LISTA D4.2 – Sound class and intonation structure extraction

Vassilis Tsiaras, Vincent Aubanel, Varvara Kandia, and Yannis Stylianou

April 21, 2012

1 Introduction

It is known that gross sound classes (e.g., vowels, fricatives, nasals) are treated differently with respect to the degree of modification that humans apply to maintain intelligibility under quiet and noisy conditions. Therefore, speech modifications should not be applied monolithically to the signal, but instead should be context-sensitive, dependent on the units and structures in the signal.
This deliverable will focus on analysis of both prosodic and segmental properties of speech. More specifically, one goal is to develop a detector of sound classes. Using mel frequency cepstrum coefficients (MFCC) and their derivatives, a broad phonetic class detector has been developed in Matlab. Training and testing of the detector was performed using the TIMIT database. A second goal is to build a hierarchical description of speech rhythm and intonation based on prosodic structure analysis tools. For this purpose, the rhythmogram representation has been employed as well as a recent approach to intonation extraction (prosogram). Similar to the broad phonetic class detector, the hierarchical description of speech has been developed in Matlab using some utilities of R for visualization purposes. The results from both segmental and prosodic analyses will be helpful in determining the modification strategy given a segment of speech.

2 Sound Class Extraction

The context for this work is the possibility of detecting in an automatic way phonetically important events in continuous speech signals which contribute in maximizing the intelligibility of the heard speech in adverse listening conditions. For instance, studies have shown that selective reinforcement of bursts and vocalic onsets can provide significant improvements to the intelligibility of the subsequently degraded speech signal, even for the same overall signal-to-noise ratio. These phonetically important events are referred to as potential enhancement regions (PER) in running speech [1]. The detection of PER is related with the recognition of broad phonetic class (BPC) which is also important in a wide variety of contexts like speech recognition. BPC recognition may actually seen as the first step for detecting PERs. The present study assumes that the speech signal has been generated by a sequence of five types of BPCs: vowels/semivowels, nasals/flaps, fricatives, stops and silence. The assignment of individual phonemes to BPCs follows the categorization used by Sainath et al. [2]. For an alternative categorization see the work of Scanlon et al. [3]. Vowels, nasals, fricatives and silence are modeled with a three-state hidden Markov model (HMM), while for stops, the corresponding MFCCs are modeled by a single state HMM since their duration is usually short. HMMs [4] have a rich theoretical foundation and have been extensively applied to a wide variety of statistical pattern recognition tasks in speech processing and elsewhere. The main motivation for using HMMs is that they provide a structured, flexible, computationally tractable model of describing sequence data. The probability density function of the observed spectral feature vectors given a state of a HMM, is modeled with Gaussian mixtures (GMM) [5, 6]. There are alternative methods that produce estimates of the conditional probabilities which have been shown to offer improvements over GMM, in predicting the label of the hidden state [7, 8]. However, these methods

are computationally expensive relative to evaluating a set of Gaussian mixtures and their estimates of the observation conditional probabilities cannot easily be compared across frames since they are based on the decision scores from the classifiers themselves. For these reasons, their use in HMM software systems is limited.
A major property of conventional HMM-based formulations is their use of homogeneous, frame-based feature vectors. Such a representation may not be able to adequately capture phonetic boundaries. However the combination of the BPC recognition system with simple strategies of segmentation produces in an automatic way the potential enhancement regions (PER) in running speech. Therefore, the first step for extracting the sound class is the segmentation of the signal using a measurement of signal stationarity based on energy smoothness and degree of spectral prediction [9]. The segmentation is conducted without any explicit phonetic knowledge. The second step includes segmentation enhancement and segments identification (frame labeling); the BPC of each segment is determined by the majority of the labels of the corresponding frames.
The above techniques have been implemented in Matlab and the corresponding toolbox is described in Section 3.

2.1 Feature Vectors

To obtain steady-state measurements of the spectra from continuous speech, short-time spectral analysis is performed. First, the speech is divided into frames by a 16-msec window progressing at a 8-msec frame rate. Next, the average energy, 13 Mel-Frequency Cepstrum Coefficients (MFCC) as well as their first derivatives are extracted from the speech frames. Finally, the 28-dimensional feature vector is a reduced representation of the corresponding speech frame. Let T be the number of frames and $O = \{o_1, o_2, \ldots, o_T\}$ be the feature vectors extracted from the continuous speech. The sequence O is called the observation sequence and each o_i, $i \in \{1, \ldots, T\}$, is a D-dimensional feature vector, where D is equal to 28.

2.2 Classification

Given a set of acoustic observations $O = \{o_1, o_2, \ldots, o_T\}$ associated with a speech waveform, the goal of the classification is to find the sequence of BPC sub-phoneme units $Q = \{q_1, \ldots, q_T\}$ that most likely produced the given observation sequence. In other words, we want to maximize $Q^* = \arg\max_Q P(O|Q)$. Here, each sub-phoneme unit is represented as a state from an HMM and the conditional probability of an observation given a hidden state is modeled by Gaussian mixtures.

2.3 Hidden Markov Models

Hidden Markov models are especially known for their application in temporal pattern recognition such as speech and are characterized by the following:
1) N, the number of states in the model. In this work each vowel/semivowels, nasal, fricative and silence is modeled by a three-state HMM and each stop with one-state HMM (see Fig. 1). These five HMMs are connected into a larger HMM with $N = 13$ states (see Fig. 2). The individual states are denoted as $S = \{s_1, \ldots, s_N\}$, and the state at time t as q_t.
2) M, the number of distinct observation symbols per state. We denote the individual symbols as $V = \{v_1, v_2, \ldots, v_M\}$. In this work M is equal to 5 since the observed symbols are one of Vowels/Semivowels (VS), Nasals/Flaps (NF), Fricatives (F), Stops (St) and Closures/Silence (CS) (i.e., $V =$ {VS, NF, F, St, CS}.
3) The state transition probability distribution $A = \{a_{ij}\}$, where

$$a_{ij} = P(q_{t+1} = s_j | q_t = s_i), \quad 1 \leq i, j, \leq N \tag{1}$$

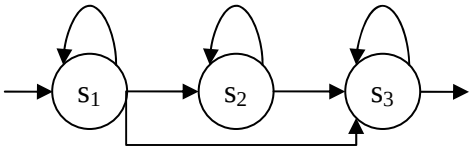

Figure 1: Transitions between states in the vowels/semivowels HMM.

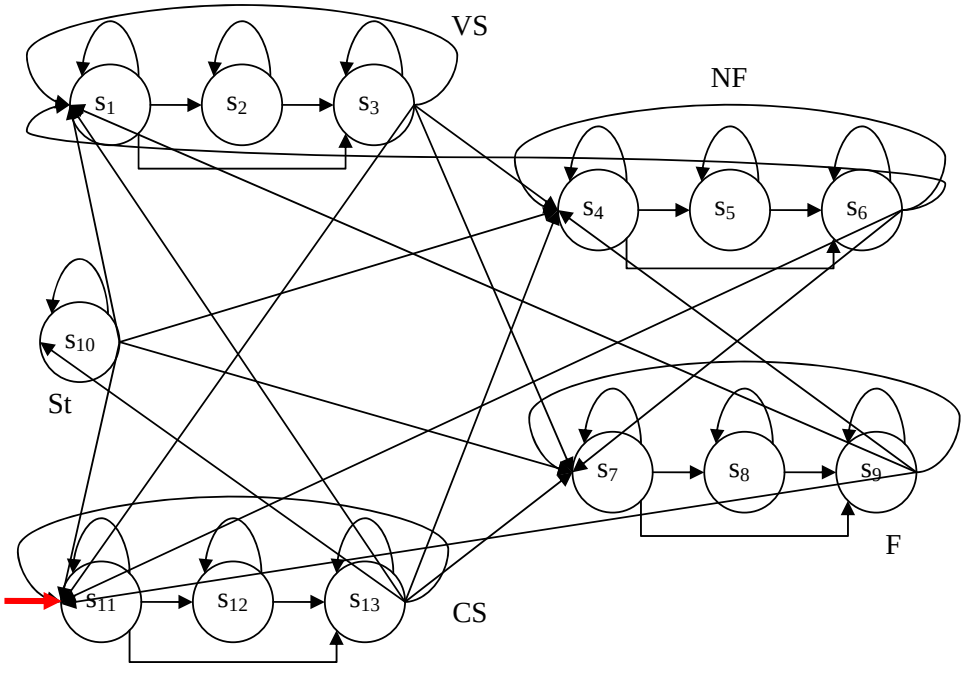

Figure 2: Transitions between states in the larger HMM. Observed symbols are: Vowels/Semivowels (VS), Nasals/Flaps (NF), Fricatives (F), Stops (St) and Closures/Silence (CS).

Table 1: *Relative frequencies of state transitions in train TIMIT database.*

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}	s_{11}	s_{12}	s_{13}
s_1	0.44	0.55	0.01	0	0	0	0	0	0	0	0	0	0
s_2	0	0.85	0.15	0	0	0	0	0	0	0	0	0	0
s_3	0.20	0	0.44	0.1	0	0	0.1	0	0	0	0.15	0	0
s_4	0	0	0	0.32	0.6	0.08	0	0	0	0	0	0	0
s_5	0	0	0	0	0.67	0.3	0	0	0	0	0	0	0
s_6	0.39	0	0	0.01	0	0.32	0.08	0	0	0	0.18	0	0
s_7	0	0	0	0	0	0	0.45	0.53	0.02	0	0	0	0
s_8	0	0	0	0	0	0	0	0.85	0.16	0	0	0	0
s_9	0.36	0	0	0.01	0	0	0.03	0	0.45	0	0.15	0	0
s_{10}	0.2	0	0	0.01	0	0	0.02	0	0	0.76	0.02	0	0
s_{11}	0	0	0	0	0	0	0	0	0	0	0.4	0.57	0.03
s_{12}	0	0	0	0	0	0	0	0	0	0	0	0.87	0.13
s_{13}	0.1	0	0	0.03	0	0	0.1	0	0	0.32	0.03	0	0.43

The relative frequencies of the state transition in the train TIMIT database are estimates of the state transition probabilities. These relative frequencies are shown in Table 1 rounded to 2 decimal digits.
4) The observation symbol probability distribution in state j, $B = \{b_j(k)\}$, where:

$$b_j(k) = P(v_k \text{ at } t | q_t = s_j), \quad 1 \leq j \leq N, \quad 1 \leq k \leq M \tag{2}$$

5) The initial state distribution $\pi = \{\pi_i\}$ where:

$$\pi_i = P(q_1 = s_i), \quad 1 \leq i \leq N \tag{3}$$

All the sentences of TIMIT database start with a silence. Therefore $\pi_{11} = 1$ and $\pi_i = 0$, for $i \neq 11$.
In the following we use the notation

$$\lambda = \{A, B, \pi\} \tag{4}$$

to indicate the complete parameter set of the model.

2.4 Viterbi Algorithm

To find the single best state sequence, $Q = \{q_1, q_2, \ldots, q_T\}$, for the given observation sequence $O = \{o_1, o_2, \ldots, o_T\}$, we define the quantity

$$\delta_t(i) = \max_{q_1, \ldots, q_{t-1}} P(q_1, q_2, \ldots, q_t = i, o_1, o_2, \ldots, o_t \mid \lambda) \tag{5}$$

i.e., $\delta_t(i)$ is the best score (highest probability) along a single path, at time t, which accounts for the first t observations and ends in state s_i. By induction we have

$$\delta_{t+1}(j) = (\max_i \delta_t(i) a_{ij}) b_j(o_{t+1}) \tag{6}$$

To retrieve the state sequence, we need to keep track of the argument which maximized (6), for each t and j. We do this via the array $\psi_t(j)$. The complete procedure for finding the best sequence can now be stated as follows:
1) Initialization:

$$\delta_1(i) = \pi_i b_i(o_1) \quad 1 \leq i \leq N \tag{7}$$

$$\psi_1(i) = 0 \tag{8}$$

2) Recursion:

$$\delta_t(j) = \max_{1 \leq i \leq N} (\delta_{t-1}(i) a_{ij}) b_j(o_t), \quad 2 \leq t \leq T, \quad 1 \leq j \leq N \tag{9}$$

$$\psi_t(j) = \text{argmax}_{1 \leq i \leq N} (\delta_{t-1}(i) a_{ij}), \quad 2 \leq t \leq T, \quad 1 \leq j \leq N \tag{10}$$

3) Termination:

$$P^* = \max_{1 \le i \le N}(\delta_T(i)) \tag{11}$$

$$q_T^* = \text{argmax}_{1 \le i \le N}(\delta_T(i)) \tag{12}$$

4) Path (state sequence) backtracking:

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \quad t = T-1, T-2, \ldots, 1 \tag{13}$$

2.5 Gaussian Mixture Models

The HMM require the estimate of the observation symbol probability distribution. In this work the conditional probability of an observed symbol given a state of HMM is modeled with GMMs. A Gaussian mixture model is a weighted sum of K component Gaussian densities as given by the equation,

$$p(o|\lambda) = \sum_{i=1}^{K} w_i g(o|\mu_i, \Sigma_i) \tag{14}$$

where o is a D-dimensional continuous-valued data or features vector, w_i, $i = 1, \ldots, K$, are the mixture weights, and $g(o|\mu_i, \Sigma_i)$, $i = 1, \ldots, K$, are the component Gaussian densities. Each component density is a D-variate Gaussian function of the form,

$$g(o|\mu_i, \Sigma_i) = \frac{1}{(2\pi)^{\frac{D}{2}} |\Sigma_i|^{\frac{1}{2}}} \exp\{-\frac{1}{2}(o-\mu_i)^T \Sigma_i^{-1}(o-\mu_i)\} \tag{15}$$

with mean vector μ_i and covariance matrix Σ_i. The mixture weights satisfy the constraint that $\sum_{i=1}^{K} w_i = 1$. The complete Gaussian mixture model is parameterized by the mean vectors, covariance matrices and mixture weights from all component densities. These parameters are collectively represented by the notation,

$$\theta = \{\theta_1, \ldots, \theta_K\} \quad \text{where} \quad \theta_i = \{w_i, \mu_i, \Sigma_i\} \tag{16}$$

In this work K was set to 20 and diagonal covariance matrices were used.

2.6 Maximum Likelihood Parameter Estimation

There are several techniques available for estimating the parameters of a GMM. One popular method is the maximum likelihood (ML) estimation. The aim of ML estimation is to find model parameters which maximize the likelihood of the GMM, given the training data. For a sequence of T' training vectors $O = \{o_1, o_2, \ldots, o_{T'}\}$ the GMM likelihood can be written as $p(O|\theta) = \prod_{t=1}^{T'} p(o_t|\theta)$. Note that in general the number of training vectors T' of a GMM is different from the number of observations T of the HMM. The ML parameter estimates are obtained iteratively using a special case of the expectation-maximization (EM) algorithm [10]. The basic idea of the EM algorithm is, beginning with an initial model θ, to estimate a new model $\bar{\theta}$, such that $p(o|\bar{\theta}) \ge p(o|\theta)$. The new model then becomes the initial model for the next iteration and the process is repeated until some convergence is reached. On each EM iteration, the following re-estimation formulas are used which guarantee a monotonic increase in the model's likelihood value:
Mixture weights:

$$\bar{p}_i = \frac{1}{T'} \sum_{t=1}^{T'} p(i|o_t, \theta) \tag{17}$$

Means:

$$\bar{\mu}_i = \frac{\sum_{t=1}^{T'} p(i|o_t, \theta) o_t}{\sum_{t=1}^{T'} p(i|o_t, \theta)} \tag{18}$$

Variances (diagonal covariance):

$$\bar{\sigma}_{ij}^2 = \frac{\sum_{t=1}^{T'} p(i|o_t, \theta) o_{tj}^2}{\sum_{t=1}^{T'} p(i|o_t, \theta)} - \bar{\mu}_{ij}^2, \quad j = 1, \ldots, D \tag{19}$$

where $\bar{\sigma}_{ij}$, o_{tj} and $\bar{\mu}_{ij}$ denote the jth elements of the vectors $\bar{\sigma}_i$, o_t and $\bar{\mu}_i$ respectively.
The a posteriori probability for component i is given by

$$P(i|o_t, \theta) = \frac{w_i g(o_t|\mu_i, \Sigma_i)}{\sum_{k=1}^{K} w_k g(o_t|\mu_k, \Sigma_k)} \tag{20}$$

For the initialization, the training data of each GMM was clustered, using the BIRCH algorithm [11], into K classes. The class means and variances then served as the initial model for EM training. BIRCH clustering algorithm runs faster than LBG and K-means algorithms since it performs one scan of the data and according to our experiments its results are comparable to those of LBG and K-means in terms of quality and superior in terms of stability.

2.7 Segmentation

A number of studies have shown that transitions in speech are important for speech intelligibility in both normal and hearing impaired subjects and speech perception in more demanding communication situations such as speech in noise [12] [13] [14] [15]. Most of the works investigated the detection of speech transient information are related to speech coding and they do not address the relation of the transient information to speech intelligibility and speech modifications.
During the first year of the project we re-implemented and partially tested a time-frequency criterion which was initially used to control in an automatic way the time-scale modification of speech [9]. This tool is used as a first segmenter of speech, before identifying the speech frames. The method combines two criteria; one is defined in time domain and the other in the frequency domain.
The time domain criterion(C_n^1) is defined as the normalized transition rate or the RMS values computed from two contiguous frames $n-1$ and n:

$$C_n^1 = \frac{|E_n - E_{n-1}|}{E_n + E_{n-1}} \tag{21}$$

where E_n is the RMS value of the frame centered at sample n.
The frequency domain criterion is based on the gradient of the regression line for the evolution of Line Spectrum Frequencies (LSFs) over time. At instant n, within the time interval $[n \pm M]$, the gradient for the LSF, y_l, is given by:

$$g_n^l = \frac{\sum_{m=-M}^{M} m\, y_l(n+m)}{\sum_{m=-M}^{M} m^2} \tag{22}$$

for $l = 1, \cdots, P$ where P is the number of estimated LSFs. Then, the measurement of transition rate is given by:

$$m_n = \sum_{l=1}^{P} (g_n^l)^2 \tag{23}$$

Finally, a criterion combining the time and frequency criteria is defined as:

$$C_n^3 = \frac{2}{1 + e^{-\beta\, m_n - \alpha\, C_n^1}} - 1 \tag{24}$$

where β and α are determined by normalizing the criterion between 0 and 1.
Determining a function $f(t)$ as

$$f(t) = \begin{cases} \sim 0 & \text{when a speech segment is stationary} \\ \sim 1 & \text{when a speech segment is non-stationary} \end{cases} \tag{25}$$

then the stationarity sensitive time scale factor $\bar{b}$ can be controlled by $f(t)$ in the following way:

$$\bar{b} = 1 + d(t)\, b \tag{26}$$

where $d(t) = 1 - f(t)$ and b is the initial time-scale modification factor. Conducted listening tests using 23 listeners showed that in 90% of responses listeners preferred the stationarity sensitive time scaled modified signal [9].

3 Sound Class Detector: Toolbox User Guide

The Sound Class detector has a training and a testing phase. Next, we provide information of the process. However, at this stage the software release will concern the direct segmentation of a speech signal to broad phonetic classes using pre-determined structures as these were constructed during training. Training and testing was performed on the TIMIT database.

3.1 External Libraries

The software toolbox consists of a set of Matlab functions together with two external libraries, which are:

1. The VOICEBOX (Speech Processing Toolbox for MATLAB) [1], which is mainly written by Mike Brookes and is distributed under the GNU Public License.

2. The Hidden Markov Model (HMM) Toolbox for Matlab (HMMall) [2], which is written by Kevin Murphy and is distributed under the MIT License.

The licenses of both external libraries permit redistribution and use in source and binary forms, with or without modifications. Note that a modification has been made in file *melcepst.m* of the the included version of VOICEBOX. The change concerns the handling of "short" frames (i.e., the frames whose length is smaller than the chosen frame length). These frames appear when a segment itself is "short". For example, in TIMIT train database, there are 4104 phoneme segments (out of total 177080 phoneme segments) with length less than 256 samples.

3.2 Matlab Files and Functions

The Matlab files used for training and testing, are:

- installExternalPackages.m
  The corresponding function installs the external packages.

[1] http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html
[2] http://www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html

- setToolboxPaths.m
  The corresponding function sets the matlab paths of the external libraries and of the functions that are in directories gmm and sftr. sftr corresponds to the first stage segmentation [9].

- calculateSegmentFeatures.m
  The corresponding function calculates the feature vectors of a continuous speech signal y.

- calculateTrainFeatures.m
  The corresponding function calculates feature vectors from the whole database. Either the train or the test database. These feature vectors are used to train the classifiers and to measure the accuracy of each classifier

- constructModelGMM.m
  The corresponding function constructs the Gaussian Mixture Models

- estimateHMMtransitionProbabilities.m
  The corresponding function estimates the transition probabilities for a 13 states HMM model when it is called with isBPC = true and for a 171 states HMM when it is called with isBPC = false. Each phoneme of vowels, nasals, fricatives, and silence is modeled with a 3 states HMM. Each phoneme of stops is modeled with a single state HMM.

- getPhonemeClasses.m
  The corresponding function returns the phonemes in each broad phonetic class (BPC).

- probabilityGMM.m
  The corresponding function calculates estimates of the probabilities that each one of T feature vectors belongs to one of Q states. Each state is modeled with a Gaussian Mixture Model.

- segmentSignal.m
  The corresponding function segments a speech signal using the sftr function. See the sftr documentation.

- plotSignalAndLabeling.m
  The corresponding function plots the signal and the true as well as the calculated segments and labeling.

- labelSegments.m
  The corresponding function segments a speech signal and labels each segment as vowel, nasal, fricative, stop or silence.

- testLabelSegments.m
  The corresponding function tests the labelSegments function.

- testGMMaccuracy.m
  The corresponding function tests the accuracy of the classification using GMM models.

3.3 Paths and Directories

Also there are three directories. The gmm, the sftr and the output directories.

- The gmm directory contains the functions that implement the GMM models.
  birch.h and birch.c implement the BIRCH clustering algorithm.
  GMM.m: The corresponding function implements the GMM models.

- The sftr directory contains the functions that implement the sftr function which is used for segmenting the signal [9].
- The output directory is used for storing intermediate results as well as the models of the classifiers. Output contains two subdirectories: train and test.
  In the train subdirectory are stored the results that are from the train speech database.
  In the test subdirectory are stored the results that are based on the test database.

3.4 General Usage

The main function is labelSegments. Run it by typing
segments = labelSegments(y, fs)
in the Matlab command line. Examples of its usage exists in file testLabelSegments.m

predictedSegmentLabels = labelSegments(y, fs, frameLength, θ, m, s, π, A, isBPC)

Input arguments:

- y: the speech signal
- fs: sampling frequency
- frameLength: The length of the frame e.g., 256 samples. Note that models for 256 and 400 samples windows exist in the output directory
- θ: GMM model parameters
- m: average value of the train features
- s: standard deviation of the train features
- π: HMM prior probability
- A: HMM transition probability
- isBPC: Boolean. If it is true then the broad phonetic classes model with 13 states is used. If it is false then the individual phoneme classes model with 171 states is used.

Note that when the arguments model, θ, m, s, π, and A are omitted then their values are loaded from the corresponding files in the output/train directory. Similarly, when the argument frameLength and isBPC are omitted then default values are assigned to them.

Output arguments:

- predictedSegmentLabels: ($T \times 3$). For the t-th segment keeps its label $\in \{1, 2, 3, 4, 5\}$ = {vowels, nasals, fricatives, stops, silence} in predictedSegmentLabels(t, 1) its start position (in samples) in predictedSegmentLabels(t, 2) and its end position in predictedSegmentLabels(t, 3)

Table 2: *Confusion matrix of GMM classification.*

	VS	NF	F	St	CS
VS	258803	22575	3688	729	4065
NF	2235	30698	583	212	1405
F	1386	2869	80238	5933	3488
St	409	889	6534	19077	1509
CS	2925	7620	4770	2843	133771

3.5 Some Results Produced with the Toolkit

Figures 3 and 4 show the speech wave form of a sentence of TIMIT database. Also they show the ground truth segmentation and labeling on the horizontal line with vertical coordinate equal to 0.45. The horizontal line with vertical coordinate equal to 0.4 is the segmentation and labeling produced by the labelSegments function. The symbols with vertical coordinate equal to 0.3 denote the labels of the GMM classification. The symbols with vertical coordinates equal to 0.35 denote the labels of HMM sequence prediction. Black color corresponds to vowels/semivowels, red color corresponds to nasals/flaps, blue color corresponds to fricatives, green color correspond to stops and yellow color corresponds to closures/silence. Table 2 shows the confusion matrix of the prediction of the hidden states using only the

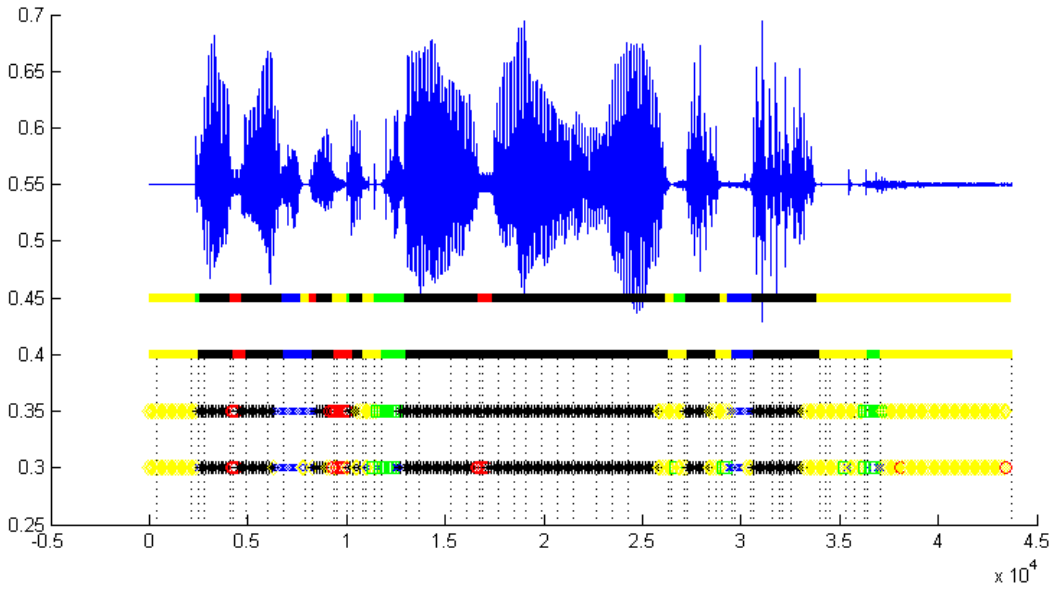

Figure 3: Results of segmentation.

GMMs. The accuracy of the prediction is 0.8721. Table 3 shows the confusion matrix of the prediction of the hidden states using HMMs. The accuracy of the prediction is 0.9405. Note the improvement of the accuracy when the predictions of GMMs are fed into Viterbi algorithm to recover an optimal state sequence.

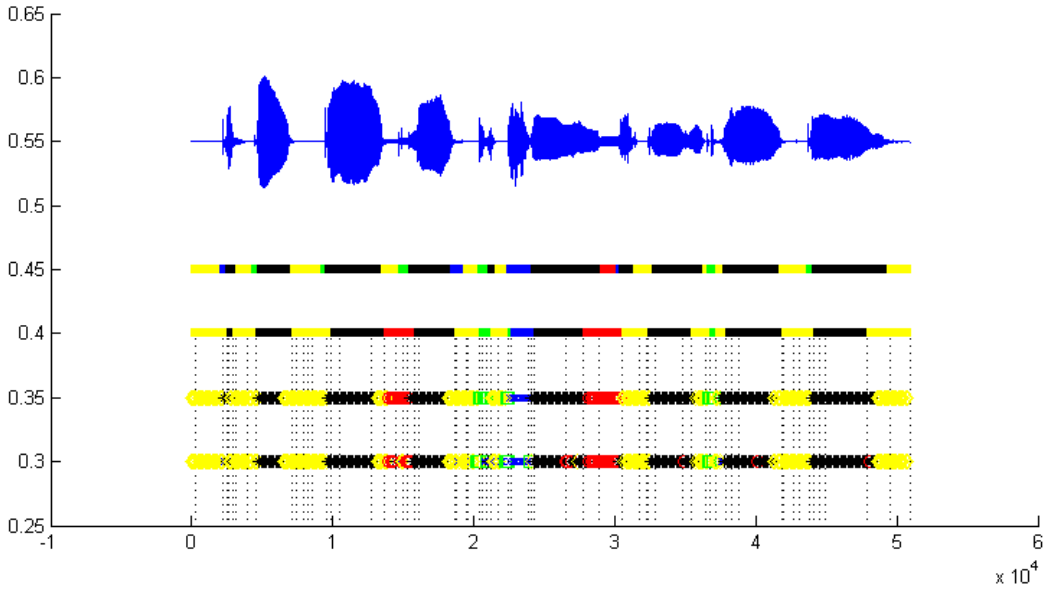

Figure 4: Results of segmentation.

Table 3: *Confusion matrix of HMM classification.*

	VS	NF	F	St	CS
VS	277211	10419	691	248	1291
NF	892	33681	69	48	443
F	924	1891	85581	3294	2224
St	125	311	1607	25528	847
CS	2410	3824	1742	2361	141592

3.6 Usage of the released version, BPC

In this section we will provide details for the released version which will be referred to as Broad Phonetic Classifier (BPC). This will allow a user to directly start segmenting speech signals without being necessary to pass through the training stage. In other words, model parameters estimated on the TIMIT database will be used for any input signal.
As previously, the main function is labelSegments. Simply it runs as
segments = labelSegments(FileName)
in the Matlab command line.

Input arguments:

- FileName: is the name of a speech signal. This is supposed to be a wav file, located at the directory Signals. The name of the file is provided without its extension, i.e., .wav

The analysis is performed with a window of 256 samples, using HMM with 13 states. As it was mentioned before, the released code loads a series of mat files (constructed during the training stage on the TIMIT) for the GMM model parameters, the average value of the train features, the standard deviation of the train features, the HMM prior probability and the HMM transition probability.
Output arguments:

- predictedSegmentLabels, as before
- .lab file: it creates a transcription file which compatible with wavesurfer. The transcription (text) file is saved to the same directory of the source speech file with the extension .lab
- A plot with the speech file using different colors per acoustic class; black for vowels, red for nasals, blue for fricatives, green for stops, and yellow for silences. An output of the tool based on the above distribution of colors is shown in Fig. 5. Note that this signal is not part of the TIMIT database.

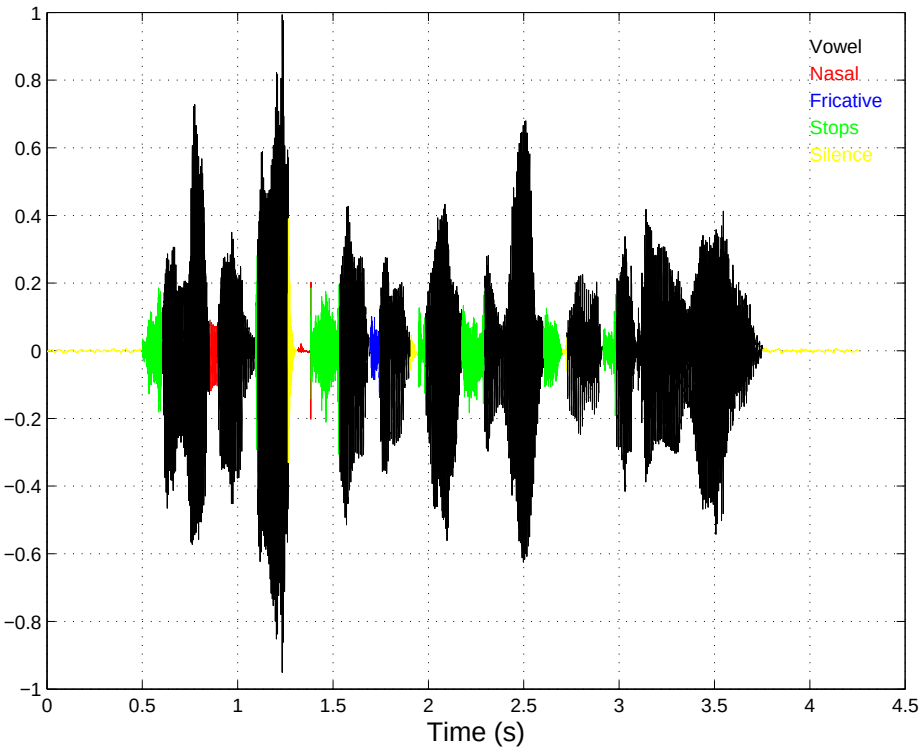

Figure 5: Results from Broad Phonetic Classifier.

In the current release there are two c files which require to be compiled as mex files. In the released code, executables have been prepared for a windows system at 32 bits.

4 Intonation structure extraction

The aim of this tool is to automatically extract aspects of intonational structure of speech, to allow temporal modification with the view of improving intelligibility in different masking conditions.
The software presented in this document identifies relevant locations in the speech signal for temporal modification. It draws on two complementary approaches of prosodic modeling of speech, the rhythmogram for a hierarchical modeling of speech rhythm and the prosogram which provides a perceptually motivated stylisation of pitch contour. The locations identified as best candidate for temporal modification are then used for elongating the signal and inserting pauses.
Section 4.1 describes the set of tools developed to extract rhythmic structure and stylised pitch contour, and the parameter selection that gives best results. An application of this information to modify the temporal structure of speech is reported in Section 4.2. The list of relevant software files necessary to run the example is given in Section 4.3.

4.1 Intonation extraction

4.1.1 Rhythmic structure extraction

The rhythmogram was developed by Todd and coll. [16, 17, 18], and aims at modeling the representation of rhythm in the brain. It operates as a multiscale auditory-based filtering process, which produces a characterisation of speech in terms of the relative prominences at different time scales. It consists of three stages: auditory periphery modeling, multiscale filtering and peak extraction. It has proven useful for modeling rhythmic characteristics of music [17] and poetry [17, 18], differentiation of languages in terms of proportion of voicing ratio [18] and detection of syllable prominences in French and Italian [19].
The multiscale peak detection operated by the rhyhtmogram provides the locations of relative energy prominences at different time scales. Specifically, the iterative bottom-up procedure of peak tracking allows to relate the globally salient prominences (i.e. at a larger time scale) to the more fine-grained local energy bursts in the signal. A contiguous peaks detection was done to link the global prominences with the local positions in the signal. A similar procedure is applied to the valleys given by the rhythmogram analysis, giving the locations of the silences at different time scales. Parameter values adjustment is presented in section 4.1.3.
The functions dealing with rhythmogram extraction are located in the `rhythm` directory of the software. Rhythmogram extraction is performed by `comp_rhythm.m`, while `reshape_rhythm.m` groups contiguous peaks into stems (see also Table 5).

4.1.2 Stylised pitch contour extraction

The prosogram is a set of praat scripts which provide a stylisation of pitch contours of a speech signal from the raw fundamental frequency ($F0$) signal. It was developed by Piet Mertens [20, 21] and has the particular property of giving a representation of pitch variation which is perceptually motivated. Specifically, it abstracts from the raw $F0$ curve by transforming the variations into segments, which slopes are constrained by values obtained from perceptual studies. Whereas it doesn't model higher level of prosodic hierarchy (such as pitch accents), it had been argued to provide a segmentation of speech into syllables.
The main function which perform stylised pitch contour extraction is `get_prosogram.sh` at the root folder of the software directory.

4.1.3 Parameter selection

Temporal modification of speech is done by inserting elongation and silences by tracking down the highest peaks (resp. valleys) in the rhythmogram to lower levels and recording the corresponding time locations in the signal. Targeting the highest peaks amounts to selecting the most prominent portions of the signal (less prominent in the case of valleys), and selecting the corresponding time location at the bottom of the stem ensures quality segmentation.
The rhythmogram analysis is mainly controlled by 3 parameters, determining the number of filters (`NUMFILTERS`) and their range of widths (from `MIN_WIDTH` to `MAX_WIDTH`). Increasing `MAX_WIDTH` allows to find higher-level prominences (e.g. "sentence" prominences). For the set of sentences used here, a value of 500 ms typically yields one highest peak. Below this value, peaks are clipped. `MIN_WIDTH` has been set to 10 ms, a value chosen to yield good precision for segmentation.
Two thresholds for filter widths have been set: a **detection threshold** below which peaks/valleys are ignored, and a **segmentation threshold**, a value at which the time location is recorded. They differ for elongations and silences, as these phenomena are different in speech: detecting lower valleys doesn't often yield to a silence at a finer detail level: it's often related to an attenuation, whereas a peak almost always lead to a local prominence, such as a voiced vowel. The detection thresholds were set to 100 ms for peaks, and 200 ms for valleys. Segmentation thresholds also give different results for selecting the most appropriate points for elongation and silence insertion: local silent portion wich allow pauses to be inserted are best attained with lowest values of filter width, whereas these levels sometimes lead to local energy peaks which don't represent well the best mid-point of the prominence. Segmentation thresholds were set to 20 ms for peaks and 10 ms for valleys.

4.2 Application

This section describes the application of the code to a set of sound files, with extraction of rhythm and stylised pitch contours, and visual representation of the results. Insertion of elongations and pauses is then applied at the locations pointed by the highest stems of peaks and valleys respectively. Three profiles of parameter values for the rhythmogram are used to exemplify rhythmic modification of speech at different levels. The profiles vary the maximum number of elongations and silences, and their duration, as shown in Table 4.

	parameter	profile 1	profile 2	profile 3
elongations	max number	3	4	5
	duration (ms)	50	100	200
silences	max number	1	2	3
	duration (ms)	50	100	200

Table 4: Profiles used for elongations and silences insertions

Figure 6 shows the results of rhythmogram analysis and stylised pitch countour extraction for the first one of the five sound files used in this example. Overlaid vertical lines show the locations at which elongations and silences were inserted, for the three profiles.
Plots of the results such as shown in Figure 6 is generated with a set of `R` scripts, which main function is `plot.R`. Resulting sound files are joined in folder `sig_out` for playback.
The function which perform the insertion of elongations and silences following rhythmogram analysis on the set of five sound files joined as example is `do_example.m`.

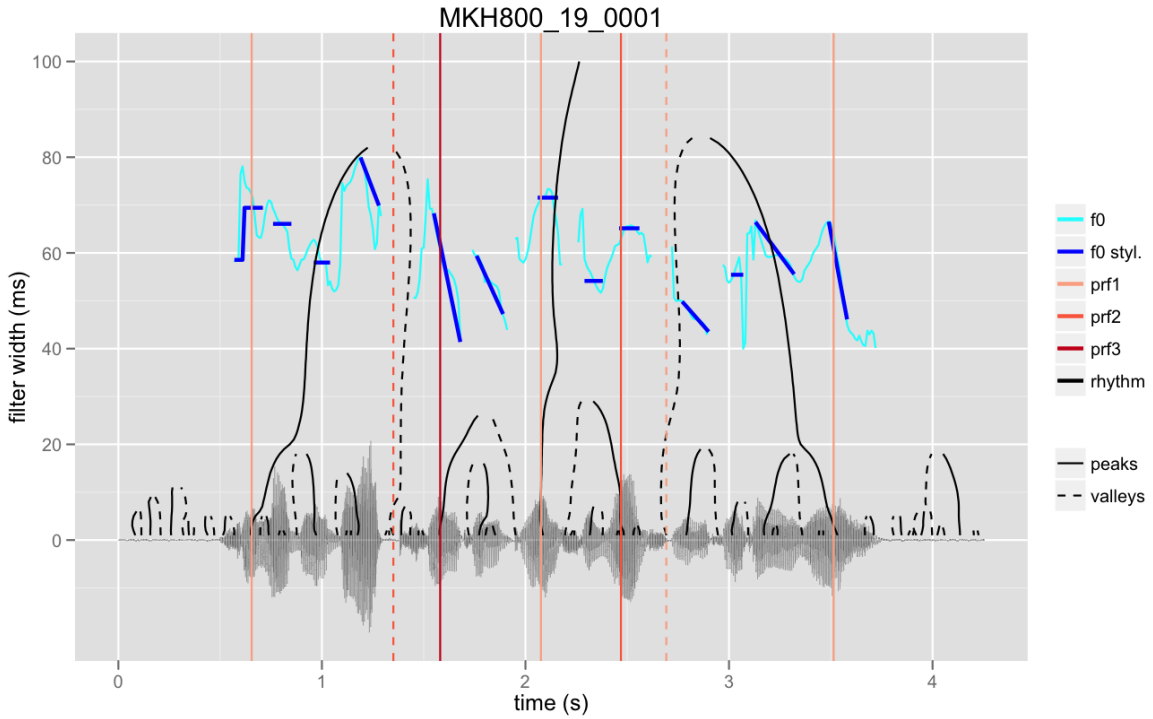

Figure 6: Results of rhythmogram analysis and stylised pitch countour extraction for the sentence "She had your dark suit in greasy wash water all year". Waveform is shown in grey, and rhythmogram stems in black, contrasting peaks (solid lines) and valleys(dashed lines). The rhythmogram stems are build by linking contiguous peaks at different filter widths (y axis). Fundamental frequency is shown in light blue and corresponding stylised pitch contour in dark blue. the range of visible pitch is 80–182 Hz. Vertical lines show the insertion points for elongations and silences, following highest stems for peaks and valleys respectively. Because increasing profile number have an increasing number of insertion locations, the total number of insertions for a profile is the sum of insertions for this profile plus the insertion locations of lower order profiles.

4.3 List of files

Table 5 lists the main files and directories which constitute the software for rhythmic structure and stylised pitch countour extraction, elongation and silences insertion, and plotting of the results, for a set of five sound files from the TIMIT database.

file/directory	description
`do_example.m`	main function for example treatment: performs rhythmogram analysis and inserts elongation and silences for three different profiles.
`get_prosogram.sh`	performs pitch stylisation extraction
`plot.R`	plots the result of rhythmogram analysis and stylised pitch countour extraction, along with the insertion points for elongation and silences
`plot/`	additional `R` functions and plot outputs in `.png` format
`prosogram/`	additional files needed to extract stylised pitch countour
`rhythm/`	additional files to compute the rhythmogram and insert elongations and silences
`rhythm/params/`	definitions of profiles for elongations and silences insertion, and global rhythmogram parameters
`sig/`	soundfiles used as examples, as well as analysis files generated in the process of rhythmic extraction and stylised pitch countour extraction (rhythmogram stems for peaks and valleys, pitch and stylised pitch contours)
`sig_out/`	resulting modified soundfiles with insertion of elongation and silences, as well as time information of insertion (time refers to the original sound files).

Table 5: List of files and directories for the software

References

[1] M. Huckvale, "A syntactic pattern recognition method for the automatic location of potential enhancement regions in running speech," *Speech, Hearing and Language: work in progress*, vol. 10, 1997.

[2] T. Sainath, D. Kanevsky, and B. Ramabhadran, "Broad phonetic class recognition in a hidden Markov model framework using extended Baum-Welch transformations," in *Proceedings of IEEE ASRU-2007*, Kyoto, Japan, pp. 306–311.

[3] P. Scanlon, D. P. W. Ellis, and R. B. Reilly, "Using broad phonetic group experts for improved speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 3, pp. 803–812, 2007.

[4] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[5] D. A. Reynolds and R. C. Rose, "Robust text-independent speaker identification using gaussian mixture speaker models," *IEEE Transactions on Speech and Audio Processing*, vol. 3, no. 1, pp. 72–83, 1995.

[6] D. A. Reynolds, "Automatic speaker recognition using gaussian mixture speaker models," *The Lincoln Laboratory Journal*, vol. 8, no. 2, pp. 173–192, 1995.

[7] D. Guo and S. Z. Li, "Content-based audio classification and retrieval by support vector machines," *IEEE Transactions on Neural Networks*, vol. 14, no. 1, pp. 209–215, 2003.

[8] T. N. Sainath, B. Ramabhadran, M. Picheny, D. Nahamoo, and D. Kanevsky, "Exemplar-based sparse representation features: From timit to lvcsr," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 8, pp. 2598–2613, 2011.

[9] D. Kapilow, Y. Stylianou, and J. Schroeter, "Detection of non-stationarity in speech signals and its application to time-scaling," in *Proc. Eurospeech*, vol. 5, 1999, pp. 2307–2310.

[10] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society. Series B*, vol. 39, no. 1, pp. 1–38, 1977.

[11] T. Zhang, R. Ramakrishnan, and M. Livny, "Birch: A new data clustering algorithm and its applications," *Data Mining and Knowledge Discovery*, vol. 1, no. 2, pp. 141–182, 1997.

[12] W. Strange, J. Jenkins, and T. Johnson, "Dynamic specification of coarticulated vowels," *J. Acoust. Soc. Am.*, vol. 74, pp. 695–705, 1983.

[13] E. Kennedy, H. Levitt, A. Neuman, and M. Weiss, "Consonant-vowel intensity ratios for maximizing consonant recognition by hearing-impaired listeners," *J. Acoust. Soc. Am.*, vol. 103, pp. 1098–1114, 1998.

[14] S. Nagarajan, X. Wang, M. Merzenich, C. E. Schreiner, P. Johnston, W. Jenkins, S. Miller, and P. Tallal, "Speech modifications algorithms used for training language learning-impaired children," *IEEE Trans. on Rehabilitation Eng.*, vol. 6, no. 3, pp. 257–268, 1998.

[15] S. D. Yoo, J. R. Boston, A. El-Jaroudi, C. C. Li, J. D. Durant, K. Kovacyk, and S. Shaiman, "Speech signal modification to increase intelligibility in noisy environments," *J. Acoust. Soc. America*, vol. 122, pp. 1138–1149, 2007.

[16] N. P. M. Todd and G. J. Brown, "A computational model of prosody perception," in *ICSLP*, Yokohama, Japan, 1994, pp. 127–130.

[17] ——, "Visualization of Rhythm, Time and Metre," *Artificial Intelligence Review*, vol. 10, pp. 253–273, 1996.

[18] C. S. Lee and N. P. M. Todd, "Towards an auditory account of speech rhythm: application of a model of the auditory 'primal sketch' to two multi-language corpora," *Cognition*, vol. 93, no. 3, pp. 225–254, 2004.

[19] B. Ludusan, A. Origlia, and F. Cutugno, "On the use of the rhythmogram for automatic syllabic prominence detection," in *Interspeech*, 2011, pp. 1–4.

[20] P. Mertens, "The Prosogram: Semi-Automatic Transcription of Prosody Based on a Tonal Perception Model ," in *Speech Prosody*, Nara, Japan, 2004.

[21] ——, "Un outil pour la transcription de la prosodie dans les corpus oraux ," *Traitement Automatique des Langues*, vol. 45, no. 2, pp. 109–130, 2004.