performance after time mark refinement, though as can be expected it is not as pronounced.

(ms)	**All examples** %	**After refine (f20,p20)** %	**After refine (f3,p15)** %	**After refine (f2.5,p10)** %	**After refine (f2,p15)** %
0	0.0	4.35	3.22	2.31	3.45
[-4 , 4]	30.12	40.42	33.83	32.45	33.29
[-8 , 8]	53.85	62.35	59.15	55.40	58.62
[-10 , 10]	59.85	68.07	66.56	63.42	66.41
[-15 , 15]	76.00	77.08	78.09	75.69	78.01
[-16 , 16]	78.27	78.13	79.84	77.15	79.64
[-20 , 20]	84.53	83.07	85.47	83.54	85.31
[-25 , 25]	88.96	87.76	90.05	88.42	89.99
[-30 , 30]	91.67	90.94	92.97	91.44	92.92
[-40 , 40]	95.03	94.83	96.12	94.99	96.12
[-50 , 50]	96.95	96.88	97.72	96.96	97.72
[-60, 60]	98.13	98.05	98.63	98.12	98.62

Table 6.1

The basic problem with the Euclidean distance is that it is scaling variant. This means that if the variances of the vector components differ much from each other, the components with large variance dominate the distance value.

Next we compared some of the experiments and created a plot which shows the improvement on the boundary placement after the refinement procedure, for a search window that is 15ms and various values of frame shift in the feature extraction procedure.

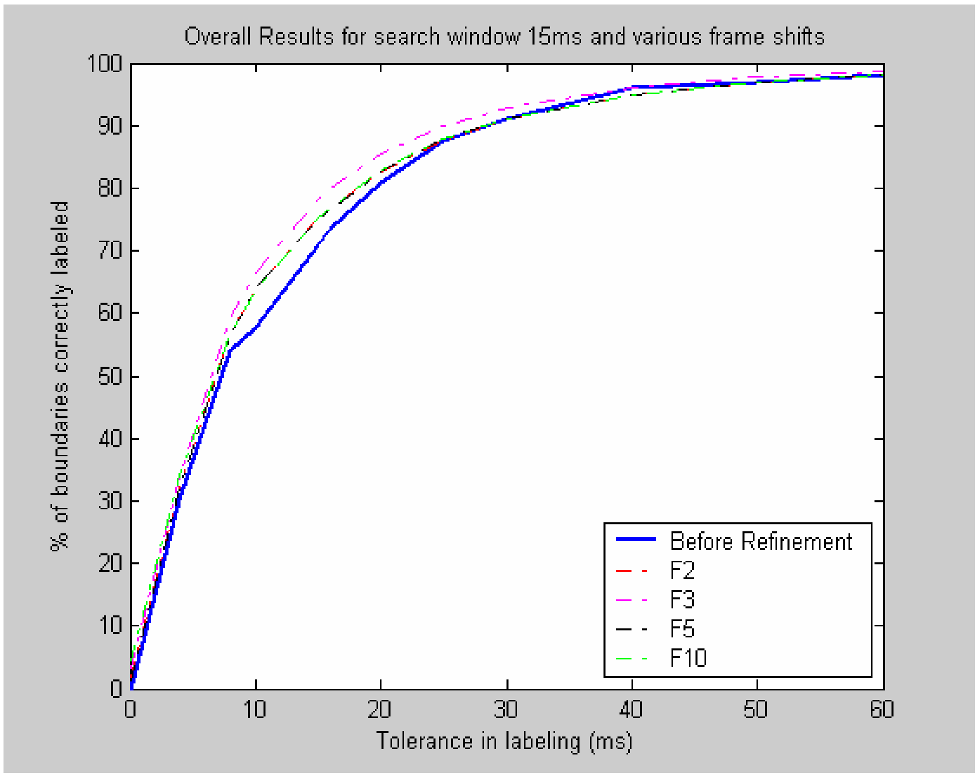


Figure 6.10. The improvement of the refinement process

As we can see from this figure the refinement process is dependent of the value of the frame shift in feature extraction. As this value gets smaller, the algorithm can accomplish better results in refining the boundaries. The best results are when the frame shift is 3ms. The values of 2ms is too small and as we saw from graphic examples results in many spurious peaks near the actually boundary point which are in fact misleading the refinement in many phoneme cases.

Chapter 7

Confidence measures

Introduction

There are many circumstances in speech recognition and speech understanding where there is a need to associate measures of confidence with a utterance. These are objective measures that can be derived from an utterance and which can be used to help determine where a given word, phrase or sentence hypothesis corresponds to an actual occurrence of that event.

Successful measures for speech recognition should adequately predict when recognition fails (i.e., the measure should correlate highly with the actual system performance). Assigning similar confidence scores to time-aligned speech poses several new challenges. First, there exists a group of possible boundary misalignments rather than a binary "correct/incorrect" recognition decision. Second, some errors are more significant than others (e.g., consider misalignments involving stop-to-vowel compared with vowel-to-vowel transitions). Ideally the confidence measure should provide user feedback on the quality of the boundary assigned to each phonetic transition.

Speech Recognition confidence Scores

A speech recognition confidence score should reflect how confident the speech recognizer is in the recognition of an utterance. Confidence scores are often given in the range 0-100 or 0-1.One question here is what this figure really means. Often, the only thing to know for sure when using a commercial speech recognizer is that high scores mean "confident" and low scores mean "not confident".

Confidence Accuracy

The most common use of confidence is to compare it to a threshold. Phonemes with confidence scores below the threshold are rejected, and those with a score higher than the threshold are accepted. Alternatively, one could define a grey zone where the utterance is implicitly or explicitly confirmed (see section 3.1.1).
The threshold used should be tuned according to some empirical data. There is a trade-off that has to be made between the number of *correct rejections* and *correct acceptances*. These terms are explained in Table 7.1 :

	Correctly recognized	**Falsely Recognized**
Above threshold	*Correct acceptance*	*False acceptance*
Below threshold	*False Rejection*	*Correct Rejection*

Table 7.1: Classification of acceptances and rejections

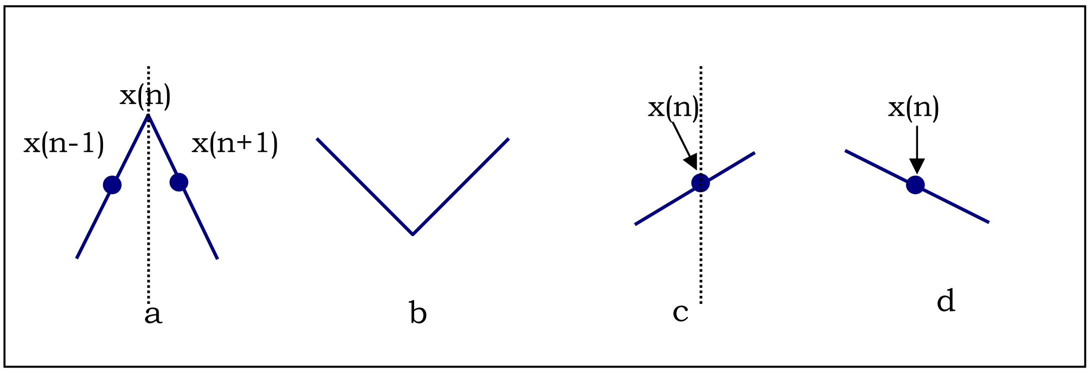
A high threshold will give a large number of rejections, but also a low number of acceptances. A low threshold will instead give a low number of rejections and a high number of acceptances.

The tuning should aim at finding the lowest total number of false acceptances and false rejections. This is often close to the so-called *equal error rate*, where the number of false acceptances equals the number of false rejections. An intuitive measure of confidence accuracy is how low the equal error rate can get. However, this measure is not in line with the assumption that the confidence score should reflect probability of correctness. If the confidence should reflect the probability as perfectly as possible, recognitions with confidences around 50% would be correctly judged in 50% of the cases, and those with a confidence of 0% or 100% would be correctly judged in 100% of the cases. Thus, the equal error rate would be as high as 25%, given the hypothetical case of an even distribution of confidence scores.

How confidence scores are calculated

On the Euclidean distance measures plots, we distinguish the following cases:

A peak (a), a valley (b) and on (c) and (d) we have no peaks just a straight line. Valleys are never an issue (we only look for peaks in our algorithm).



Assume that x(n) is the array with the Euclidean distances. Our confidence measure is calculated by the second derivative in the position of x(n) as shown in the figures above. This position is either the highest peak or just the position of the candidate boundary. Cases like (a) give us a negative confidence value. When cases (c) or (d) are encountered, a score is calculated and it has a positive value. The second derivative at x(n) point is given by the following equation

C_1 = x(n+1) -2 *x(n) + x(n-1) (Simple confidence measure)

We can expand this and get some other ways to calculate confidence.

C_2= (x(n+1)-x(n-1)-x(n)+x(n-2))/2 (Smoothed confidence measure)

For our evaluation purposes, we take the negation of all confidence values calculated (-c = confidence) and all the negative values now are set to zero (this way we are eliminating all c and d cases for confidence scores). So we are now left with only positive values and zeros. The highest the value, the more confident our decision will be.

Confidence measures for the segmentation technique, reflects how confidence the segmentation is on boundary placement. Ideally the confidence measure should provide feedback on the quality of the boundary assigned to each phonetic transition. The only thing to know for sure is that high scores mean the boundary is placed more accurately when comparing it to the manual segmentation and low scores mean that the boundary is “far” from the actual point of segmentation.

Results

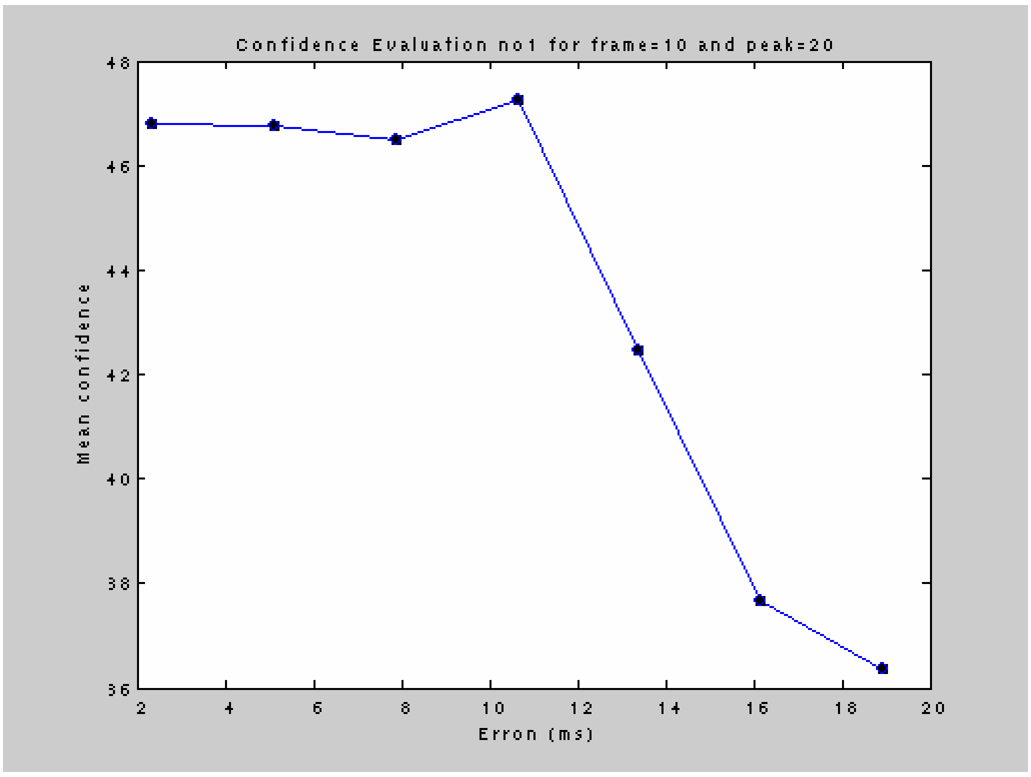
First, there exists a continuum of possible boundary misalignments rather than a binary correct/incorrect decision. Second, some errors are more significant than others (ex. Consider the misalignments involving stop-to-vowel compared with vowel-to-vowel transitions). Ideally the confidence measure should provide feedback on the quality of the boundary assigned to each phonetic transition

We are showing two sets of confidence evaluation plots. The first group of plots shows the mean confidence value calculated in various areas of the error distribution. Errors are divided into areas such as (0,5) U (-5,0) then (-10,-5) U (5,10) etc.

The second group of plots shows the mean absolute error value for the confidence distribution. These plots were constructed by grouping confidence scores into regions of (0-1), (1-2), (2-3) etc. and then calculated abs mean error for those regions.

Figures are present for both simple and smoothed confidence scoring techniques and for the various parameters of the search window and the frame shift of our experiments. Examples that did not succeed to give good results in refinement procedure (in fact the segmentation went worse for some experiments) do not give satisfying confidence evaluation plots. Only a couple of these cases are presented here the rest of cases are in the Appendix.

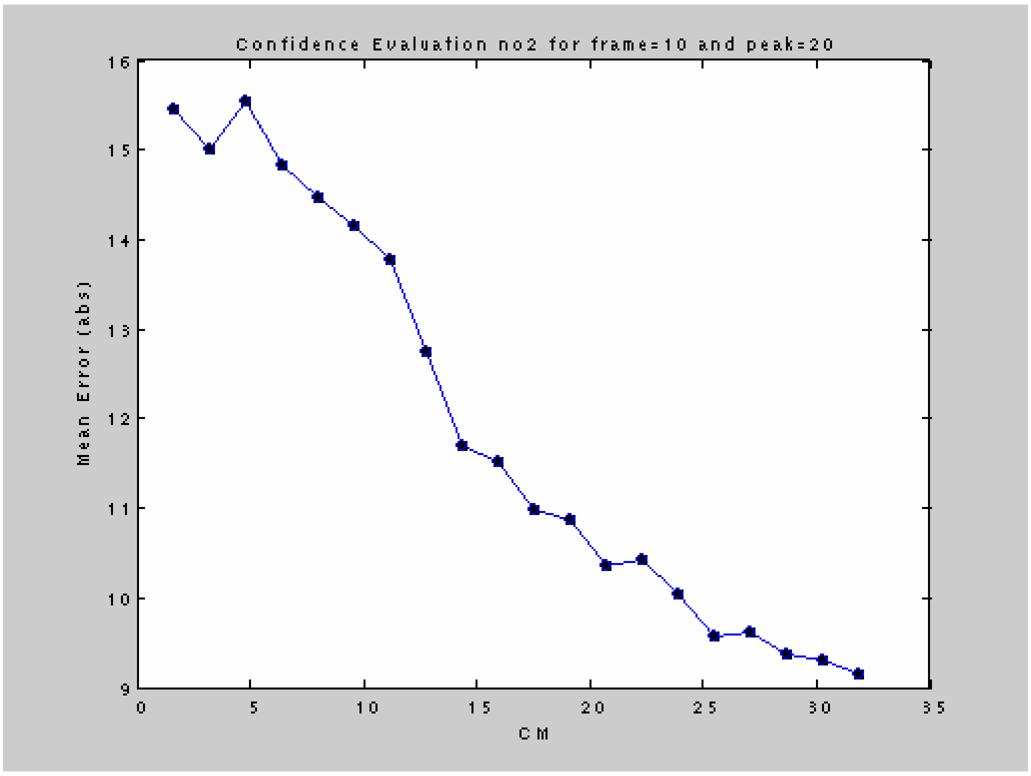
Confidence Measure (a)



Simple confidence plot

This plot all indicates that the confidence value is higher for small errors. When the confidence is evaluated within the search window of the refinement algorithm confidence scores are which meaning that the boundary placement is very close to the manual boundary. However when the all errors that are outside the search window are highly unlikely to be refined so the confidence value is decreased gets when the error is bigger then the search window limits.

Confidence Measure (b)



Simple confidence plot

This plot indicates that mean error, remains high for very low values of confidence and this means that we are not "confident" on the boundary placement for large errors. As the confidence value increases the more "confident" we can say we get and as we can see from the plot the error decreases.

Confidence Evaluation

As we mentioned before on transitions between phones who are acoustical similar or in plosive transitions the segmentation accuracy inevitably drops. In situations like this, it is important to have some ideas of which of the boundaries in a sentence are likely to be correct, which are doubtful and which are almost certainly incorrect.

To evaluate the confidence scores the number of non-detected deviations that exceed 35,70 and 100ms were counted if the lowest confidence scores were to be checked manually. The lowest confidence scores are evaluated by setting a threshold for the confidence measure and all examples that are below this threshold are rejected. In the following figure we computed the histogram for the error distribution, after rejecting all values that had a threshold bellow τ=5.5 for the example of frame=3ms, peak=15ms. We did the same procedure for various thresholds between 1 and 10. The plot in Figure 7.1 shows the error distribution of the remaining (non-rejected) examples. The plot in Figure 7.2 shows in the x-axis the percent of examples rejected due to low confidence and in the y axis the average error of the examples that were not rejected. These results show the amount of reduction in error after rejecting low confidence examples. This is important because it represents a performance improvement that was obtained without any additional information and with minimal additional complexity. The next figures display the results.

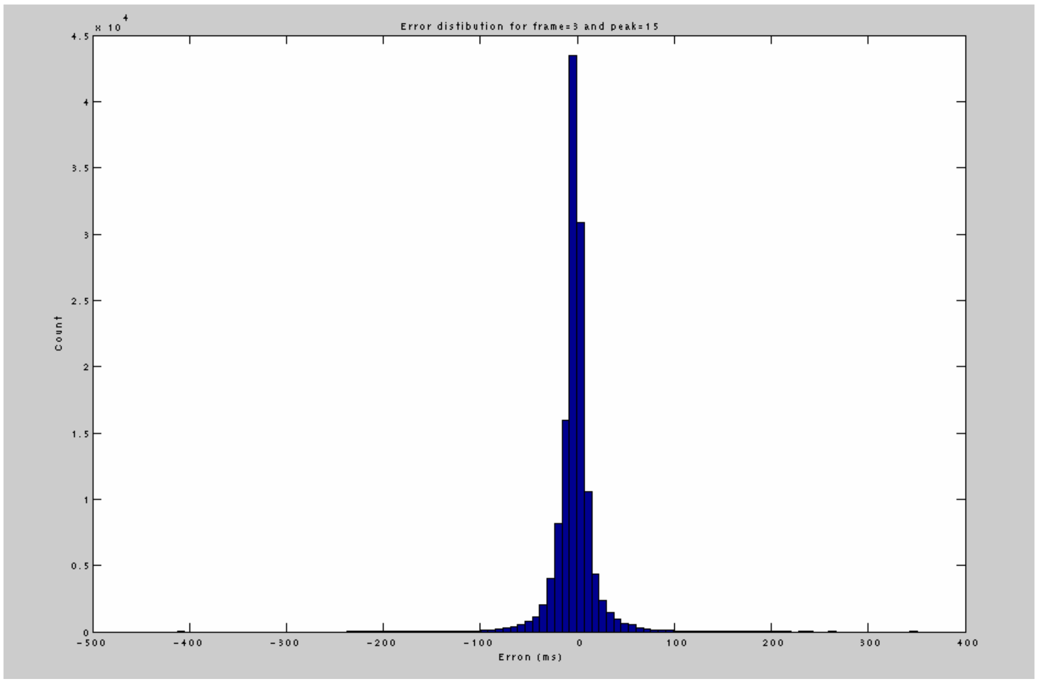


Figure 7.1: Error distribution after rejecting low confidence boundaries(threshold is 5.5)

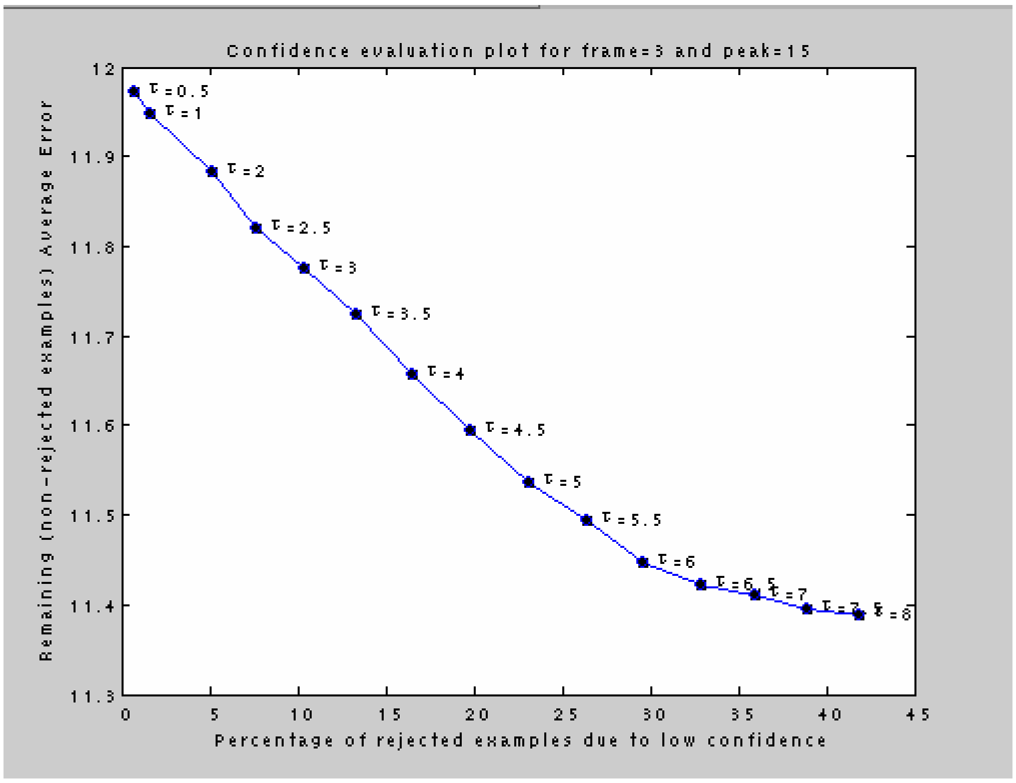


Figure 7.2: Confidence evaluation plot for various thresholds

We then compared the original segmentation with the hypothetical one that will arise after rejecting low confidence scores to see the overall improvement or the boundary placement.

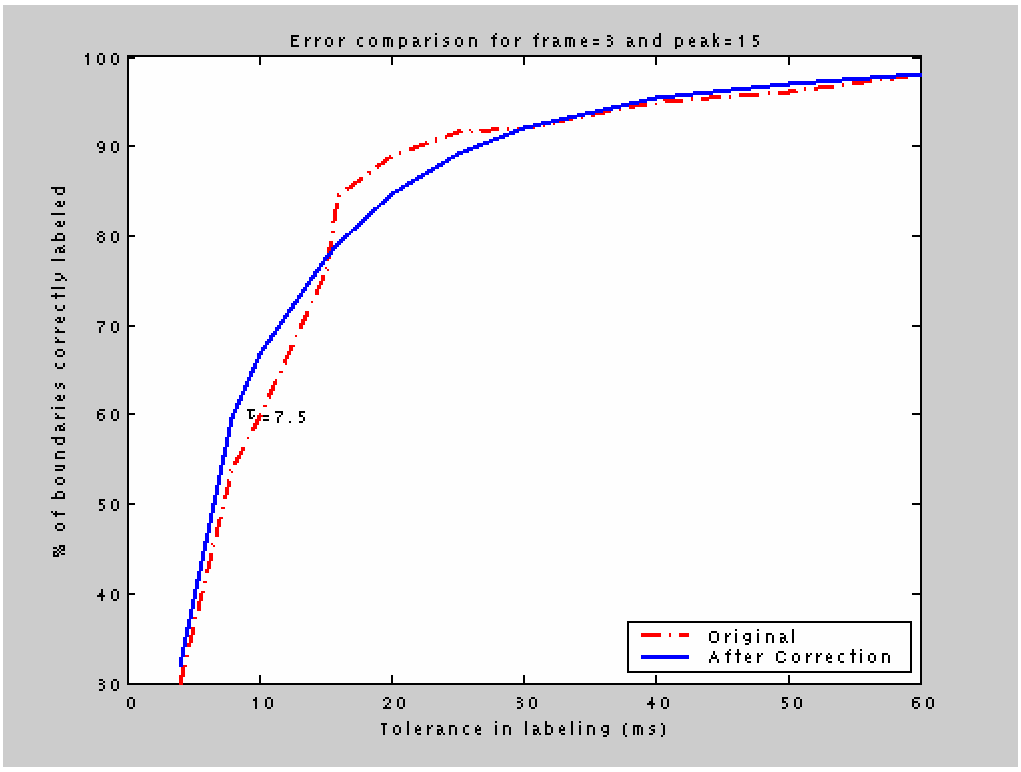


Figure 7.3: Comparison plot for threshold 7.5 between original and after post correction results

Chapter 8

Conclusions and Future Work

Conclusions

The purpose of this thesis was the creation of a specialized algorithm in the HTK recognition toolkit. This algorithm provides the ability to parameterize the process of forced-alignment as well aw tools for the examination and the connection of the results with the correspondent speech signals. We have described a two-stage speech segmentation scheme which uses HMM based boundary models at the initial segmentation was implemented with HTK and by forced alignment. This procedure was not determined to accomplish its purpose by its first implementation so a refinement procedure was conducted later based on Euclidean distance measure.

We created a graphical tool with MATLAB to visualize all data and all calculated features (LPC, Spectogram, FFT, MFCC, Euclidean distance). This tool visualizes human and automatic segmentation boundaries in one plot with the original waveform, which is very useful and really helped us supervise the whole refinement procedure and understand for specific boundary examples the reasons for the correct of incorrect refinement.

For our evaluation purposes we examined bias (context dependent and context free) post-correction techniques. Performance analysis of this scheme indicates that this technique can give encouraging results especially when the boundaries were shifted based on their context-dependent bias value.

We also calculated confidence measures to decide whether a proposed boundary is correct or incorrect. Confidence measures based on the second derivative of a peak near a candidate boundary were computed for many values of frame shift and for many search windows. All of our results were evaluated but our evaluation scheme is rather simple and although it gave some important conclusions a more proper comparative user study should take place based on our results.

We lastly conducted an advanced correction technique based on two confidence measures by simply rejecting the boundaries that were judged to be out of limits, which resulted in a significant improvement on the error rates. However we only examined one case just to show how confidence scores can be used to improve the performance of an automatic segmentation system.

Future Work

In order to define the things that can be done in the future to improve the automatic segmentation performance, one must start by thorough study of our results and the whole approaching scheme to decide in which particular areas a future work should be focused. There are many things that can be improved and many of our results can be further evaluated before even improved. What follows is a list of possible projects to continue this work.

- ***Improve the initial segmentation***

Some future work should involve the task of the initial segmentation to be repeated to get an improved segmentation file because the better the initial segmentation is, it would lead to a substantial improvement on the refinement procedure also.

Initial segmentation can be improved by adapting acoustic models. The idea is to use a small amount of adaptation data to re-estimate only parts of the system. In Chapter 3, we described how the parameters are estimated for plain continuous density HMMs within HTK, primarily using the embedded training tool HERest. This technique can produce high performance speaker independent acoustic models. However it is possible to build improved acoustic models by tailoring a model set to a specific speaker.

By collecting data from a speaker and training a model set on this speaker's data alone, the speaker's characteristics can be modeled more accurately. Such systems are commonly known as speaker dependent systems. The drawback of speaker dependent systems is that a large amount of data (typically hours) must be collected in order to obtain sufficient model accuracy.

HTK supports both supervised adaptation and unsupervised adaptation and the tool to use is HEAdapt using maximum likelihood linear regression (MLLR) and/or maximum a-posteriori (MAP) techniques to estimate a series of transforms or a transformed model set that reduces the mismatch between the current model set and the adaptation data.

Results from the adaptation methods in speech recognition are quite impressive so this alone is a good reason for trying this method to improve the initial segmentation.

- ***Improve the refinement algorithm***

As far as the refinement procedure is concerned, a better algorithm should be used. For example, a way to improve the refinement procedure is to implement a refinement algorithm that will use a Hamming window to "smooth" the peaks of the Euclidean distance measure before shifting the boundary. Remember that there were cases that the peaks near the boundary that did not occur in the same time as the actual boundary point or there we many spurious peaks near the actual boundary point, thus leading the refinement procedure into failure for many of those boundaries.

Each time the algorithm looks for peaks inside a search window. Hamming must have the same size as the search window. We can change the portion of Euclidean distances array inside the search window, leaving everything outside this region unchanged. Although the Euclidean distance has sharp peaks and valleys, the Hamming window has the effect of producing a higher peak that is more closely spaced.

A major drawback on our refinement algorithm is that it is very slow. There is no doubt that Matlab is the best tool for parameterization, but our front-end process combined with the Euclidean distance and the search for the highest peak took a little less than 10 hours to be competed on each of our experiments. Maybe a new approach should be considered, one that significantly fastens the refinement procedure.

- ***Improvements on the Confidence***

Improvements to our confidence scoring techniques are also being investigated. This includes incorporating more features in the confidence evaluation procedure. Accurate localization of a boundary between two phonetic segments depends on the acoustic characteristics of both segments and on what the key contrast between them is. If phonemes are to be partitioned into classes and grouping together classes that share the same acoustic model, it is expected that using different signal representations to build models for different boundary classes (Vowel-Vowel, Vowel-Fricative etc.) should give better results.

This system can be extended to include other speech features (features related to the acoustic characteristics of the input utterance, features related to the context of every labeled unit ,Silence detection, Voicing detection, Zero-crossing rate, the place of articulation detection) that have been developed recently to discriminate between phonemes without adding unnecessary complexity to the system [11]. The information rich acoustic features in general must be extracted. This knowledge is expected to have a profound effect on the automatic speech recognition systems whose performance can significantly improve by integrating more knowledge into their design.

Splines instead of derivatives can also be used to confidence calculation. Splines are a mathematical means of representing a curve, by specifying a series of points at intervals along the curve and defining a function that allows additional points within an interval to be calculated. This way the points x(n-1) and x(n+1) left and right of a peak (as explained in how confidence is calculated) will be positioned the same and the slope will be the same, giving a better accuracy to the confidence score.

- ***How confidence scores can be used***

In our work, we only took the confidence statistics evaluated them and showed one possible way of how they can be used to improve the segmentation. Another way we can use confidence besides rejecting low confidence boundaries, is for example to shift the boundaries based explicitly on each phonemes confidence interval. More precisely, the bias can be estimated as a function of the boundary's confidence score. Large confidence score typically will represent large shifts in the boundary placement.

More tests should be done to the confidence evaluation with various thresholds to get a more complete image of the improvement that can be accomplished in the segmentation performance. Let's not forget that setting a threshold introduces two kinds of errors: False Acceptance Rate (the rate of falsely accepted items) and False Rejection Rate (the rate of falsely rejected items). This parameters need to computed on all our results and for various threshold values.

A proper comparative user study is the only way to determine the effectiveness of the various strategies for incorporating confidence scores in the segmentation process. Unfortunately, we will have to leave this study to future work

- ***Future work on the Graphical tool***

We wanted a framework that could accommodate many kinds of functions for speech and other time-based signals in one shell, with a user interface that would be flexible enough to show only the information relevant to the task at hand. Pre-computed waveform data is cached on disk in order to speed up display, thus providing instant file load and quick access anywhere in even the largest of sound files. Our tool can be further expanded so that it can become a standard tool in the

Telecommunications laboratory for viewing and editing sound files and transcriptions.

A simple task for the tool is to add more functionality and flexibility in viewing all transcriptions (context-dependent and non-dependent) by adding a back/previous button to move to the previous file/context transcription and a "file open" menu to allow the user to open specific waveform from TIMIT. The tool now simply loads information for all transcriptions and the user can move in a "serial" order between them.

One the other hand, in order to determine the effectiveness of confidence scoring the user should be able to hand-correct boundaries based on confidence scores. The main idea behind this intuition is to present to the user of the tool the segmentation result transcriptions, starting from ones that have the lowest confidence score.

BIBLIOGRAPHY

[1] WIGTHMAN ,C.,W., AND TALKIN , D. T. The Aligner : Text-to-speech Alignment using Markov Models. In Progress in Speech Synthesis, J.P.H.V. Santen, R. W. Sproat, J. Olive, and J. Hirschberg, eds. Spinger-Verlag, New York 1997.

[2] Odell J, Ollason D, Woodland P, Young S , Jansen J, "The HTK Book for HTK V2.0", Cambridge University Press, Cambridge, UK, 1995.

[3] Pellom B. L., and Hansen J. H. L. , Automatic Segmentation and labelling of speech recorded in unknown noisy channel environments. In proceedings of the 1997 ESCA-NATO Workshop in Robust Speech Recognition for unknown Communication Channels (1997).

[4] A. Ljolje and M. D. Riley. Automatic segmentation of speech for TTS. In Proc. of Eurospeech-93, volume 2, pages 1445-1448(Berlin, 1993.)

[5] Jan Ph.H van Santen, Richard W. Sproat, "High Accuracy Automatic Segmentation", *Proc. EUROSPEECH 99.*

[6] A. Ljolje, J. Hirschberg and J.P.H van Santen, "Automatic Speech Segmentation for Concatenative Inventory Selection",*Progress in Speech Synthesis,* Springer 1997, pp 305-311.

[7] Abhinav Sethy, Shrikanth Narayanan, "Refined Speech Segmentation for Concatenative Speech Synthesis". ICSLP 2002

[8] L. R. Rabiner A tutorial in Hidden Markov Models and selected applications in Speech Recognition, Proceedings of the IEEE vol. 77 pg 257-286, February 1989

[9] L. R. Rabiner Fundamentals of Speech Recognition, Prentice Hall 1993

[10] S. Cox and R. Rose, "Confidence measures for the switchboard database," in *Proc. ICASSP*, 1996.

[11] A. M. Abdelatty, J. Van der Spiegel, Gavin Haentjens, J. Berman and P. Mueller, "An Acoustic-Phonetic Feature-based System for Automatic Phoneme Recognition in Continuous Speech", *IEEE ISCAS,* May 1999, Proc. Vol. III, pp. 118-121

[12] Matthew J Makashay, ColinW.Wightman, Ann K. Syrdal andAlistair Conkie, "Perceptual evaluation of automatic segmentation in Text-To-Speech Synthesis", *Proc. ICSLP 2000.*

[13] R. O. Duda and P. E Hart, Pattern classification and scene analysis, John Wiley and Sons, 1973.

[14] T. Schaaf and T. Kemp, ``Confidence measures for spontaneous speech recognition'', in *Proceedings of 1997 ICASSP*, Munich, April 1997, vol. II, pp. 875--878.

[15] L. Gillick, Y. Ito, and J. Young, ``A probabilistic approach to confidence estimation and evaluation'', in *Proceedings of 1997 ICASSP*, Munich, April 1997, vol. II, pp. 879--882.

[16] F.Wessel, K. Macherey, and R.Schluter, "Using word probabilities as confidence measures," in International Conference of Acoustics, Speech, and Signal Processing, Seattle, WA,May 1998, pp. 225–228.

[17] T. Schaaf and T. Kemp, "Confidence measures for spontaneous speech recognition", Proc. ICASSP, pp.875-878 (1997).

[18] Time-Frequency Features for Continuous Speech Recognition, a General Examination James G. Droppo, January 5, 1998

[19] Hosom, J.-P. (2000) Automatic Time Alignment of Phonemes Using Acoustic-Phonetic Information. PhD Thesis, Oregon Graduate Institute of Science and Technology.

[20] Sjolander, K. & Beskow, J. (2000) WaveSurfer - an open source speech tool. Proceedings of the ICSLP 2000, IV, 464-467.

APPENDIX

HTK tools we used

HLEd -G $DataBase -n $phonSet $tracing -i $phones $HLEdScript -S $Phonelist";

Parameters in the above command:
-G : inform HLEd that format of the source files is TIMIT
-n $phonSet : HLEd can be made to automatically generate a list of all new label names (HMM list) as a by-product of editing the label files by using the -n option.
-i mlf This specifies that the output transcriptions are written to the master label file mlf.
$Phonelist are …
/usr/share/SpeechDatabases/TIMIT/timit/test/dr1/faks0/si1573.phn
/usr/share/SpeechDatabases/TIMIT/timit/test/dr1/faks0/si2203.phn

foreach(<PHONES>){
HInit -m 1 $tracing -S $hlist_train -C $config -i $MaxIter -M $Fold2Store -l $_ -o $_ -I $phones $proto";}

Parameters:
-m N This sets the minimum number of training examples so that if fewer than N examples are supplied an error is reported (default value 3).
-S $hlist this loads the script file with the list of train files or test files from TIMIT.
-i N This sets the maximum number of estimation cycles to N
-M dir Store output HMM macro model files in the directory dir.
-I mlf This loads the master label file mlf.
-l s The string s must be the name of a segment label. When this option is used, HInit searches through all of the training files and cuts out all segments with the given label.
-o s The string s is used as the name of the output HMM in place of the source name. This is provided in HInit since it is often used to initialise a model from a prototype input definition.

HHEd -d $Fold2Store -w $hmmdefs $dummy $phonSet";

Parameters :

-d dir This option tells HHEd to look in the directory dir to find the model definitions

-w mmf Save all the macros and model definitions in a single master macro file mmf

$dummy is a text file containing a sequence of edit commands (here this is empty)

$phonSet defines the set of HMMs to be edited (phone list from TIMIT)

HRest $tracing -i $MaxIter -C $config -M $Fold2Store -l $_ -I $phones -H $StoredModels/$_ $_ -S $hlist ";

Parameters :
-$MaxIter : maximum number of iterations for HRest tool
$Fold2Store : where to store the new models

$phones :this is the phone list

$StoredModels : where to find the old models
-S $hlist this loads the script file with the list of train files or test files from TIMIT.

Results from the ASR Segmentation

A) OVERALL CONTEXT STATISTICS

BIAS(ms)	AF	FR	NAS	SEM/V	SIL	STOP	VOW	*
AF	-3.00	-8.24	-3.86	-2.22	1.98	0.30	5.68	-1.34
FR	-27.31	-2.05	-2.66	-4.20	-1.65	-0.84	2.04	-5.24
NAS	-4.38	-3.39	-2.23	-3.42	8.60	1.50	4.85	0.22
SEM/V	1.01	-1.43	-1.10	-4.33	3.82	1.72	6.09	0.83
SIL	-12.90	-12.07	-7.72	-10.39	0.0	-30.48	5.71	-9.70
STOP	-0.06	-4.15	-8.66	-7.79	-21.41	-31.52	-1.64	-10.75
VOW	10.70	-3.80	-3.72	-3.15	7.94	-1.72	6.28	1.79
*****	-5.13	-5.02	-4.28	-5.07	-0.11	-8.72	4.14	-3.46

Context-context results for Bias (8 mixtures)

MEAN(ms)	AF	FR	NAS	SEM/V	SIL	STOP	VOW	*
AF	3.00	17.61	7.68	8.79	12.73	7.05	8.51	9.34
FR	27.31	13.03	8.83	8.73	11.64	8.11	7.43	12.16
NAS	8.67	9.40	15.90	14.69	21.16	10.60	8.81	12.75
SEM/V	7.01	7.94	9.13	14.07	16.59	7.11	14.54	10.91
SIL	13.73	15.87	9.80	13.63	0.0	32.46	11.22	13.82
STOP	6.38	10.78	12.78	10.46	29.03	34.16	7.70	15.90
VOW	16.07	7.74	9.18	16.21	19.48	7.53	16.82	13.29
*****	11.74	11.77	10.47	12.37	15.81	15.29	10.72	12.60

Context-context results for mean error (8 mixtures)

STD(ms)	AF	FR	NAS	SEM/V	SIL	STOP	VOW	*
AF	0.00	19.54	5.97	7.24	12.49	6.98	8.27	8.64
FR	22.57	12.80	7.77	7.24	14.42	9.34	8.92	11.87
NAS	5.55	8.67	16.21	16.14	21.10	11.09	11.47	12.89
SEM/V	2.16	6.87	10.03	13.79	20.72	6.64	14.56	10.68
SIL	7.90	13.14	10.78	13.62	0.0	20.24	13.09	11.60
STOP	5.65	10.66	13.02	8.77	29.97	28.74	8.54	15.05
VOW	26.03	6.49	10.72	16.08	19.86	7.51	19.03	15.10
*****	9.98	11.17	10.64	11.84	17.28	12.93	11.98	12.26

Context-context results for Standard Deviation (abs)(8 mixtures)

BIAS(ms)	AF	FR	NAS	SEM/V	SIL	STOP	VOW	*
AF	-3	-7.19	-2.82	-2.97	5.92	0.47	5.64	-0.57
FR	-23.98	-3.59	-2.90	-5.01	0.69	-1.75	2.35	-4.88
NAS	-4.92	-3.14	-2.62	-2.51	9.12	1.88	5.20	0.43
SEM/V	3.01	-0.95	-0.52	-4.70	7.38	1.50	5.90	1.66
SIL	-12.48	-12.09	-7.51	-11.40	0.0	-29.82	5.06	-9.76
STOP	0.39	-2.5	-7.17	-6.75	-19.52	-27.75	-1.20	-9.22
VOW	10.22	-3.82	-3.39	-3.15	8.60	-2.18	6.24	1.79
*	10.19	10.77	10.13	11.31	16.95	12.65	11.51	-2.93

Context-context results for Bias (16 mixtures)

MEAN(ms)	AF	FR	NAS	SEM/V	SIL	STOP	VOW	*
AF	3.00	15.03	6.28	8.38	13.31	6.83	8.28	8.73
FR	23.98	12.13	8.75	9.30	11.75	7.85	7.62	11.62
NAS	7.08	9.03	15.10	13.47	20.26	10.31	8.64	11.98
SEM/V	6.01	7.85	8.27	13.06	18.55	6.84	13.72	10.62
SIL	13.60	15.49	9.35	14.03	0.0	31.50	11.10	13.59
STOP	6.06	10.37	11.33	9.58	27.96	30.40	7.37	14.73
VOW	16.17	7.73	8.78	15.01	19.11	7.55	15.87	12.89
*	10.19	10.77	10.13	11.31	16.95	12.65	11.51	12.02

Context-context results for mean error (16 mixtures)

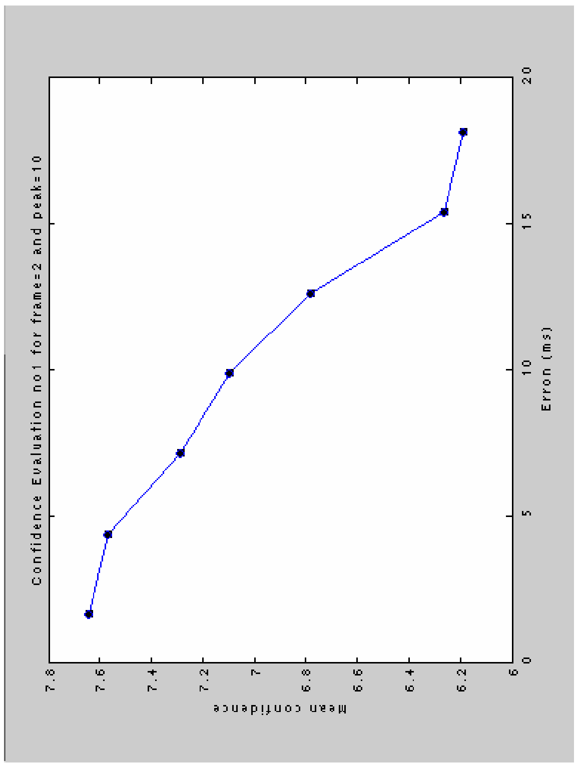
STD(ms)	AF	FR	NAS	SEM/V	SIL	STOP	VOW	*
AF	0.00	17.91	5.21	6.67	13.01	6.75	7.82	8.20
FR	24.09	12.39	8.64	7.66	13.37	8.22	8.78	11.88
NAS	5.18	8.41	14.70	14.84	20.46	10.63	11.37	12.23
SEM/V	2.77	6.72	9.58	12.86	21.09	6.33	13.60	10.42
SIL	8.36	13.10	10.58	13.92	0.0	21.79	12.93	12.05
STOP	5.20	10.39	12.18	8.27	28.66	27.48	8.18	14.34
VOW	25.71	6.43	10.04	14.93	18.34	7.36	17.89	14.39
*	10.19	10.77	10.13	11.31	16.95	12.65	11.51	11.93

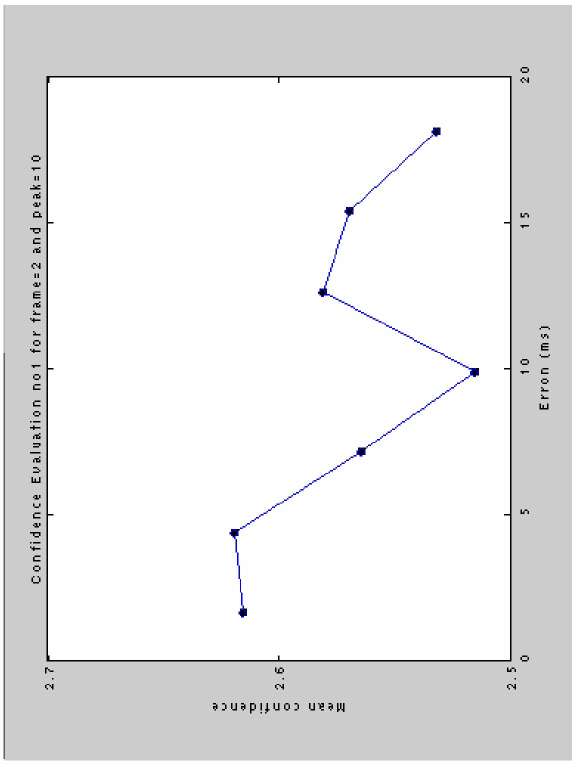
Context-context results for Standard Deviation (abs)(16 mixtures)

B) STATISTICS FOR EACH PHONEME

In the next pages we present the comparison results between the manual and the automatic segmentation transcriptions.

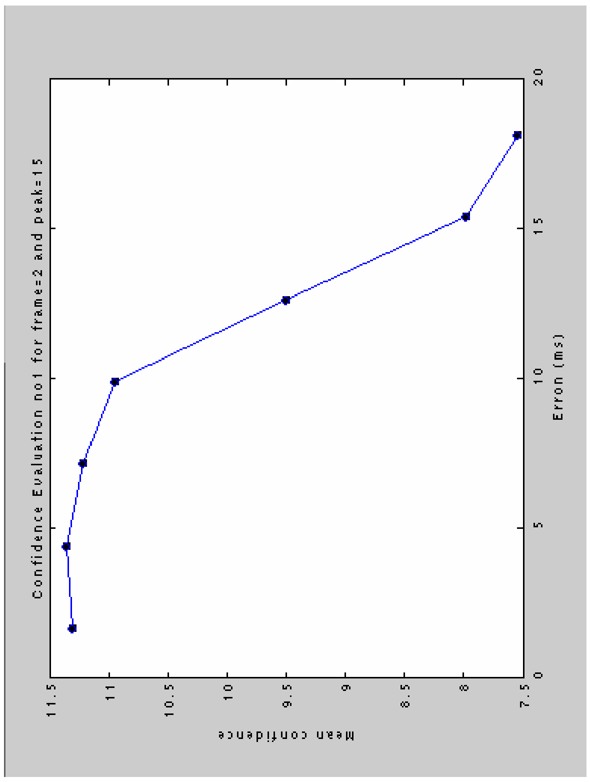
Confidence Measure (first group)

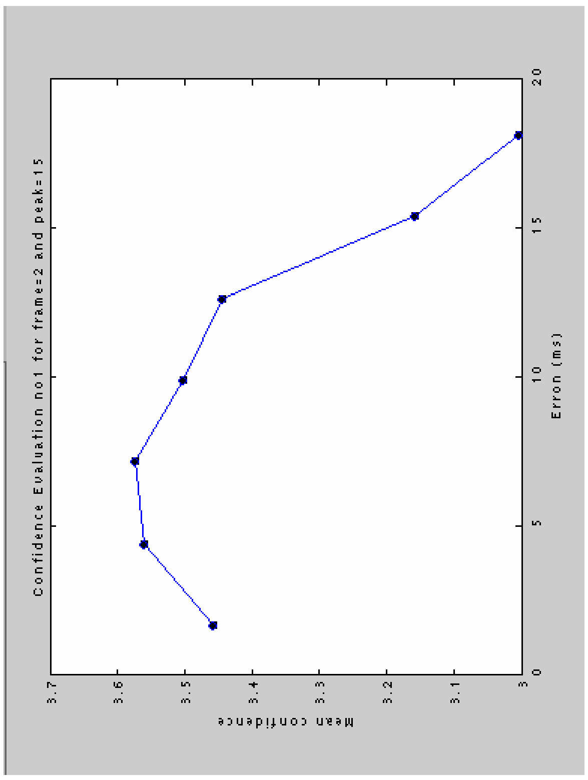


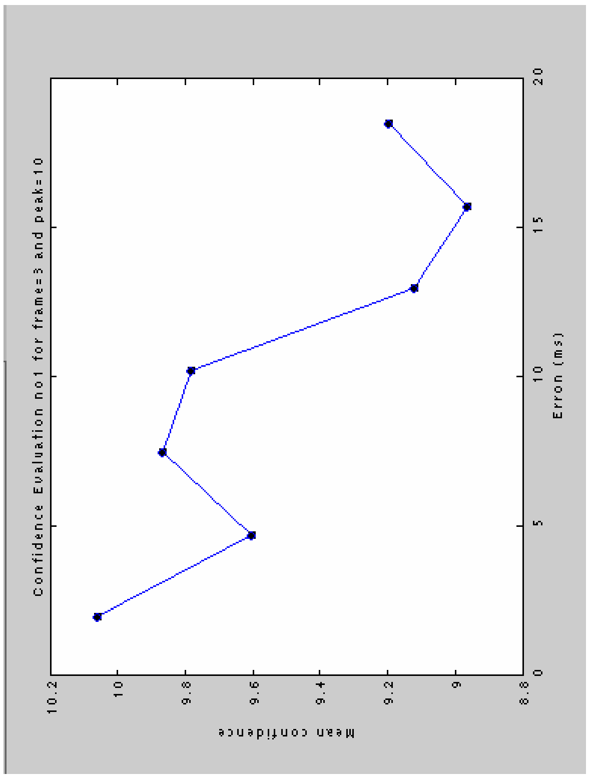


Simple confidence plot

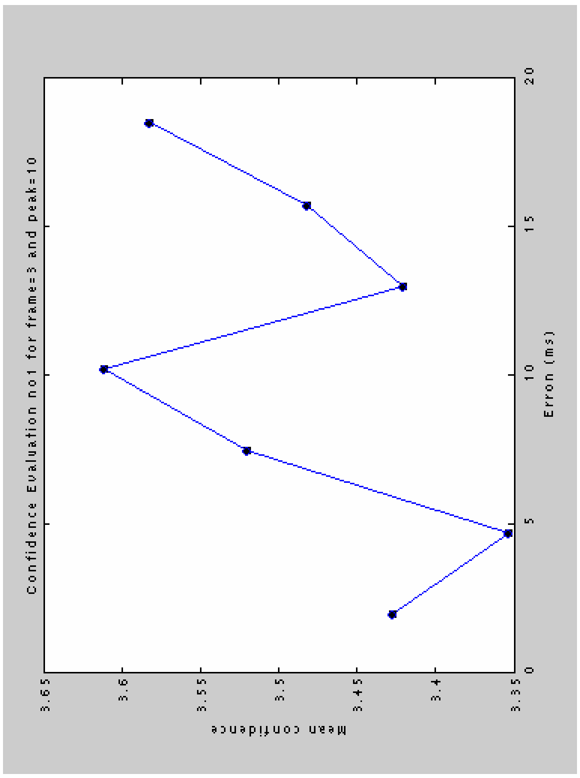
Smoothed confidence plot



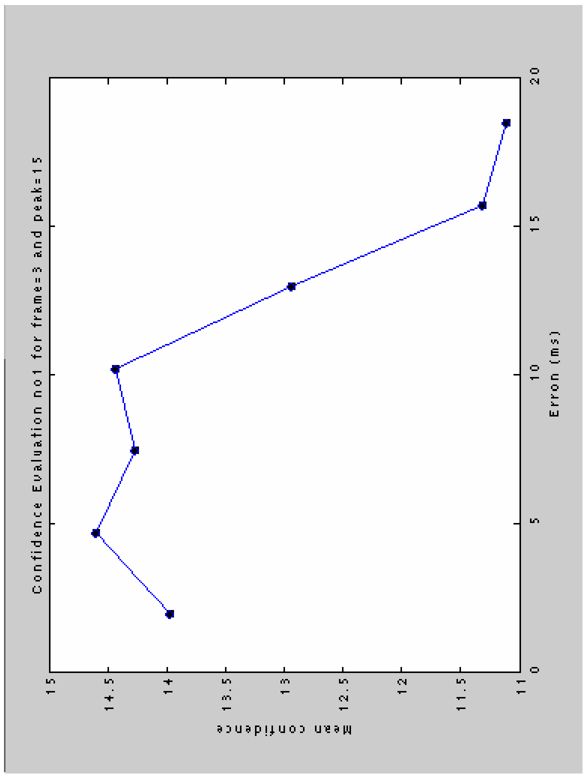




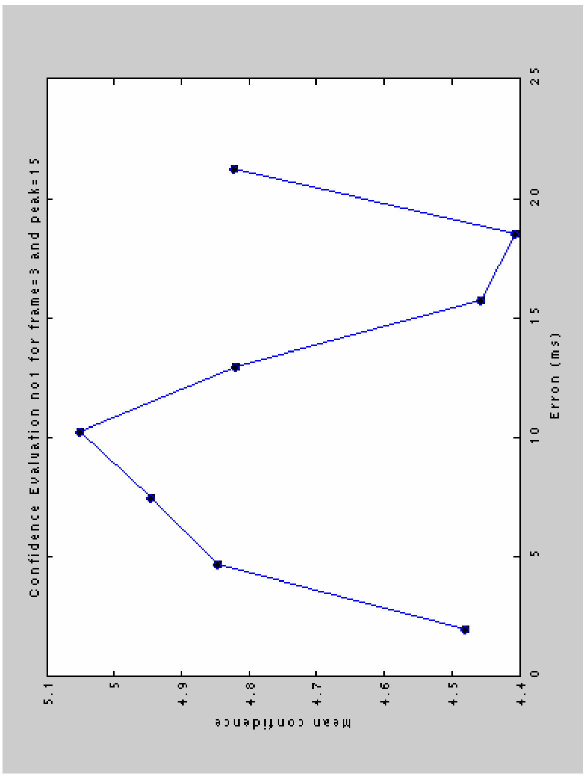
Simple confidence plot



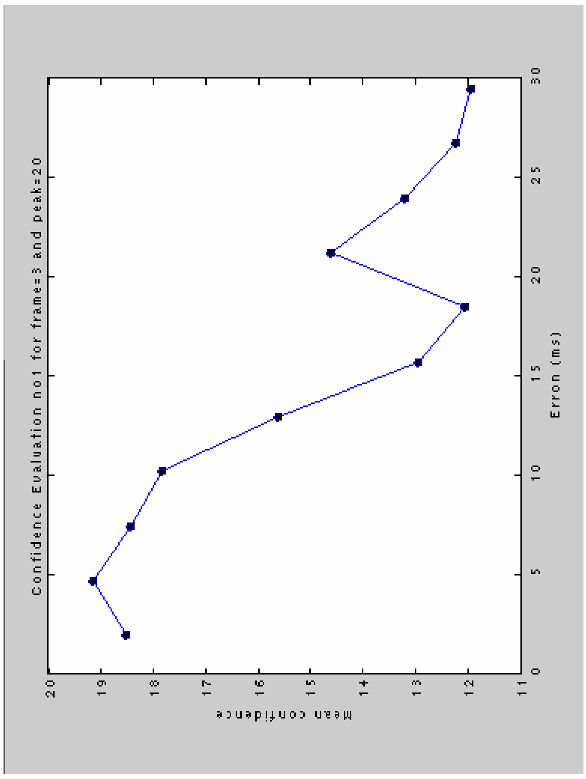
Smoothed confidence plot



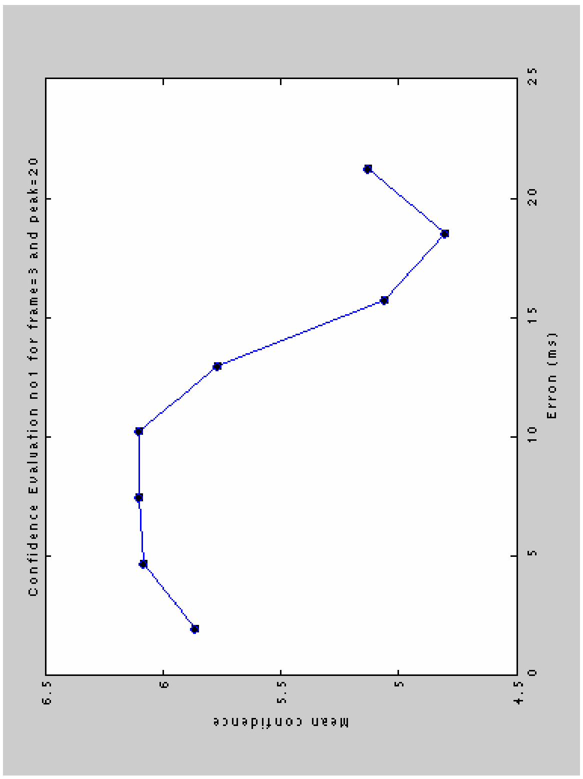
Simple confidence plot



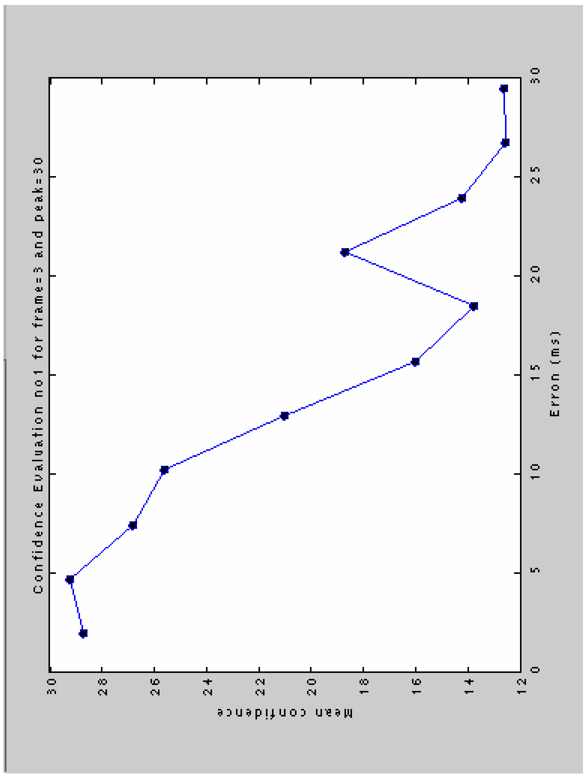
Smoothed confidence plot



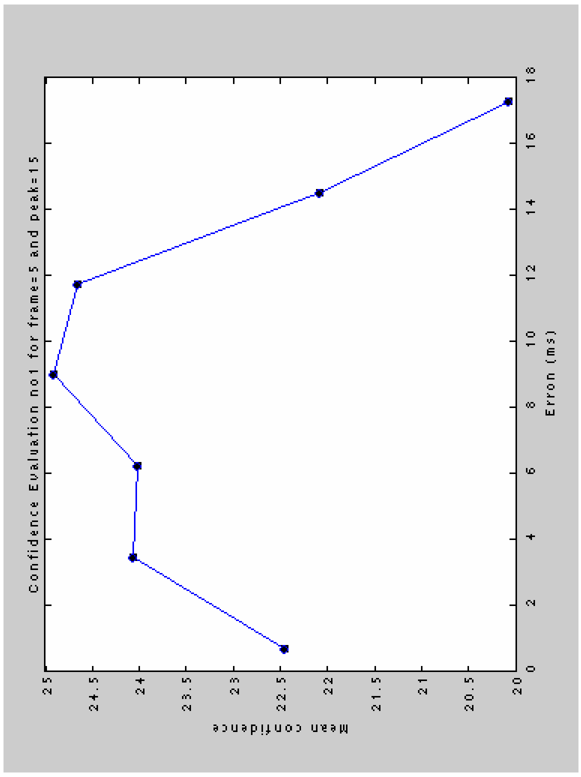
Simple confidence plot



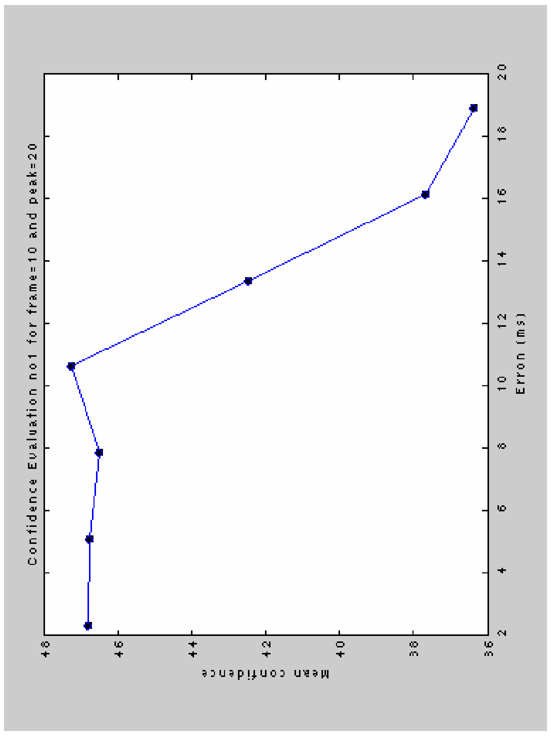
Smoothed confidence plot



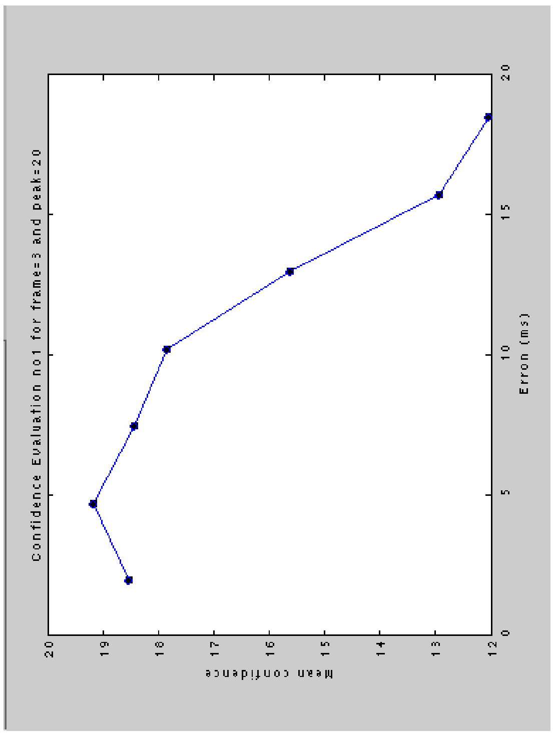
Simple confidence plot



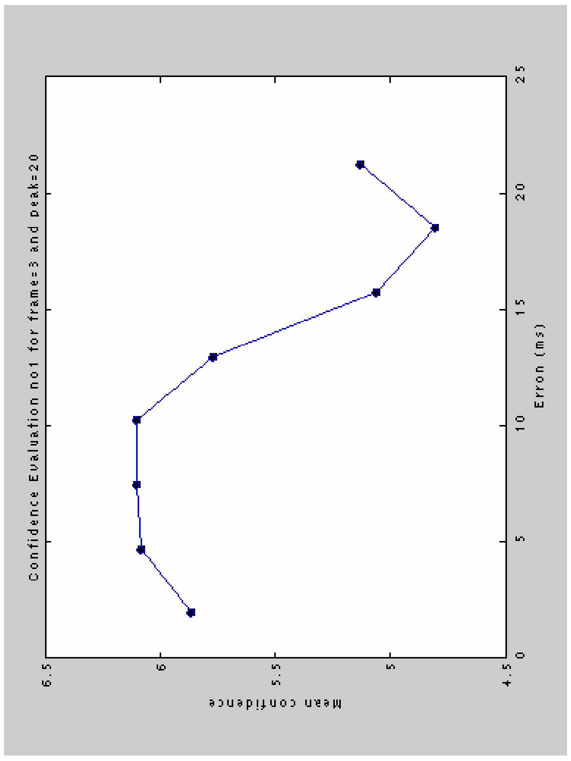
Simple confidence plot



Simple Confidence Plot

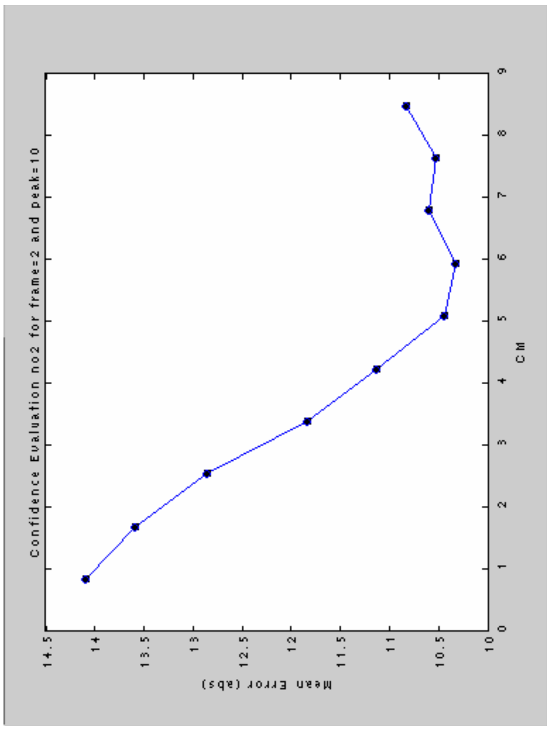


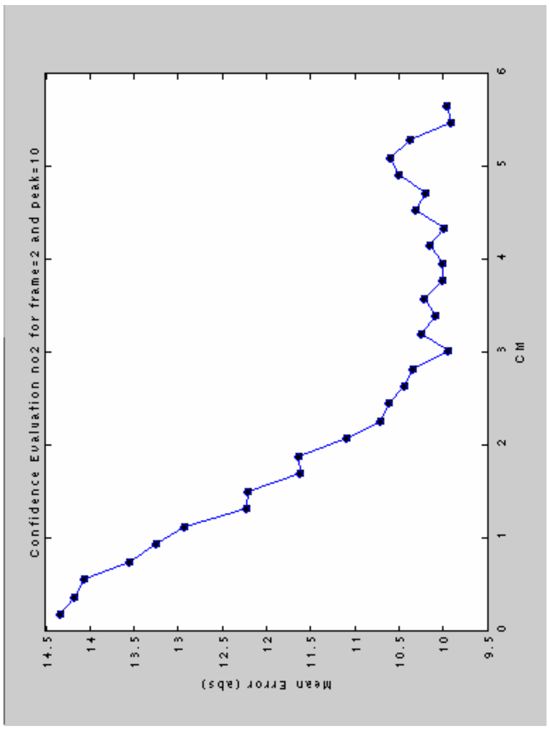
Smoothed confidence plot



Simple confidence plot

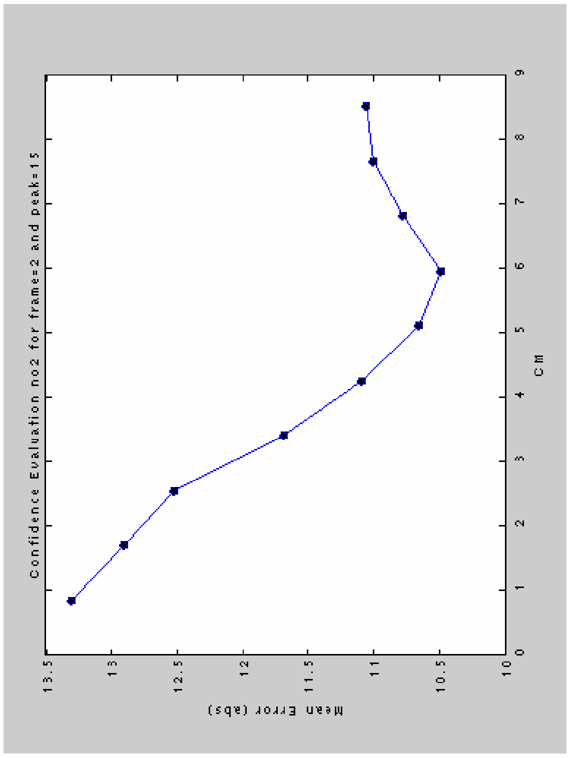
Confidence Measure (second group)

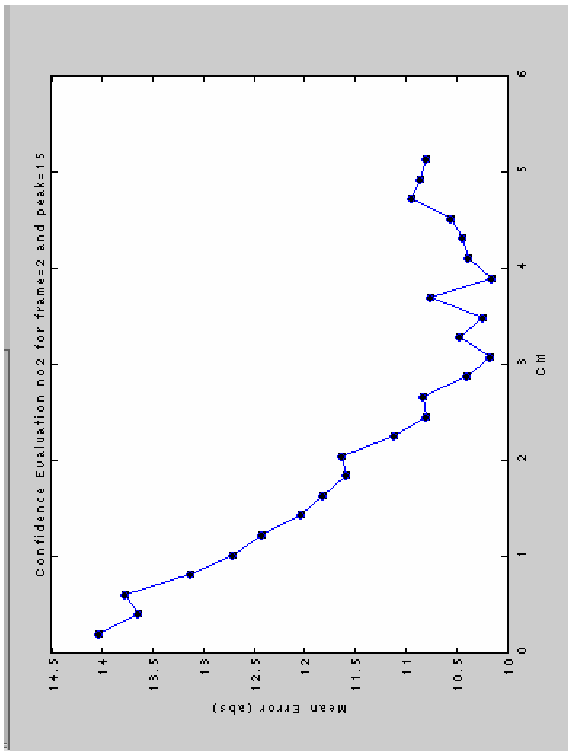


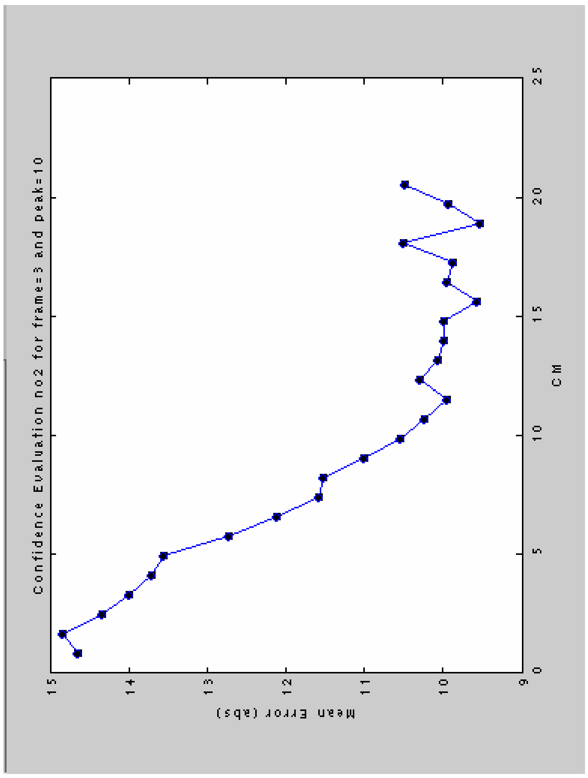


Simple confidence plot

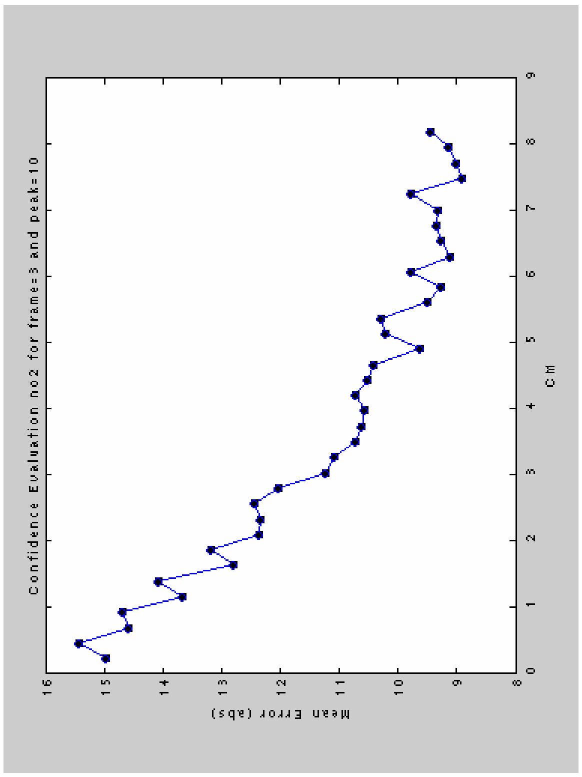
Smoothed confidence plot



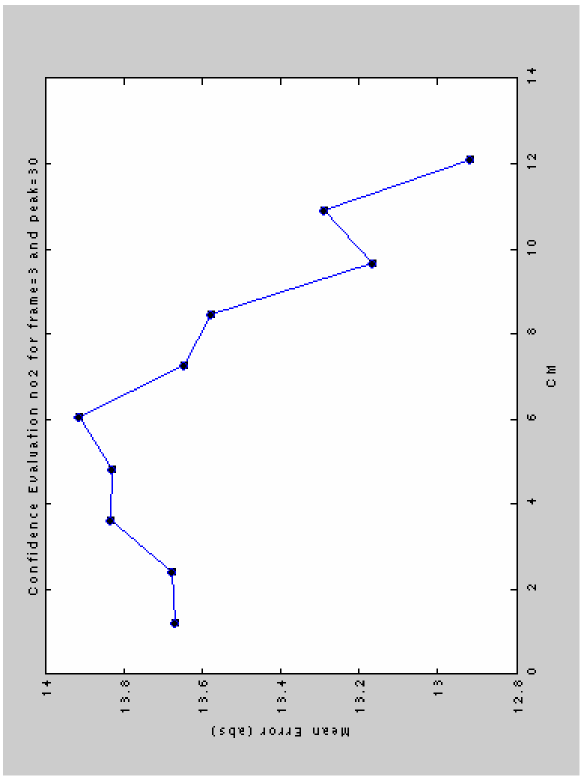




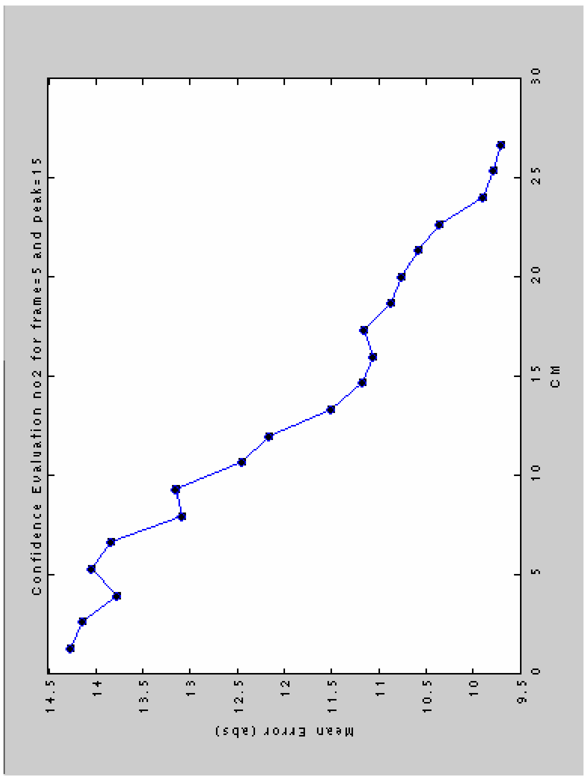
Simple confidence plot



Smoothed confidence plot



Simple confidence plot



Simple confidence plot