Extraction of Prosodic Features from Speech Signals

Toningenieur-Projekt

durchgeführt von

Florian Pokorny BSc
(florian.pokorny@student.kug.ac.at)

Graz, 2011

Institut für Elektronische Musik und Akustik
der Universität für Musik und darstellende Kunst Graz

Leiter: Dipl.-Ing. Dr.techn. Alois Sontacchi
Betreuer: Dipl.-Ing. Johannes Luig

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel[1] nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

[1] Die vorliegende Dokumentation wurde unter Verwendung von LaTeX angefertigt. Alle Abbildungen ohne Quellenangabe wurden entweder mit CorelDRAW 12 erstellt, oder aus MATLAB 7.0.4 exportiert.

Zusammenfassung

Mittels gesprochener Sprache transportiert der Mensch zwei Arten an Information. Neben der Semantik, die den Inhalt einer Äußerung umfasst, dient die Prosodie deren Gliederung und Akzentuierung. Als *prosodisch* können also jene Sprachmerkmale bezeichnet werden, die im physikalischen Signal des syntagmatischen Verlaufs nicht direkt segmentierbar sind, sondern ihre sprachliche Funktion erst durch eine Relationierung, also durch einen Vergleich mit den vorangehenden und/oder nachfolgenden Teilen des Signals erhalten. Die prosodischen Merkmale Intonation und Intensität lassen sich mit den akustischen Parametern Grundfrequenz und Schallintensität beschreiben. Der Sprechrhythmus kann durch die Detektion von Silbengrenzen erfasst werden.

Ziel dieses Projekts war es, mit Hilfe von MATLAB und/oder Praat ein Analyse/Synthese-Tool zu entwickeln, mit Hilfe dessen ein Sprachsignal hinsichtlich seiner prosodischen Merkmale analysiert werden kann. Weiters sollte ein synthetisches Signal erzeugt werden können, das nur noch die detektierten prosodischen Merkmale enthält. Diese sollten vor der Synthese in einem GUI skalierbar sein.

Abstract

Human speech carries two types of information. On the one hand, semantics covers the content of an utterance, on the other hand prosody provides the expression's structure and accentuation. Thus, speech features are termed prosodic features if their direct segmentation from the syntagmatic run is not possible and a comparison with previous and/or following segments of the signal is required. So the relation between these segments is essential for the representation of a prosodic linguistic function. The prosodic features intonation and intensity are characterized by the acoustic parameters fundamental frequency and sound intensity. Speech rhythm can be extracted by detecting the borders of syllables.

The aim of this project was the development of an analysis/synthesis tool in MATLAB and/or Praat that extracts prosodic features from a speech signal and, furthermore, creates a synthetic signal consisting of these features only. Moreover, they are scalable using a GUI before the synthesis.

Contents

1 Introduction **6**

2 Feature Analysis **7**
2.1 Pitch Extraction 7
2.1.1 Detection of Periodicity Using Praat 7
2.1.2 Resampling of the Pitch Contour 8
2.2 Intensity Extraction 9
2.2.1 Detection of Syllable Nuclei 9
2.3 Syllable Extraction 10
2.3.1 Blind Calculation of Syllable Boundaries 10
2.3.2 Use of Hand-Labeled Syllable Boundaries 15

3 Feature Synthesis **16**
3.1 Scaling and Synthesis of the Syllable Boundaries 16
3.2 Scaling and Synthesis of the Pitch Contour 17
3.3 Scaling and Synthesis of the Intensity Contour 19

4 Graphical User Interface **21**

5 Evaluation **24**
5.1 Comparison of Hand-Labeled and Calculated Syllable Boundaries 25

6 Conclusion and Outlook **30**

Bibliography **31**

Appendix **32**
Creation of Text Files with Syllable Boundary Locations 32

List of Figures

1 Smoothed pitch contour with marked pitch borders . . . 8
2 Resampled smoothed pitch contour with marked pitch borders . . . 9
3 Smoothed intensity contour with syllable nucleus candidates . . . 10
4 Histogram of the length of syllables . . . 11
5 Short-term energy function of a segment of speech . . . 11
6 Mirrored short-term energy function . . . 12
7 Inverted magnitude spectrum . . . 12
8 Hann-weighted causal portion of the root cepstrum . . . 13
9 Minimum phase group delay function . . . 14
10 Stretched group delay function with detected peaks . . . 14
11 Steps involved in the extraction of syllable boundaries . . . 15
12 Method of speech rhythm scaling . . . 16
13 Interpolation of the pitch contour dependent on scaled syllable boundaries . . . 17
14 Method of pitch scaling . . . 17
15 Method of pitch synthesis . . . 18
16 Filter used for extraction pink noise from white noise . . . 19
17 Method of intensity scaling . . . 20
18 Final synthesis signal . . . 20
19 Start screen of the GUI with marked functional areas . . . 21
20 Info screen of the GUI . . . 22
21 GUI immediately after a calculation pass . . . 23
22 Output waveform based on scale factors of 1 . . . 24
23 Output waveform based on scale factors of 0 . . . 24
24 Output waveform based on scale factors of 0.3 . . . 25
25 Speech segment with hand-labeled and detected syllable boundaries . . . 27

Listings

1 `get_pitch.praat` . . . 7
2 Classification into TPs, FNs and FPs . . . 26
3 Excerpt of the label file `A0101B.TextGrid` . . . 32

List of Tables

1 Sound files used for the syllable detection evaluation . . . 26
2 Syllable detection performance results for the default settings . . . 28
3 Syllable detection performance results for varied settings . . . 28

1 Introduction

From a linguistic point of view, there exist specific phonetic characteristics in spoken language that outreach a single segment. Thus, they are called suprasegmental features or suprasegmentals. These terms imply a difference between segmental sounds like vowels and consonants, which make up the semantic information, and features which are perceived as extending over longer periods of speech [CY95, p.328]. On the one hand, the linguistic role of suprasegmentals consists in the differentiation of meaning at a level of words, syllables or sentences. On the other hand, an important task of suprasegmental features is the rhythmic structuring of utterances. Suprasegmentals and their linguistic functions are often abstracted with the terms *prosodic features* or *prosody*. This expression has been deduced from the Greek word 'prosodía', which means 'said in addition' and induces the wrong assumption that accentuation and speech melody would be secondary attributes performed in addition to speech segments. In fact, segmental and suprasegmental features always appear together in each spoken utterance. [Gra06, p.63]

As already mentioned, prosodic features can not be directly extracted from the syntagmatic run of a speech signal. A comparison of neighboring segments in the signal is required and their relation is essential for the representation of a prosodic linguistic function.

Pitch and loudness are known as prosodic features [Gra06, p.63] which can be characterized by the acoustic parameters fundamental frequency and sound intensity. Duration represents a third prosodic feature [Gra06, p.63] that builds up the speech rhythm extractable by the detection of syllable boundaries.

At this point, the question arises how much of the prosodic information of a speech signal can be removed without losing the essential meaning of an utterance. Furthermore, it would be interesting to investigate the role and the influence of each single prosodic feature and its dominance in perception.

Therefore, an analysis and synthesis tool has been developed in MATLAB that can be applied to arbitrary speech signals. In the analysis part of the tool, a separation between voiced and unvoiced passages via measurement of periodicities and a calculation of the fundamental frequency is performed at first. Subsequently, the sound intensity is measured and hence the positions of syllable nuclei are extracted as being local maxima of the intensity contour. The locations of syllable boundaries can either be imported from text files with hand-labeled data or they can be calculated using an implemented blind detection algorithm.

In the synthesis part, a signal is created simply based on the analyzed features pitch, intensity and speech rhythm. However, the tool allows to adjust the magnitude of each feature between 100% and 0%. 100% means that the original prosodic influence is kept, whereas 0% signifies that no variation of this feature occurs any more in the whole synthesized signal and consequently this specific prosodic information is entirely neglected.

In the following sections, all important calculation methods and considerations that contributed to the successful development of the tool are explained in detail. Section 4 includes a description of the graphical user interface (= GUI) created for a comfortable use of the tool. It serves as a user manual.

2 Feature Analysis

The analysis and synthesis tool can cope with arbitrary sound files, but there are two preconditions to be fulfilled. Firstly, the sampling frequency fs of the file has to be an integer multiple of 22050 Hz in case of being higher than this value. In case of being lower the original sampling frequency is kept. Secondly, the sound file has to contain voiced parts, i.e., parts with detectable periodicities. This latter condition is assumed to be an indicator for the presence of speech within the respective sound file as every syllable is assumed to contain a *vocalic* part.

The analysis routine of the tool starts with the import of the selected sound file. Stereo files are converted into mono files in a first step. Files with sampling rates higher than 22050 Hz are resampled at that value using the MATLAB function `decimate`. This function filters the input data using an 8^{th} order Chebyshev type I lowpass filter with a cutoff frequency of $0.8 * (fs/2)/R$, before resampling at $1/R$ times the original sample rate [TM05]. $R = fs/22050$ and has to be an integer, which makes up the reason for the first condition concerning the use of arbitrary sound files mentioned above.

Hence, mono files with a sampling rate of 22050 Hz or lower original sampling rates constitute the starting point for all further analysis and synthesis steps.

2.1 Pitch Extraction

The calculation of the fundamental frequency is carried out with Praatcon for reasons of simplicity, processing speed and accuracy. Praatcon is a special version of Praat - a well-known phonetic analysis software (q.v. [BW11]) - that can be accessed from a console window, thus, from other calculation platforms like MATLAB.

2.1.1 Detection of Periodicity Using Praat

At first, a temporary copy of the original input sound file is created. Using the MATLAB function `eval`, Praatcon is launched, the created temporary sound file and two adjustable parameters are handed over and the written Praat script `get_pitch.praat` is opened and put into execution. The two mentioned parameters constitute the expected maximum and minimum pitch values in hertz. The Praat script `get_pitch.praat` is shown in listing 1.

Listing 1: `get_pitch.praat`

```
1  form settings
2  sentence filename test
3  natural  fmax     500
4  natural  fmin     60
5  endform
6
7  Read from file... 'filename$'.wav
8  To PointProcess (periodic, peaks)... fmin fmax yes yes
9  Write to short text file... 'filename$'.praat
10
11 To PitchTier... 0.02
12 Write to short text file... 'test'.praat
```

At hand, the extraction of the pitch contour is based on the detection of periodicities. The used Praat command `To PointProcess (periodic, peaks)...` analyses the selected sound file and creates a PointProcess object, i.e., a sequence of points in time. The actions of the Praat routines `Sound: To Pitch (ac)...` and `Sound & Pitch: To PointProcess (peaks)...` are combined. On the one hand, the above-mentioned acoustic periodicity detection is performed on the basis of an accurate short-term analysis of the fundamental frequency and the

harmonics-to-noise ratio, working in the lag (autocorrelation) domain, as described in Boersma [Boe93]. This method is able to achieve more accurate, noise-resistant and robust results than the original autocorrelation method or other methods based on the cepstrum or combs. On the other hand, the determined acoustic periodicity contour is interpreted as being the frequency of an underlying sequence of glottal closures in vocal-fold vibration. For every single voiced interval, i.e., parts containing detected periodicities, a number of points - representing glottal pulses - are found and their points in time are saved and handed back to MATLAB. [Boe11]

Back in MATLAB, the differences between adjacent elements of the vector - containing the detected time marks - are calculated. A separation into voiced and unvoiced parts is performed by comparing these differences representing the respective periods of oscillation to the periodic time of an oscillation constituted by the chosen expected minimum pitch. Locations in time where difference values exceed this reference interval, represent pitch borders. A correction is conducted if a detected pitch border is surrounded by two difference values exceeding the reference interval. In either case, the border is neglected.

In order to obtain the desired pitch values in Hertz, the element-wise reciprocal of the difference vector is built. Subsequently, the extracted pitch contour is smoothed using a 20-point moving average filter.

Figure 1 shows a section of a pitch contour calculated in the described way.

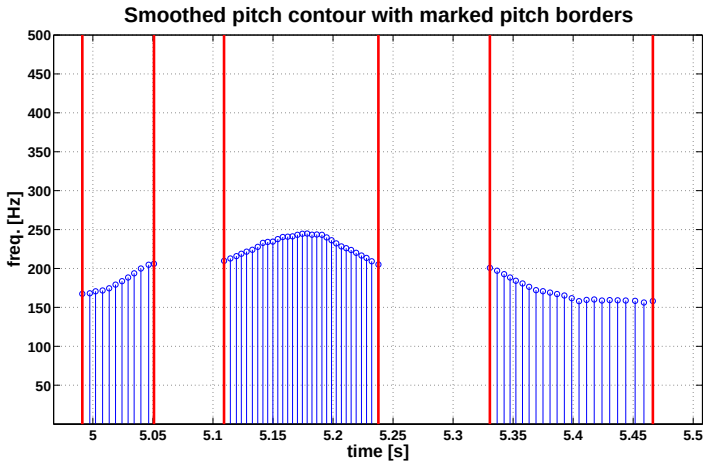

Figure 1: Section of a smoothed discrete-valued pitch contour with marked pitch borders. The blue stems illustrate the discrete smoothed pitch values, the red bars represent corrected and finalized pitch borders that mark a changeover from voiced to unvoiced parts of the signal or vice versa.

2.1.2 Resampling of the Pitch Contour

For further analysis and synthesis steps related to the fundamental frequency, the currently extracted pitch contour has to be resampled to fs. This procedure is required due to the special used method of deriving non-equidistant pitch values from pitch intervals. The step of resampling the pitch contour is performed for all voiced parts within a loop by using the MATLAB function `interp1` that interpolates an underlying vector at the points in an arbitrary array [TM05]. During one loop cycle, the voiced section between two detected consecutive pitch borders gets processed. The points in the resampling array exhibit time distances of $1/fs$. As interpolation method, the cubic one has been chosen to avoid steps and sharp bends within the contour.

Figure 2 illustrates the same section of the pitch contour from figure 1 complemented by the resampled contour.

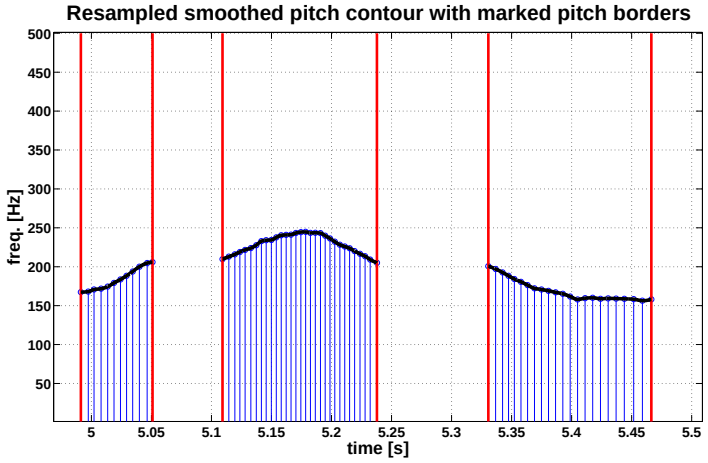

Figure 2: Section of a smoothed discrete-valued pitch contour with marked pitch borders and the contour resampled to fs plotted in black.

2.2 Intensity Extraction

The intensity contour is calculated by raising the values of the input signal element-wise to the power of two. Subsequently, smoothing is performed by using a 2^{nd} order lowpass digital Butterworth filter. Its coefficients are calculated using the MATLAB function `butter`. The adjustable cutoff frequency is set to $20/(fs/2)$ by default, i.e., 20 Hz.

The filtering operation is conducted as a zero-phase forward and reverse digital filtering procedure using the MATLAB function `filtfilt`. More precisely, the forward filtered data sequence is reversed and run back through the filter. Hence, the result has zero phase distortion and the magnitude is modified by the square of the filter's magnitude response. [TM05]

2.2.1 Detection of Syllable Nuclei

For reasons of their subsequent use in the calculation of syllable boundaries, the locations of syllable nuclei are detected based on the smoothed intensity contour and on the information about the locations of voiced and unvoiced parts.

In a first step, the median value of the smoothed intensity contour is calculated followed by the detection of peaks using the function `pickpeaks` downloaded from the Internet (q.v. [Sar11]). The minimum number of points separating two peaks is adjustable and set to 500 by default. This value has been determined experimentally. At the beginning, all detected maxima are considered as candidates for syllable nuclei locations. Subsequently, those candidates with values equal or lower than a forth of the computed median value of the smoothed intensity contour are discarded. In a next step, the smoothed intensity contour is divided into segments each consecutively extending from one valid location candidate to the next. Within a loop over all segments, the difference between a candidate value and the minimum value of the respective segment is calculated and compared to an adjustable threshold that is set to 0.0001 by default. Those location candidates with difference values exceeding the threshold are kept. Finally, all location candidates situated inside unvoiced parts of the signal, are discarded.

Figure 3 illustrates the smoothed intensity contour of a sound sample with a recorded "good morning" uttered by a female speaker.

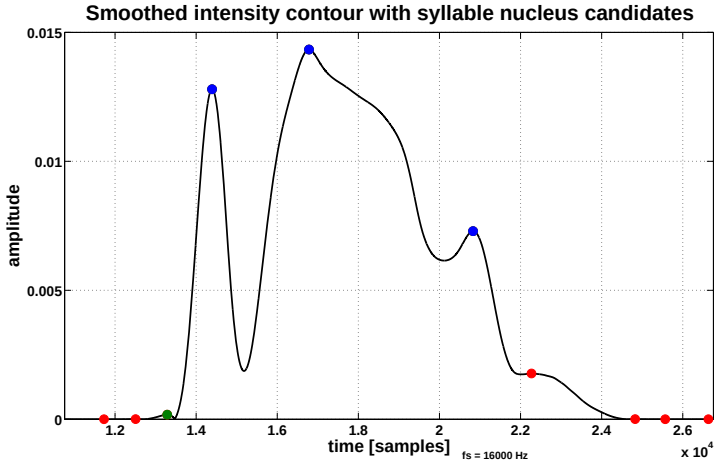

Figure 3: Smoothed intensity contour with syllable nucleus candidates. The red dots mark location candidates for syllable nuclei that have been discarded due to the threshold condition. The green dot is situated inside of an unvoiced part of the signal. The blue dots constitute all valid locations of syllable nuclei.

2.3 Syllable Extraction

For the extraction of syllable boundaries from the input speech signal, a blind syllable detection algorithm (q.v. [PNM04]) has been implemented. However, if the locations of syllable boundaries are available, e.g., in a hand-labeled data corpus, they can be imported from a text file alternatively. Moreover, such label files can be created by anyone after determining syllable boundaries manually (q.v. appendix). The alternative of using external detected syllable boundaries yields the advantage that possible inaccurate performance results of the blind detection algorithm can be bypassed.

2.3.1 Blind Calculation of Syllable Boundaries

The embedded blind syllable detection algorithm has been implemented according to descriptions in Prasad et al. [PNM04]. It is based on processing the short-term energy function of the continuous speech signal and on its characteristic of being a positive function. [PNM04, p.429] Furthermore, the property is used that poles and zeros of a minimum phase group delay function can be distinguished easily since local maxima correspond to poles while local minima correspond to zeros. [PNM04, p.434]

In short, the symmetrized and inverted short-term energy function is viewed as an arbitrary magnitude spectrum. Subsequently, the root cepstrum gets calculated, group delay processing is performed and desired syllable boundaries are determined by detecting the peaks of the computed group delay spectrum. [PNM04, p.438]

Thus, first of all the input speech signal is divided into segments each approximately containing an adjustable number of syllables. For this purpose, the lengths of 15815 syllables from the *Aix-MARSEC Project* corpus - a database of spoken British English (q.v. [ABH04]) - have been examined by creating a histogram. As shown in figure 4, most of the syllables have a length between 100 ms and 200 ms.

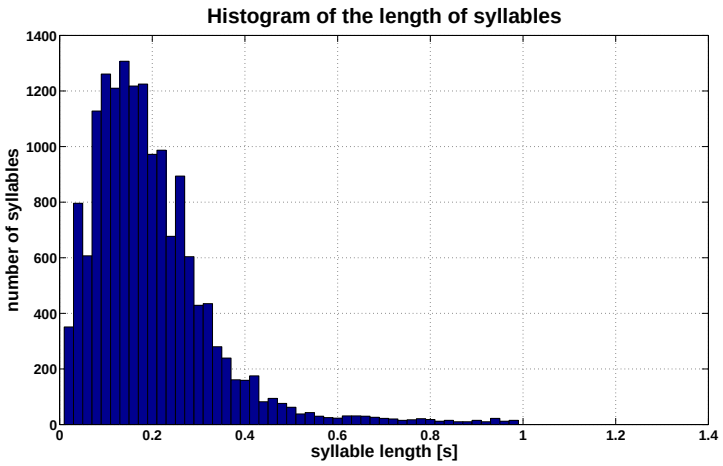

Figure 4: Histogram of the length of 15815 syllables extracted from the *Aix-MARSEC Project* corpus (q.v. [ABH04]).

For allowing a performance evaluation of the syllable detection algorithm by comparing the calculated boundaries to the available ones of the hand-labeled *Aix-MARSEC Project* corpus, the algorithm is initially optimized for the corpus data. Hence, the most frequent syllable length of 0.14 s - multiplied by the adjustable number of syllables - is used in the algorithm to determine the length of the analysis segments. This approximate number of syllables per analysis segment constitutes 5 by default. The analysis segments are processed consecutively inside a `for`-loop. The following descriptions of the implementation correspond to signal processing steps within one loop-cycle, i.e., they deal with processes applied to one analysis segment.

According to Prasad et al. [PNM04, p.438], the short-term energy function has to be obtained at first. In this case, the respective segment of the already computed, smoothed intensity contour is taken. Subsequently, zero values are replaced with the detected minimum value unequal zero in order to avoid points of discontinuity within the inverted signal computed later on. Figure 5 illustrates the short-term energy function of one processed segment of a speech sample.

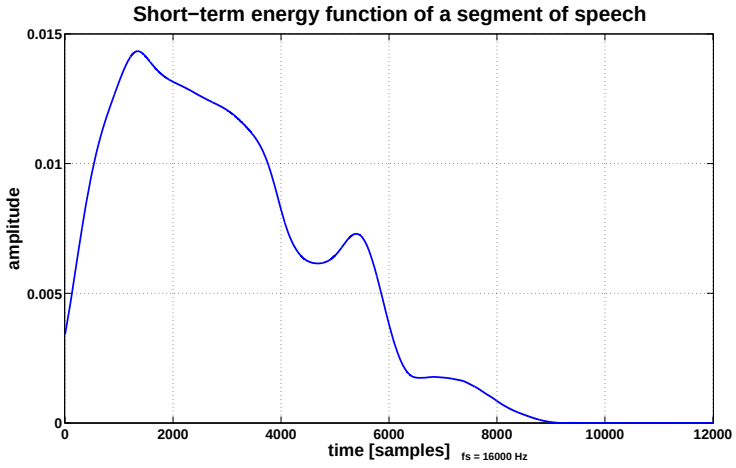

Figure 5: Short-term energy function of a segment of speech with corrected zero values.

In a next step, a symmetric function is constructed by producing a lateral inversion of the short-term energy function about the Y-axis [PNM04, p.438] and by attaching this mirrored

sequence to the original function. The result can be seen in figure 6 for the same speech segment as in figure 5.

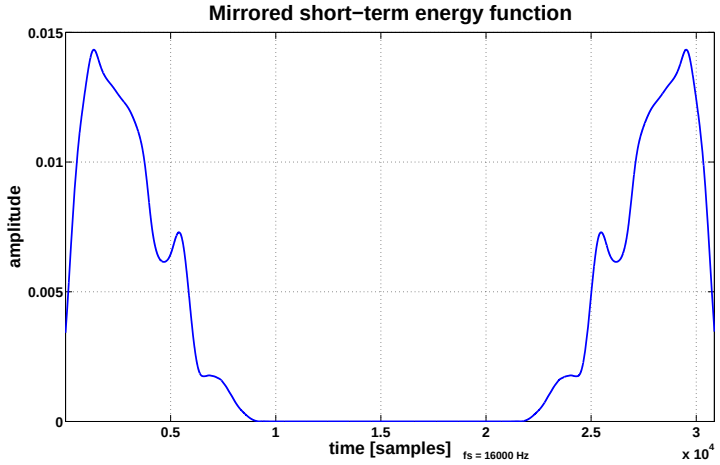

Figure 6: Mirrored short-term energy function attached to the original function.

Due to the fact that the created sequence is a positive and symmetric function, it can be viewed as an arbitrary magnitude spectrum [PNM04, p.438]. This magnitude spectrum is weighted by an adjustable exponential factor that highly affects the detection performance of the algorithm dependent on the analyzed sound file. It is set to 1 by default.

Subsequently, the weighted magnitude spectrum is inverted. Figure 7 shows the result of the inversion.

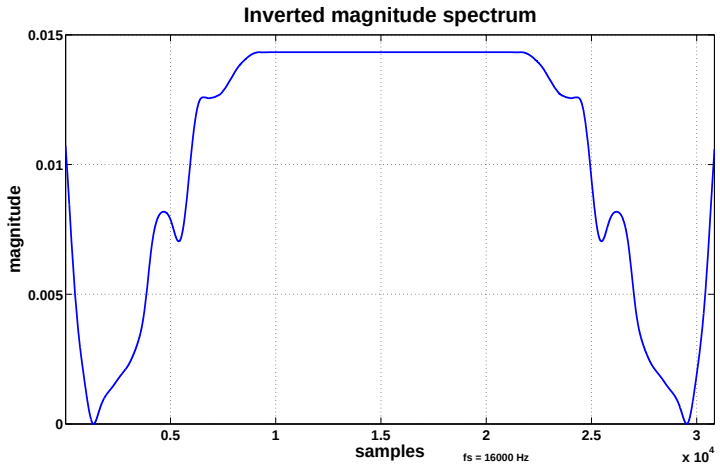

Figure 7: Inverted magnitude spectrum originally derived from the short-term energy function.

At this point, the root cepstrum is calculated by processing an inverse discrete Fourier transform (= IDFT). As the causal portion of the root cepstrum exhibits minimum phase properties [PNM04, p.438] an adaptive window scale factor is introduced that guarantees the cut-off of the acausal part of the root cepstrum. More precisely, the window scale factor determines the size of the window that is cut out of the causal part of the root cepstrum starting from its first value. According to Prasad et al. [PNM04, p.438], this window size S_W is defined in (1).

$$S_W = \frac{size\ of\ short-term\ energy\ function}{window\ scale\ factor} \tag{1}$$

The adaptive window scale factor is initially set to 1 which causes the whole causal portion of the root cepstrum being used for the extraction of syllable boundaries. Later on, the number of detected syllable boundaries is compared to the number of the already extracted syllable nuclei which should equal to each other. In case of a mismatch, the adaptive window scale factor is increased by 1, the respective smaller part of the causal portion of the root cepstrum is extracted and all forthcoming calculation steps are conducted again within a `while`-loop. Namely, the remaining signal is multiplied by the second half of a Hanning window as described in Murthy and Yegnanarayana [MY91, p.212]. Thus, values closer to the end of the extracted causal portion of the root cepstrum are faded out more and more. Figure 8 illustrates a Hann-weighted causal portion of the root cepstrum of the same speech segment already presented in the previous figures.

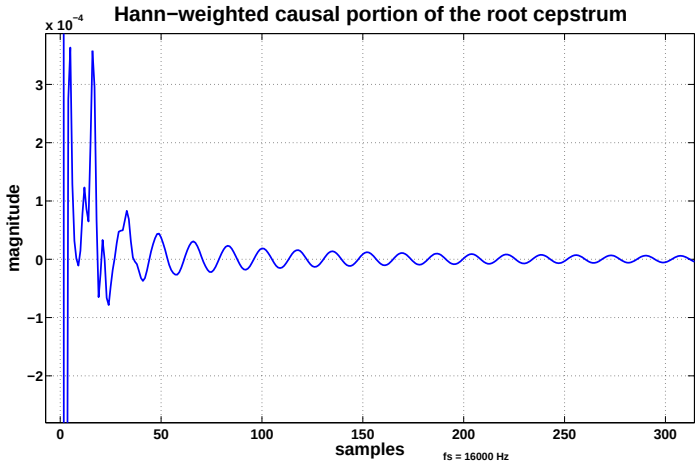

Figure 8: Causal portion of a speech segment's root cepstrum multiplied by the second half of a Hanning window.

Subsequently, the minimum phase group delay function is calculated. In general, group delay is defined as the negative derivative of phase [PNM04, p.432] as shown in the equation below.

$$\tau = -\frac{d\varphi}{dt} \tag{2}$$

Thus, two processing steps are performed. Firstly, the phase spectrum is computed, i.e., a DFT is performed and the phase is extracted from the result's causal portion. Secondly, the forward difference of the phase function is calculated. Figure 9 shows the result.

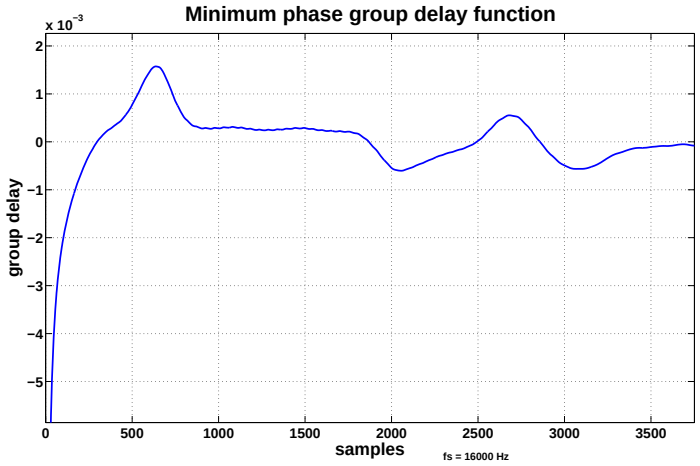

Figure 9: Minimum phase group delay function of a speech segment.

As the positive local maxima in the minimum phase group delay function approximately correspond to syllable boundaries [PNM04, p.439], peak detection is performed using the MATLAB function `peakdet` downloaded from the Internet (q.v. [Aki11]).

An adjustable threshold affects the sensibility of the peak detector. More precisely, to be considered as a valid peak, a detected local maximum must be preceded by a value lower than this threshold. It is set to 0.001 by default.

As already mentioned above, the number of detected positive peaks is compared to the number of syllable nuclei. The step-wise increase of the window scale factor by 1 in case of a mismatch of numbers is aborted at the latest at a window scale factor's value of 100. In this particular case of no match of numbers, peak positions are selected from that adaptive cycle with the smallest occurring mismatch.

Because the length of the group delay function does not equal the length of the original analysis segment caused by the truncation of the root cepstrum and by the processed DFTs, the group delay function gets accordingly scaled linearly at this point using the MATLAB function `interp1`. The peaks - corresponding to syllable boundaries - are recalculated. In figure 10 the stretched group delay function is shown.

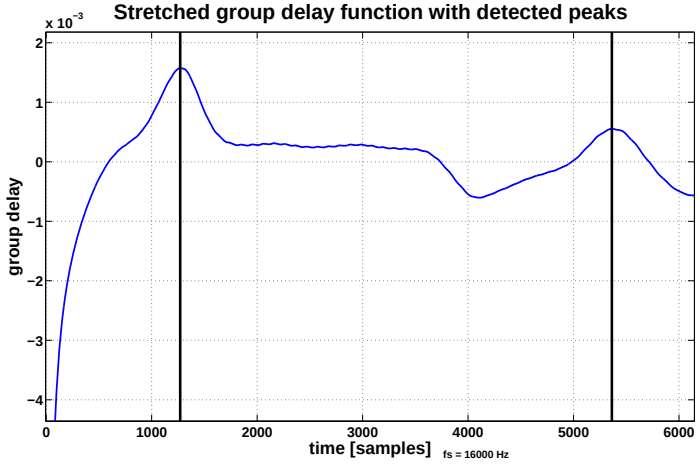

Figure 10: Stretched group delay function with detected peaks for a segment of speech.

At this point, the `for`-loop - used for the segment-wise calculation of peaks within the minimum phase group delay function - ends and the detected peaks from all analysis segments are merged to one data vector. Finally, a correction of possibly twice detected syllable boundaries in neighboring analysis segments is performed. This correction is necessary because the analysis segments have not been weighted using any window functions. Thus, a search interval of half the most frequent syllable length from the histogram is defined. Subsequently, the number of detected syllable boundaries inside this search interval - placed centered above the segment borders - is reduced to one syllable boundary. In other words, as a syllable boundary at the edge of subsequent analysis windows may be detected twice, these double entries are removed in this post-processing step.

In summary, figure 11 shows all important steps involved in the extraction of syllable boundaries for a segment of speech according to Prasad et al. [PNM04].

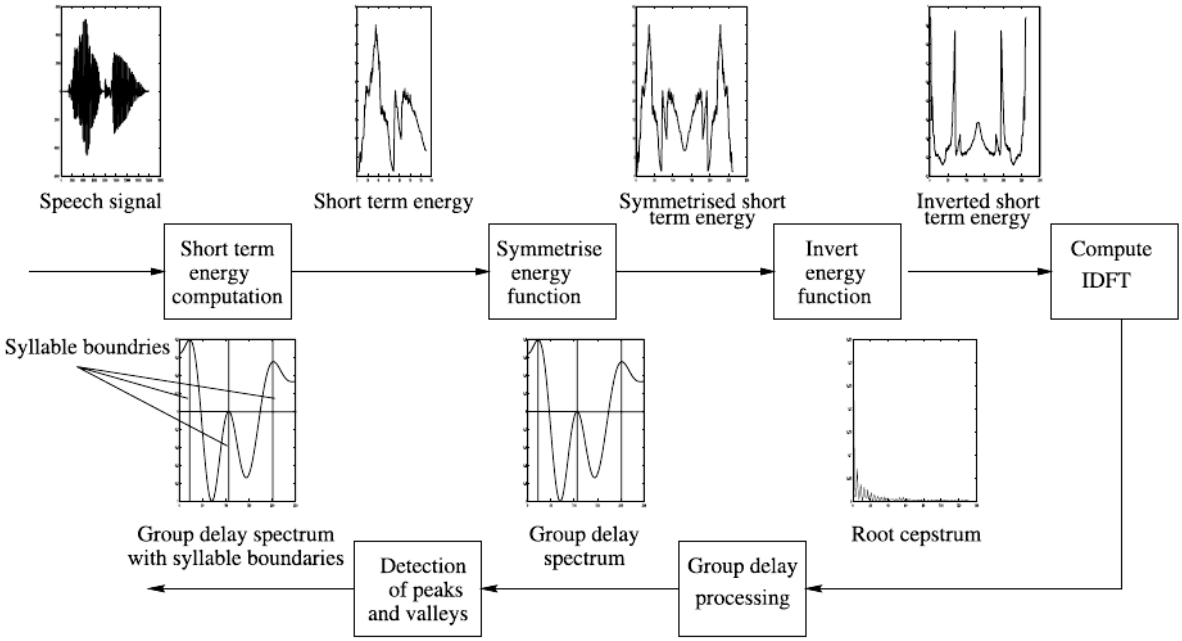

Figure 11: Steps involved in the extraction of syllable boundaries for a segment of speech. [PNM04, p.438]

2.3.2 Use of Hand-Labeled Syllable Boundaries

Hand-labeled syllable boundaries can be imported if formatted as a column of time values in seconds and if saved as a text file with the ending *.TextGrid*. Those hand-labeled data files used for testing the import of syllable boundaries also contained other labeled phonetic characteristics like phonemes or word boundaries. Therefore, the correct rows of the data column have to be selected after the whole data column is being read from the text file using the MATLAB function `textscan`. For this purpose, the column is scanned for the expressions *Syllables* and *Abercrombie*. *Syllables* marks the top of the desired data section containing the locations of syllable boundaries, *Abercrombie* marks the top of the next characteristic's data section, i.e., the ending of the desired section. The required formatting for the creation of own text files containing the locations of external detected syllable boundaries is described in the appendix.

3 Feature Synthesis

Inspired by the methods presented in McAulay and Quatieri [MQ86], a basic synthesis algorithm has been implemented that creates a signal based on the extracted prosodic features pitch, intensity and speech rhythm. More precisely, the fundamental frequency contour of the analyzed signal is synthesized considering the computed locations of syllable boundaries and weighted by the extracted intensity contour. As already mentioned in section 1, the tool offers the opportunity of adjusting the magnitude of each of these features between 100% and 0%.

3.1 Scaling and Synthesis of the Syllable Boundaries

As a first and fundamental step of the synthesis algorithm, the syllable boundary positions are scaled. Since the first and the last detected syllable boundary normally mark the beginning of the first and the ending of the last uttered word in the speech signal, their locations stay unmodified at all times. However, the speech rhythm in between of those fixed borders can be scaled linearly between two states by adjusting the syllable scale factor. A scale factor of 1 means that the locations of the syllable boundaries in the synthesized signal remain the same as in the analyzed signal, i.e., the original speech rhythm is preserved. A syllable scale factor of 0 causes equidistant syllable boundaries. Figure 12 illustrates the method of rhythm scaling by showing the two extreme states - due to scale factors of 1 and 0 - and an additional state in between for a short sound sample.

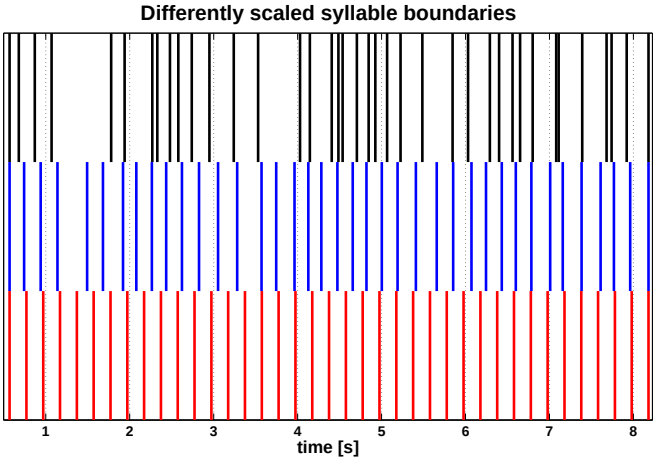

Figure 12: Syllable boundaries of a short speech signal scaled by different scale factors (black: factor = 1; blue: factor = 0.3; red: factor = 0).

As the analyzed pitch and intensity sequences have to keep their temporal course during a syllable, their contours are cubically interpolated depending on the scaled locations of the syllable boundaries using the MATLAB function `interp1`. In other words, some parts of the pitch and intensity contours potentially get stretched and others get shrinked. Figure 13 shows both the pitch contour of a speech sample with the original locations of syllable boundaries and the interpolated pitch contour caused by syllable boundaries scaled with a factor of 0.

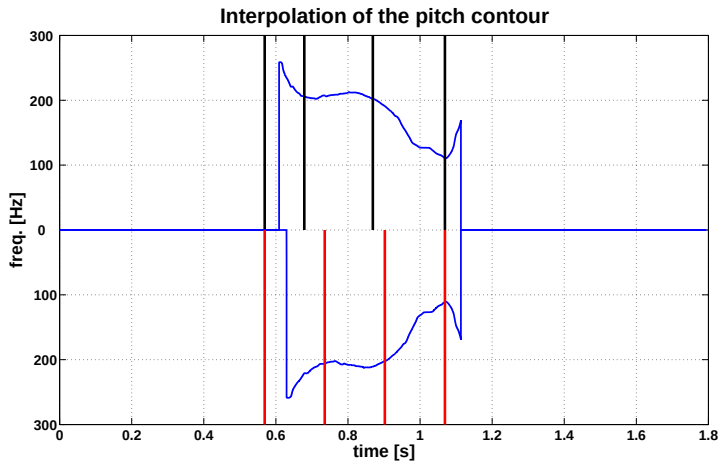

Figure 13: Pitch contour of a speech sample with the original locations of syllable boundaries colored in black and the interpolated pitch contour caused by the original syllable boundaries scaled with a scale factor of 0 colored in red.

Therefore, also the pitch borders are shifted in time due to the interpolation procedure.

3.2 Scaling and Synthesis of the Pitch Contour

In the next step of the synthesis algorithm, the interpolated pitch contour is scaled depending on the adjustable pitch scale factor. For this purpose, the element-wise differences between the discrete pitch values and the mean value of the contour are computed. Subsequently, this difference vector is multiplied by the pitch scale factor and the contour gets recreated by an element-wise addition of the values of the scaled difference vector and the contour's mean value. Thus, a pitch scale factor of 1 keeps the original pitch contour, whereas shrinking scale factors continually compress the pitch dynamic of the contour more and more. A scale factor of 0 causes a constant pitch contour at the height of the mean value of the original contour. Figure 14 illustrates the pitch contours of a short sound sample scaled with different factors.

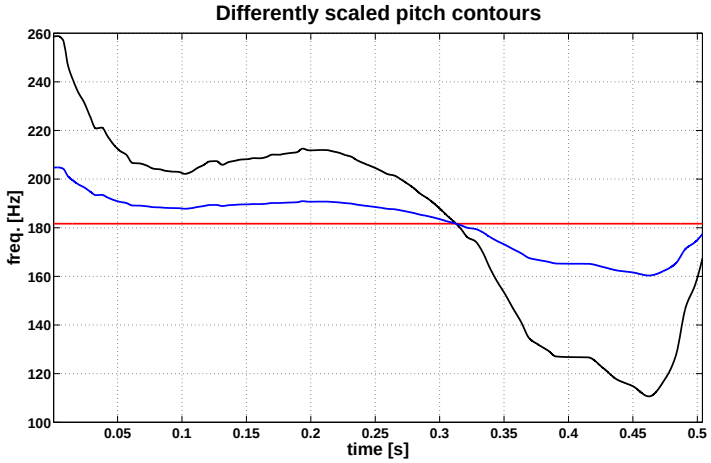

Figure 14: Pitch contours of a short sound sample scaled with different scale factors (black: factor = 1; blue: factor = 0.3; red: factor = 0).

The following processing step represents the core of the synthesis algorithm, i.e., the signal sequence gets generated. All voiced parts of the analyzed signal are processed consecutively within a `for`-loop. In doing so, the scaled contour of the extracted pitch is synthesized as a sine wave modulated in frequency. In intervals of 20 ms, the current pitch value is extracted from the scaled pitch contour and the sine wave's period gets modified accordingly. As these consecutive sine wave segments of 20 ms are created separately, the wave's offset in phase has to be computed for each connection point to guarantee a continuous crossover from one synthesis segment to the next. Furthermore, the synthesized signal values are smoothed over 200 µs inside windows of 400 µs placed centered above these segment borders. Each synthesized voiced part is faded to zero at the end using half a triangular window with a length of 2 ms. In figure 15, the synthesized signal of a voiced part of a speech sample can be regarded.

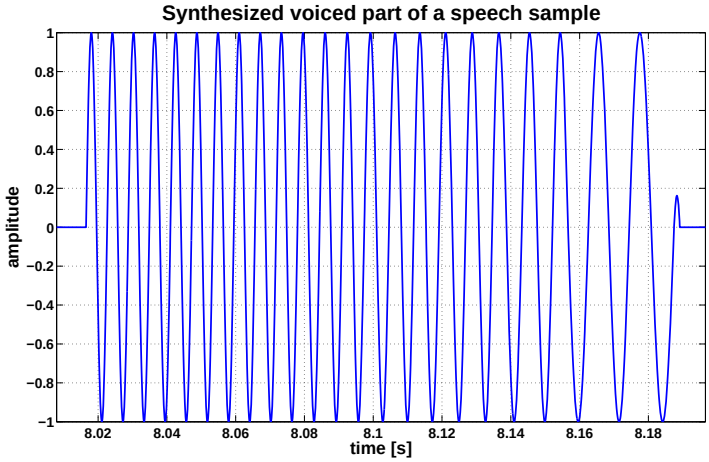

Figure 15: Voiced part of a speech sample synthesized based on its original pitch and rhythm contour.

Unvoiced parts of the signal - exceeding an adjustable threshold in intensity - are synthesized by pink noise. Pink noise sequences are extracted by filtering white noise that is generated using the MATLAB function `randn`. Figure 16 shows both the magnitude and phase responses and the pole-zero plot of the used 3^{rd} order IIR-filter that causes the desired decrease of -3 dB/*octave*. Furthermore, a white noise signal before and after the filtering process is illustrated.

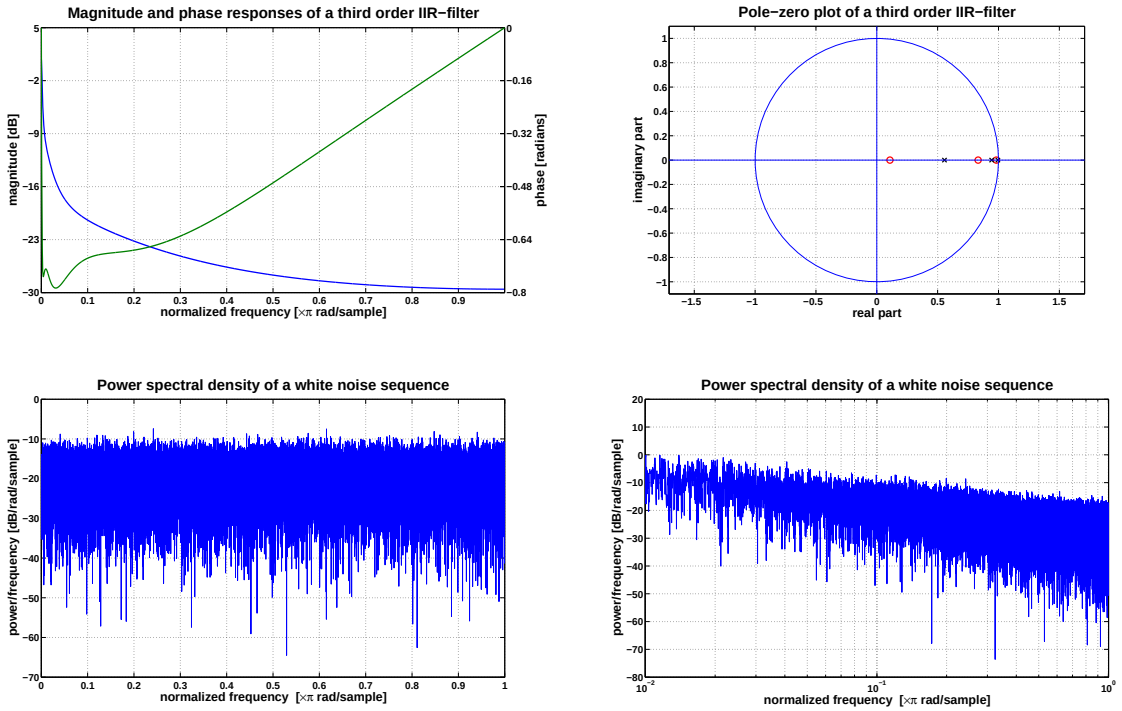

Figure 16: The magnitude and phase responses of the 3^{rd} order IIR-filter used for extracting pink noise from white noise is shown in the upper left. Its pole-zero plot in the upper right. The power spectral density of a white noise sequence - estimated via a periodogram - can be regarded on the bottom left and the same sequence run through the filter on the bottom right in a logarithmic scale.

The pink noise signal is normalized to -0.1 dBFs.

3.3 Scaling and Synthesis of the Intensity Contour

In the last step of the synthesis algorithm, the extracted and interpolated intensity contour is applied to the signal. At first it is normalized to the maximum absolute value of the input signal's waveform. Subsequently, scaling of the contour dependent on the value of an adjustable scale factor is performed in the same way, as already described in section 3.2 concerning the scaling of the pitch contour. Thus, a scale factor of 1 means that the original intensity contour is kept, whereas a factor of 0 causes a constant contour at the height of the original contour's mean value. Figure 17 illustrates the intensity contours of a short sound sample, again scaled with different factors.

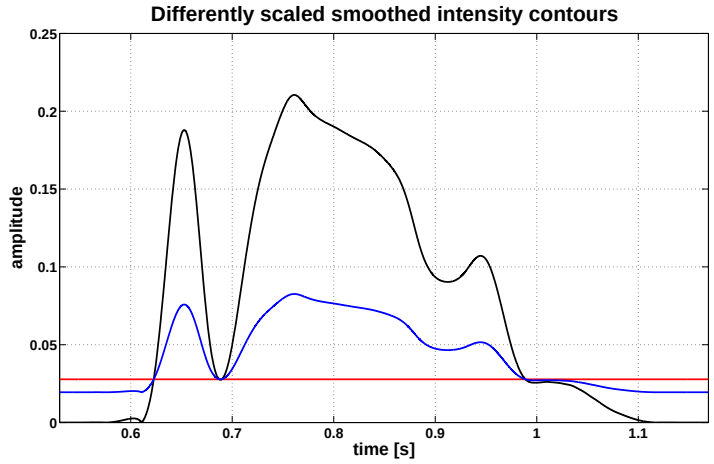

Figure 17: Intensity contours of a short sound sample scaled with different scale factors (black: factor = 1; blue: factor = 0.3; red: factor = 0).

If desired, the scaled intensity contour can get normalized to -0.1 dBFs for a more comfortable playback later on. Finally, the generated signal - consisting of regions of silence, regions of pink noise and regions of sine waves modulated in frequency - is weighted in intensity by an element-wise multiplication of the signal with the scaled and possibly normalized intensity contour. Figure 18 shows the final synthesized output signal of a short speech sample.

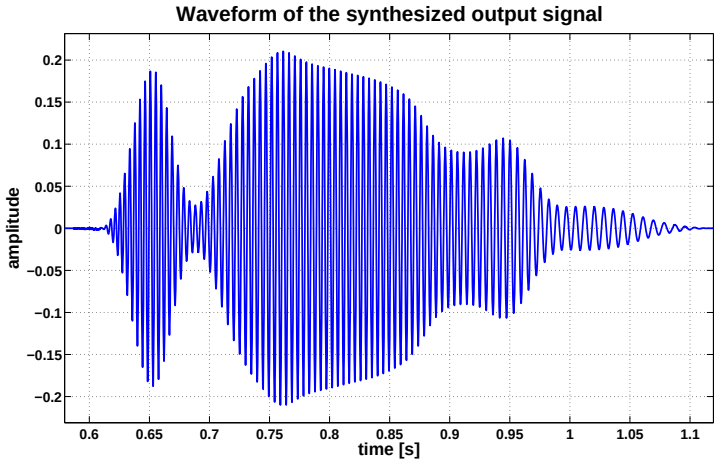

Figure 18: Speech sample synthesized based on its original pitch, intensity and rhythm contour.

4 Graphical User Interface

In order to offer a comfortable use of the analysis and synthesis tool, a GUI has been created. To guarantee an error-free use of the GUI, the following files have to constitute the contents of MATLAB's current directory:

```
documentation.pdf
get_pitch.praat
praatcon.exe
prosody.fig
prosody.m
prosody_info.fig
prosody_info.m
prosody_run.m
```

Furthermore, a folder named *pictures* contains the required *jpg*-files for the GUI. This folder also needs to be located inside the current directory. The GUI gets started by running the MATLAB script `prosody_run.m`. Figure 19 illustrates its start screen. Moreover, the single functional areas of the GUI are marked by numbered and colored arrows.

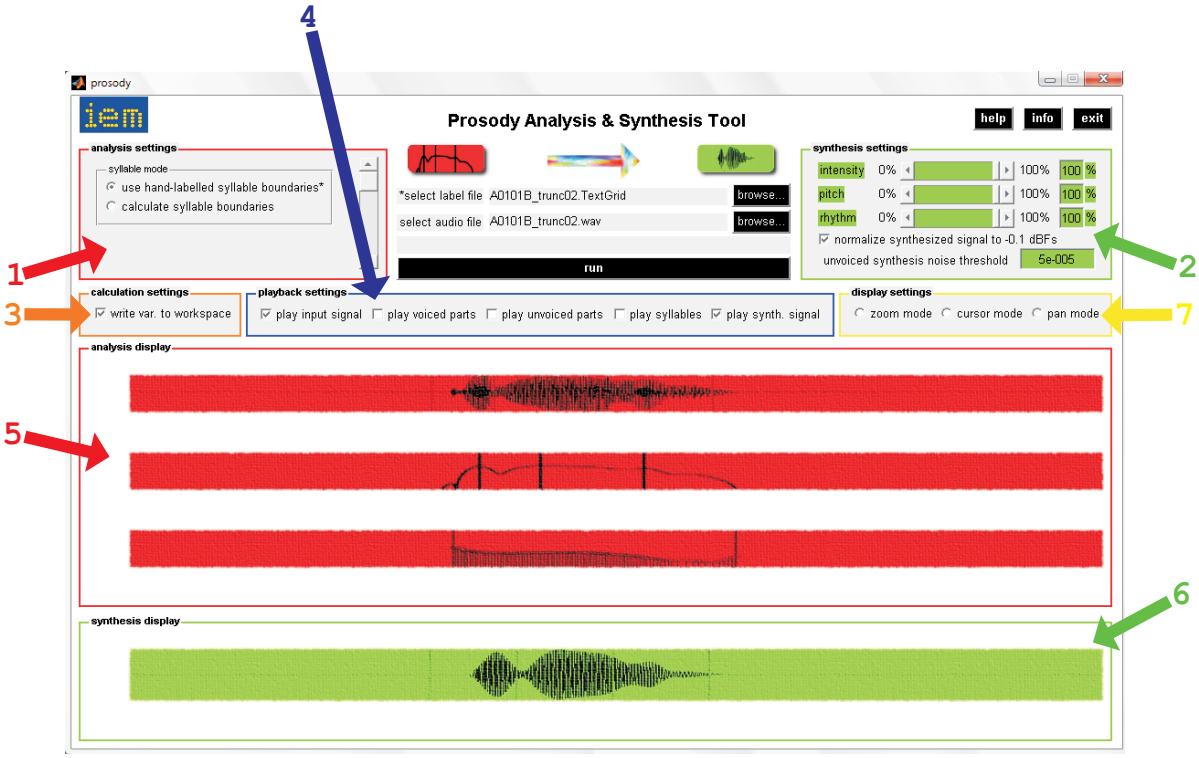

Figure 19: Start screen of the GUI with marked functional areas.

Apparently, functional areas are bordered in different colors. Red objects are always related to the analysis part of the tool, whereas green objects concern the synthesis part of it. Push-buttons are colored in black. The elongated gray area above the *run* button centered in the upper part of the GUI serves as a message window. Immediately above, an arbitrary input sound file in *wav* format and if available, the corresponding label file in *TextGrid* format - containing the locations of syllable boundaries - can be selected within a file dialog box by pushing the respective *browse...* buttons. The initial file path in the dialog box is MATLAB's

current directory. It is important to keep in mind the convention that the names of the sound file and the label file have to equal each other except for the ending. Furthermore, the sampling rate of the input sound file has to fulfill the conditions already mentioned in section 2.

The red panel on the upper left - marked by the red arrow number 1 - contains all setting values concerning the analysis part of the tool. At first, the syllable extraction mode can be chosen via radio buttons. In case of not choosing the calculation mode, a valid label file - containing the locations of syllable boundaries - has to be selected. By moving the slider on the right of the panel, the further analysis setting get accessible to adjust. The functions of all those setting values have been described in detail in section 2.

The panel on the upper right of the GUI - colored in green and marked by the green arrow number 2 - contains the synthesis settings. The intensity scale factor, the pitch scale factor, as well as the rhythm scale factor can be adjusted between 100% and 0% by using the respective slider, or by typing the number into the respective green number box. Via a checkbox, it can be chosen if the synthesized signal gets normalized to -0.1 dBFs. Moreover, the noise threshold that determines which unvoiced regions of the signal get synthesized with pink noise - mentioned in subsection 3.2 - can be adjusted within this panel.

All default setting values have been determined experimentally in order to achieve optimal results when using sound files from the *Aix-MARSEC Project* corpus. In cases of violations of any conventions while changing settings, the message window displays a warning message.

By ticking the checkbox within the orange panel marked by the orange arrow number 3, all important variables get written into MATLAB's workspace during the next calculation pass.

The blue panel in the center that is marked by the blue arrow number 4 contains several checkboxes. The user can tick those signals he wants to listen to. The original input signal, a signal just containing the voiced parts of the input signal, another one just containing the unvoiced parts of it, a signal with the emphasized detected syllable boundaries and the synthesized output signal are available. In case of more than one selection, the respective signals are played in the listed order.

By pushing the black *run* button the calculation gets started and the required computation steps of the analysis and synthesis tool get executed according to the selected settings.

The red arrow number 5 and the green arrow number 6 mark those areas, where analysis and synthesis results get displayed. Inside the yellow panel - marked by the yellow arrow number 7 - the zoom mode, the cursor mode, as well as the pan mode are available for a more comfortable viewing of plotted results.

By pushing the *help* button at the top right of the GUI, the documentation at hand gets opened as representing a kind of user manual in extracts.

Figure 20 shows the window that gets opened by pushing the *info* button next to the *help* button.

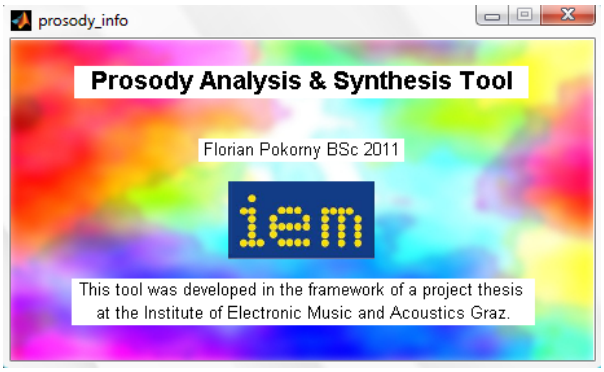

Figure 20: Info screen of the GUI.

Figure 21 illustrates the GUI immediately after a calculation pass.

Figure 21: GUI immediately after a calculation pass.

By pushing the *exit* button top right, the GUI gets closed.

5 Evaluation

The use of the tool allows to compare the effects of differently combined and scaled magnitudes of prosodic features in a speech signal. On the one hand, the analyzed features as well as the synthesized output signal can be examined visually by surveying the plots. On the other hand, a perceptive evaluation of the extracted results is possible by listening to the respective signals. Figure 22 illustrates the output signal synthesized based on all scale factors set to 1.

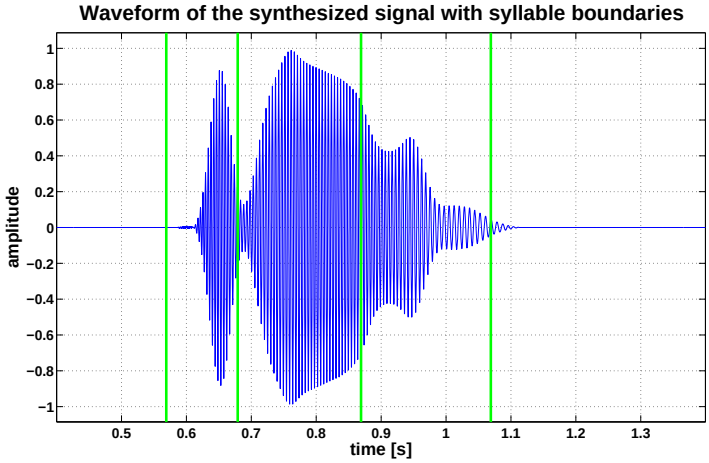

Figure 22: Output waveform synthesized based on the pitch, intensity and rhythm scale factors set to 1.

The shown signal contains the unscaled analyzed prosodic information of the input sound file. Hence, a considerable correlation between the synthesis signal and the original input signal is perceivable when listening to them. Furthermore, it is remarkable, how much of the prosodic information of an utterance is perceived consciously in case of a loss of the semantic information.

In figure 23, another output signal can be seen. However, now the scale factors have been set to 0 for the synthesis.

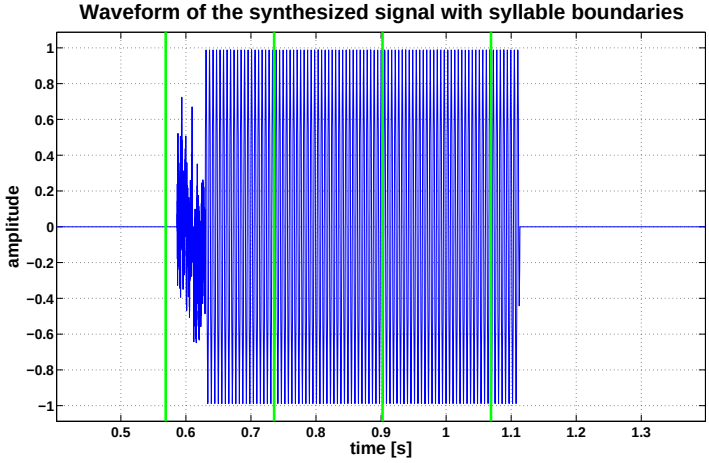

Figure 23: Output waveform synthesized based on the pitch, intensity and rhythm scale factors set to 0.

In contrast to the previous signal example, the whole prosodic information got lost in this case. As a respective prosodic run rarely occurs in everyday speech, the signal sounds very unusual.

In the end, figure 24 shows an output signal synthesized based on the scale factors set to 0.3.

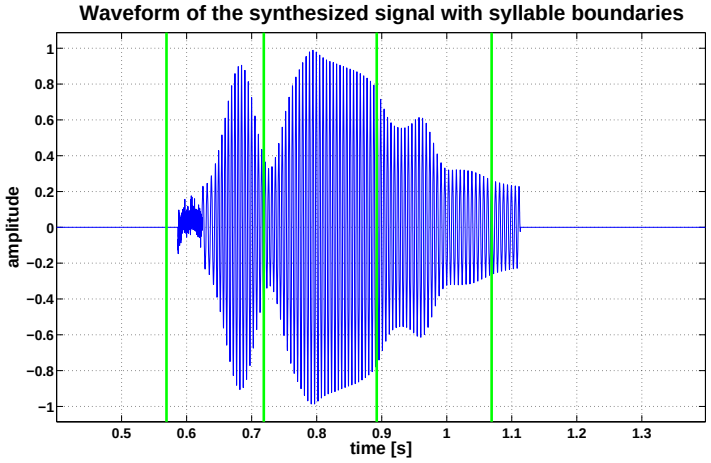

Figure 24: Output waveform synthesized based on the pitch, intensity and rhythm scale factors set to 0.3.

There is a perceivable correlation between this signal and the corresponding original input signal, even though the synthesized signal sounds monotonous and unemotional.

It is noticeable that the unvoiced parts - synthesized with pink noise sequences - come to the fore more and more when continually reducing the intensity scale factor. The influence of the scaled intensity contour on the synthesis signal seems to be weaker in perception than the influences of rhythm and pitch. Thus, a variation of the intensity scale factor is perceived the easiest when the other prosodic features are minimized.

5.1 Comparison of Hand-Labeled and Calculated Syllable Boundaries

As the algorithm for the extraction of syllable boundaries represents an important but also sensitive and complex implementation part of the analysis and synthesis tool, its performance rate got evaluated by comparing detected syllable boundaries to hand-labeled boundaries from the *Aix-MARSEC Project* corpus. The detection results from 10 sound files (see table 1) have been examined. In a first pass, syllable boundaries were calculated by using the algorithm with the default settings. For a second pass, the approximate number of syllables per analysis segment, the syllable weighting factor as well as the syllable threshold factor (see subsection 2.3.1) have been modified. The following table 1 presents an overview of the used sound files and some of their attributes.

	file name	sex of speaker	length [s]	# labeled bounds
	A0101B.wav	female	58.64	310
	A0102B.wav	female	56.82	284
	A0103B.wav	female	59.57	285
	A0104B.wav	female	53.83	246
	A0105B.wav	female	60.33	291
sum			**289.19**	**1416**
	A1101G.wav	male	53.12	264
	A1102G.wav	male	57.22	305
	A1103G.wav	male	59.77	337
	A1104G.wav	male	60.58	309
	A1105G.wav	male	33.2	171
sum			**263.89**	**1386**
total sum			**553.08**	**2802**

Table 1: Sound files used for the performance evaluation of the syllable detection algorithm taken from the *Aix-MARSEC Project* corpus (q.v. [ABH04]).

As shown in table 1, 5 recordings from a female speaker and 5 recordings from a male speaker with a total number of 2802 hand-labeled syllable boundaries have been selected. The locations of the hand-labeled bounds served as reference locations representing the real boundaries. In order to evaluate the locations of the detected bounds, the number of *true positives* (= TPs) and *false negatives* (= FNs) was counted by scanning an adjustable tolerance interval Δ round the hand-labeled locations. The presence of at least one detected bound within such an interval represented one TP, an absence represented one FN. The number of *false positives* (= FPs) was calculated by subtracting the number of TPs from the number of all detected bounds. Hence, the FPs represented those detected boundaries that were not situated within such a tolerance interval round the reference locations. Listing 2 shows the routine for the classification into TPs, FNs and FPs.

Listing 2: Classification into TPs, FNs and FPs

```
TP = 0;
FN = 0;
candidates1 = bounds_detected-delta;
candidates2 = bounds_detected+delta;
for k = 1:length(bounds_labeled)
    temp = sum((bounds_labeled(k) >= candidates1) & (bounds_labeled(k) <= candidates2));
    if temp > 0
        TP = TP+1;
    else
        FN = FN+1;
    end
end

FP = length(bounds_detected)-TP;
```

The following figure 25 illustrates a part of a sound file from table 1 above with both the hand-labeled and the detected boundary locations. Furthermore, the classification into TPs, FNs and FPs is shown.

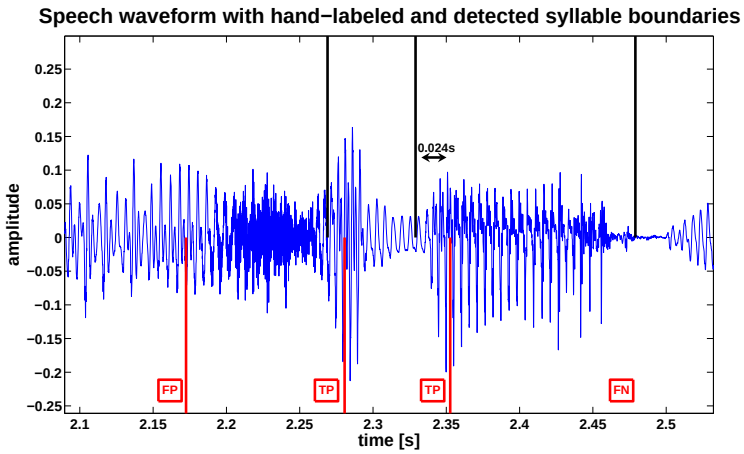

Figure 25: Waveform of a segment of speech with both the hand-labeled syllable boundaries in black color and the detected ones in red color. The classification into TPs, FNs and FPs holds for a tolerance interval Δ of at least 0.024 s.

For the evaluation of the detection algorithm's performance rate, the quantifying parameters *precision* (= P) and *recall* (= R) as well as the *F-measure* have been computed according to Makhoul et al. [MKSW99, p.250].

$$P = \frac{\#\,TPs}{\#\,TPs + \#\,FPs} \tag{3}$$

$$R = \frac{\#\,TPs}{\#\,TPs + \#\,FNs} \quad {}^{2} \tag{4}$$

$$F = \frac{PR}{(1-\alpha)P + \alpha R} \overset{\alpha=0.5}{=\!\!=} \frac{2PR}{P+R} \tag{5}$$

P demonstrates the relation between the number of correctly detected syllable boundaries and the total number of detected boundaries, i.e., how many of the detected boundaries are classified as correct. In contrast, R depicts the relation between the number of correctly detected boundaries and the number of reference boundaries. Hence, a high *recall* can also be achieved by just generating a very high number of arbitrary boundary locations in time.

In order to obtain one single quantifying parameter by combining P and R, the *F-measure* is defined as their weighted harmonic mean [MKSW99, p.250]. As representing the most popular value, α was selected to equal 0.5 [MKSW99, p.250].

In addition, the *correct detection ratio* (= CDR) has been calculated according to Luig [Lui09].

$$CDR = \frac{\#\,TPs}{\#\,TPs + \#\,FNs + \#\,FPs} \tag{6}$$

[2] also used in Aversano et al. [AEEM01, p.518]

Table 2 shows the results for the syllable boundaries detected using the algorithm with the default settings and for different values of Δ.

female speaker	$\Delta = 0.02$ s	$\Delta = 0.04$ s	$\Delta = 0.08$ s
P	0.17	0.35	0.55
R	0.23	0.48	0.76
F	0.2	0.4	0.64
CDR	0.11	0.25	0.47
male speaker	$\Delta = 0.02$ s	$\Delta = 0.04$ s	$\Delta = 0.08$ s
P	0.16	0.33	0.62
R	0.2	0.41	0.76
F	0.18	0.37	0.68
CDR	0.1	0.23	0.52
mixed speakers	$\Delta = 0.02$ s	$\Delta = 0.04$ s	$\Delta = 0.08$ s
P	0.16	0.34	0.58
R	0.22	0.45	0.76
F	0.19	0.39	0.66
CDR	0.1	0.24	0.49

Table 2: Results for the syllable boundaries detected using the algorithm with the default settings ($number\ of\ syllables/analysis\ window = 5$, $syllable\ weighting\ factor = 1$, $syllable\ threshold\ factor = 0.001$) for different tolerance intervals Δ.

In table 3 the evaluation results for syllable boundaries detected using different settings as before are shown.

female speaker	$\Delta = 0.02$ s	$\Delta = 0.04$ s	$\Delta = 0.08$ s
P	0.23	0.44	0.7
R	0.21	0.4	0.63
F	0.22	0.41	0.67
CDR	0.12	0.26	0.5
male speaker	$\Delta = 0.02$ s	$\Delta = 0.04$ s	$\Delta = 0.08$ s
P	0.25	0.43	0.67
R	0.26	0.46	0.7
F	0.25	0.44	0.68
CDR	0.14	0.29	0.52
mixed speakers	$\Delta = 0.02$ s	$\Delta = 0.04$ s	$\Delta = 0.08$ s
P	0.24	0.44	0.68
R	0.23	0.43	0.67
F	0.24	0.43	0.68
CDR	0.13	0.27	0.51

Table 3: Results for the syllable boundaries detected using the algorithm with varied settings ($number\ of\ syllables/analysis\ window = 10$, $syllable\ weighting\ factor = 2$, $syllable\ threshold\ factor = 0.01$) for different tolerance intervals Δ.

The slightly higher values of the CDR from the second pass compared to those from the first pass are caused by a reduction of the number of FPs as a consequence of the higher *syllable threshold factor*.

In principle, the results at hand seem to imply a very low performance rate of the implemented syllable detection algorithm. However, it should be kept in mind that the algorithm is about

an absolutely blind syllable detection algorithm for continuous speech that exclusively obtains the approximate number of syllable nuclei as an auxiliary information. By modifying the method for extracting these syllable nuclei and by introducing some adaptive routines for the selection of the best setting parameters for each single analysis segment of an input signal, the algorithm's performance rate can presumably be increased.

6 Conclusion and Outlook

The created analysis and synthesis tool is a valuable instrument to examine the influences of prosodic features on the information content transported via spoken language. The GUI permits a comfortable use of the tool. Since the prosodic features pitch, intensity and rhythm are considered separately, the individual role and perceptive dominance of each of them can be discovered by the user.

Certainly, there are several implementation steps within the tool that need to get modified and improved for more robust and accurate results. For instance, the algorithm for extracting the syllable boundaries suffers from inaccuracy and a rather high proportion of wrong detections. Hence, the use of adaptive cycles to determine the best adjustable calculation parameters depending on the quality of the input sound file could yield an advancement.

Furthermore, the necessity of A-weighting the input intensity has been contemplated in the course of the implementation.

Finally, as the documented tool just synthesizes the fundamental frequency of an input sound file, the re-synthesis of the whole speech signal after scaling the extracted prosodic features represents a highly interesting challenge for future work. For this purpose, the methods presented in McAulay and Quatieri [MQ86] constitute a suitable approach for the extension of the tool at hand.

References

[ABH04] Auran, Cyril and Bouzon, Caroline and Hirst, Daniel. The aix-marsec project: An evolutive database of spoken british english. In *Speech Prosody 2004, International Conference*. ISCA, 2004.

[AEEM01] Aversano, Guido and Esposito, Anna and Esposito, Antonietta and Marinaro, Maria. A new text-independent method for phoneme segmentation. In *Circuits and Systems, 2001. MWSCAS 2001. Proceedings of the 44th IEEE 2001 Midwest Symposium on*, volume 2, pages 516–519. IEEE, 2001.

[Aki11] Akien, Geoffrey. peakdet(v, delta, x), MATLAB Central 2011. http://www.mathworks.com/matlabcentral/fileexchange/26288-co2gui-lab-control-andautomation/content/peakdet.m (November 11, 2011).

[Boe93] Boersma, Paul. Accurate short-term analysis of the fundamental frequency and the harmonics-to-noise ratio of a sampled sound. In *Proceedings of the Institute of Phonetic Sciences*, volume 17, pages 97–110, 1993.

[Boe11] Boersma, Paul. *Praat Manual: Version 5.2.18*, 2011.

[BW11] Boersma, Paul and Weenink, David. Praat: doing phonetics by computer [computer program], 2011. http://www.praat.org/ (November 11, 2011).

[CY95] Clark, John and Yallop, Colin. *Phonetics and Phonology*. Blackwell Publishing Ltd, 2nd edition, 1995.

[Gra06] Grassegger, Hans. *Phonetik Phonologie*. Jürgen Tesak / Schulz-Kirchner Verlag, 3rd edition, 2006.

[Lui09] Luig, Johannes. Investigations on a robust feature set for classification of speech under stress. Master's thesis, Graz University of Technology, 2009.

[MKSW99] Makhoul, John and Kubala, Francis and Schwartz, Richard and Weischedel, Ralph. Performance measures for information extraction. In *Broadcast News Workshop'99 Proceedings*, pages 249–252. Morgan Kaufmann Pub, 1999.

[MQ86] McAulay, Robert J. and Quatieri, Thomas F. Speech analysis/synthesis based on a sinusoidal representation. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 34(4):744–754, 1986.

[MY91] Murthy, Hema A. and Yegnanarayana, B. Formant extraction from group delay function. *Speech Communication*, 10(3):209–221, 1991.

[PNM04] Prasad, Kamakshi V. and Nagarajan, T. and Murthy, Hema A. Automatic segmentation of continuous speech using minimum phase group delay functions. *Speech Communication*, 42(3-4):429–446, 2004.

[Sar11] Saragiotis, Christos. pickpeaks(x,npts,dim,mode), MATLAB Central 2011. http://www.mathworks.com/matlabcentral/fileexchange/27811-peaks-picking/content/pickpeaks.m (November 11, 2011).

[TM05] The MathWorks, Inc. *MATLAB Help: Version 7.0.4.365 (R14) Service Pack 2*, 2005.

Appendix

Creation of Text Files with Syllable Boundary Locations

Locations of external detected syllable boundaries can be imported from a text file with the ending *.TextGrid*. They have to be saved as a column of time values in seconds. It is important to consider the formatting that the first time value constituting a syllable boundary is located in the fifth row below the expression *Syllables*. The last value has to be located in the third row above the next characteristic's caption *Abercrombie*. Possible transcriptions within the desired data section are removed automatically. Time values that occur twice due to a possible syllable-wise formatting style always marking the starting point and the end point of one syllable are reduced to one single occurrence using the MATLAB function `unique`.

Listing 3 shows an excerpt of a valid text file fulfilling all mentioned format conventions. It has been taken from the *Aix-MARSEC Project* corpus. The three dots inside the listing just mark the continuation of the data column in each case. They are not part of the original text file.

Listing 3: Excerpt of the label file `A0101B.TextGrid`

```
...
57.651000000000003
58.6417
""
"IntervalTier"
"Syllables"
0
58.6417
311
0
0.56899999999999995
"_"
0.56899999999999995
0.67900000000000005
"' g U d"
0.67900000000000005
0.869
"' m O:"
0.869
1.069
...
"' D @U"
57.097000000000001
57.207000000000001
"D I"
57.207000000000001
57.651000000000003
"END"
57.651000000000003
58.6417
""
"IntervalTier"
"Abercrombie"
0
58.6417
173
0
0.56899999999999995
...
```