AUTOMATED ALIGNMENT OF SONG LYRICS FOR PORTABLE AUDIO DEVICE DISPLAY

BY BRIAN MAGUIRE

A thesis submitted to the

Graduate School - New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Master of Science

Graduate Program in Electrical and Computer Engineering

Written under the direction of

Prof. Lawrence R. Rabiner

and approved by

_____

_____

_____

New Brunswick, New Jersey

October, 2008

ABSTRACT OF THE THESIS

Automated Alignment of Song Lyrics for Portable Audio Device Display

by Brian Maguire
Thesis Advisor: Prof. Lawrence R. Rabiner

With its substantial improvement in storage and processing power over traditional audio media, the MP3 player has quickly become the standard for portable audio devices. These improvements have allowed for enhanced services such as album artwork display and video playback. Another such service that could be offered on today's MP3 players is the synchronized display of song lyrics. The goal of this thesis is to show that this can be implemented efficiently using the techniques of HMM based speech recognition. Two assumptions are made that simplify this process. First, we assume that the lyrics to any song can be obtained and stored on the device along with the audio file. Second, the processing can be done just once when the song is initially loaded, and the time indices indicating word start times can also be stored and used to guide the synchronized lyrical display. Several simplified cases of the lyrical alignment problem are implemented and tested here. Two separate models are trained, one containing a single male vocalist with no accompaniment, and another containing the same vocalist with simple guitar accompaniment. Model parameters are varied to examine their effect on alignment performance, and the models are tested using independent audio files containing the same vocalist and additional vocal and guitar accompaniment. The test configurations are evaluated for objective accuracy, by comparison to manually determined word start times, and subjective accuracy, by carrying out a perceptual test in which users rate the perceived quality of alignment. In all but one of the test configurations evaluated here, a high level of objective and subjective accuracy is achieved. While well short of a commercially viable lyrical alignment system, these results suggest that with further investigation the approach outlined here can in fact produce such a system to effectively align an entire music library.

Table of Contents

List of Figures

List of Tables

Introduction

In recent years, the popularity of portable MP3 players such as Apple's iPod and Microsofts Zune has grown immensely. As the storage capacity and processing power of such devices continues to expand, so does the ability to offer added features that enhance the users experience. Consumers can already view album artwork, watch videos, and play games on their portable units; far exceeding the capabilities of portable CD players and other traditional media. Another highly desirable feature that could be realized on today's MP3 players is real time display of song lyrics. An automated system that outputs the lyrics on screen in synchrony with the audio file would allow the user to sing along to popular songs, and easily learn the words to new songs. This would greatly enhance the multimedia experience beyond just listening to music.

The goal of this thesis research is to investigate the question as to whether this lyrical transcription can be realized accurately and efficiently using modern techniques of automatic speech recognition. The basic idea is to utilize a large vocabulary speech recognition system that has been trained on the vocal selections (perhaps including accompanying instruments) of one or more singers in order to learn the acoustic properties of the basic sounds of the English language (or in fact any desired language). The resulting basic speech sound models can then be utilized to align the known lyrics of any song with the corresponding audio of that song, and display the lyrics on the portable player in real-time with the music. This problem is therefore considerably simpler than the normal speech recognition problem since the spoken text (the lyrics) is known exactly, and all that needs to be determined is the proper time alignment of the lyrics with the music.

The speech recognition system that is used for this lyrical alignment task works as follows. In order to train the acoustic models of the sounds of the English language (the basic phoneme set), we need a training set of audio files along with text files containing the corresponding correct lyrical transcriptions. The audio files are first converted to an appropriate parametric representation, such as MFCCs (mel frequency cepstral coefficients), while the text files are translated to a phonetic representation. The resulting feature vectors and phonetic transcriptions are used to initialize a set of Hidden Markov Models based on a uniform segmentation of each of the training files. This initial set

of HMMs provides a set of statistical models for each of the 39 phonemes of the English language, as well as a model for background signal, often assumed to be silence. The initial HMM models are used to determine a new optimal segmentation of the training files, and a refined set of HMM estimates is obtained. After several iterations of re-segmenting and re-training, the segmentation of the audio files into the basic phonemes corresponding to the phonetic transcription of the lyrics converges, and the training phase is complete. The resulting set of HMMs is now assumed to accurately model the properties of the phonemes, and can be used to perform alignment of independent data outside the training set. In the problem at hand, these converged models can now be used to align songs stored on an MP3 player. In order for the resulting set of HMM models to be artist independent, thereby allowing transcription of an entire music library with just one trained set of models, the training set must contain a wide variety of artists representing a broad range of singing styles and performances.

In the above approach to this problem of automatic alignment of lyrics and music, two assumptions are made that will simplify the solution by exploiting the storage and processing capacity of modern MP3 players. First, we assume that along with the audio file itself, we can store the text of the song lyrics as metadata on the device. Lyrics for most popular songs are freely available online, and the space required to store this data is very small ($\tilde{1}$ kB) relative to the MP3 audio file itself ($\tilde{4}$ MB). With an accurate lyrical transcription available, the problem at hand reduces to one of finding an optimal alignment of the known lyrics to the given audio file, rather than a true large vocabulary recognition of the lexical content in the audio. The second assumption is that the alignment between a set of lyrics and the corresponding music can be performed just once, perhaps be verified manually, and then stored along with the lyrics as an array of time indices corresponding to times when each word within the lyrics begins in the audio file. In doing so, the potential bottleneck of real-time lyrical alignment and processing is eliminated. Much like Apple's iTunes currently loads album artwork and determines song volumes, the processing could be done every time a new song is downloaded. Then, during playback, the player device simply has to read the next time index and highlight the next word in the transcription. An approach to this storage of both song lyrics and timing information is given in [1].

In the experiments presented here, we will demonstrate the above approach on several simplified versions of the general problem of aligning song lyrics and music. We first consider the case of an artist dependent model with no musical accompaniment. For this simple case, the training data contains music and lyrics from only a single male vocalist, and the resulting set of HMM phoneme models will be used to align longer test audio files of this same vocalist to their transcribed lyrics. To further test the capability of this approach, alignment of a second set of test data containing a vocal

harmony will be performed using these same trained models. The harmony is performed by the same vocalist in time with the song's melody, but at a different pitch. Finally, a second set of training data containing music and lyrics from the same vocalist along with simple guitar accompaniment will be used to train a second set of HMM models. Independent test data containing this vocalist with guitar accompaniment is aligned using both the previous all-vocal HMM phoneme models, as well as this second set of HMM models. The resulting performance using these two very different models is compared.

In each of the above scenarios, the objective accuracy of the alignment is assessed by comparing the automatically generated alignment times of each word in the lyrics with the true word start times as determined by manual inspection of the test files. In another set of tests we measure the subjective accuracy by administering a perceptual test which emulates the display of an MP3 player. The test audio files are played using the audio output of the computer, and the test lyrics are displayed according to the automatically generated alignment. Participants in this subjective quality test listen to the music and observe the alignment of the lyrics, then rate how closely the lyrical alignment on screen matches the lyrical transitions heard in the music. The objective and subjective evaluation results are compared to see where this overall lyrical alignment approach produces significant errors, and which errors most effect perceived quality of the resulting alignment.

The test configurations outlined above fall well short of a commercially viable system for lyrical alignment on a modern MP3 player. Substantial complications are introduced when considering a system that is independent of both the vocalist and the musical accompaniment. Nevertheless, by achieving a high level of alignment accuracy, we hope to show that with further investigation the approach used here could become the first step in producing such a system.

Chapter 1
Background

A block diagram of the overall lyrical alignment system is shown in Figure 1.1 below. The system can logically be separated in to two segments, model training and independent alignment. In the model training segment, the training audio and text data is converted to appropriate representations and used to estimate the parameters of the phonetic models. In the independent alignment segment, audio and lyrics of songs outside the training set are converted to the same representations, and are aligned using the converged model estimates of the training step.

1.1 Representation of Audio

In order to train and test the lyrical alignment system described above, the audio files must be converted from the .wav format to an appropriate parametric representation. We assume that the audio files are created at a sampling rate of 44.1 kHz, representative of most high quality audio files. The first step in the processing is a reduction in sampling rate down to a more compact 16 kHz rate. Standard signal processing techniques are used to effect this change in sampling rate in Matlab. The next step of the processing is to block the audio file into frames and transform each frame into a spectral representation using an appropriate FFT routine. The FFT filter bands are converted to a mel frequency scale consistent with psychophysical measures as to the contribution of various spectral bands to our perception of speech. Finally the mel-scale spectral components are converted to a set of mel frequency cepstral coefficients (MFCC), since the MFCC coefficients have been shown to perform well in subword unit speech recognition [2].

In addition to the MFCC coefficients used in the spectral representation of each frame of audio, we also use a set of delta cepstrum coefficients as part of the feature vector in several cases. These delta cepstrum coefficients provide an approximate time derivative of the MFCC coefficients. The inclusion of such dynamic cepstral features has been shown to improve recognition performance [3].

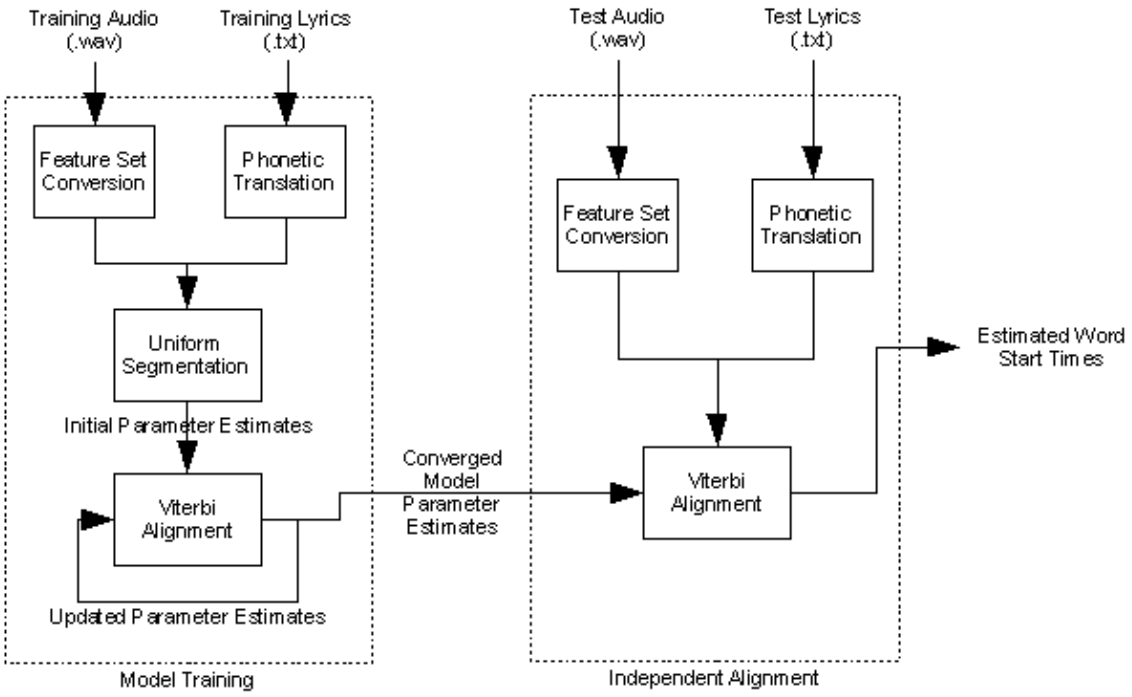


Figure 1.1: Overall block diagram of lyrical alignment system

1.1.1 Computation of MFCCs

In general, the mel frequency spectral coefficients of a segment of an audio signal are found by computing a high resolution short time log magnitude spectrum and mapping the spectral components to a set of mel frequency scale filters. A discrete cosine transform then provides the inverse Fourier transform of the mel frequency spectral coefficients, thereby providing a set of mel frequency cepstral coefficients. The Matlab implementation of the signal processing for determining mel scale ceptstral coefficients is based on code provided by Slaney [3] as part of the auditory analysis toolkit that is available freely over the Internet. The processing occurs in several steps as shown in Figure 1.2.

The input audio signal is passed through a pre-emphasis filter designed to flatten the spectrum

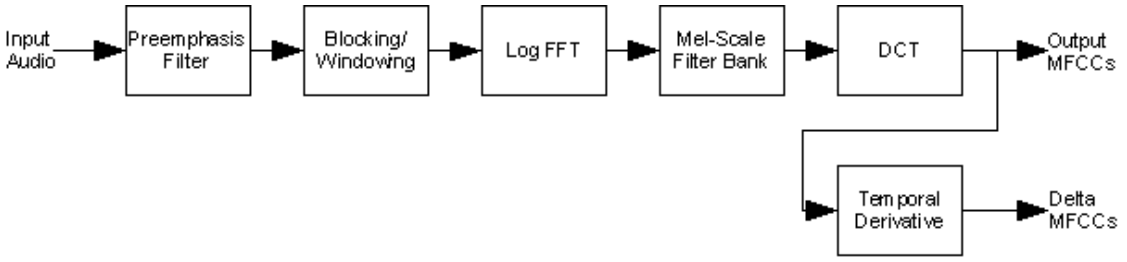


Figure 1.2: Block diagram of computation of MFCC feature set

of the speech signal. The spectrally flattened audio signal is then segmented into blocks, which generally overlap by up to 75%. A Hamming window is applied to each audio segment, also known as a frame, thereby defining a short time segment of the signal. In this implementation, the audio signals are downsampled to a 16 kHz rate, and a window length of 640 samples (40 msec) with a frame shift of 160 samples (10 msec) is used.

The next step in the processing is to take an FFT of each frame (windowed portion) of the signal. The resulting high resolution log magnitude spectrum of each frame is approximately mapped to a mel scale representation using a bank of mel spaced filters. Thirteen linearly spaced filters span the low frequency content of the spectrum, while twenty seven logarithmically spaced filters cover the higher frequency content. This mel frequency scaling is modeled after the human auditory system. By decreasing emphasis on the higher frequency bands, the lower frequency spectral information that is best suited for improved human perception of speech is emphasized.

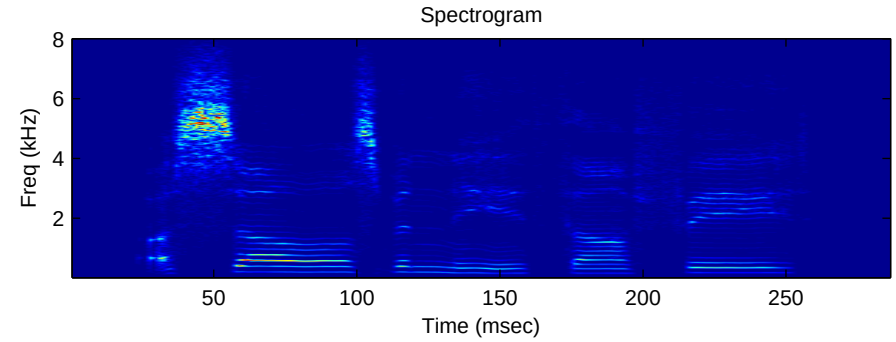
This mel scale spectrum (as embodied in the mel scale frequency bank) is converted to a log spectral representation and the set of mel frequency cepstral coefficients is computed using a discrete cosine transform of the mel scale log spectrum, thereby providing an efficient reduced-dimension representation of the signal [5].

In the experiments presented here, 13 MFCCs are used to form the feature vector for each frame. In some of our experiments we utilize a feature set consisting of the 13 MFCC coefficients along with a set of 13 delta MFCC coefficients. The first MFCC coefficient is the log energy of the frame, and is included in the feature vector.

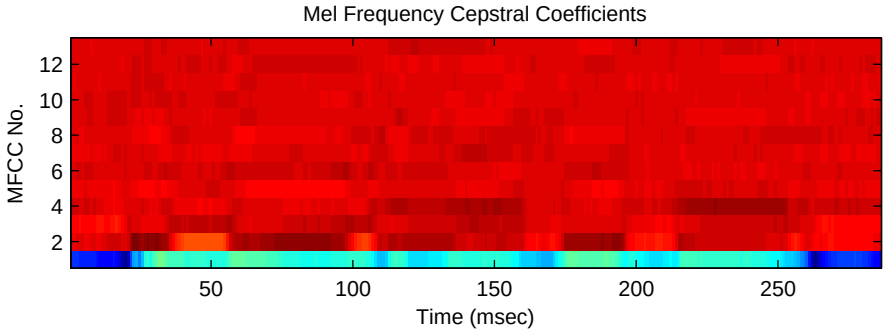
Figure 1.3(a) shows a spectrogram of an input sample of duration 2.8 seconds (286 frames). Figure 1.3(b) shows the corresponding MFCCs. Note the similarity in strong MFCCs among neighboring frames corresponding to the same sounds. Also, note the drop in log energy (first MFCC) between the vocal signal and the beginning and ending silence.

1.1.2 Pitch Independence

One notable property of MFCCs as a parametric representation of an audio signal is that they are largely independent of pitch [6]. Figure 1.4(a) shows the spectrogram of an input vocal sample. The horizontal lines indicate the pitch contour of the notes being sung. Figure 1.4(b) shows the interpolated and reconstructed spectrogram after computation of the MFCCs on the same input sample, showing the data approximated by the feature vectors. While the formant frequencies that define the phonemes are still clear, the horizontal lines that define the pitch are smoothed significantly.



(a)



(b)

Figure 1.3: Vocal sample 'Our souls beneath our feet'; (a) Input Spectrogram, (b) Corresponding MFCCs

In general speech recognition applications, this is valuable as fundamental pitch varies significantly between speakers [7]. In the case of sung vocals, this is even more significant as a single vocalist can cover multiple octaves of pitch within a single song or even a single line of music. The alignment to lyrics needs to be blind to this variation, as only the phonetic content of the signal is important for accurate alignment.

1.2 Hidden Markov Model

With our input audio files converted to an appropriate feature vector format, we can now begin to develop our formal lyrical alignment model. We assume that we can describe the speech sounds within the music using a basic set of 39 phonemes of English and an additional sound that represents the background signal (or silence). Thus for a training file with lyrics of the form 'Up in the air', we represent the resulting sound by the symbolic phonetic representation of the form:

/AH//P/ /AH//N/ /DH//AH/ /EH//R/

Here we will utilize a set of 40 Hidden Markov Models (HMM), one to represent each of these sounds of the English language and one for background signal [8]. Our first task in implementing

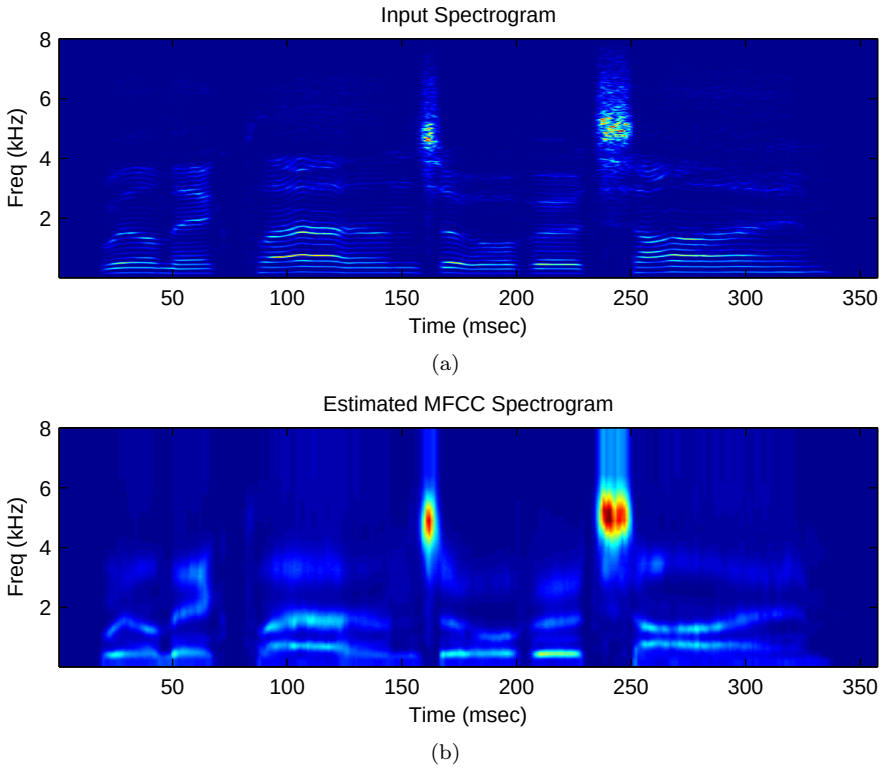


Figure 1.4: Vocal sample 'We'll make plans over time'; (a) Input Spectrogram, (b) Spectrogram as approximated by MFCC computation

a lyrics alignment algorithm is the training process, in which we estimate the parameters of this set of HMMs using an appropriate training set of music and lyrics. As will be shown later, the background signal (or silence) model is of particular importance when considering the inclusion of musical accompaniment. In such a case, there is a distinction between vocal silence, where background sounds are still present, and true silence.

The training set for estimating the parameters of the 40 HMMs consists of a set of music audio files along with corresponding accurately labeled transcriptions. The transcriptions contain a sequence of words, and are converted to a sequence of phonetic labels using a pronunciation dictionary [9], where initially there is assumed to be no silence between words. Each phoneme HMM consists of a sequence of states and within each state there is a statistical characterization of the behavior of the MFCC coefficients in the form of a Gaussian distribution. The HMMs are assumed to obey a left-right state model. An example of a three state model for the phoneme /AY/ is shown in Figure 1.5. The basic assumption of a left-right state model is that the system can remain in a given state for only a finite number of time slots (frames of MFCC coefficients). Hence if the duration of the /AY/ sound is T frames, the alignment of the T frames with the 3 states of the model of Figure 1.5 can only be 1 of a small number of possibilities, e.g., frames 1 and 2 in state 1, frames, 3-8

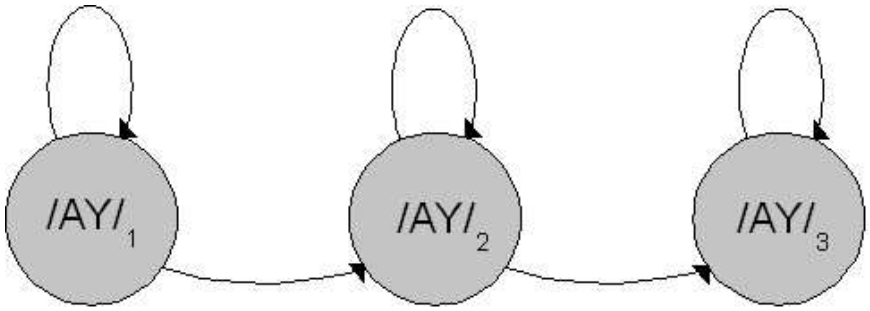


Figure 1.5: Three state model of phoneme /AY/

in state 2, frames 9-T in state 3 etc. It is the goal of the training procedure to determine the optimal alignment between frames of the sound and states of the HMM. Once that optimal alignment is determined, the statistical properties of the frame vectors, within each state of each HMM model can be determined, thereby enabling optimal training of the phoneme and background HMMs.

1.2.1 HMM Initialization

Before beginning iterations to refine the HMM models, an initial estimate of the statistical parameters within each state of the HMM must be provided. A simple initialization procedure is to assume that initially there is a uniform segmentation of the training data into phoneme HMM states. With this assumption, each audio file in the training set is first transformed into frames of MFCC feature vectors, and an approximately equal number of MFCC frames is assigned to each state in the phonetic transcription of each file. The training files are assumed to have a region of silence at the beginning and end of each audio file. Figure 1.6 shows examples of a uniform segmentation for the case of one and three state HMMs for the utterance 'Up in the air'. Note that the region initially labeled as silence is very accurate, leading to a very good initial estimate of the parameters of the silence model. This is beneficial in later stages of refining the model, especially once silence between words is allowed.

After performing this uniform segmentation on all training files, frames of MFCC data are now assigned to each model state. A mean and variance is computed for each element of the MFCC feature vector, thereby effectively defining a Gaussian distribution which characterizes each state of the HMM models. As discussed earlier, the choice of MFCCs as a feature set is beneficial here as the discrete cosine transform produces a vector that is sufficiently decorrelated, allowing us to perform our calculations on 13 independent Gaussian random variables rather than a single multivariate distribution.

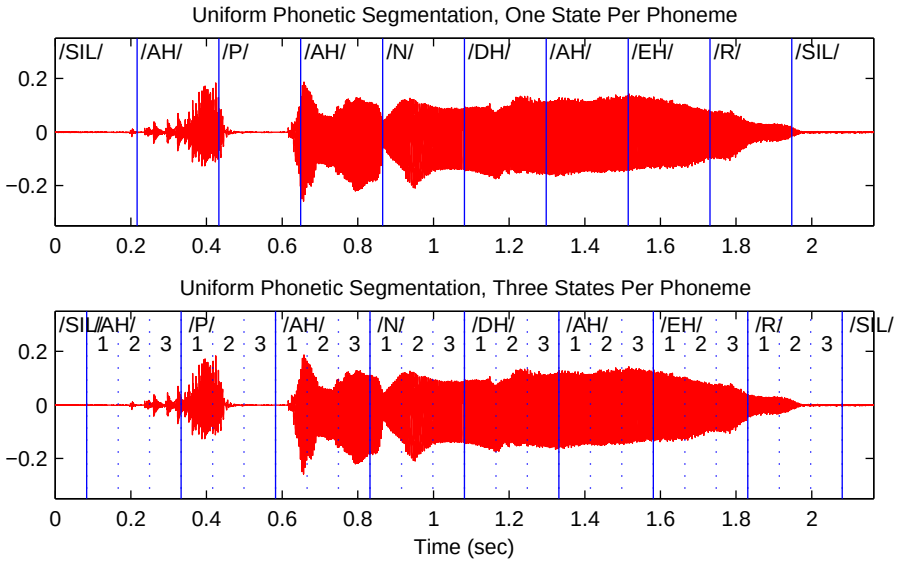


Figure 1.6: Comparison of uniform segmentation of vocal sample 'Up in the air' using one and three states per phoneme

1.2.2 HMM Training

Once initial HMM model estimates (Gaussian distribution means and variances) are computed for each state of the HMM models, the process of aligning HMM models with MFCC frames is refined (essentially a re-training process) using a Viterbi alignment. This maximizes the likelihood of the alignment over the full set of training files. Again, the known phonetic transcriptions are used, but now the possibility of silence between words is allowed as shown in Figure 1.7. For this model of phoneme concatenation, the last state in each word can stay in the same state, transition to silence, or skip the silence and transition to the first state of the following word.

The Viterbi alignment of MFCC frames to states of the HMM phonetic models proceeds as follows for each training file. First the log likelihood of each input frame 'i' belonging to state 'j' of the phonetic transcription is computed using the Gaussian distribution obtained from the initial uniform segmentation by the formulation:

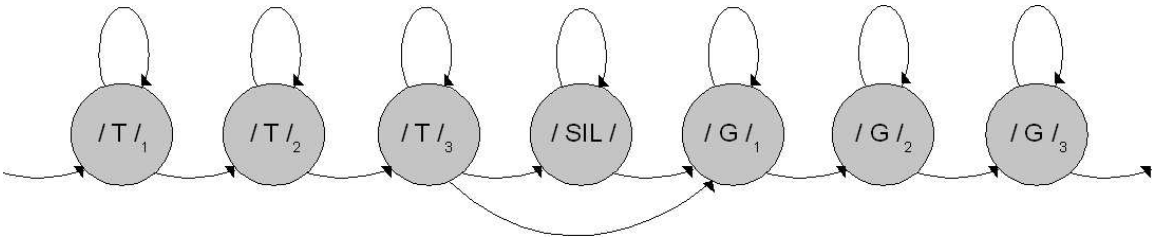


Figure 1.7: Illustration of optional silence state between words

$$p(j,i) = \log \prod_{d=1}^{13} \frac{1}{\sqrt{2\pi\sigma_{j,d}^2}} \exp\left(\frac{-(x_{i,d} - \mu_{j,d})^2}{2\sigma_{j,d}^2}\right) \tag{1.1}$$

The Viterbi algorithm aims to determine the segmentation of MFCC frames among the states of the phonetic transcription that maximizes this log likelihood over the entire phrase. The transitions among states are not only constrained by the assumed left-right model, but also by the beginning and ending states. The first and last frames of MFCCs must of course be assigned to the first and last phoneme states respectively. Thus the initial accumulated log likelihood is simply the likelihood of the first frame belonging to the first state of the transcription (most often silence). For all states thereafter, the new accumulated log likelihood $\delta_i(j)$ is computed as the sum of the maximum likelihood of all possible preceding states and the likelihood of the current frame 'i' belonging to the current state 'j'. The index of the preceding state which maximized this likelihood is recorded as $\psi_i(j)$. These formulations are as follows:

$$\delta_i(j) = \max_{j-1 \leq k \leq j} (\delta_{i-1}(k)) + p(j,i) \tag{1.2}$$

$$\psi_i(j) = \arg \max_{j-1 \leq k \leq j} (\delta_{i-1}(k)) \tag{1.3}$$

Upon reaching the final state of the transcription, there is only one allowed transition; to the final MFCC frame of the audio. By following the entries of the matrix $\psi_i(j)$ backwards, the optimal path aligning the MFCC frames with the phonetic states is traced as each entry indicates the preceding state which maximized the likelihood. An example of an initial uniform path and the subsequent optimal Viterbi path is shown in Figure 1.8. The beginning and ending path constraints are also shown for clarity.

After performing this Viterbi alignment for all files in the training set, an updated estimate of the model statistics is computed. As with the initial segmentation, all training audio frames are now segmented by phoneme and state, and a new mean and variance can be computed for each HMM model and state. With this updated model, another iteration of the Viterbi alignment is performed, and HMM model statistics are again refined. This process is performed until the total accumulated log likelihood over all the training files converges (i.e., doesnt change from one iteration to the next). Figure 1.9 shows the improved segmentation of words in an audio file from the initial uniform segmentation through several iterations of the Viterbi alignment.

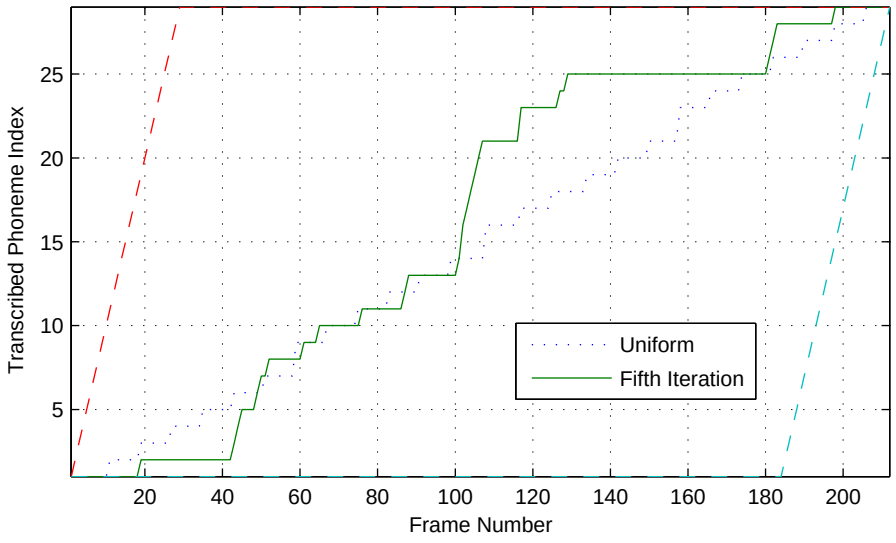


Figure 1.8: Comparison of intial uniform path and optimal path after five iterations of the Viterbi algorithm for vocal sample 'Up in the air'

1.2.3 Aligning Independent Test Data

Once the set of HMMs have sufficiently converged based on iterations of the training data, they can be used to align independent test samples not contained within the training set. In this case the test set consists of songs stored on an MP3 player. Using the prior assumption that we can reasonably store a text transcription of the song lyrics along with the audio file, the alignment process is identical to the Viterbi algorithm performed on the training data above. The text transcription is converted to a phonetic transcription using a pronunciation dictionary, and the audio is transformed to a string of feature vectors. Using the model parameters from the final training iteration, the optimal path of phonetic states across the input feature vectors is computed. The key difference is that the Viterbi alignment is only performed once, i.e., no further iterations can be used to adapt the model to the new data. In the experiments to be presented, the process is simplified by using test data containing the same vocalist as the training data. In a viable system for commercial use, a wide range of vocalists and musical styles must be included in the training data to allow for a model which performs accurate alignment independent of the artist.

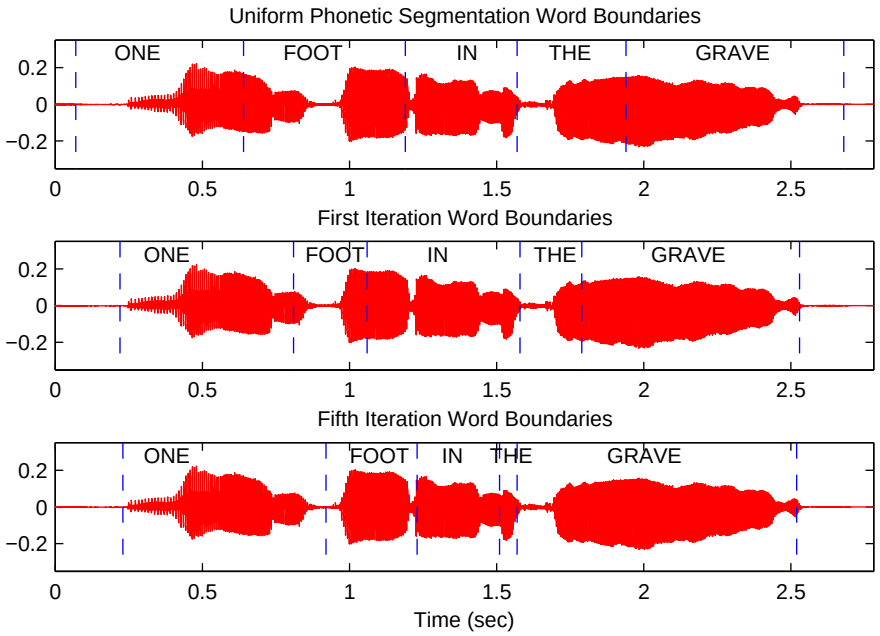


Figure 1.9: Comparison of word alignment for vocal sample 'One foot in the grave' through intial segmentation and several iterations of the Viterbi algorithm

Chapter 2
Implementation and Results

This system to align song lyrics to audio is implemented in Matlab and tested here in a series of experiments. Using an appropriate set of data, the training step is implemented using several different choices of model parameters in order to determine the best performing model. The results of each training run are analyzed to ensure that the phonetic models have converged to a reasonable estimate. Then, several different sets of independent test data are aligned and evaluated using the converged training models. Objective scores are obtained by comparing aligned word start times to the ground truth, namely the start times obtained by manual alignment of the test data. Subjective scores are obtained using a perceptual test where participants grade the perceived quality of the alignment between text and audio. The objective and subjective results are compared for consistency, and analyzed to determine the effectiveness of the implementation.

2.1 Matlab Implementation

A series of Matlab functions were written to perform the model training and testing described above. The input audio was converted to MFCC parameters using the implementation provided by Slaney [3], and lyrical transcriptions translated to phonetic representations using the SPHINX dictionary [9]. Model training is divided in to two steps: intialization and Viterbi iterations. The function 'initialize.m' first calls the appropriate functions to convert the inputs, then calls 'uniform.m' to break the frames of MFCCs in to uniform blocks based on the length of the file's phonetic transcription. The frames are then assigned to the appropriate phoneme and state in a Matlab data structure. Once completed for all training files, a mean and variance is computed for all frames assigned to each phoneme state, thus providing an initial estimate of the HMM model parameters.

The function 'iteration.m' is the main calling function to perform the iterations of the Viterbi algorithm. Again this function uses the frames of MFCCs and phonetic transcriptions, and passes them to 'viterbi.m' which computes the likelihood of each frame belonging to each HMM state, then the δ and ψ matrix entries. From this the optimal path is determined, and frames are assigned to

each phoneme state. Again, once completed for all training files, a mean and variance is computed for each phoneme state and the HMM model parameters are updated. This function is repeated until the model converges as discussed previously.

Independent test data is aligned using the Viterbi algorithm as implemented in 'iteration.m'. The key difference is that the alignment is performed only once, so there is no need to compute updated means and variances for the phoneme states. Rather, the optimum path is returned, and passed along with the outputs of 'wordinds.m', which determines the start and end points of words within the phonetic transcription, to the function 'compstarttimes.m'. This final function generates a text file containing an array of time indices indicating the start time of each word in the transcription, which is in turn read by the GUI to produce output text aligned with the audio files. The full implementation of these Matlab functions can be seen in Appendix A.

2.2 Data

For simplicity, the data for these experiments was based on the features from a single male vocalist. All data was taken from audio recordings done on a PC-based multi-track audio system. The full recordings contained a main vocal track, a guitar instrumentation track, and, in several cases, a second harmony vocal track. This allowed for identical vocal samples to be used with various backgrounds by inclusion or exclusion of the additional tracks. The harmony vocal track contained the same male vocalist singing identical lyrics, but at a varied pitch from the main vocal track.

2.2.1 Training Data

Due to the melodic nature of the sung vocals, it was important to train the models on vocal samples occurring in context with natural variation in pitch, timing, and duration. Therefore, rather than having a prepared list of training utterances sung by a participant, as is often the case in the training of a speech recognition system, the training data was obtained by dividing full length songs in to small sections. The lyrics of these sections were then each transcribed in order to be converted to its phonetic transcription.

Two sets of training data were used in the experiments to follow. The initial training set consisted of just the single male vocalist with no background or instrumentation. There were 183 files with length ranging from 1 to 6 seconds and 1 to 7 words each, obtained from 5 complete songs. The transcriptions of the entire training set are detailed in Appendix B. Nearly all the files began and ended with silence, allowing the uniform segmentation to proceed as discussed earlier. In the few

cases where a single vocal line is split and insufficient silence existed at the beginning or end of the sample, the corresponding phonetic transcription was adjusted accordingly.

The second training set consisted of both the vocalist and an accompanying guitar. The same 5 songs were divided in the same fashion, which yielded 183 files with identical vocal content to those in the initial training set. This second set was used to train a separate set of HMMs, on which similar vocal and accompaniment test data were aligned. Although the same instrument was used in all 5 songs, there was some stylistic variation as several songs contained quiet picked guitar while others contained louder full chord strumming.

It should be noted that this training set was significantly smaller than one that would be found in a general large vocabulary speech recognition system. As will be shown in the experiments that follow, due to the alignment constraint of our problem, satisfactory performance was still achieved in most cases. The drawbacks of this small data set will be seen in the training step as model complexity is expanded.

2.2.2 Test Data

The test data was obtained in a fashion similar to the training data. 4 different songs from the same vocalist were used to generate 8 test samples with length ranging from 14 to 27 seconds, and 18 to 40 words. As was done above in generating two separate training sets with identical vocal data, three separate test sets were created. The first contained the single vocalist with no background, the second contained the vocalist with guitar accompaniment, and the third contained the vocalist with a vocal harmony (and no guitar). Note that two of the full song audio files contained no vocal harmonies, so the harmony set contained only six files.

While most real songs to be aligned will be significantly longer than the test samples used here, this has little bearing on the quality of the alignment. To prove this assertion, a full 2 minute song containing 118 words was aligned to a fully trained model in several different ways. First, the song was aligned as one complete audio file and one complete transcription. Then, the song was split in two, and the two files were aligned separately. This was done again with the original file split in to four and eight separate files. Figure 2.1 shows the resulting distribution of objective word start time errors for these four cases. The distributions were very similar, indicating that the alignment quality was not dependent on the length of the audio samples. It will be shown in the results of our model testing that the alignment process quickly recovers from most errors.

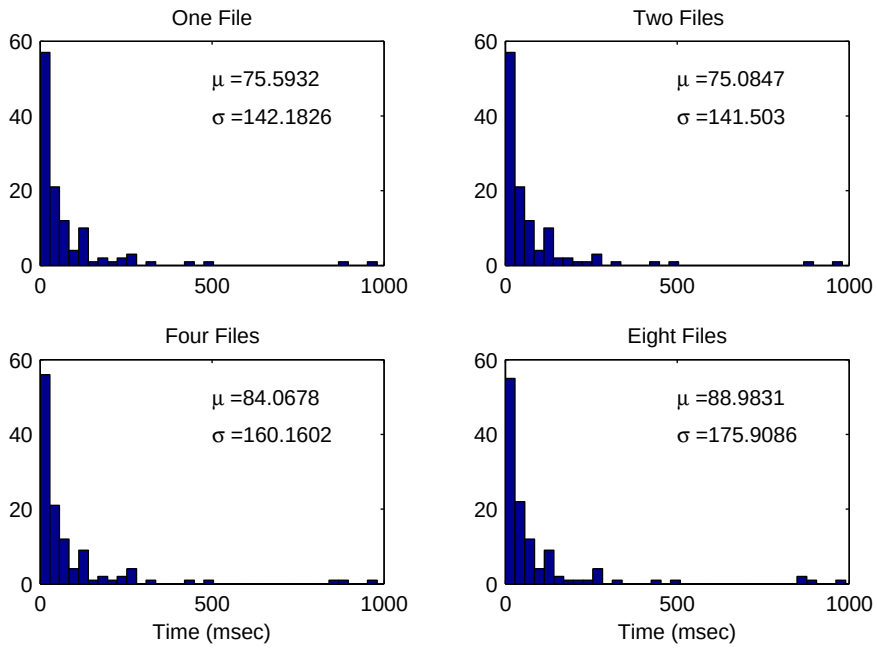


Figure 2.1: Distribution of word start time errors for sample song aligned as one, two, four, and eight separate files

2.3 Model Training Results

Two types of acoustic models were trained using two different sets of training data. The first training set (called the all-vocal case) contained only the male vocalist recordings, while the second training set (called the mixed-recording case) contained recordings of the vocalist with guitar accompaniment. For the all-vocal case, several different choices of model parameters were investigated in order to determine the best set of model parameters. Only one mixed-recording model was trained.

2.3.1 All-Vocal Case Models

Four sets of HMMs were trained using the all-vocal case training set described above. The HMMs differed in the number of features (13 for using just MFCC features or 26 when using MFCC plus delta MFCC features), the number of HMM states for each phoneme (1 or 3) and finally the number of Gaussian mixtures (1 or 2) in the statistical model for each state of each HMM model. The ultimate goal was to determine which combination of model parameters performed best in objective and subjective tests of performance.

Table 2.1 lists the parameters of each of the four HMMs. Models 1 and 2 used a 13 element MFCC feature vector, while Models 3 and 4 used 26 element MFCC feature vectors. Model 1 consisted of a single state per phoneme HMM while Models 2-4 used 3 state HMM models. All models used

Model	Number Features	States per Phoneme	Gaussian Mixtures
1	13	1	1
2	13	3	1
3	26	3	1
4	26	3	2

Table 2.1: Description of all-vocal case model parameters

a single state HMM to represent background signal (or silence). Finally Models 1-3 used a single Gaussian mixture to characterize the statistical properties of the feature vector in each state of each phoneme HMM, whereas Model 4 used a 2 mixture Gaussian in each state of each phoneme HMM.

For each of the four models of Table 2.1, the audio files were segmented and converted into the appropriate feature set, and the word transcriptions were converted to phonetic transcriptions using a word pronunciation dictionary (based on the word pronunciations from the SPHINX system [8]).

Initial model estimates (means and variances of the single or 2 Gaussian mixture models) were obtained by uniformly segmenting each training utterance into HMM states corresponding to the phonemes within each utterance and then determining the mean and variance of all feature vectors assigned to a common HMM state for the entire training set of utterances. Following the uniform segmentation step, the iterations of the model training procedure began. As discussed in detail in Chapter 2, each full iteration of model training entailed Viterbi alignment of the feature vectors of the 183 training files to the concatenation of the current HMM models corresponding to the known phonetic transcription. From the set of alignment paths for all utterances in the training set, the HMM model estimates (means and variances) were updated at each iteration, and a total accumulated log likelihood was computed as the sum of the log likelihoods of each training file. Training model iterations continued until the sum of log likelihoods converged to a constant value. Figure 2.2 shows the accumulated log likelihood scores for the first 5 iterations for each of the four models of Table 2.1. By the end of the fifth iteration, all four of these models were converged.

Figure 2.2 shows that Models 1 and 2, which used only a thirteen element feature vector, had lower total log likelihood scores than Models 3 and 4. Similarly we see that models with 3 states per phoneme (Models 2, 3, and 4) provided higher log likelihood scores than models with just 1 state per phoneme (Model 1). Finally we see that the model with 2 Gaussian mixtures per state (Model 4) had a somewhat lower log likelihood score than the model with the same parameters but only 1 Gaussian mixture per state (Model 3).

To further compare the effectiveness of the resulting set of phoneme HMMs, it is instructive to examine the likelihood scores of each of the final converged models in more detail. Since the goal

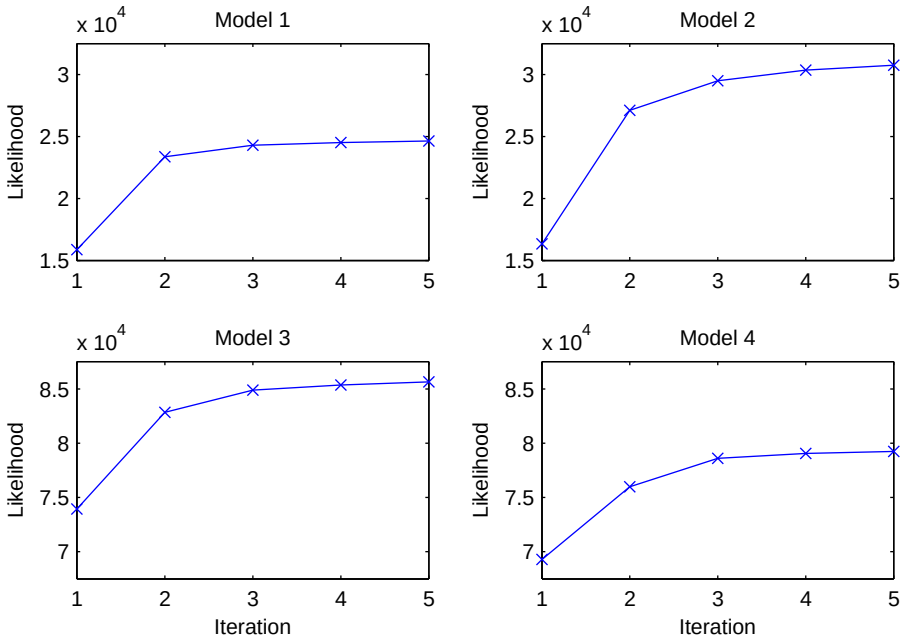


Figure 2.2: Convergence of total log likelihood over five training iterations for the four all-vocal training models

of the alignment process was to accurately determine locations within each training file of word transitions, it would be expected that the accumulated log likelihood over individual words would give some indication as to how well this goal was achieved. Figure 2.3 shows the distribution of word likelihood scores for the four models, as well as the mean and standard deviation. It can be seen from Figure 2.3 that the word distribution scores for Model 2 have a significantly larger mean than the word distribution scores for Model 1, with comparable variances. Hence it appears that HMM models with 3 states per HMM give better scores than HMM models with one state per HMM. It can also be seen that the word distribution scores for Models 3 and 4 have much higher means than for Models 1 and 2, indicating that the use of the 26 parameter feature vector gives better scores than the 13 parameter feature vector. Finally we see that the word distribution scores when using two Gaussian mixtures are actually lower than when using a single Gaussian mixture per state, indicating that there may not be sufficient data to define clearly two Gaussian mixtures per state.

As mentioned earlier, the training set size used to define HMM parameters was rather small, especially when compared to the training set sizes used in modern speech recognition systems. The 183 files in the training set for the all vocal models contained just 2311 phonemes from 699 words. The distribution of these 2311 training set phonemes was far from uniform, resulting in a substantial variation of phoneme counts and phoneme log likelihood scores for each of the 39 possible phonemes.

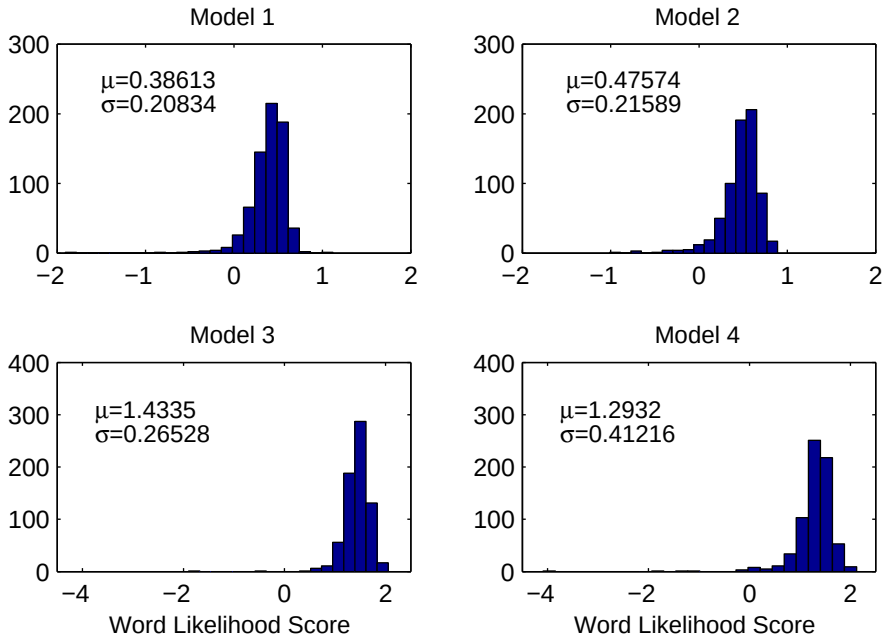


Figure 2.3: Distribution of word scores after five training iterations for the four all-vocal training models

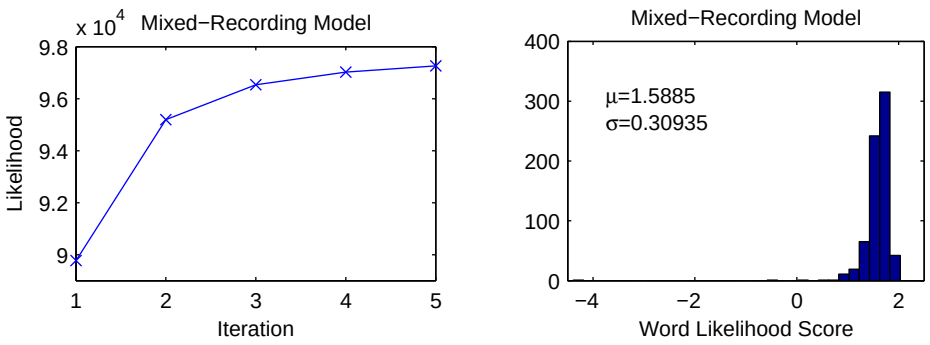
As with any assisted machine learning task, the use of more labeled training data should logically lead to better representations of the sounds of English and ultimately to better performance in the task at hand, namely aligning the music file to the corresponding lyrics file.

To test this assertion, Table 2.2 shows a list of the 39 phonemes of English, ordered by count in the overall training set, showing the average log likelihood score for each phoneme and for each of the 4 models that were trained. The average log likelihood scores of Table 2.2 suggest that there is a strong correlation between the number of occurrences of a phoneme in the training set and its average log likelihood score.

It is interesting to note that while recognition accuracy for most speech recognition tasks improves with the incorporation of additional Gaussian mixture densities in each HMM state [10], the results of Table 2.2 tend to indicate that using a two mixture model in fact degraded performance slightly from that obtained using a single Gaussian mixture per state. The likely cause of this was a lack of sufficient data to accurately train the two mixture model. By increasing HMM model complexity using additional Gaussian mixtures and increased number of states, the small amount of training data is often insufficient for providing reliable and robust model estimates. This appears to be the case for most of the phonemes of Table 2.2 where there are less than 50 occurrences for about half the phonemes. For these phonemes the accuracy and reliability of the means and variances of the

Phoneme	Count	Model 1	Model 2	Model 3	Model 4
N	223	0.365	0.428	1.563	1.356
AH	181	0.171	0.100	1.204	1.004
T	150	0.137	0.226	0.955	0.809
R	126	0.148	0.182	0.752	0.609
D	125	0.083	0.148	0.745	0.691
L	111	0.131	0.123	0.757	0.727
S	106	0.308	0.366	0.827	0.801
AY	86	0.211	0.218	0.738	0.720
AE	78	0.208	0.252	0.707	0.605
IY	72	0.149	0.161	0.600	0.531
EH	68	0.190	0.215	0.642	0.565
F	67	0.128	0.205	0.492	0.424
IH	64	0.130	0.139	0.490	0.429
K	59	0.080	0.094	0.368	0.290
EY	57	0.129	0.161	0.523	0.493
ER	54	0.086	0.120	0.397	0.363
M	53	0.095	0.114	0.364	0.319
UW	53	0.046	0.051	0.373	0.327
AO	51	0.116	0.092	0.433	0.365
V	50	0.101	0.149	0.343	0.128
OW	50	0.095	0.126	0.416	0.365
W	50	0.071	0.108	0.338	0.293
Z	49	0.109	0.112	0.311	0.259
DH	49	0.060	0.100	0.319	0.283
B	46	0.050	0.084	0.285	0.247
AA	35	0.087	0.097	0.286	0.255
P	31	0.069	0.067	0.214	0.200
HH	26	0.047	0.006	0.182	0.150
NG	25	0.053	0.076	0.162	-0.080
G	25	0.030	0.054	0.175	0.134
AW	23	0.056	0.085	0.203	0.156
Y	20	0.025	0.032	0.128	0.119
TH	12	0.017	0.034	0.083	0.067
JH	10	0.022	0.028	0.075	0.054
CH	9	0.020	0.031	0.058	0.005
UH	9	0.013	0.018	0.073	-0.004
SH	7	0.030	0.027	0.061	0.050
OY	1	0.006	0.007	0.011	0.000
ZH	0	0.000	0.000	0.000	0.000

Table 2.2: Average log likelihood scores for all occurences of each phoneme in the all-vocal training set models



(a) (b)

Figure 2.4: Results of mixed-recording model training; (a) Convergence of total log likelihood over five training iterations, (b) Distribution of word likelihood scores after fifth training iteration

two mixture distributions is significantly less than that of the one mixture distributions.

2.3.2 Mixed-Recording Case Model

For simplicity, only one set of HMMs was trained using the mixed-recording case training data which contained both vocals and guitar accompaniment. Based on results obtained in the all vocal models case, models utilizing 26 parameter feature vectors with 3 HMM states per phoneme were used, with just a single Gaussian mixture for each state distribution.

Figure 2.4 shows the curve of how the log likelihood scores converged during the training phase along with the distribution of word likelihood scores after five training iterations on this training set. The training appears to converge to a model with comparable log likelihood to that of the all vocal training set models. The most significant difference between the mixed-recording case and the models from the all-vocal case was the background/silence model. Whereas utterances in the vocal training set contained beginning and ending silence in each file and the training alignment procedure allowed for optional silence between words (due to the vocalist taking breaks), the guitar training set never contained true silence. Rather, it contained sections without vocals, but still with some musical accompaniment as the guitar was continually played. This difference will be evident in the model testing results, as attempts to align mixed-recording samples on the all-vocal case models show significant misclassifications of silence.

2.4 Explanation of Testing

The training procedure of the previous section described the method for creating phoneme HMM models with different feature vectors, number of states per phoneme model, and number of mixtures

Test Config.	Test Data	Training Data	Number Features	States per Phoneme
1	All-Vocal	All-Vocal	13	1
2	All-Vocal	All-Vocal	26	3
3	Harmony	All-Vocal	26	3
4	Mixed-Recording	All-Vocal	26	3
5	Mixed-Recording	Mixed-Recording	26	3

Table 2.3: Data and model parameters for five test configurations

per HMM state. Using two different training sets we produced 4 models based on strictly vocal training utterances and one model based on a training set that consisted of vocals and guitar accompaniment. In this section we describe a series of objective and subjective tests for evaluating how well several of the resulting models could align independent test samples of audio and transcribed lyrics.

Table 2.3 shows the five configurations of test and training data that were used in evaluating the efficacy of the lyrics alignment procedure based on HMM models of the phonemes.

The first two test configurations represented tests designed to determine the ability of the basic training models (trained from all-vocal training data files) to align independent test files containing only the single male vocalist. Models 1 and 3 were used in these first two test configurations. Model 1, trained using 13 parameter feature vectors and using just one HMM state per phoneme, was used as a baseline comparison for the more complex Model 3 with a 26 parameter feature vector and three HMM states per phoneme. Test configuration 3 used the set of test files that contained both the main male vocalist melody and an additional harmony vocal track and evaluated performance using training Model 3. The final two test configurations of Table 3.3 used the mixed-recording (vocals with guitar accompaniment) set of test data along with either all-vocal training Model 3 (test configuration 4) or the mixed-recording training model (test configuration 5). This pair of comparisons served to show how well the all vocal model handled alignment of vocals with simple accompaniment, and how well the trained guitar model isolated and learned from the vocals in the combined guitar and vocal files.

2.4.1 Objective Testing

For each file in the test set of data, an alignment between the music file and the given lyrics was obtained by aligning the set of feature vectors of the music file with the concatenated set of HMM states corresponding to the sequence of words (and phonemes), noting the aligned location

of each word in the file. Objective testing of these five configurations of test and training data was performed by comparing the computed start times for each word in each test file to the 'ground truth' as determined by manual alignment of the audio to the lyrics. There is a limit to the perceptual precision of human alignment of lyrics to music, and therefore there exists some inherent inaccuracy in the manual alignments on which the comparison is based. As such, computed start time errors of several milliseconds may in fact be more correct than the 'ground truth' as determined by human listening and observing music waveforms. In observing the distribution of start time errors however, a models accuracy should be determined not by these fine errors, rather by the number of word location errors above a reasonable perceptual threshold.

Since the test data was all derived from the same audio recordings, and the timing of the vocals is independent of the chosen model parameters, for simplicity it was assumed that all test configurations were governed by the same ground truth alignment times. Thus the manual alignment was performed just once and used as the basis of comparison for all five tests. As will be seen, most notably in the harmony test configuration, the addition of a second vocal or musical accompaniment can slightly shift the perceived start of certain words, so the assumption that the 'ground truth' was independent of the test set was not completely accurate and needed to be taken into some consideration for these configurations. This effect was not significant enough to alter our assessments of model performance.

2.4.2 Subjective Testing

Subjective testing of the time alignments produced by the five test configurations was carried out by administering a perceptual test using a GUI designed to emulate the display of a portable mp3 player. Figure 2.5 shows a screen shot of this display. There were a total of 20 audio selections and as the music audio was played, the display highlighted the estimated word being sung according to the model aligned word start times.

The subject was given specific instructions before the test outlining the problem description and the system of scoring, reproduced in Appendix C. Playing through each of 20 files, one at a time, the subject was asked to assign a rating score from 1 to 4, with 1 being total misalignment of lyrics, and 4 being (perceptually) perfect alignment of lyrics. The first five files in all tests were identical, and served as a perceptual calibration on the subjects ratings. The alignments of these first five samples were intentionally skewed to produce a subjective rating with a desired score. These first five samples covered the range of possible alignment quality (rating scores of 1 to 4), and prepared the subject for the expected range of alignments within the test suite.

In order to obtain an equal number of rating scores for all five training models aligning all the

Figure 2.5: Screen shot of GUI used in perceptual test

eight test files, three different tests were created. Since each test contained fifteen actual test samples (after calibration), this allowed for nine rating scores for each training model across the three tests. Table 2.4 shows the logical organization of each test. Note the files in the perceptual test are given generic labels T1 through T20. The same song could be repeated in a test, but always with an alignment generated by a different model, or with alternate accompaniment.

2.5 Analysis of Test Results

The subjective and objective tests of the previous section were carried out and the results were compiled to determine the overall accuracy of this approach to the lyrical alignment problem. Comparisons were made across the various configurations described in Table 2.3 in order to establish a correlation between the objective accuracy and perceptual scores. Within each model the errors were analyzed to determine the source of alignment inaccuracy, and which errors were most perceptually significant. Appendix D contains the complete word start time errors for all words in the test set.

Before discussing results of the various tests, it is worth a brief discussion of the perceptual calibration results. Table 2.5 shows the results for the five calibration files used in the perceptual test (labeled T1 to T5 in the table). Recall that a score of 4 represented perfect alignment and a score of 1 represented gross mis-alignment errors.

As discussed earlier, each calibration file was intentionally skewed from its true timing (word alignment) in order to simulate various levels of accuracy in word alignment. The variation was

	Test 1		Test 2		Test 3	
	File	Test Set	File	Test Set	File	Test Set
T6	1	3	2	5	8	2
T7	3	2	6	1	3	5
T8	4	5	8	3	7	3
T9	5	2	3	4	4	1
T10	7	1	1	2	7	4
T11	6	4	5	5	2	2
T12	4	3	4	2	1	1
T13	2	4	2	3	6	4
T14	5	5	1	4	4	3
T15	8	1	5	1	4	4
T16	3	3	3	1	7	2
T17	6	2	8	4	6	1
T18	1	5	7	3	8	5
T19	2	1	6	5	3	3
T20	5	4	3	2	7	5

Table 2.4: Contents of three versions of the perceptual test

done with an intended target score in mind. Table 2.5 shows the target score and average score across all 12 participants for each of the 5 files, along with the content of the background signal. In all but one case, the average score was above the target score, indicating that the participants were slightly more accepting of errors than anticipated. What was important however was that the relative scores were fundamentally ordered as expected. This indicates that the participants perception of good and bad alignment was in line with what was expected, albeit with a bias towards slightly higher baseline scores.

2.5.1 Test Configurations 1 and 2: All-Vocal Test Results

We begin by discussing results for the configurations using only the single vocalist in both training and testing data sets. Figure 2.6 shows the distribution of word start time errors, the difference between manually identified and system generated start times for all words in the test set, for test configurations 1 and 2. The difference between the distributions is clear. Test configuration 2 had

File	Background	Target Score	Avg Score
T1	Guitar	3	3.33
T2	None	4	3.92
T3	Harmony	1	1.67
T4	Guitar	2	2.50
T5	None	3	3.42

Table 2.5: Results of perceptual calibration

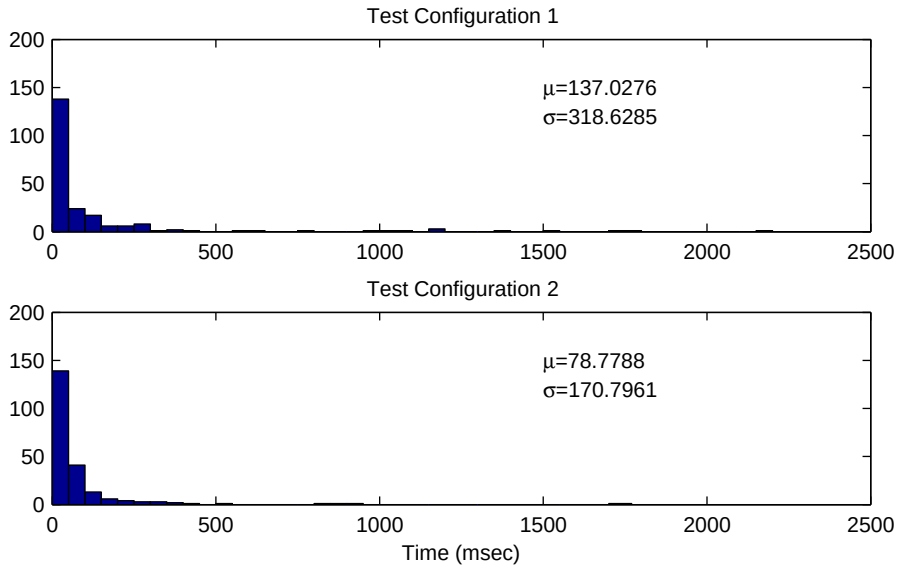


Figure 2.6: Distribution of objective word start time errors for test configurations 1 and 2, using all-vocal training and test data

a mean error of 79 msec and a standard deviation of 171 msec. This shows both a smaller average error and more narrow distribution of errors than test configuration 1, which had a mean error of 137 msec and standard deviation of 319 msec. Test configuration 1 produced several errors of duration longer than 1 sec, while configuration 2 produced only one of such errors. These objective results support what was shown in the model training results; the expansion of the MFCC feature vector from 13 to 26 elements and the inclusion of three states per phoneme HMM rather than just one improves the objective performance of the alignment system.

Figure 2.7 shows the overall results of the perceptual tests for these two configurations. These subjective scores show a similar improvement in performance in test configuration 2, which received an average perceptual score of 3.7. Test configuration 1 received an average score of 3.4. Also of note, out of 36 responses, test configuration 2 received just one rating of 2 (fair), whereas test configuration 1 received three. These results combine to suggest a strong correlation between objective and subjective scores.

In order to observe where serious alignment errors occur and determine how they effect perceived alignment quality, an analysis of the objective and subjective scores for each individual file and the corresponding word start time errors is necessary. Table 2.6 shows the average perceptual score for each of the eight test files used in test configuration 1, along with the counts of word start time errors sorted into bins of varying size: 0-99 msec, 100-249 msec, 250-499 msec, 500-999 msec, and 1000 msec and greater.

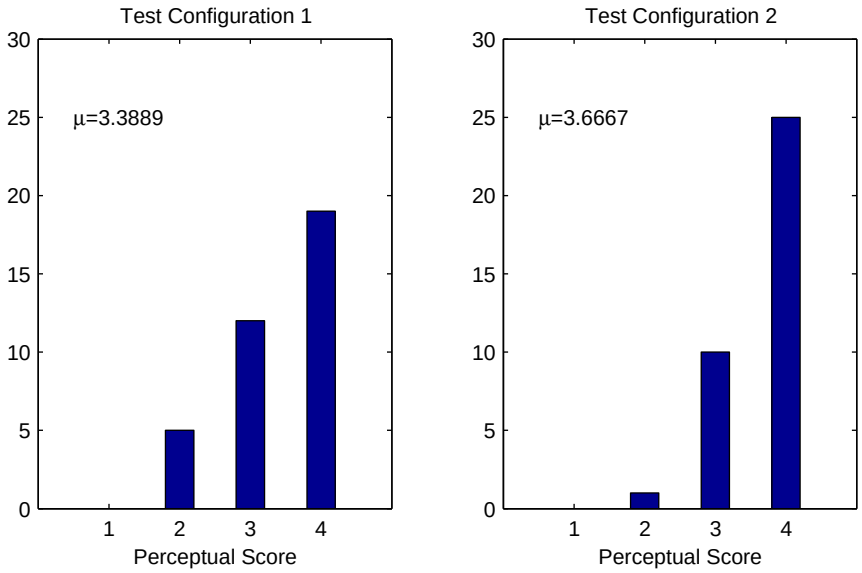


Figure 2.7: Distribution of subjective perceptual scores for test configurations 1 and 2, using all-vocal training and test data

File	Words	Percept. Scores	Word Start Time Errors (msec)				
			0-99	100-249	250-499	500-999	1000+
2	41	4.00	33	7	1	0	0
7	18	4.00	15	3	0	0	0
8	21	4.00	18	2	1	0	0
3	24	3.50	20	4	0	0	0
6	36	3.50	27	5	3	1	0
4	29	3.00	16	6	3	2	2
1	20	2.75	12	4	2	1	1
5	28	2.25	16	3	2	0	7

Table 2.6: Comparison of perceptual score and word start time errors for each individual file in test configuration 1

Files 2, 7, and 8 all received a perfect score of 4.0 in the perceptual test, meaning every listener judged the highlighted word changes in the user GUI as aligning perfectly with the music. File 7 had no word start time errors greater than 250 msec; in fact the largest start time error was 150 msec. All the listeners found this to be an acceptable perceptual start time error, and each gave File 7 a perfect score. Files 2 and 8 had maximum start time errors of 280 msec and 380 msec respectively. A single isolated start time error in this range did not appear to hinder the perceived quality of the alignments and again each listener gave these test files perfect scores of 4.0. File 2 had substantially more short start time errors than the other 2 aligned files that were rated perfect, but this increase was proportional to the increase in the total number of words in the file, and therefore did not negatively affect the resulting perceptual score. Curiously File 3 received a slightly lower perceptual score (3.50) even though it had a distribution of errors similar to that of File 7, with no errors greater than 250 msec. There are two errors which appear to cause this perceptual degradation. First, the longest error of 240 msec occurred during the last word of the file, while in all the other cases of large start time errors there was some chance to recover before the listener had to decide on an appropriate rating score. Second, word 18 ('AND') started 90 msec late while word 19 ('RELEVANT') started 40 msec early. Individually both were reasonably small start time errors, but when they occur together they gave the impression of rushing through the phrase. Similar combined errors occurred in this and other test files, but this appeared to be the most egregious occurrence of this phenomenon. This result indicates that although short function words have a lesser effect on perception of the meaning of vocal content, they are significant in the evaluation of timing accuracy.

The perceptual score of test File 6 was the same as that of test File 3 even though it appeared to have more objective start time errors. Several of the errors occurred in a similar fashion to the one described above, most notably the article 'A' appeared to be nearly omitted in several cases. The longest start time error was 590 msec early, and occurred on word 29, 'GROUND'. The preceding word is a drawn out article 'THE' (true duration 540 msec) which is voiced as one continuous phrase. This musical continuity is what distracts the user from the error, and prevents the perception of a long objective error.

Files 1 and 4 received identical scores from all listeners but one. Even with a small pool of listeners, this is an insignificant difference. Both files had a start time error at the first word of a line of lyrics that followed a long pause between lines. This is a common error that occurs throughout several of the configurations tested, and appears to be quite noticeable to the listeners and thereby significantly reduced the perceptual score ratings whenever it occurred.

File	Words	Percept. Scores	Word Start Time Errors (msec)				
			0-99	100-249	250-499	500-999	1000+
1	20	4.00	18	2	0	0	0
2	41	4.00	40	1	0	0	0
8	21	4.00	18	3	0	0	0
3	24	3.88	20	4	0	0	0
5	28	3.50	20	5	2	1	0
7	18	3.50	12	4	2	0	0
4	29	3.25	19	3	3	3	1
6	36	3.00	28	3	5	0	0

Table 2.7: Comparison of perceptual score and word start time errors for each individual file in test configuration 2

File 4 also exhibited a short rushed phrase early on in the file, spanning three erroneous words, but this did not appear to have a significant effect on perceived alignment as exhibited in the subjective scores. File 5, on the other hand, showed what appears to be a brief complete failure of the alignment process. In this case 6 of the 7 most significant start time errors occurred in a row, and gave the impression of complete misalignment in the middle of the file (words 12 through 17). Surprisingly, the alignment recovered soon after and provided several more well matched lines of lyrics before another noticeable error occurred in the final phrase.

Table 2.7 shows the perceptual scores and the corresponding word start time errors for all eight files under test configuration 2. The correlation between subjective and objective scores again appears to be real, as the highest scored files have no word start time errors over 250 msec. Files 1, 2, and 8 all received perfect scores, while file 3 received just one score of 3 (good). The subjective scores given to files 5 and 7 are similar to the scores given files in configuration 1 with comparable distributions of objective errors.

There does appear to be some variation among the lower scores though. The start time errors in file 4 occur in similar locations to those in this same file under test configuration 1. Most notably word 26 (‘OF’) begins the last line in the lyrics, and starts well over a second early. Figure 2.8 shows the waveform of this segment of the utterance, as well as the estimated phonetic alignment. While the vocal audio contains a significant pause between words ‘TIRED’ and ‘OF’, the alignment in fact skips the pause between these two and assigns the time interval of the long pause to the following word ‘BEING’. The problem is that the word ‘OF’ is very weakly articulated, and, as can be seen in the alignment shown in Figure 2.8 is taken to be simply part of the long silence. There is no strong harmonic content to represent the voiced portion of the word, /AH/, therefore the alignment inaccurately assigns the minimum three frames (one per state of the phoneme) in what should have remained the stop consonant /D/. Notice though that the system recovers quickly with accurate

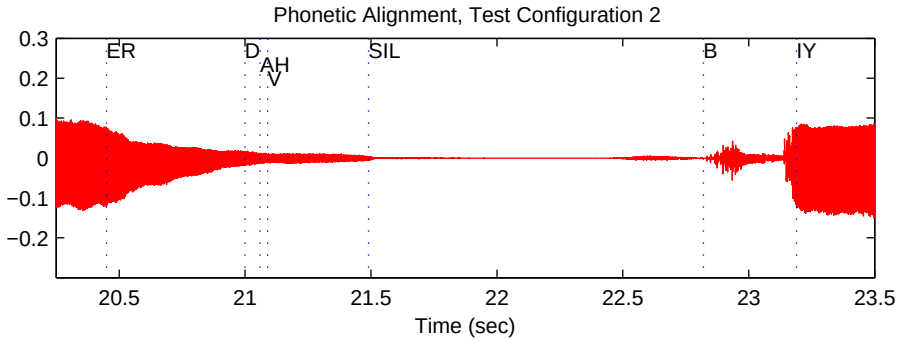


Figure 2.8: Incorrect phonetic alignment of silence portion in test file 4, '...tired /SIL/ of being...'

alignment of the following phonemes, /B/ and /IY/.

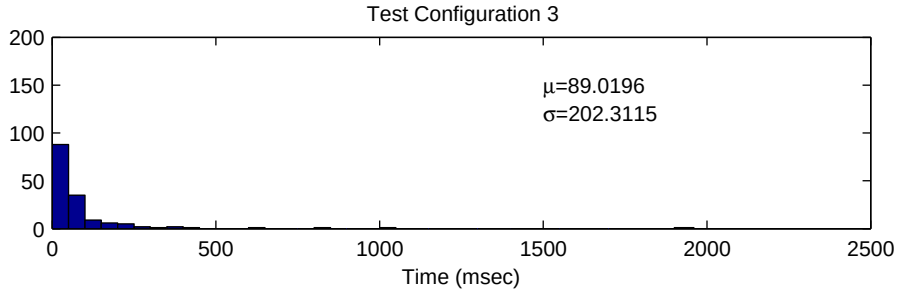
While care was taken to provide ideal training and test data in order to establish the viability of this system, the above example shows a major problem that can become an issue in any real implementation, namely the inconsistency of vocals, especially across different singers. Failures of this nature must be avoided by using adequate training of the acoustic phoneme models that allow for varied pronunciation and emphasis of each phoneme.

Finally, file 6 curiously received the lowest score for this test configuration even though it had none of the long duration errors exhibited in the alignment of File 4, and none of the magnified combinations of errors discussed above. In this case, the five errors above 250 msec were disbursed throughout the file. While one error of this length would be acceptable to most listeners, five such errors occurring in three distinct sections of a fairly short audio file appear to limit the quality of the alignment. While the average perceptual score was still good, this example shows that even without the most egregious errors as exhibited in several of the test files, consistent inaccuracy in start time location can and will negatively affect the perceived quality of the overall time alignment.

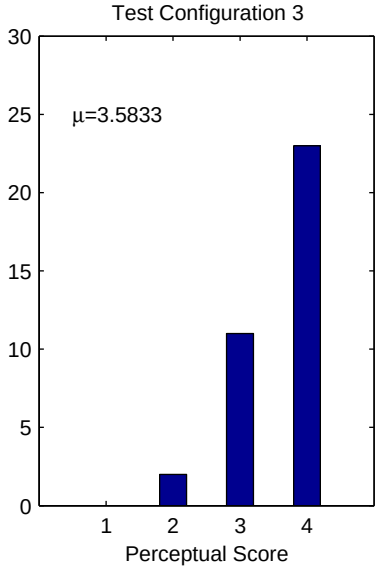
2.5.2 Test Configuration 3: Harmony Results

Test configuration 2 above appears to provide very good alignment quality for all but one of the test files. To further test the performance of the all-vocal model used, test configuration 3 included a second vocal track containing the same vocalist for which the model was trained singing the same lyrics, but at a different pitch so as to harmonize with the original vocals. As noted earlier, test files 5 and 6 contained no vocal harmony, and thus were omitted from this portion of the testing.

Figure 2.9(a) shows the objective results, the distribution of word start time errors, for this test configuration. With a mean error of 89 msec and standard deviation of 202 msec, the distribution of word start time errors appears quite similar to that of test configuration 2, especially in the pattern



(a)



(b)

Figure 2.9: Results for test configuration 3; (a) Distribution of objective word start time errors, (b) Distribution of subjective perceptual scores

File	Words	Percept. Scores	Word Start Time Errors (msec)				
			0-99	100-249	250-499	500-999	1000+
2	41	4.00	39	2	0	0	0
8	21	4.00	18	3	0	0	0
3	24	3.88	19	5	0	0	0
7	18	3.75	13	3	2	0	0
1	20	3.50	17	2	1	0	0
4	29	2.75	17	5	3	2	2

Table 2.8: Comparison of perceptual score and word start time errors for each individual file in test configuration 3

of the most significant errors, namely those above 500 msec. However, the distribution is not quite as compact for smaller errors, most notably in the range from 0 to 100 msec. As discussed earlier, this is likely because the addition of a second voice singing the same lyrics shifts the true start times from those determined from the single vocalist and used throughout these tests. Although the second vocal was sung along to the first, there was some natural variation in timing even for a trained musician. This slight degradation of time accuracy is insignificant, as it was shown in the all-vocal model tests that timing errors in this range were acceptable to the perceptual test participants. The similarity in objective results between test configurations 2 and 3 suggests that the addition of a vocal harmony had little bearing on the performance of the system.

The distribution of perceptual test rating scores for this test configuration are shown in Figure 2.9(b). The average score of 3.6 from this model was slightly lower than that of the single vocalist, although still on average quite good. A more detailed look at the results for each individual file shows to what extent this additional voice effects this configuration test results. Table 2.8 shows the average perceptual ratings score for each file, and the corresponding distribution of word start time errors.

Files 2, 3, 7 and 8 all received very high perceptual rating scores. In comparing the results in Table 2.8 to those in Table 2.7, it is clear that all four of these files have nearly identical distributions of word start time errors as achieved using test configuration 2 with just a few insignificant exceptions. For example, file 2 shows two words with errors in the range of 100 to 249 msec, whereas before there was only one. The disparity occurs in word 12 (‘AND’), where previously the error was 90 msec, and here it was 100 msec. This slight difference has no effect on the perceived quality of the lyrical alignment.

File 1 on the other hand had a noticeable start time error that was not present before the introduction of the vocal harmony. Specifically, word 3 (‘IN’) was off by only 30 msec in test configuration 2, but was off by 290 msec in test configuration 3. Perceptually the alignment clearly

hesitated before the transition from the preceding word ('ALONG'), thereby accounting for the decrease in perceptual rating scores. Since the harmony track was not in perfect synchronization, the /NG/ sound was shifted slightly in time, and auditorally this deemphasized the true occurrence of this nasal consonant. Recall from Table 2.2 that /NG/ was among the lesser occurring phonemes in the training set, which likely explains the difficulty in proper alignment of this unclear occurrence.

Much like in test configurations 1 and 2, file 4 contained comparable errors in the same two locations, although perceptually it was judged more harshly under this test configuration. Again, the last line began almost 2 seconds early, skipping the entire pause that came before this line of the lyrics. Words 13 through 15 again appeared rushed, but the alignment quickly recovered. This is the only file under this configuration that yielded an unacceptable alignment, although it should be noted that its objective errors were nearly identical to those of Configuration 2, again indicating that the addition of the vocal harmony had little effect on the vocal alignment.

2.5.3 Test Configurations 4 and 5: Mixed-Recording Test Results

To further test the capabilities of this alignment system, the final two tests included simple guitar accompaniment in the audio files. As shown in Table 2.3, test configuration 4 used the original all-vocal trained models with this new mixed-recording test set. Test configuration 5 used the mixed-recording models, trained on audio containing both vocals and guitar. These simple cases served to show whether the all-vocal model could align audio content beyond just simple vocals, or if a model trained on a range of accompaniment could provide better performance. As an initial comparison, Figure 2.10 shows the word start time error distributions for these two models.

Test configuration 4 shows a mean of 164 msec and standard deviation of 317 msec, while Test configuration 5 shows a mean of 96 msec and standard deviation of 160 msec. It is immediately obvious from these distributions that the model trained with guitar accompaniment provided much more accurate alignment than the model trained with just the vocal input. In fact, the model trained with both vocal and guitar accompaniment shows just one more long error than the all-vocal test configuration 2 using comparable model parameters. Figure 2.11 shows the distribution of perceptual scores for both of these configurations. The mean rating score for test configuration 4 was 3.11 whereas the mean rating score for test configuration 5 was 3.5. It can be seen that test configuration 4 received the lowest average rating score of all the configurations. Again comparing to the all-vocal test configuration 2, configuration 4 performed as expected. With only one additional long duration error, the number of scores of 2 (fair) was unchanged. However with a higher average error, indicating an increase in the short duration errors, fewer perfect scores of 4 were assigned by

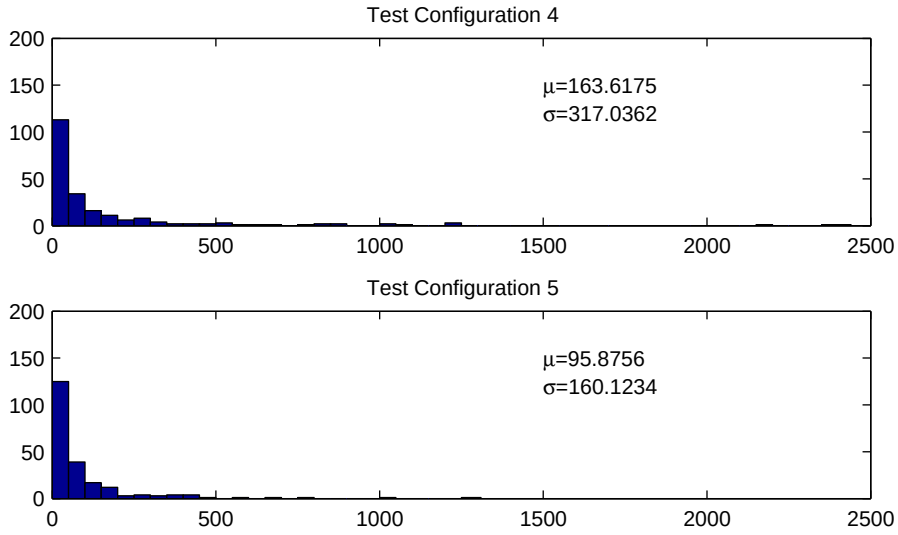


Figure 2.10: Distribution of objective word start time errors for test configurations 4 and 5, using mixed-recording test data

File	Words	Percept. Scores	Word Start Time Errors (msec)				
			0-99	100-249	250-499	500-999	1000+
8	21	4.00	18	2	1	0	0
1	20	3.75	13	4	1	2	0
2	41	3.75	35	3	3	0	0
7	18	3.75	12	3	2	1	0
6	36	2.63	26	5	2	2	1
3	24	2.50	13	3	2	4	2
4	29	2.50	14	8	3	2	2
5	28	2.50	13	7	5	0	3

Table 2.9: Comparison of perceptual score and word start time errors for each individual file in test configuration 4

the group of listeners.

As with the previous configurations, it is constructive to see where each file failed and succeeded, and which errors yielded the lowest perceptual scores. Table 2.9 shows the comparison of average perceptual score and word start time errors for each individual file tested using test configuration 4.

Four of the eight files received average scores below 3, while the ratings for the other four test files were near perfect scores of 3.75-4.0. In all four of the low scoring files, there were long pauses that led to long duration errors. This is a logical result for this test configuration. The silence model was trained to recognize true silence, but the mixed-recording test data includes a background guitar still playing through all vocal breaks. Figure 2.12 shows an example of this type of error in file 5. Shown is the ending of word 17 (‘ANYHOW’), which is followed by a pause, then the phrase

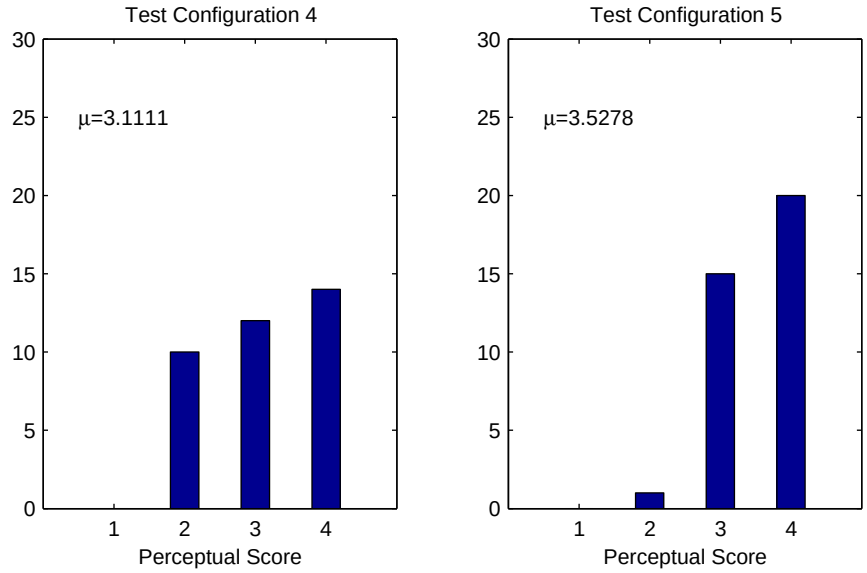


Figure 2.11: Distribution of subjective perceptual scores for test configurations 4 and 5, using mixed-recording test data

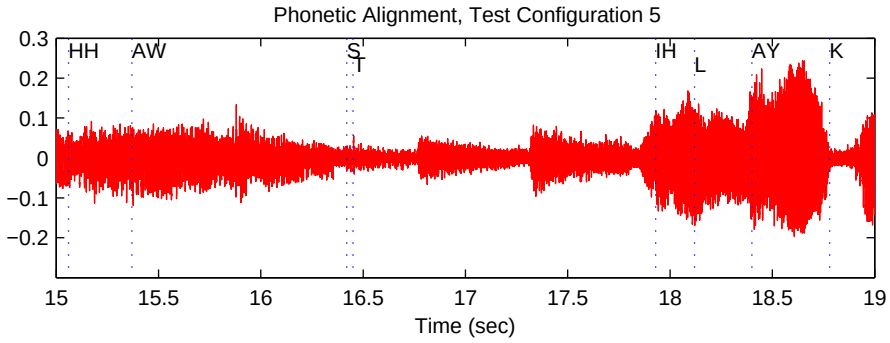


Figure 2.12: Incorrect phonetic alignment bypassing optional silence state in portion of test file 5, ''...anyhow, /SIL/ still I can't shake...'

'STILL I CAN'T SHAKE...'. Unlike the alignment error in Figure 2.8 where the silence was placed between the wrong words, here the silence was bypassed altogether. The consonants /S/ and /T/ begin as soon as the phoneme /AW/ ends, as the guitar that follows is distinct and different from the trained silence model. Notice though that the alignment does not assign any voiced phonemes to the harmonic strummed guitar. This is significant as it shows effective differentiation between the guitar and voice sections of the audio. It is for this reason that this configuration is able to identify phonemes and perform adequate alignment in segments with no long silence.

This main problem associated with test configuration 4 is eliminated in the second guitar based test configuration 5. Since the models are now trained with guitar accompaniment, the frames

File	Words	Percept. Scores	Word Start Time Errors (msec)				
			0-99	100-249	250-499	500-999	1000+
2	41	3.75	36	4	1	0	0
3	24	3.75	19	4	1	0	0
5	28	3.75	17	9	2	0	0
8	21	3.75	19	2	0	0	0
1	20	3.50	14	4	2	0	0
7	18	3.50	10	6	2	0	0
4	29	3.00	19	4	2	3	1
6	36	3.00	26	3	6	0	1

Table 2.10: Comparison of perceptual score and word start time errors for each individual file in test configuration 5

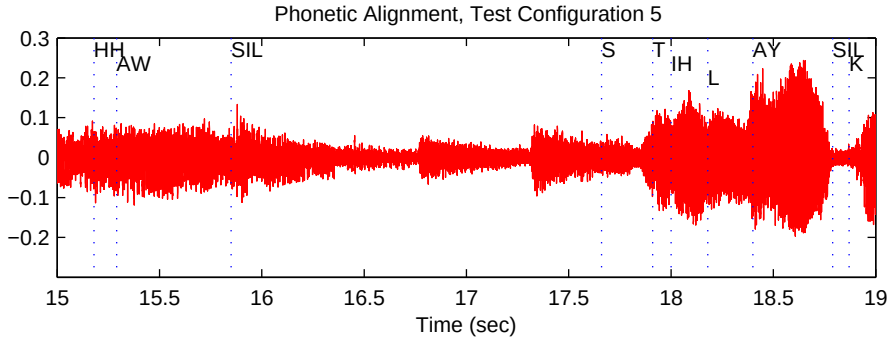


Figure 2.13: Corrected phonetic alignment of silence in portion of file 5, ''...anyhow, /SIL/ still I can't shake...'

assigned to background or silence are no longer true background/silence; rather they contain a continuation of the guitar. So as the test files are aligned, the model should assign silence to any portions with no vocal content, regardless of background. Table 2.10 shows the details of the perceptual scores for each test file along with the distribution of word start time errors.

Files 3 and 5, which in configuration 4 had long duration errors due to misplaced silence, now receive rating scores of 3.75 under this test configuration. The long duration errors have been eliminated, and those errors that remain are deemed acceptable by nearly all the listeners. Figure 2.13 shows this improvement in the previously discussed file 5.

File 4 continues to give the same errors found under the all-vocal configurations due to the poor enunciation of the vocalist. The only additional error introduced in this configuration occurred in file 6. Again it was an incorrect alignment surrounding a vocal pause. The reason for this is explained in the content of the file. While all the other test files contained a strummed guitar, file 6 had softer guitar picking as accompaniment. Although generated by the same instrument, these backgrounds were of significantly different timbre and spectral content. This is a return to the main problem

of configuration 4, where the system was presented with content it was not trained for over what should be vocal silence. This example illustrates the complexity of a real world implementation based on this type of model, namely that various performances on many different instruments must be accounted for in order to achieve a successful artist independent lyrical alignment system.

Chapter 3
Conclusions

The goal of this thesis research was to show that the techniques of HMM based speech recognition could be applied to audio files in order to align songs to their transcribed lyrics. A high level of accuracy was obtained in several of the scenarios presented here. In the ideal case of a single male vocalist with no accompaniment aligned to a set of speech phoneme models trained from the music and lyrics of that same vocalist, all file alignments were deemed acceptable to listeners in a formal perceptual test. While these subjective scores were passable, several noticeable objective errors occurred, indicating a need for further improvement in order to obtain a perfect alignment system. One possible improvement is the expansion of the MFCC feature vector used to represent the training and test audio. Also, while inclusion of multiple Gaussian mixture distributions did not appear to improve model accuracy in the experiments described here, improved performance should occur with a much larger training data set. A significant expansion of the training set also would also be necessary to create an artist independent system, where the trained models could accurately align songs featuring different vocalists and different instrumental backgrounds.

For the configurations containing musical accompaniment, a major source of word alignment error occurs in the misclassification of long periods of silence in the audio. To improve performance and eliminate this class of gross errors, it would be necessary either to train an improved background/silence model, or possibly consider the design of multiple background/silence models that represent the range of backgrounds for which the system is intended to work with. For example, with the training data used above, two 'silence' models could be trained: one for vocal silence with background (guitar) still present, and one for true silence. An alternative solution would be to create a trained system to distinguish vocal segments from vocal silence regions [11]. Early detection of background/silence regions and their elimination from subsequent signal processing would likely alleviate silence misclassification errors. Another substantial source of error is likely to come from the inclusion of a diverse range of musical backgrounds. While the model trained to include the guitar accompaniment successfully aligned the test data with similar background, further investi-

gation is required to see if it is feasible to train a model that can recognize vocals amidst varying instrumentation. This could prove difficult due to the excessive number of vocal and instrument combinations that can be present in even a small collection of songs. The necessary training data to account for all possible cases may be unreasonable. In that case, further signal processing steps must be taken to first isolate the vocals from the other instruments contained in the song before training and alignment are attempted.

The inclusion of a synchronized display of lyrics on a portable MP3 player would enhance the audio experience for the consumer. These initial results show a high level of accuracy on several simplified versions of this problem. With the addition of expanded training data and further audio preprocessing, the system outlined here could provide a viable implementation for automatic alignment of lyrics and music spanning an entire music library.

Appendix A

Source Code

A.1 intialize.m

```
% function
% [phones,features,uniformchanges]=initialize(states,feat,numfiles)
%
% function to initialize speech model: computes feature vectors and
% performs uniform segmentation of each input file, then assigns vectors to
% appropriate phonemes and computes initial estimates of means and vars.
% note wav/text directories must be specified at lines 42 and 48!
%
% inputs: states=number of states per phoneme feat=feature vector index
% (see 'featurevecs' function for details) numfiles=number of training
% files
%
% outputs: phones=phoneme data structure, contains initial model estimates
% and assigned vectors features=feature vector data structure, contains
% feature vectors for each file uniformpath=cell array, indices of
% uniformly segmented phonetic boundaries
%
% calls m-files: featurvecs, uniform

function [phones,features,uniformchanges]=initialize(states,mix,feat)

% read in list of phonemes
phonelist=textread('phones.txt','%s');

% intitialize phoneme data structure, #state substates for each phoneme
% each substate needs label, frames (to contain vectors assigned to
% substate)
%   and nextind (index at which to assign next vector)
for i=1:39
    phones(i).label=phonelist(i);
    for j=1:states
        phones(i).sub(j).nextind=1;
    end
end
% also need silence state!  only one substate
phones(40).label={'SIL'};
phones(40).sub(1).nextind=1;

% begin iteration for each file
for filecount=1:numfiles
    % read in wave file (specify directory of files from root)
    wavname=strcat('c:/project/kellvox/wavdata/S',num2str(filecount),'.wav');
    [y,fs]=wavread(wavname);
    % call featurevecs to compute feature vector for each frame, save in
    % features struct
    features{filecount}=featurevecs(y,feat,fs);

    % read in text file with phonetic transcription, no silence!
    txtname=strcat('c:/project/kellvox/phoneticnosil/S',num2str(filecount),'.txt');
    inphones=textread(txtname,'%s','delimiter',' ');
    lenphon=length(inphones);
    begendsil=zeros(2,1); % indicator of begin/end silence
    if ismember(inphones(1),'SIL')
        begendsil(1)=1;
    end
    if ismember(inphones(end),'SIL')
        begendsil(2)=1;
    end

    % pass features to uniform function, returns structure 'temp',
    %   contains phones
```

```
    [temp,uniformchanges{filecount}]=...
        uniform(features{filecount},lenphon,states,begendsil);

    tempind=1;
    % if begins with silence, assign first block to silence
    if begendsil(1)==1
        startadd=phones(40).sub(1).nextind;
        endadd=startadd+size(temp{1},2)-1;
        phones(40).sub(1).frames(:,startadd:endadd)=temp{1};
        phones(40).sub(1).nextind=endadd+1;
        tempind=2;
    end

    % now add remaining blocks to the appropriate phoneme frames member
    % starting at second block in temp (if first already added to silence!)
    for i=1+begendsil(1):lenphon-begendsil(2)
        % find index of phoneme to add frames to
        phonind=find(ismember(phonelist,inphones(i)));
        % add a block to each of the substates of that phoneme
        for j=1:states
            startadd=phones(phonind).sub(j).nextind;
            endadd=startadd+size(temp{tempind},2)-1;
            phones(phonind).sub(j).frames(:,startadd:endadd)=temp{tempind};
            phones(phonind).sub(j).nextind=endadd+1;
            % increment temp index here so each substate gets the next block
            tempind=tempind+1;
        end
    end

    % if ends with silence, add the last block to silence as well
    if begendsil(2)==1
        startadd=phones(40).sub(1).nextind;
        endadd=startadd+size(temp{tempind},2)-1;
        phones(40).sub(1).frames(:,startadd:endadd)=temp{tempind};
        phones(40).sub(1).nextind=endadd+1;
    end

    % keep all the inputs and structures, clear everything else to avoid
    % any confusion
    keep phonelist states mix feat phones uniformchanges features filecount;
end

% now compute mean and variance of frames assigned to each substate,
%   to be replaced with mixture function call
for i=[1:38 40]
    if i==40
        subs=1;
    else
        subs=states;
    end
    for j=1:subs
        tempvecs=phones(i).sub(j).frames;
        phones(i).sub(j).mean=mean(tempvecs,2);
        phones(i).sub(j).var=var(tempvecs')';
        clear tempvecs;
    end
end
```

A.2 uniform.m

```
% function [temp,changes]=uniform(feats,lengthin,states,begendsil)
%
% returns (close to) uniform length blocks of spectral vectors contained
%   in feats in cell array 'temp', called by 'initialize' function
%
% inputs: feats=block of feature vectors lengthin=length of corresponding
% phonetic transcriptiont states=number of states per phonem
% begendsil=indicator of presence of beginning/ending silence
%
% outputs: temp=cell array containing (close to) uniform blocks of feature
% vectors changes=indices of uniform changes

function [temp,changes]=uniform(feats,lengthin,states,begendsil)

% #states blocks per phone, plus start and end silence as necessary
numblocks=(lengthin-sum(begendsil))*states+sum(begendsil);

featsize=size(feats,2);
uniformlength=featsize/numblocks; % true uniform length, to be rounded!

lendiff=ceil(uniformlength)*numblocks-featsize;
truncinds=randperm(numblocks);
truncinds=truncinds(1:abs(lendiff));

featstart=1;
for i=1:numblocks-1
    blocklength=ceil(uniformlength);
    if ismember(i,truncinds)
        blocklength=blocklength-1;
    end
    featend=featstart+blocklength-1;
    temp{i}=feats(:,featstart:featend);
    featstart=featend+1;
    changes(i)=featstart;
end
featend=featsize;
temp{numblocks}=feats(:,featstart:end);
```

A.3 iteration.m

```
% function
% [phones,phonepath,numpath,scorepath]=iteration(phones,features,states)
%
% function to perform iteration of speech hmm training: resets frames
% assigned to each phones, then computes viterbi path across each
% file/transcription to determine most likely path, and assigns feature
% vectors to phones accordingly.  finally computes updated mean and
% variance estimates.
%
% inputs: phones=phoneme data structure created by initialize function
% features=matrix of feature vectors computed by initialize function
% states=number of states per phone
%
% outputs: phones=updated phoneme data structure (updated mean and var
% estimates) numpath=optimal numeric path through each file
% phonepath=optimal phonetic path through each file scorepath=accumulated
% log likelihood along optimal path

function [phones,numpath,phonepath,scorepath]=iteration(phones,features,states,mix)

numfeats=size(phones(1).sub(1).frames,1); % number of features
phonelist=textread('phones.txt','%s'); % read in list of 39 phones
% reset frames and nextind in phone structure
for i=1:40
    if i==40
        jj=1;
    else
        jj=states;
    end
    for j=1:jj
        phones(i).sub(j).frames=zeros(numfeats,1);
        phones(i).sub(j).nextind=1;
    end
end

% begin iteration for each file
for filecount=1:183
    % read in text file of phonetic transcription, leave the silence in
    % this time!
    textfile=strcat('c:/project/kellvox/phoneticsil/S',num2str(filecount),'.txt');
    phoneticin=textread(textfile,'%s');
    % obtain current files feature vectors from features cell array
    feats=features{filecount};

    % expand phonetic transcription to include #states instances of each
    % phoneme,
    %   still 1 instance of silence
    % also create means and vars matrices, contains mean and var for each
    % state
    ref=1;
    for i=1:length(phoneticin)
        if find(ismember(phoneticin(i),'SIL'))
            phonetic{ref}='SIL';
            means(:,ref)=phones(40).sub(1).mean;
            vars(:,ref)=phones(40).sub(1).var;
            subst(ref)=1;
            ref=ref+1;
        else
            ind=find(ismember(phonelist,phoneticin(i)));
            for j=1:states
                means(:,ref)=phones(ind).sub(j).mean;
                vars(:,ref)=phones(ind).sub(j).var;
                phonetic{ref}=char(phoneticin(i));
                subst(ref)=j;
                ref=ref+1;
            end
        end
    end

    % pass means, vars, feats, and phonetic to viterbi function
    %   return temp cell array with length(phonetic) blocks of vectors, as
    %   well as phonetic path and complete path accumulated score
    [temp,phonepath{filecount},numpath{filecount},scorepath{filecount}]...
        =viterbi(means,vars,feats,phonetic);
    % assign returned feature vectors to appropriate phonemes
    for i=1:length(temp)
        len=size(temp(i).frames,2);
        phn=phonetic(i);
        if find(ismember(phn,'SIL'))
```

```
            nxt=phones(40).sub(1).nextind;
            phones(40).sub(1).frames(:,nxt:nxt+len-1)=temp(i).frames;
            phones(40).sub(1).nextind=nxt+len;
        else
            ind=find(ismember(phonelist,phn));
            nxt=phones(ind).sub(subst(i)).nextind;
            phones(ind).sub(subst(i)).frames(:,nxt:nxt+len-1)=temp(i).frames;
            phones(ind).sub(subst(i)).nextind=nxt+len;
        end
    end
    keep phones phonelist phonepath numpath scorepath states mix features;
end

% now compute mean and variance of frames assigned to each substate,
%   to be replaced with mixture function call
for i=1:40
    if i==40
        subs=1;
    else
        subs=states;
    end
    for j=1:subs
        tempvecs=phones(i).sub(j).frames;
        phones(i).sub(j).mean=mean(tempvecs,2);
        phones(i).sub(j).var=var(tempvecs')';
        clear tempvecs;
    end
end
```

A.4 viterbi.m

```
% function
% [temp,phonepath,numpath,scorepath]=viterbi(means,vars,feats,phonetic)
%
% function uses viterbi dynamic programming algorithm to compute optimal
% path assigning input audio frames to phonetic substates
%
% inputs: means, vars=meand and variance for each phoneme based on current
% model estimate feats=feature vectors of current file being aligned
% phonetic=phonetic transcription of current file
%
% outputs: temp=cell array with blocks of frames to be assigned to
% appropriate phonemes numpath=optimal numeric path phonepath=optimal
% phonetic path scorepath=accumulated log likelihood over optimal path

function [temp,phonepath,numpath,scorepath]=viterbi(means,vars,feats,phonetic)

% for each possible state, compute log likelihood of each frame create
% totalstate x frames matrix of likelihoods, across which
%   optimal path will be computed
len=size(feats,2);
for i=1:size(means,2)
    % repeat this states mean and var, same dimensions as feats
    meantemp=repmat(means(:,i),1,len);
    vartemp=repmat(vars(:,i),1,len);
    % compute log likelihood (of each element in each frame!)
    probmattemp=(-(feats-meantemp).^2)./(2*vartemp)-log(sqrt(vartemp));
    % sum along the columns to give an overall likelihood for each frame
    %   to be in this current state
    probmat(i,:)=sum(probmattemp)./size(feats,1);
end

indsil=find(ismember(phonetic,'SIL'));
% compute delta and psi matrices of accumulated log likelihoods and back
% pointers
[delta,psi]=viterbimat(probmat,indsil);

% use backpointers (psi matrix) to determine optimum path also generate
% numpath and phonepath
[m,n]=size(probmat);
numpath=zeros(n,1);
numpath(n)=m;
scorepath(n)=delta(m,n);
phonepath(n)=phonetic(m);
for i=1:n-1
    numpath(n-i)=psi(numpath(n+1-i),n-i);
    scorepath(n-i)=delta(numpath(n-i),n-i);
    phonepath(n-i)=phonetic(numpath(n-i));
end
% assign cepstral vectors to states
for i=1:m
    ind=find(numpath==i);
    temp(i).frames=feats(:,ind);
end
```

A.5 viterbimat.m

```
% function [delta,psi]=viterbimat(probmat,indsil2)
%
% function computes delta and psi matrices for use in viterbi algorithm

function [delta,psi]=viterbimat(probmat,indsil2)

% m=total states, n=total frames
[m,n]=size(probmat);
delta=zeros(m,n);
psi=zeros(m,n);

delta(1,1)=probmat(1,1);
psi(1,1)=1;

% first 'm' frames
for j=2:m
    delta(1,j)=probmat(1,j)+delta(1,j-1);
    psi(1,j)=1;
    for i=2:j-1
        temp(1)=probmat(i,j)+delta(i,j-1); % came from same state
        temp(2)=probmat(i,j)+delta(i-1,j-1); % came from previous state
        % if this is state after silence, could have skipped silence!
        if sum(ismember(indsil2,i-1)) & i>2
            temp(3)=probmat(i,j)+delta(i-2,j-1);
        end
        [maxprob,ind]=max(temp);
        delta(i,j)=maxprob;
        psi(i,j)=i+1-ind;
        clear temp;
    end
    delta(j,j)=probmat(j,j)+delta(j-1,j-1);
    psi(j,j)=j-1;
end

% middle frames (all states possible!)
for j=m+1:n-m+1
    delta(1,j)=probmat(1,j)+delta(1,j-1);
    psi(1,j)=1;
    for i=2:m
        temp(1)=probmat(i,j)+delta(i,j-1); % came from same state
        temp(2)=probmat(i,j)+delta(i-1,j-1); % came from previous state
        % if this is state after silence, could have skipped silence!
        if sum(ismember(indsil2,i-1)) & i>2
            temp(3)=probmat(i,j)+delta(i-2,j-1);
        end
        [maxprob,ind]=max(temp);
        delta(i,j)=maxprob;
        psi(i,j)=i+1-ind;
        clear temp;
    end
end

% final 'm' frames
lower=max(n-m+2,m+1);
for j=lower:n
    delta(1,j)=probmat(1,j)+delta(1,j-1);
    psi(1,j)=1;
    for i=j-(n-m):m
        temp(1)=probmat(i,j)+delta(i,j-1); % came from same state
        temp(2)=probmat(i,j)+delta(i-1,j-1); % came from previous state
        % if this is state after silence, could have skipped silence!
        diff1=j-i;
        diff2=n-m;
        if sum(ismember(indsil2,i-1)) & i>2 & diff1<diff2
            temp(3)=probmat(i,j)+delta(i-2,j-1);
        end
        [maxprob,ind]=max(temp);
        delta(i,j)=maxprob;
        psi(i,j)=i+1-ind;
        clear temp;
    end
end
```

A.6 wordinds.m

```
% function fileinds=wordinds(states)
% function to determine the start and end state indices of each word in the training files.
% note text file directory must be specified in line 17
%
% inputs:
% states=number of states per phoneme
%
% outputs:
% fileinds=structure containing word and phoneme start and end indices

function fileinds=wordinds(states)
for filecount=1:183
    % read in phonetic transcription with silence
    textfile=strcat('c:/project/kellvox/phoneticsil/S',num2str(filecount),'.txt');
    phonin=textread(textfile,'%s');

    % expand to #states per phone, still one state for silence!
    ind=1;
    phbndind=1;
    for i=1:length(phonin)
        if isequal(phonin{i},'SIL')
            phonst{ind}=phonin{i};
            ind=ind+1;
        else
            for j=1:states
                phonst{ind+j-1}=phonin{i};
            end
            phonboundaries(phbndind,:)=[ind ind+states-1];
            ind=ind+states;
            phbndind=phbndind+1;
        end
    end

    % determine beginning and ending indices of words based on silence separation
    % find silence indices
    indsil=find(ismember(phonst,'SIL'));
    if ~isequal(phonin{1},'SIL')
        indsil=[0 indsil];
    end
    if ~isequal(phonin{end},'SIL')
        indsil=[indsil ind];
    end
    fileinds(filecount).wordbounds=[(indsil(1:end-1)+1)' (indsil(2:end)-1)'];

    % words start after silence, end before
    fileinds(filecount).phonebounds=phonboundaries;
    fileinds(filecount).phonetic=phonst;
    clear phonst phonin phonboundaries;
end
```

A.7 compstarttimes.m

```
% function compstarttimes(files,fileinds)
%
% compute relative word start times and write to text file, used in GUI
% inputs: files=number of test files, fileinds=fileinds structure
% outputs: text file containing arrays of relative start times
% note output text file must be set at line 31!

function compstarttimes(files,fileinds)
starttime=zeros(150,files);
for a=1:files
    path=numpath{a};
    startstates=fileinds(a).wordbounds(:,1);
    for i=1:length(startstates);
        startind=min(find(path==startstates(i)));
        starttime(i,aa)=0.01*startind;
    end
    endstate=fileinds(a).wordbounds(end,2);
    endind=max(find(path==endstate));
    starttime(i+1,aa)=0.01*endind;
end

difftime=zeros(150,files);

for i=1:files
    difftime(1,i)=starttime(1,i);
    for j=2:149
        difftime(j,i)=starttime(j,i)-starttime(j-1,i);
    end
end

fid=fopen('c:/project/thesis/wordtimes.txt','wt');
for i=1:files
    temp=difftime(:,i);
    strtemp=num2str(1000*temp(1));
    for j=2:length(temp)
        if temp(j)>0
            strtemp=strcat(strtemp,',',num2str(1000*temp(j)));
        end
    end
    strtemp=strcat(strtemp,'\n');
    fprintf(fid,strtemp);
    clear strtemp;
end
fclose(fid);
```

Appendix B

Training Data

File	Length (sec)	Lyrics
S1	3.23	SAVE ME FROM SIN
S2	3.01	BUT I'M NOT GETTING IN
S3	3.61	WE'LL MAKE PLANS OVER TIME
S4	3.02	AND I'LL GET ON JUST FINE
S5	3.40	IN A CORIDAL DISPLAY
S6	2.03	WE'LL MEET BUT FOR A
S7	2.54	DAY REACHING HEIGHTS
S8	1.80	UNDEFINED
S9	3.83	I WISH THEY COULD BE MINE
S10	2.21	IF I AM RIGHT FOR
S11	1.85	ANYONE ANYONE
S12	1.46	I AM
S13	2.19	THEN I'LL BE SURE THAT
S14	3.63	EVERYONE FITS IN TO MY PLAN
S15	2.16	BELIEVE IN TIME IN
S16	1.38	ANYONE DO THE
S17	2.72	BEST I CAN
S18	1.90	WE WALK DOWN
S19	1.52	YOUR STREET
S20	2.91	OUR SOULS BENEATH OUR FEET
S21	3.54	SUCH AN ARDENT REPLY
S22	2.84	AND A SLOW AWKWARD GOODBYE
S23	1.60	PROMISE WORTH
S24	1.90	KEEPING TO
S25	3.04	BUT EVENTUALLY WE'LL PROVE

Table B.1: Details of training data files S1-S25

File	Length (sec)	Lyrics
S26	3.74	TAPER OFF FADE AWAY
S27	4.17	WITH WORDS I'LL NEVER SAY
S28	2.96	IF I AM RIGHT FOR ANYONE
S29	2.46	ANYONE I AM
S30	2.25	THEN I'LL BE SURE THAT
S31	3.49	ANYONE FITS IN TO MY PLAN
S32	2.10	BELIEVE IN TIME IN
S33	1.38	EVERYONE DO THE
S34	1.59	BEST I CAN
S35	1.93	OH I CAN
S36	2.18	OH I CAN
S37	2.40	STEADY AND SLOW
S38	3.77	BUT NOW ITS EVERYWHERE I GO
S39	2.60	FLY BY NIGHT AND SEE
S40	3.93	CROSS LINES AND BOUNDARIES ALONE
S41	2.16	UP IN THE AIR
S42	3.61	ISLANDS DOTTING THE BLUE EYE STARE
S43	2.97	FACE FACE DOWN TO THE GROUND ITS
S44	3.54	THE SOUND OF WHAT NEVER WAS THERE
S45	2.98	SLOW MOTION PARADE
S46	2.79	KNOW I CAN CHANGE
S47	2.78	ONE FOOT IN THE GRAVE
S48	3.92	HANDS BEHIND YOUR BACK
S49	2.13	BLACK WHITE AND GREY
S50	4.00	STILL OTHER COLORS WILL NEVER LAY
S51	2.64	KIDS AND CARS AND DREAMS
S52	3.80	FORTUNES LOST TO THE SEA AND DECAY
S53	2.52	OLD TRI STATE
S54	3.41	CARRIED OVER AND UNDER I BREAK
S55	3.18	WAVES THAT CRASH AND BURN TAKE
S56	3.66	TURNS AND CHOKE ON THE WAKE
S57	3.05	SLOW MOTION PARADE
S58	2.82	OH YOU'LL CHANGE
S59	3.10	ONE FOOT IN THE GRAVE YOU'RE
S60	2.07	NOT THERE
S61	2.31	OH NO YOU CAN'T
S62	1.95	BELIEVE
S63	4.17	ALL THESE TURNS AND TRIUMPHS STARE
S64	4.21	OVERSHADOWED BY DEMAND
S65	1.60	COUNTLESS EYES
S66	3.82	AND GREEDY HANDS
S67	4.83	FACE TO FACE WITH YOUR DESIGN
S68	4.15	OH MY GOD JESUS I'M FINE
S69	5.94	CROSS THE COUNTRY IN MY MIND
S70	3.76	DRIVE AN HOUR AFTER DAWN

Table B.2: Details of training data files S26-S70

File	Length (sec)	Lyrics
S71	1.92	THIS MORNING
S72	2.55	FOUR TIMES STRONG
S73	2.88	SEE THE FACES SO
S74	4.72	FAMILIAR IN TOO LONG
S75	3.90	MAKE BROTHERS OUT OF UNKNOWN MEN
S76	4.64	AND THE BOYS WHO FOLLOW THEM
S77	2.78	WHO GROW UP TO BUILD
S78	4.87	COURAGEOUS HEARTS AND HANDS
S79	1.11	AND A
S80	3.20	LONG LOST FRIEND
S81	3.20	LONG LOST FRIEND
S82	2.89	LONG LOST FRIEND YOU WILL
S83	4.41	BE FOUND AGAIN
S84	3.91	FOLLOW SUNS THROUGH WINTER STREETS
S85	4.80	IN THE BITING AIR WE GRIEVE
S86	3.02	AND THE WORDS FLOAT BY
S87	4.56	AND SINK AND DIE AND LEAVE
S88	1.63	SILENT PRAYERS
S89	2.22	GO UNHEARD
S90	4.80	AND A PANIC UNDESERVED
S91	2.80	MAKES TWO TODAY
S92	4.66	AND FOR MORE ALL UNNERVED
S93	1.34	ANOTHER
S94	3.20	LONG LOST FRIEND
S95	3.24	LONG LOST FRIEND
S96	3.00	LONG LOST FRIEND WHO WILL
S97	4.31	BE FOUND AGAIN
S98	3.30	LONG LOST FRIEND
S99	3.31	LONG LOST FRIEND
S100	2.94	LONG LOST FRIEND WHO WILL
S101	4.31	BE FOUND AGAIN
S102	1.93	GRACE
S103	3.19	OF AN ASTRONAUT
S104	3.70	TO NEVER DO WHAT'S NOT
S105	4.08	IN YOUR NATURE
S106	1.93	STAY
S107	3.28	ABOVE THE EARTH ALIVE
S108	3.49	AND EVERYDAY YOU TRY
S109	4.11	TO NOT BE A STRANGER
S110	1.80	OH NO
S111	4.80	DESCEND THE WHOLE WAY DOWN
S112	4.63	TO DRIVE MYSELF TO GROUND
S113	3.93	FREE FALL UNTIL
S114	2.80	I'M SAFE
S115	3.56	TO BE SAFE

Table B.3: Details of training data files S71-S115

File	Length (sec)	Lyrics
S116	1.98	GRACE
S117	3.24	YOU'LL NEVER LIVE THE LIFE
S118	3.60	THAT NARROW PATH THATS RIGHT
S119	4.17	UNDER BLUE SKIES
S120	1.80	TAKE
S121	3.05	A FINGER TIP OR TWO
S122	3.52	A HAND WILL NEVER DO
S123	4.00	NEVER MAKE RIGHT
S124	4.05	AT A PROGRAM SPEED
S125	5.00	BUT NEVER CONQUER NEEDS
S126	3.86	INTELLIGENCE
S127	2.66	ERASED
S128	3.60	TO BE SAFE
S129	1.72	STRENGTH TO KEEP
S130	4.25	BOTH FEET ON THE GROUND
S131	4.30	ISN'T REALLY A FINAL
S132	2.30	PLACE
S133	1.88	ANYWHERE THAT
S134	4.20	COULD BE A CERTAIN HOLD
S135	4.10	IS JUST REALLY AN EMPTY
S136	2.30	SPACE
S137	3.21	OF AN ASTRONAUT
S138	3.50	NEVER DO WHAT'S NOT
S139	4.21	IN YOUR NATURE
S140	3.75	ON OPEN PALMS
S141	2.23	FLOAT IN TO THIS STATE
S142	4.41	NORTH AND SOUTH ARE ONE
S143	3.29	ABOVE THE BLUE BAY
S144	2.96	BRIDGES AND HIGHWAYS
S145	1.59	CARRIED ME
S146	4.41	TOO TIRED WAS I TO RUN
S147	3.42	AND YOUR SUN WAS BRIGHT
S148	3.01	FADING IN TO NIGHT
S149	2.50	BURNING IN THE MORNING
S150	3.40	TO MAKE LIGHT
S151	2.97	CAROLINA
S152	1.77	I HAVE LOVED YOU
S153	1.37	ALL THESE YEARS
S154	2.78	IT WAS A THICK SKIN
S155	3.87	FOR ME TO LIVE IN
S156	3.61	YOU SWALLOWED WHOLE
S157	2.44	FACE THE FACES NOW
S158	4.32	FAMILIAR FAR FROM HOME
S159	3.20	COLD SEASONS FOLLOW
S160	2.90	HOLED OUT AND HOLLOW

Table B.4: Details of training data files S116-S160

File	Length (sec)	Lyrics
S161	1.60	DISAPPEAR
S162	4.30	IN TO WHAT I HAVE KNOWN
S163	3.80	COLD NIGHTS SPENT DOWN
S164	3.10	CLOSER TO THE GROUND
S165	2.63	SCREAMING FROM A ROOF TOP
S166	3.51	TO HEAR THE SOUND
S167	3.15	CAROLINA
S168	1.73	I HAVE LOVED YOU
S169	1.37	ALL THESE YEARS
S170	2.81	IT WAS A THICK SKIN
S171	3.43	FOR ME TO LIVE IN
S172	3.00	CAROLINA
S173	3.37	YOUR HAND HELD ME THROUGH IT ALL
S174	3.38	FOREVER
S175	3.90	AND EVER
S176	3.39	GREENER DREAMS THEY PULL ME BACK
S177	2.89	IN TO THIS STATE I'VE KNOWN
S178	3.54	FROM THE SIDE ITS PARALLAX
S179	1.42	DISAPPEAR
S180	5.22	IN TO WHAT I HAVE GROWN
S181	3.91	CAROLINA
S182	3.65	CAROLINA
S183	3.31	CAROLINA

Table B.5: Details of training data files S161-S183

Appendix C

Perceptual Test Instructions

The purpose of this listener test is to determine the accuracy of several models in aligning lyrics to audio files. The display (depicted below) is designed to simulate a potential graphical display of an mp3 player, and your task is to assess how well the lyrics follow the music. You will play through twenty samples ranging from 15 to 40 seconds each. There will be several with just a single vocalist, and several with simple background (ie acoustic guitar or vocal harmonies). Please play through the files one at a time, and give an alignment rating immediately after each, with 4 being the highest score and 1 the lowest. Consider the ratings as follows:

- 4 - Excellent - Lyrics match audio and you could easily follow and sing along
- 3 - Good - Can still follow lyrics with just a few misalignments
- 2 - Fair - Significant alignment errors, difficult to follow lyrics
- 1 - Failure - Alignment completely off, can't follow lyrics at all

Again, listen to each file once by selecting it from the list and pressing the PLAY button, then record your score (1-4) on the attached sheet. Please record the test number specified on screen at the top of the sheet. Thank you for your assistance with this test.

Appendix D

Test Data

No.	Word	Config. 1	Config. 2	Config. 3	Config. 4	Config. 5
1	GET'	30	200	200	190	240
2	ALONG'	60	0	0	110	80
3	IN'	80	30	290	10	40
4	TIME'	440	30	30	20	0
5	WITH'	1710	10	0	840	300
6	SOME'	40	10	10	40	40
7	PIECE'	10	60	40	10	80
8	OF'	10	10	10	20	10
9	MIND'	20	80	80	70	80
10	THOUGH'	990	30	30	370	40
11	WHAT'	180	60	60	40	40
12	I'	0	0	40	20	10
13	AM'	150	190	190	180	190
14	LEAVING'	100	80	80	0	70
15	IS'	230	30	10	50	170
16	NOT'	20	10	10	10	10
17	WHATS'	0	30	30	10	180
18	LEFT'	30	20	20	40	50
19	BEHIND'	290	50	50	140	30
20	/SIL/	10	0	0	850	430

Table D.1: Word start time errors for four test configurations of file T1

No.	Word	Config. 1	Config. 2	Config. 3	Config. 4	Config. 5
1	MILES'	10	10	40	280	280
2	AWAY'	0	0	0	10	10
3	FROM'	120	80	70	30	70
4	HIS'	0	80	90	40	50
5	COUNTRY'	0	30	0	50	60
6	THIS'	230	10	10	10	50
7	CITY'	120	80	80	0	30
8	IT'	50	30	20	20	60
9	SPEAKS'	10	10	40	80	0
10	SO'	110	30	70	10	80
11	TERSE'	100	60	10	270	10
12	AND'	10	90	100	90	40
13	FRANK'	120	20	10	70	150
14	AND'	10	10	20	290	10
15	FEET'	110	70	10	150	190
16	ONLY'	50	50	40	50	40
17	CARRIED'	10	80	80	90	10
18	BUT'	70	60	50	60	10
19	NEVER'	0	0	0	10	20
20	WERE'	280	180	230	0	20
21	MARRIED'	10	10	0	10	10
22	RIGHT'	40	40	40	10	40
23	TO'	20	0	0	20	30
24	THIS'	50	20	30	20	20
25	TIME'	10	30	30	40	30
26	AND'	10	70	80	0	20
27	PLACE'	70	60	70	240	20
28	AND'	10	10	10	220	10
29	WORDS'	60	30	40	60	10
30	FLOAT'	50	20	20	20	30
31	ON'	50	40	30	40	140
32	LEVITY'	90	40	50	0	10
33	DROWNED'	10	0	20	10	20
34	IN'	0	20	10	10	30
35	HOME'	20	10	30	70	60
36	REMEDIES'	10	70	20	0	50
37	BLUE'	20	30	10	60	30
38	AND'	10	60	50	20	10
39	DINGY'	10	10	40	40	0
40	GREY'	80	30	60	30	80
41	/SIL/	30	10	0	50	150

Table D.2: Word start time errors for four test configurations of file T2

No.	Word	Config. 1	Config. 2	Config. 3	Config. 4	Config. 5
1	DECADES’	30	10	40	90	70
2	ONCE’	20	30	30	60	40
3	STRADDLED’	60	20	20	20	10
4	NOW’	20	70	70	50	60
5	ALL’	110	130	110	110	120
6	WILL’	50	30	10	350	20
7	UNRAVEL’	80	70	80	550	50
8	AND’	50	20	10	10	30
9	SLIP’	0	0	160	90	0
10	BY’	0	0	30	10	0
11	NAKED’	10	10	10	0	80
12	HANDS’	130	130	90	150	140
13	NOT’	0	0	0	560	490
14	SENSELESS’	0	10	20	60	20
15	JUST’	0	10	10	20	10
16	SENTIMENT’	90	90	220	10	10
17	SINCERE’	10	20	10	40	20
18	AND’	90	70	80	800	60
19	RELEVANT’	40	20	20	1030	30
20	LATE’	10	0	0	1230	70
21	AND’	40	110	120	680	30
22	SO’	0	0	0	50	20
23	UNPLANNED’	240	160	230	250	160
24	/SIL/	100	20	10	180	180

Table D.3: Word start time errors for four test configurations of file T3

No.	Word	Config. 1	Config. 2	Config. 3	Config. 4	Config. 5
1	I'	20	70	70	10	60
2	DON"T'	150	80	70	80	80
3	WANT'	280	310	370	320	320
4	TO'	40	250	240	150	570
5	BE'	10	0	10	70	90
6	TOO'	10	20	20	20	0
7	CAREFUL'	70	60	60	150	150
8	I'	0	0	0	10	10
9	DON"T'	110	0	10	70	50
10	CARE'	20	100	90	100	30
11	IF'	20	30	60	20	40
12	YOU"RE'	10	10	10	200	210
13	THE'	630	520	630	870	690
14	ONLY'	1060	910	1020	1080	1040
15	ONE'	790	860	840	880	770
16	WHO"S'	70	10	60	350	10
17	TIRED'	200	30	20	140	170
18	OF'	300	10	20	30	50
19	BEING'	90	90	90	30	40
20	ALONE'	110	140	160	170	60
21	OH'	10	10	10	0	0
22	AND'	0	80	140	80	0
23	NOW'	230	120	170	110	100
24	I"M'	10	0	90	10	10
25	TIRED'	10	0	10	0	60
26	OF'	1380	1770	1960	1040	20
27	BEING'	310	310	350	10	20
28	ALONE'	130	10	120	150	10
29	/SIL/	20	50	340	410	390

Table D.4: Word start time errors for four test configurations of file T4

No.	Word	Config. 1	Config. 2	Config. 3	Config. 4	Config. 5
1	YOU'	5	5	—	285	145
2	TAKE'	0	20	—	10	10
3	MY'	0	50	—	50	170
4	ARM'	20	110	—	110	120
5	WHISPER'	0	0	—	160	190
6	SOFTLY'	10	10	—	10	0
7	YOU'	180	160	—	160	30
8	CONFIDE'	10	30	—	20	40
9	I'	20	20	—	2170	20
10	WANTED'	270	130	—	2440	120
11	YOU'	40	30	—	210	30
12	TRULY'	1770	10	—	0	20
13	YOU'	1180	290	—	300	300
14	NEVER'	2200	10	—	10	10
15	KNEW'	1530	210	—	210	200
16	ME'	1160	10	—	20	10
17	ANYHOW'	1030	40	—	60	60
18	STILL'	10	10	—	1230	10
19	I'	100	90	—	90	90
20	CAN"T'	0	80	—	80	10
21	SHAKE'	10	90	—	390	370
22	THOUGHTS'	180	30	—	460	170
23	OF'	0	10	—	10	0
24	YOU'	40	100	—	100	10
25	ARE'	0	0	—	10	80
26	ALL'	1180	850	—	40	10
27	AROUND'	370	250	—	130	130
28	/SIL/	10	20	—	440	100

Table D.5: Word start time errors for four test configurations of file T5

No.	Word	Config. 1	Config. 2	Config. 3	Config. 4	Config. 5
1	AND’	0	10	—	10	0
2	IN’	40	90	—	90	30
3	THE’	0	0	—	10	0
4	NIGHT’	70	60	—	60	60
5	TIME’	10	0	—	0	0
6	YOU’	10	20	—	100	60
7	WANDER’	40	30	—	30	60
8	AROUND’	190	410	—	470	350
9	NOISE’	0	10	—	300	430
10	LIKE’	260	20	—	240	270
11	A’	110	320	—	80	80
12	GIANT’	10	280	—	20	10
13	BUT’	0	30	—	70	10
14	ONLY’	220	40	—	40	50
15	THE’	280	30	—	30	10
16	SOUND’	20	20	—	10	10
17	STEP’	10	10	—	30	10
18	LIKE’	20	10	—	30	40
19	A’	260	400	—	50	220
20	GHOST’	40	260	—	10	420
21	AND’	50	120	—	120	80
22	BREATHE’	30	50	—	30	30
23	AT’	70	110	—	110	110
24	MOST’	0	0	—	10	10
25	UNTIL’	10	70	—	50	70
26	YOU’	20	30	—	50	20
27	MEET’	40	10	—	20	0
28	THE’	40	10	—	190	70
29	GROUND’	590	30	—	640	370
30	AND’	20	10	—	1210	1310
31	FLATTEN’	70	30	—	540	380
32	OUT’	220	20	—	70	30
33	TO’	0	10	—	10	150
34	BE’	160	130	—	10	30
35	SO’	20	20	—	20	10
36	/SIL/	40	70	—	60	80

Table D.6: Word start time errors for four test configurations of file T6

No.	Word	Config. 1	Config. 2	Config. 3	Config. 4	Config. 5
1	FOR’	10	160	170	160	30
2	THE’	30	10	0	10	20
3	BEST’	150	100	110	20	20
4	I”LL’	60	130	100	120	100
5	PRAY’	40	40	40	60	110
6	AND’	10	10	30	10	320
7	HOPE’	30	30	30	70	120
8	THAT’	10	100	50	130	10
9	SOMEDAY’	20	20	30	50	10
10	THESE’	50	10	10	260	70
11	MEMORIES’	0	50	10	40	420
12	WILL’	0	10	10	0	10
13	KEEP’	10	10	70	80	0
14	ME’	100	30	30	10	130
15	LIVE’	140	250	260	290	150
16	AND’	0	0	0	0	10
17	AWAKE’	10	60	30	70	130
18	/SIL/	10	400	400	540	20

Table D.7: Word start time errors for four test configurations of file T7

No.	Word	Config. 1	Config. 2	Config. 3	Config. 4	Config. 5
1	CUT’	20	10	70	320	20
2	YOU’	110	50	50	50	100
3	DOWN’	20	10	20	0	10
4	PAST’	0	0	170	0	50
5	YOUR’	10	100	80	0	70
6	DEFENSES’	30	10	10	0	20
7	I’	10	10	10	10	10
8	WILL’	80	70	80	70	70
9	MAKE’	0	0	20	0	10
10	YOU’	20	10	30	0	50
11	LOVE’	10	60	70	30	30
12	ME’	40	90	100	20	90
13	SENSE’	30	30	40	20	20
14	LESS’	0	20	20	10	50
15	NOW’	20	40	40	40	30
16	I”M’	30	0	10	20	10
17	TIRED’	10	10	20	10	20
18	OF’	380	0	10	10	10
19	BEING’	50	130	120	30	40
20	ALONE’	140	140	10	170	50

Table D.8: Word start time errors for four test configurations of file T8

References

[1] L. Alboresi and M. Furini, 'Audio-Text Synchronization Inside mp3 Files: A New Approach and Its Implementation,' *IEEE Consumer Communications and Networking Conference*, 2004.

[2] S. Davis and P. Mermelstein, 'Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences,' *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 29, no.4, 1980.

[3] S. Furui, 'Speaker-Independent Word Recognition Using Dynamic Features of Speech Spectrum,' *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 34, no. 1, 1986.

[4] M. Slaney, 'Auditory Toolbox, Version 2,' 1998. [Online]. Available: http://cobweb.ecn.purdue.edu/m̃alcolm/interval/1998-010/.

[5] N. Ahmed, T. Natarajan and K.R. Rao, 'Discrete Cosine Transform,' *IEEE Transactions on Computers*, vol. 23, no.1, 1974.

[6] M. Slaney, 'Auditory Toolbox Technical Report,' Interval Research Corporation, Tech. Rep. #1998-010, 1998.

[7] T. Rossing, *The Science of Sound*, New York: Addison-Wesley, 1990.

[8] L.R. Rabiner, 'A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition,' *Proceedings of the IEEE*, vol. 77, no. 2, 1989.

[9] *CMU Pronouncing Dictionary*, Carnegie Mellon University. [Online]. Available: http://www.speech.cs.cmu.edu/cgi-bin/cmudict.

[10] L.R. Rabiner and B. Juang, *Fundamentals of Speech Recognition*, Upper Saddle River, NJ: Prentice Hall, 1993.

[11] A. Berenzweig and D. Ellis, 'Locating Singing Voice Segments Within Music Signals,' *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2001.