GlottHMM

Analysis and synthesis software for statistical parametric speech synthesis

Version 1.0.9
June 18, 2012

Tuomo Raitio
tuomo.raitio@aalto.fi

Contents

1 Introduction

This manual describes speech analysis and synthesis tool GlottHMM. GlottHMM is primarily intended to be used as a vocoder in statistical parametric speech synthesis, but it can be used also for speech analysis and modification. GlottHMM is based on the source-filter theory of speech production, but the main difference compared to other common vocoder techniques is that it utilizes glottal inverse filtering in order to separate the contributions of the voice source and the vocal tract filter. This physiologically oriented speech parametrization technique aims towards high-quality and flexible speech synthesis.

1.1 Background

In 2007, a collaborative project between Aalto University and University of Helsinki was begun in order to develop a physiologically motivated vocoder for hidden Markov model (HMM) based speech synthesis. In 2008, as a result of a Master's thesis, the first version of GlottHMM was born. Since then, GlottHMM has been developed further to a full scale vocoder.

At the moment, GlottHMM is published only as an internal version for research purposes, but the aim is to publish GlottHMM for public use at a later date. The authors wish that the GlottHMM package will foster both research and commercial development of speech synthesis technology, and thus to publish the program as open source if possible.

1.2 License and distribution

This version of GlottHMM is intended to be used within the EU's FP7 project *Simple4All*, and it can be used freely for research purposes. The redistribution of this version of the program is not encouraged. The GlottHMM package is intended to be released for public use at a later date.

For possible commercial use, the author wishes to note that the GlottHMM package contains methods that may be under a pending patent application by Nokia Corporation. For more information, see "Method, apparatus and computer program product for providing improved speech synthesis" (application number: FI2009/050414). However, the intention is to release GlottHMM as free software, but the form of the license is yet undetermined (at least MIT or BSD based license will be provided). For more information about the license or distribution, contact one of the authors (see next section).

Note that GlottHMM is not a finished product and many feature combinations have not been tested. Simple4All participants are encouraged to experiment with the system and provide feedback.

1.3 Authors and contact

The GlottHMM program is mainly written by Tuomo Raitio, with a great deal of help and advice from Antti Suni, who has been developing the program especially from the statistical point of view.

Tuomo Raitio
Department of Signal Processing and Acoustics
Aalto University, Espoo, Finland
tuomo.raitio@aalto.fi

Antti Suni
Institute of Behavioural Sciences
University of Helsinki, Helsinki, Finland
antti.suni@helsinki.fi

1.4 Acknowledgements

The authors would like to thank Martti Vainio and Paavo Alku who have been successfully leading this research. The authors would also like to thank all other collaborators during the project. The

research has been funded by the Academy of Finland, Aalto University, European Community's Seventh Framework Programme (Simple4All project), Tekes, Nokia, Nokia Foundation, Emil Aaltonen Foundation, HPY Research Foundation, and the Research and training foundation of TeliaSonera Finland Oyj.

2 Getting started

This section gives the overview of GlottHMM and helps to get started with it. Sections 3, 4, and 5 give more detailed description of the current implementation of GlottHMM. The GlottHMM vocoder and results are also described in several conference papers [1, 2, 3, 4] and in one journal article [5], but this document describes the latest version of GlottHMM in detail.

GlottHMM consists of three main components: Analysis, Synthesis, and a pulse library constructor. Analysis is a program that reads speech files and converts them into suitable speech parameters and extracts glottal pulses. Synthesis reads the speech parameters given by Analysis or HMMs and reconstructs synthetic speech using either single glottal flow pulse or a pulse library. The pulse library can be constructed by a Matlab function that links the extracted pulses and the pulse parameters. The files of the GlottHMM package are listed in Table 1.

2.1 Installing

GlottHMM is mostly written in standard C. The following libraries outside standard C are required for compiling and using Analysis and Synthesis:

- GNU Scientific library (GSL) v.1.15 (`www.gnu.org/software/gsl`)
- Libconfig v.1.3.2 (`www.hyperrealm.com/libconfig`)
- Libsndfile v.1.0.21 (`www.mega-nerd.com/libsndfile`)

This version of GlottHMM works at least with the above mentioned versions of the libraries, other versions have not been thoroughly tested.

For GCC C linker, the following libraries must be included (-l): `m, gsl, config, sndfile, gslcblas`. Files `create_pulse_library.m` and `plot_params.m` are Matlab functions, and require Matlab to be installed. The functions are tested with Matlab version R2011b, but they should work with earlier versions as well.

Component	Filename	Description
Analysis	Analysis.c	Source code for speech analysis (main)
	AnalysisFunctions.c	Source code for speech analysis (functions)
	AnalysisFunctions.h	Source code for speech analysis (header file)
	hp_16khz	High-pass filter coefficients for 16 kHz speech
	hp_44khz	High-pass filter coefficients for 44.1 kHz speech
	hp_filter_design.m	Matlab function for designing filters
Synthesis	Synthesis.c	Source code for speech synthesis (main)
	SynthesisFunctions.c	Source code for speech synthesis (functions)
	SynthesisFunctions.h	Source code for speech synthesis (header file)
	gpulse	Glottal pulse file
An/Syn	config_def	Default configuration file
Pulselib	create_pulse_library.m	Matlab function for creating pulse libraries
Add-on	plot_params.m	Matlab function for illustrating speech parameters
Help	glotthmm_manual.pdf	GlottHMM manual (this file)

Table 1: Files in the GlottHMM package.

In addition, configuration file for Analysis and Synthesis need to be modified according to your environment, e.g., defining paths for certain files. Also Matlab function `create_pulse_library.m` need to adjusted to the environment.

2.2 Analysis

Both Analysis and Synthesis are evoked by executing them from the command line. As input, both Analysis and Synthesis require a file name to be analyzed/synthesized and a configuration file in which the Analysis/Synthesis process is defined. The configuration settings are described detailed in Appendix A.

For example, speech parameters of `speechFile.wav` are extracted by executing the following command:

```
>> Analysis speechFile.wav configurationFile
```

As a results, speech features are written to parameter files `speech_file.speechFeatureX`. The extracted speech features can be defined in the configuration file. The data format of the parameter files is either ASCII or binary, defined in the configuration file. Descriptions and help of Analysis is evoked by executing the program without parameters.

Both Analysis and Synthesis can be given two different configuration files, one for default parameters, and one for user defined parameters. Thus, the most important and frequent settings can be defined in an additional user configuration file, which override the default configurations. For example, the following command line would run Analysis with the user configuration settings:

```
>> Analysis speechFile.wav configDef configUsr
```

The default configuration file must include all the settings, but the user configuration file may contain only part of the settings, or need not to be given at all. The use of two separate configuration files is intended for easy tuning of the most common parameters in user configuration file, while the others can be left unchanged in default configurations. The most important settings in the configuration for Analysis are for example

- Define sampling frequency
- Define order of the all-pole model
- Define voicing thresholds (zero-crossings, gain, max/min F0)
- Define spectral modeling technique
- Define extracted speech features

2.3 Synthesis

Speech defined by the parameters `speechFile.speechFeatureX` is reconstructed by executing the following command:

```
>> Synthesis speechFile configDef configUsr
```

As a result, Synthesis reads the speech features, and synthesizes a speech file `speechFile.syn.wav`. The input parameters for Synthesis can be defined in the configuration files, similarly to Analysis.

Descriptions and help of Synthesis is evoked by executing the program without parameters. The most important settings in the configuration for Synthesis are:

- Select between analysis-synthesis and HMM synthesis using parameter `USE_HMM`. Speech parameters are smoothed for analysis-synthesis.
- Define used speech features
- Select between single pulse technique vs. pulse library
- Select target/concatenation costs and parameter weights for pulse library
- Select post-filtering method and strength

2.4 Configuration files

There are two types configuration files that can be used with GlottHMM. The default configuration file, provided with the GlottHMM package, includes all the possible configuration parameters. The name of the default configuration file is given as a second argument in the command line. However, user defined configuration file may be given to Analysis and Synthesis (third argument), in which the most common or frequently changed parameters are given. The second configuration file will override the parameters in the default file. Thus, only the parameters that require change to the default value need to be defined separately, which makes the use of the program easier.

The configuration file format is defined in the libconfig library (`www.hyperrealm.com/libconfig`). Note that integer parameters must be defined without decimals (e.g. sampling frequency is 16000), and decimal numbers must include the decimal point (e.g. frame length in milliseconds is 25.0)!

Parameters in the configuration file are roughly divided into three categories: Common, Analysis, and Synthesis. Analysis and Synthesis both share common parameters, but the two latter ones are defined separately for Analysis and Synthesis. For detailed description of the configuration file parameters, see Appendix A.

2.5 Constructing and using pulse libraries

Pulse libraries can be constructed by using the Matlab function `create_pulse_library.m`. The function runs Analysis that extracts speech parameters from selected files, and gathers the pulses and corresponding parameters into a pulse library. In order to construct pulse libraries, the configuration setting `EXTRACT_PULSELIB` must be set to `true`, and user also needs to define paths to Analysis, original wav files, and configuration file, and set the number of extracted files.

Usually only a few wav files is enough for a pulse library. However, in order to get a representative set of glottal flow pulses, speech files with different speaking styles may be required. If the pulse library gets large and thus synthesis gets slower, the size of a large pulse library can reduced by k-means clustering which selects only N centroids of the pulse library.

Pulse libraries are used by setting the configuration file parameter `USE_PULSE_LIBRARY` to `true` and typing the name of the pulse library to parameter `PULSE_LIBRARY_NAME`.

3 Analysis

This section describes the operation of Analysis. Analysis is executed from the command line by typing `Analysis arg1 arg2 (arg3)`. The first argument is the name of the speech file and the second argument is the default configuration (config) file name. Optionally, a third argument can be given, defining the name of the user config file that overrides the parameters in the default config file. Before analysis, user must define e.g. the sampling frequency and the polarity of the speech signal in the config file.

In addition, a high-pass filter coefficient file is required. The purpose of the high-pass filter is to reduce low-frequency components that may cause large fluctuations in the estimated glottal flow signal. The high-pass filter should be a finite impulse response (FIR) filter that has a cut-off frequency from 30 to 80 Hz, depending on the speech data, and a roll-off large enough. In the GlottHMM package, a 301-tap FIR filter coefficient file is given for 16 kHz speech signals (`hp_16khz`) and a 901-tap FIR filter coefficient file for 44.1 kHz speech (`hp_44khz`). Different types of filters can be easily created for example with Matlab. A simple script for designing FIR filters with Matlab is also provided with the package (`hp_filter_design.m`).

3.1 Technical description

The flow chart of Analysis is shown in Figure 1. Analysis is based on frame-by-frame analysis of speech signals. In the beginning Analysis, the speech signal is high-pass filtered. The high-pass filtering coefficient file is defined in the config file (`HPFILTER_FILENAME`). The high-pass filtering can also be set on/off in the config file (`HP_FILTERING`).

After the high-pass filtering, the speech signal is windowed with two types of windows. A short frame (around 25 ms) is used to evaluate energy (in decibels) of the speech signal and vocal tract and voice source spectra. A longer frame (25–50 ms, depending on lowest F0) is used to evaluate other parameters, which need slightly longer analysis frame. The lengths of the frames can defined in the config file (`FRAME_LENGTH, F0_FRAME_LENGTH`) in milliseconds. The shift between two adjacent frames is defined in `FRAME_SHIFT` (around 5 ms).

Glottal inverse filtering (GIF), i.e., estimating the glottal source signal from a speech signal, is applied for both frames. Iterative adaptive inverse filtering (IAIF) (see Figure 2), or a modified version of it (see Figure 3) is used for GIF. GIF outputs the estimated vocal tract filter and the estimated voice source signal.

Inside GIF, various spectral modeling method can be utilized. Normally, basic linear prediction (LP/LPC) is used, but also weighted linear prediction (WLP) or extended linear prediction (XLP) may be used. Weighted linear prediction [6], or stabilized version of it, SWLP [7] use a function to weight the autocorrelation. Usually, the autocorrelation is weighted by the short time energy (STE) window of the signal, thus emphasizing high energy parts, but also other weights can be used, such as weight based on glottal closure instants (GCI). (S)WLP is beneficial especially with high-pitched (female) voices since the spectrum is less distracted by the harmonics of the excitation signal. (S)WLP may also give more accurate estimates of the vocal tract spectrum. XLP [8] has similar properties to WLP, but in the current implementation it uses only STE weight. Both methods, WLP and XLP can possibly produce unstable filters, and thus a stabilized version of them (SWLP, SXLP) can by used (`LP_STABILIZED`). However, the effect of weighting is reduced in the stabilization process. Thus, it is up to experimentation whether possible unstable or badly estimated filters will be averaged out in statistical modeling in order to produce stable and natural synthesis filters.

The order of all-pole modeling (LP) depends on the sampling frequency, but also on the speech material. Usually 20th–30th order model is appropriate for 16 kHz speech, but higher sampling rates may require higher orders from 30 to 50. However, it is better to use warping of the spectral model with higher sampling rates than increase the order (e.g., for 44.1 kHz speech, WARPING_VT

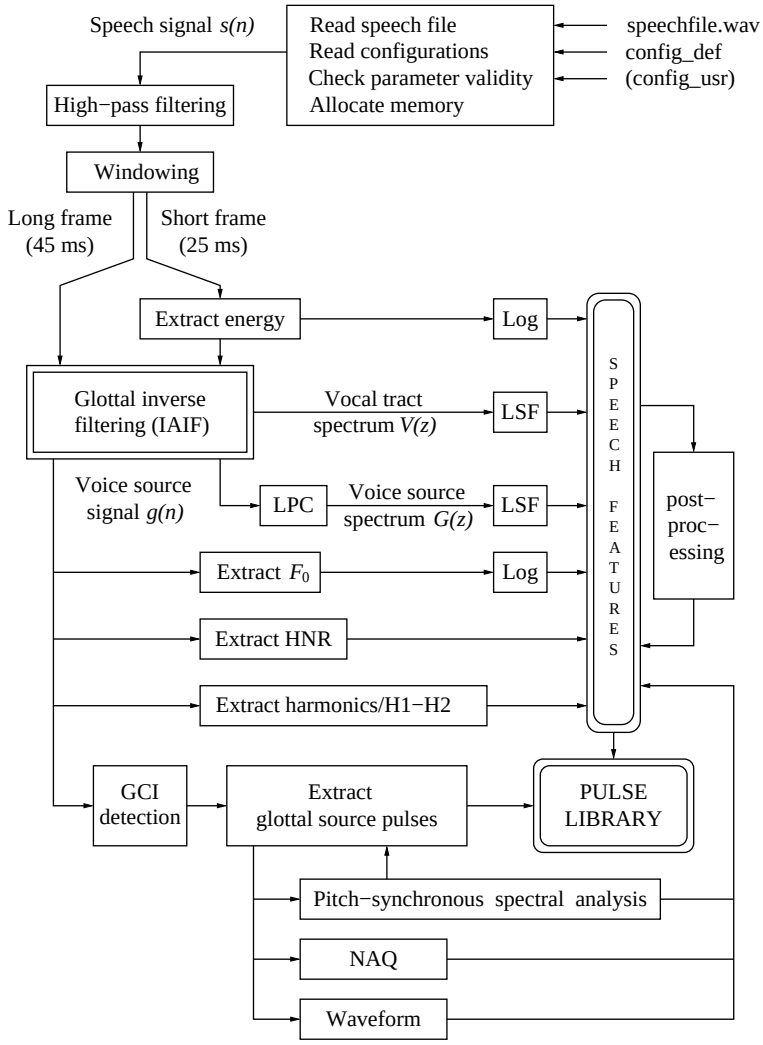

Figure 1: Flow chart of speech parametrization.

can be set to 0.4 and 30th order model can be used). The accuracy of LP analysis and statistical modeling decrease if the order is more than 50.

From the estimated voice source signal (short frame), LP is used to estimate the spectrum of the voice source, i.e., the spectral tilt and the more detailed spectral structure as well. This spectral model is used in synthesis either to vary the voice source spectrum (single pulse technique) or to select appropriate glottal pulses from library. Both vocal tract and voice source LP coefficients are converted to line spectral frequencies (LSF) which provide stability and low spectral distortion in statistical modeling.

The voice source signal estimated from the longer frame is used to estimate the rest of the speech features. First, fundamental frequency is estimated with the autocorrelation method. Although this is very simple technique, it yields robust estimates of F0 if used for glottal inverse filtered signal. The resolution of F0 estimation (sample based estimation) is increased by parabolic interpolation of the autocorrelation peak position. User must define the minimum and maximum F0 in the config

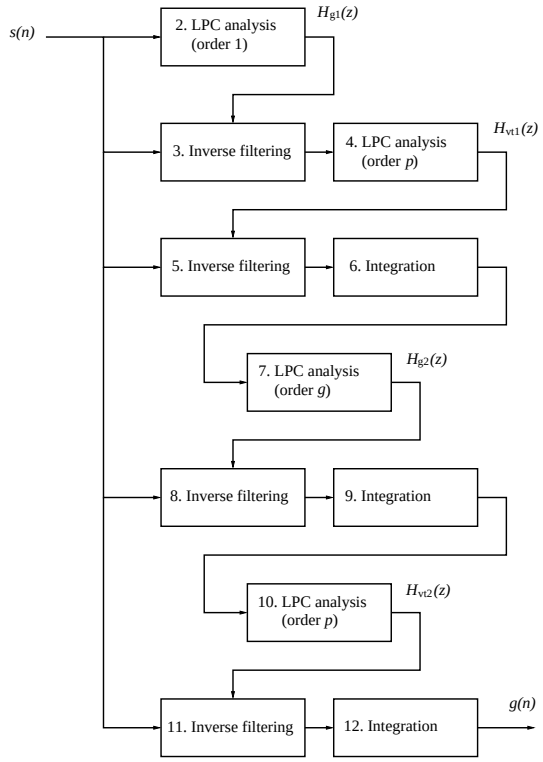

Figure 2: Iterative adaptive inverse filtering (IAIF) method.

file, and thus values outside this range are classified as unvoiced. Voicing threshold is used to help in voiced/unvoiced decision; frames whose low-frequency energy is less than the given threshold are classified as unvoiced. Also zero-crossing rate (ZCR) is used to help the classification; frames with ZCR values exceeding the defined threshold are classified as unvoiced. The F0 vector is finally median filtered, and small isolated voiced/unvoiced gaps are fixed. Finally, a post-processing algorithm is applied in order search for possible discontinuities (`RELATIVE_F0_THRESHOLD`) in F0 trajectories, and to estimate a new trajectory by weighted linear fitting over a defined number of F0 values (`F0_CHECK_RANGE`).

Alternatively, an external F0 file can be used as an input for the system. The input vector must be in ASCII format, each F0 value on a separate line. The F0 file need not to be exactly of correct length, but an interpolation for the F0 vector is applied. The interpolation process may produce occasional very small F0 values and thus the lowest F0 of the interpolated F0 vector is limited to F0_MIN.

A harmonic analysis is performed in order to extract the harmonic-to-noise ratio (HNR). This feature describes the breathiness or irregularity of the voice. First, fast Fourier transform (FFT) is performed to the signal frame, after which a peak picking algorithm is applied to find the harmonic peaks. The magnitudes of the peaks and the harmonic valleys are estimated. Envelopes of the peak and valley magnitudes are formed and smoothed. The difference between these vectors define the HNR at each frequency. The HNR values are averaged according to the equivalent rectangular bandwidth (ERB) scale to a specific number of frequency bands, defined in the config file (`HNR_CHANNELS`). In addition, the first N harmonic magnitudes are saved to describe the low-frequency properties of the source. The number of harmonic is defined in config file (`NUMBER_OF_HARMONICS`). The harmonic

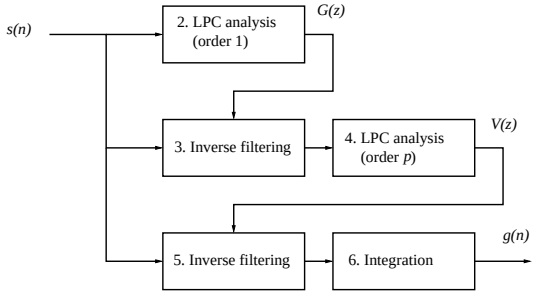

Figure 3: Modified version of the iterative adaptive inverse filtering (IAIF) method.

values are saved as a difference to the first harmonic magnitude in dB. Thus, the first value of the harmonics vector is the difference between the first and the second harmonic, called H1-H2. This value is also saved as a separate speech feature (H1H2).

Glottal closure instants (GCI) of the long frame are estimated first by searching for the minimum of the differentiated glottal volume velocity signal. The GCIs are located in the minima or very close to the minima of the inverse filtered signal. From this point, other GCIs are searched backward and forward at fundamental period (T0) intervals. If an appropriate minima is found in the vicinity of T0, it is classified as a GCI. Note that the polarity of the speech signal must be correct in order estimate GCIs correctly. The polarity can be changed by setting the config parameter `INVERT_SIGNAL` to `true`.

After GCI detection, each complete two-period glottal flow derivative is extracted. Although this segment is composed of two glottal flow pulses, the segment is simply called as *pulse* in this document. If the T0 of the pulse is not close enough to the F0, the pulse segment is discarded. The threshold for discarding pulses can be tuned with the config parameter `MAX_PULSE_LEN_DIFF`. The maximum length of the pulse is defined in `PULSEMAXLEN`. Accepted pulses are windowed with the Hann window so that the GCI of the two-period segment is in the middle, and thus they can be easily concatenated in the synthesis stage. The energy of the pulses is normalized and the pulses are stored to a matrix. In addition, a resampled versions of the pulses are stored in order to be able to compare the concatenation cost in the synthesis stage. The length of the resampled pulse is defined in `RESAMPLED_PULSELEN`.

Two additional parameters are extracted from each pulse. First, the basic shape of the pulse is stored by downsampling the pulse only to a few samples of length (around 20). The samples in the center (others are close to zero) are stored to matrix to describe the pulse shape. Secondly, the normalized amplitude quotient [9] of the pulses are estimated. These two parameters are used as a target cost in synthesis when selecting appropriate pulses from the library.

If config file parameter `PITCH_SYNCHRONOUS_ANALYSIS` is set to `true`, the glottal inverse filtering is performed pitch-synchronously for each pulse. This enables individual spectral parameters for each pulse instead of frame-wise parameters, and also more accurate estimates of the pulses. Additionally, pitch-synchronous analysis reduces the effect of the harmonics of the voice source to the vocal tract estimate. Thus, pitch-synchronous analysis is recommended especially for high-pitched voices. (S)WLP or (S)XLP is not used in pitch-synchronous analysis, regardless of the spectral modeling settings.

Finally, the speech features are post-processed using, e.g., median filtering in order to remove possible small errors and outliers.

3.2 Speech features

As a results of Analysis, speech features are written to parameter files `speech_file.speechFeatureX`. The extracted speech features can be specified in the configuration file. The data format of the parameter files is either ASCII or binary, defined in the configuration file. All possible speech features are listed in Table 2.

Additionally, if `EXTRACT_PULSELIB` is set to `true` for constructing a pulse library, the parameters listed in Table 3 are also extracted. The infofile in Table 2 contains 15 values which define some of the analysis settings. The structure of the infofile described in Table 4. Finally, the result of the glottal inverse filtering, the complete voice source signal, can be extracted by setting `EXTRACT_SOURCE = true`. The voice source is saved both to parameter `SourceSignal` and as a sound file `SourceSignal.wav`.

Speech feature	Description	Typical order
F0	Fundamental frequency	1
Gain	Energy of frame (dB)	1
LSF	Vocal tract spectrum	20–30
LSFsource	Voice source spectrum	10–20
HNR	Harmonic-to-noise ratio	5–10
Harmonics	Magnitude differences of harmonics	5–10
H1H2	Difference between H1 and H2	1
NAQ	Normalized amplitude quotient	1
Waveform	Downsampled waveform	10–20
infofile	Information about analysis	15 values

Table 2: Speech features and the typical order of the parameters.

Speech feature	Description
PULSELIB.pulses	Library pulses
PULSELIB.rspulses	Resampled library pulses
PULSELIB.pulselengths	Actual lengths of the library pulses
PULSELIB.pulseinds	Sample indices indicating the beginning time of the pulse in the original speech file
PULSELIB.pulsepos	Frame indices indicating from which frame the pulses were extracted
PULSELIB.gain	Energy of each pulse
PULSELIB.lsf	Vocal tract spectra of the pulse frame
PULSELIB.lsfsource	Voice source spectra of the pulse/pulse frame
PULSELIB.hnr	Harmonic-to-noise ratio of the frame
PULSELIB.harmonics	Magnitude differences of harmonics
PULSELIB.h1h2	Difference between H1 and H2
PULSELIB.naq	Normalized amplitude quotient of the pulse
PULSELIB.waveform	Downsamples waveform of the pulse

Table 3: Features for constructing pulse libraries.

Line	Description	Example value
1	Frame length (ms)	25.0
2	Frame shift (ms)	5.0
3	Number of extracted frames	535
4	LP order for vocal tract	30
5	LP order for voice source	20
6	Warping coefficient for VT	0.0
7	Warping coefficient for voice source	0.0
8	Number of HNR channels	5
9	Number of harmonics	10
10	Number of extracted pulses	1256
11	Maximum length of pulses (ms)	45.0
12	Length of resampled pulses (ms)	10.0
13	Number of samples in waveform parameter	10
14	Sampling frequency (Hz)	16000
15	Data format (1:ASCII, 2:Binary)	1

Table 4: Structure of the infofile.

3.3 Creating pulse libraries

GlottHMM package contains a Matlab script `create_pulse_library.m` for creating pulse libraries. The Matlab script executes Analysis and gathers all the pulse information in one folder.

First, the config parameter `EXTRACT_PULSELIB` must be set to `true` in the config file. Then paths to Analysis executable, working directory, wav-file directory, and the name of the config file need to be defined in the Matlab script. The number of files (N) from which the pulse library is constructed can be set, and the first N files in the wav-directory will be used for building the pulse library. The library will be saved to folder `pulse_libraries/pulse_library_name`. Since a single speech file usually contains a large number of glottal flow pulses, a reasonable size pulse library can be constructed with using only 5 to 10 speech files. However, in order to get a representative set of glottal flow pulses, speech files with different speaking styles may be required. Naturally, male voices contain much fewer pulses than female voices, and thus the size of the pulse library become very large with only a few female speech files. For large pulse libraries, binary data formant is recommended for faster reading in Synthesis. Since the glottal pulse library becomes large with only a few speech files, the pulse library might need some pruning. A simple k-means clustering method which selects only N centroids of the pulse library can be used for reducing the size of the pulse library, while trying to maintain the variety of different types of pulses.

4 HMM training and parameter generation

GlottHMM package does not provide direct means to build HTK (Hidden Markov Model Toolkit) compatible features. In order to integrate the GlottHMM vocoder to any speech synthesis system, some additional scripts are required.

4.1 Stream structure

The currently used configurations for HMM training are described below, but experimentation is encouraged.

- Four different streams:
  1. Vocal tract LSF (30 coefficients) + Gain (1 coefficient)
  2. Source LSF (5–20 coefficients)
  3. Harmonic-to-noise ratio, HNR (5–10 bands)
  4. Fundamental frequency, F0 (1 dimension)
- Additional parameters (H1-H2, NAQ, Waveform) can be trained for pulse selection.
- Only F0 has been modeled with multi-space distribution (MSD), but it is also possible for HNR and source LSFs (and H1-H2, NAQ, Waveform) too since they are redundant in unvoiced sections.

4.2 Feature extraction

For feature extraction, the following procedures are required:

- Combine vocal tract LSF and gain
- Add dynamic (delta and acceleration) features
- Handle MSD streams
- Build HTK-compatible features

Modification from, e.g., STRAIGHT-based scripts should not be very difficult

4.3 HTS training and clustering

The configurations need to be modified to suit selected model structure, otherwise no special modifications are required. The stream weights should preferable be set to vocal tract LSF, F0, and duration.

4.4 Parameter generation

HMGenS can be applied. Model structure-independent `HTS_engine` version exists, but it requires further work to be included in Simple4All framework.

In order to alleviate muffled speech caused by over-smoothing, post-filtering (formant enhancement) should be used, possibly in combination with global variance (GV) parameter generation. Typically small amount of GV (< 0.5) combined with LSF-based post-filtering (0.5) leads to good results.

5 Synthesis

This section describes the operation of Synthesis. Synthesis is executed from the command line by typing `Synthesis arg1 arg2 (arg3)`, where the arguments are the name of the speech file without `.wav` extension, default config file, and optional user config file. Before synthesis, either analysis of the same file must be performed, or parameters must be generated from HMMs. For analysis-synthesis, parameter `USE_HMM` should be set to `false`, and for speech synthesis using statistical modeling, the parameter should be set to `true`. For analysis-synthesis the speech parameters are slightly smoothed in time, but for HMM-based synthesis this is not performed. Additionally, instead of synthesizing a single file at a time, a synthesis list, defined in the config file (`SYNTHESIZE_MULTIPLE_FILES`, `SYNTHESIS_LIST`), can be used to synthesize multiple files at once.

5.1 Technical description

Synthesis is based of first creating the voiced and unvoiced excitations and then filtering the combined excitation with the vocal tract filter. The voiced excitation can be constructed with two different method: (1) Using a single glottal flow pulse extracted from natural speech, which is interpolated and scaled in magnitude to create an appropriate voice source. (2) Using a pulse library, consisting of various glottal flow pulses extracted from natural speech, from which the best candidates are selected for creating the voice source.

5.1.1 Synthesis with single pulse technique

Synthesis with a single pulse technique is illustrated in Figure 4. The process begins with creating the voiced and unvoiced excitation. The unvoiced excitation is composed of white noise scaled by the energy feature. The voiced excitation is constructed by interpolating the glottal flow pulse according to F0 and scaling it according to magnitude, and concatenating the pulses for voiced sections. The glottal flow pulse is a one-period length pulse extracted from natural speech. The path and name of the glottal flow pulse is defined in the config file parameter `GLOTTAL_PULSE_NAME`. In the GlottHMM package, a pulse extracted from 16 kHz male voice is provided (`gpulse`). This pulse can be used for synthesizing both male and female voices, since the pulse is further modified to fit the intended speaker. It might be beneficial to use pulses from the original speaker, but extracting an appropriate pulse is not straightforward due to possible discontinuity between the beginning and the end of the single pulse.

Before concatenating the pulses, noise is added according to the harmonic-to-noise ratio (HNR). This is performed for each pulse by calculating the FFT of the pulse, and adding a random component to each frequency bin of the spectrum according to the ERB-based HNR measure, and reconstructing the pulse by IFFT. Thus for each frequency band, a correct amount of noise is added. However, in order to prevent too much noise in the low frequencies (due to possible errors in HNR analysis), a low frequency limit (`NOISE_LOW_FREQ_LIMIT`) is set so that below that frequency, no noise is added. An appropriate value might vary from 1000 to 2000 Hz.

After adding the noise, the spectrum and HNR of each pulse is estimated again. This is done by reconstructing an analysis frame from the pulses, and evaluating the LP spectrum and HNR of the frame with the same methods as was performed in Analysis. Thus, the source spectrum and HNR of the excitation can be later modified to correspond to the desired one.

Once the voiced excitation is created, the spectrum of the excitation is modified in order to match it to the spectrum given in the voice source spectrum parameter. This is called spectral matching. This is performed by filtering the whole voiced excitation with an infinite impulse response (IIR) filter, consisting of the inverse of the estimated spectrum of the reconstructed excitation, and the the actual given voice source spectrum. This process can be described by first flattening (whitening)

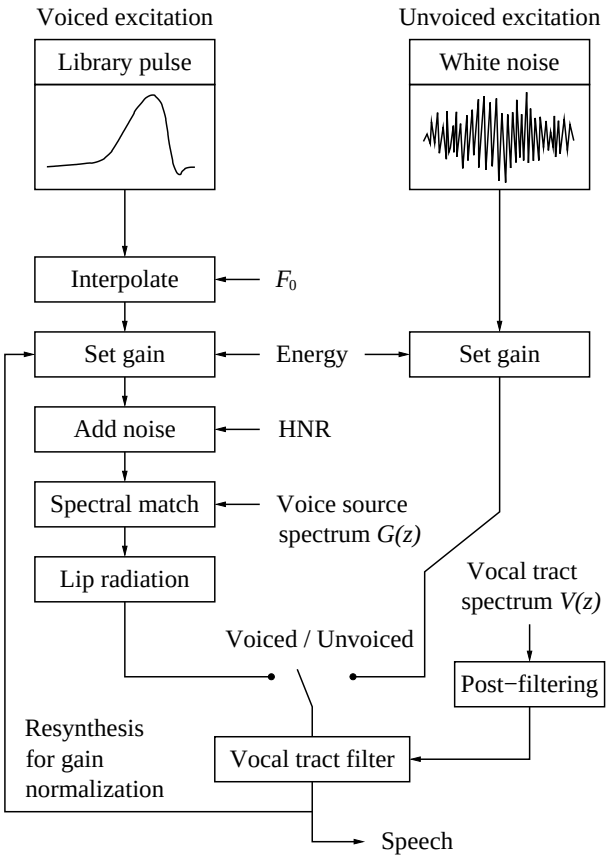

Figure 4: Illustration of synthesis with the single pulse technique.

the reconstructed excitation, and then applying the desired voice source spectrum. This gives the created excitation the desired spectral tilt and the more detailed spectral variation that should exist in natural voice source. Finally, the lip radiation is applied to the reconstructed glottal flow, which means differentiating the signal.

The vocal tract filter coefficients are post-filtered, meaning that the spectral structure, the formants, are enhanced in order to alleviate the oversmoothing of the statistical modeling. There are two methods for post-filtering, which can be set on/off in the config file with parameter `POSTFILTER_METHOD`. Options for the parameter are `"NONE"`, `"LSF"`, and `"LPC"`. The LSF-based method [10] slightly shifts the vocal tract LSFs in order to sharpen the most prominent formants. The LPC-based method [11] modifies the spectrum generated by the LP coefficients so that formants are more prominent, and then re-evaluates LP coefficients through the autocorrelation method. The strength of the post-filtering is adjusted by the config parameter `POSTFILTER_COEFFICIENT`. The range of the formant enhancement coefficient is [0,1] so that smaller value results in stronger post-filtering.

The voiced and unvoiced excitations are finally combined and filtered with the vocal tract filter. However, the spectral matching and filtering may cause variations in the signal energy, and thus the synthesis is performed again with normalized gain values.

5.1.2 Synthesis with pulse library technique

The pulse library technique is based on selecting the best matching pulses according to target and concatenation costs in order to reconstruct a realistic voice source. In this sense, the technique resembles the classical unit-selection framework, where speech units are selected from a database and concatenated to create speech. However, the fundamental difference between the conventional unit-selection framework and the pulse library technique is that the number of units required for natural sounding synthetic speech is very low in the pulse library technique. This is because the two components, the glottal source and the vocal tract filter, are separated by glottal inverse filtering, and thus only the varying context or modes of the voice source need to be stored into a pulse library, and the variation due to vocal tract filter can be modeled by the HMM.

Synthesis with pulse library can be used by setting the configuration file parameter `USE_PULSE_LIBRARY` to `true` and typing the path and name of the pulse library to parameter `PULSE_LIBRARY_NAME`. Synthesis with a single pulse technique is illustrated in Figure 5. The pulse library consists of two-period segments of the glottal flow derivative waveform. These pulses are described by the speech features extracted in the analysis phase. These *pulses* are selected for the voiced sections of the excitation signal according to target and concatenation costs. Minimizing the target cost of the voice source parameters ensures that a pulse with desired properties (fundamental period, spectral tilt, the amount of noise) is most likely to be chosen. The target cost consists of the error between the speech features between the library pulses and the given target parameters. For each target parameter (LSF, LSFsource, Harmonics, HNR, Gain, F0, Waveform, H1H2, and NAQ) an individual weight may be defined in the config file setting `PARAMETER_WEIGHTS`.

The concatenation cost consists of the root mean square (RMS) error between the consecutive downsampled pulses. Minimizing the concatenation cost ensures that adjacent pulse waveforms do not differ substantially from each other, possibly producing abrupt changes in the excitation signal leading to a harsh voice quality. In order to prevent selecting the same pulse in a row, possibly resulting in a bad speech quality, an bias can be set to the concatenation error with the config file parameter `PULSE_ERROR_BIAS`.

The weights of the target and concatenation costs can be defined with `TARGET_COST` and `CONCATENATION_COST` in the config file. The selection process is performed for each continuous voiced section and optimized with the Viterbi algorithm.

Since F0 is included in the target cost, a pulse with approximately correct fundamental period will be chosen, and thus interpolation of the pulse is not usually required. However, interpolation of the library pulses can be used by setting the config parameter `USE_PULSE_INTERPOLATION` to `true`. Also the HNR of the pulses should ideally be correct, but the noise addition can by turned on with the parameter `ADD_NOISE_PULSELIB`.

After the selection process, only the energy of the pulse is equalized to the energy measure given by the HMM. The excitation signal is finally generated by overlap-adding the selected pulses according to the current F0 value. Thus a pulse train comprising a series of individual glottal source pulses is generated.

The rest of the pulse library technique is similar to single pulse technique, consisting of combining the voiced and unvoiced sound sources, post-filtering of the vocal tract filter, filtering, and finally resynthesis for gain normalization.

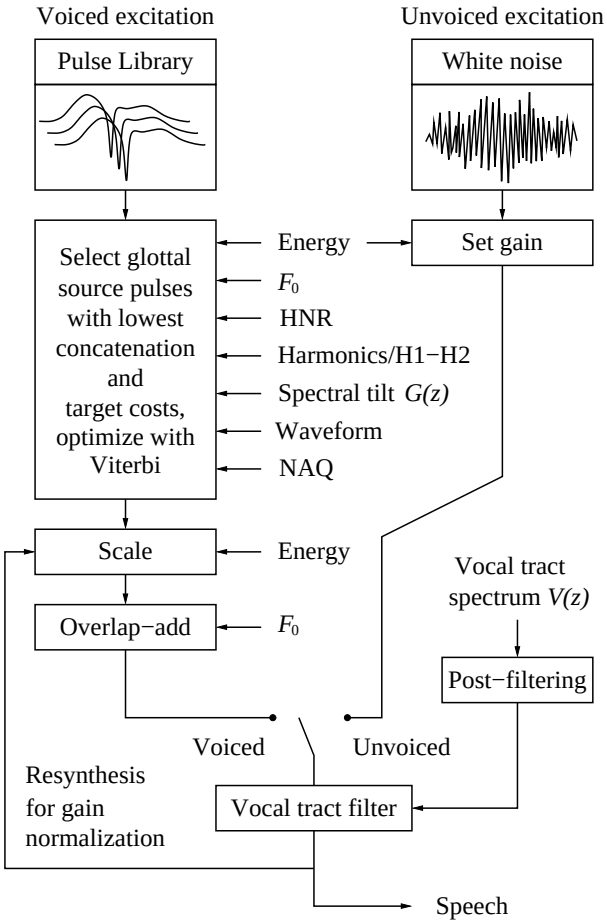

Figure 5: Illustration of synthesis with the pulse library technique.

References

[1] T. Raitio, A. Suni, H. Pulakka, M. Vainio, and P. Alku, "HMM-based Finnish text-to-speech system utilizing glottal inverse filtering," in *Proc. Interspeech*, 2008, pp. 1881–1884.

[2] A. Suni, T. Raitio, M. Vainio, and P. Alku, "The GlottHMM speech synthesis entry for Blizzard Challenge 2010," in *The Blizzard Challenge 2010 workshop*, 2010, http://festvox.org/blizzard.

[3] T. Raitio, A. Suni, H. Pulakka, M. Vainio, and P. Alku, "Utilizing glottal source pulse library for generating improved excitation signal for HMM-based speech synthesis," in *Proc. ICASSP*, 2011, pp. 4564–4567.

[4] A. Suni, T. Raitio, M. Vainio, and P. Alku, "The GlottHMM entry for Blizzard Challenge 2011: Utilizing source unit selection in HMM-based speech synthesis for improved excitation generation," in *The Blizzard Challenge 2011 Workshop*, 2011, http://festvox.org/blizzard.

[5] T. Raitio, A. Suni, J. Yamagishi, H. Pulakka, J. Nurminen, M. Vainio, and P. Alku, "HMM-based speech synthesis utilizing glottal inverse filtering," *IEEE Trans. on Audio, Speech, and Lang. Proc.*, vol. 19, no. 1, pp. 153–165, Jan. 2011.

[6] C. Ma, Y. Kamp, and L. Willems, "Robust signal selection for linear prediction analysis of voiced speech," *Speech Comm.*, vol. 12, no. 1, pp. 69–81, 1993.

[7] C. Magi, J. Pohjalainen, T. Bäckström, and P. Alku, "Stabilised weighted linear prediction," *Speech Comm.*, vol. 51, no. 5, pp. 401–411, May 2009.

[8] S. Keronen, J. Pohjalainen, P. Alku, and M. Kurimo, "Noise robust feature extraction based on extended weighted linear prediction in LVCSR," in *Interspeech*, 2011, pp. 1265–1268.

[9] Paavo Alku, Tom Bäckström, and Erkki Vilkman, "Normalized amplitude quotient for parametrization of the glottal flow," *The Journal of the Acoustical Society of America*, vol. 112, no. 2, pp. 701–710, 2002.

[10] Z.-H. Ling, Y.J. Wu, Y.-P. Wang, L. Qin, and R.-H. Wang, "USTC system for Blizzard Challenge 2006: an improved HMM-based speech synthesis method," in *Blizzard Challenge Workshop*, 2006.

[11] T. Raitio, A. Suni, H. Pulakka, M. Vainio, and P. Alku, "Comparison of formant enhancement methods for HMM-based speech synthesis," in *SSW7*, Sep. 2010, pp. 334–339.

A Description of configuration file parameters

In the following list, all the configuration file parameters and their purpose are described. The default values and/or the recommended range/possible options are shown for each parameter. Note again, that integer parameters must be defined without decimals (e.g. sampling frequency is 16000), and decimal numbers must include the decimal point (e.g. frame length in milliseconds is 25.0).

Analysis and Synthesis: Common parameters:

SAMPLING_FREQUENCY: Sampling frequency in Hz (16000)

FRAME_LENGTH: Analysis frame length in milliseconds (25.0)

UNVOICED_FRAME_LENGTH: Analysis frame length for unvoiced segments in milliseconds (20.0)

F0_FRAME_LENGTH: Analysis frame length for F0 evaluation in milliseconds (45.0)

FRAME_SHIFT: Analysis frame shift in milliseconds (5.0)

LPC_ORDER: Order of the spectral model of the vocal tract (30)

LPC_ORDER_SOURCE: Order of the spectral model of the voice source (20)

WARPING_VT: Warping coefficient for vocal tract model (0.0). Default value produces linear spectral resolution; positive values produce finer resolution in the low frequencies; negative values produce finer resolution in the high frequencies. The value must be in the range [-1,1].

WARPING_GL: Warping coefficient for voice source spectral model (0.0). Default value produces linear spectral resolution; positive values produce finer resolution in the low frequencies; negative values produce finer resolution in the high frequencies. The value must be in the range [-1,1].

HNR_CHANNELS: Number of harmonic-to-noise ratio (HNR) frequency channels (5). Channels are distributed according to the equivalent rectangular bandwidth (ERB) scale in the frequency domain.

NUMBER_OF_HARMONICS: Number of harmonics to be modeled in the harmonics parameter (10).

SEPARATE_VU_SPECTRUM: Extract and use separate spectra for voiced and unvoiced segments (false).

DIFFERENTIAL_LSF: Use differential line spectral frequencies (LSFs) (false). Differential LSFs improves the sharpness of the formants, but the formant positions are less accurate.

LOG_F0: Convert F0 parameter to natural logarithmic scale.

DATA_FORMAT: Data format for writing speech features ("ASCII"). Reading and writing binary data ("BINARY") is faster and saves some space.

Noise reduction:

NOISE_REDUCTION_ANALYSIS: Use noise reduction in Analysis.

NOISE_REDUCTION_SYNTHESIS: Use noise reduction in Synthesis.

NOISE_REDUCTION_LIMIT_DB: Limit for noise reduction in dB; values of gain lower than the limit will be reduced (0.0, depends on speech material).

NOISE_REDUCTION_DB: Amount of noise reduction in dB (30.0).

Analysis: General parameters:

PITCH_SYNCHRONOUS_ANALYSIS: Option to use pitch-synchronous analysis (true).

INVERT_SIGNAL: Set this true if the polarity of the speech signal is inverted (false). This is especially important when using pulse library.

HP_FILTERING: Use high-pass filtering in order to remove possible low-frequency ripple in the glottal inverse filtering (true).

HPFILTER_FILENAME: Define high-pass filter file name (hp_16khz). The filter coefficients must in ASCII format, one coefficient in each line. The filter must be a FIR filter with a preferred cut-off frequency around 30–70 Hz, depending on the voice.

Analysis: Parameters for F0 estimation:

F0_MIN: Minimum fundamental frequency (28.0).

F0_MAX: Maximum fundamental frequency (500.0).

VOICING_THRESHOLD: Voicing threshold with respect to gain in the low-frequency band (0–1000 Hz) (100.0). Speech segments that have very low gain in the low frequencies are classified as unvoiced according to this relative measure. Increasing the value will results in more frames to be classified as unvoiced.

ZCR_THRESHOLD: Zero-crossings threshold (120.0). Speech segments that have more zero crossings than the threshold value are classified as unvoiced according. Decreasing the ZCR threshold will result in more frames to be classified as unvoiced.

USE_F0_POSTPROCESSING: Post-processing of fundamental frequency vector (true). Processing refines the F0 vector by median filtering, filling small gaps, and estimating weighted linear F0 estimates in the case of discontinuities in the vector.

RELATIVE_F0_THRESHOLD: Discontinuity measure in F0 post-processing (0.5). Relative F0 jumps greater that this value will be considered unnatural and will be fixed.

F0_CHECK_RANGE: Range (in frames) within which the F0 post-processing will operate when estimating and correcting possible incorrect F0 values (10).

USE_EXTERNAL_F0: Use external F0 file in Analysis. All evaluations that require fundamental frequency are thus based on the given values. The name of the external F0 file is indicated in the next parameter.

EXTERNAL_F0_FILENAME: Name of the external F0 file.

Analysis: Parameters for extracting pulse libraries:

PULSEMAXLEN: Maximum length of the two-period library pulse in milliseconds (45.0).

RESAMPLED_PULSELEN: Length of the resampled library pulse in milliseconds (10.0).

WAVEFORM_SAMPLES: Number of the samples which describe the shape of the library pulse (10).

MAX_PULSE_LEN_DIFF: Maximum relative difference between pulse length and the estimated T0 (0.05). If the relative difference is greater that the defined value, the pulse is discarded. The goal is to discard possibly misaligned pulse segments due to failure in the detection of glottal closure instant (GCI).

MAX_NUMBER_OF_PULSES: Maximum number of pulses to be stored per speech file (6000). This is just to save memory, value may be increased with long speech files.

Analysis: Parameters for spectral modeling:

USE_IAIF: Use iterative adaptive inverse filtering (IAIF) (true).

LPC_ORDER_GL_IAIF: Order of the LPC analysis for the voice source inside IAIF (8).

USE_MOD_IAIF: Use modified version of the IAIF method (true). This simpler glottal inverse filtering method gives less varying decomposition, but may not be as accurate as full IAIF.

LP_METHOD: Select between different spectral modeling methods, linear prediction (LPC), weighted linear prediction (WLP), or extended linear prediction (XLP) ("LPC"/"WLP"/"XLP").

LP_STABILIZED: Stabilize spectral models if WLP or XLP is used (true).

LP_WEIGHTING: IF WLP is used, select between short time energy (STE) weight and glottal closure instant (GCI) weight ("STE"/"GCI").

FORMANT_PRE_ENH_METHOD: Select formant enhancement method for pre-enhancement. There are two methods, LSF-based and LPC-based. Usually pre-enhancement is not used, and enhancement is used in synthesis stage ("NONE"/"LSF"/"LPC").

FORMANT_PRE_ENH_COEFF: Formant enhancement coefficient (0.5). The effect gets stronger as the value is decreased. Range: [0,1].

FORMANT_ENH_LPC_DELTA: Tuning parameter for the LPC-based formant enhancement method (20.0). Defines the width of the area around formants that is left unmodified in the power spectrum.

Analysis: Select parameters to be extracted:

EXTRACT_F0: Extract fundamental frequency (true).

EXTRACT_GAIN: Extract gain of the speech signal (true).

EXTRACT_LSF: Extract vocal tract spectrum LSFs (true).

EXTRACT_LSFSOURCE: Extract voice source spectrum LSFs (true).

EXTRACT_HNR: Extract harmonic-to-noise ratio of the voice source (true).

EXTRACT_HARMONICS: Extract the magnitude differences between N first harmonics of the voice source (true).

EXTRACT_H1H2: Extract the magnitude difference between the first and the second harmonics of the voice source (true).

EXTRACT_NAQ: Extract the normalized amplitude quotient (NAQ) of the glottal pulses (true).

EXTRACT_WAVEFORM: Extract the approximate shape of the glottal waveform (true).

EXTRACT_INFOFILE: Write infofile that describes the used analysis parameters (true).

EXTRACT_PULSELIB: Extract all parameters needed to construct a pulse library (false). This is only required when constructing a pulse library.

EXTRACT_SOURCE: Extract the estimated glottal flow signal to file and to a wav file (false). This is only additional option for speech analysis.

Synthesis: General parameters:

SYNTHESIZE_MULTIPLE_FILES: Switch for synthesizing multiple files at once.

SYNTHESIS_LIST: Full path and name of a synthesis list, e.g., "/home/user1/syn/synlist".

USE_HMM: Select between direct analysis-synthesis (false) and HMM modeling (true). For direct analysis-synthesis some smoothing of the parameters is applied in order to produce more natural sounding speech.

Synthesis: Choose excitation technique and related parameters:

USE_PULSE_LIBRARY: Switch for using a pulse library (false/true).

GLOTTAL_PULSE_NAME: Full path and name of a single pulse, e.g., "/home/user1/syn/pulselib".

PULSE_LIBRARY_NAME: Full path and name of a pulse library, e.g., "/home/user1/syn/pulse".

NORMALIZE_PULSELIB: The means of the pulse library parameters are normalized according to synthesis parameters (true).

USE_PULSE_CLUSTERING: Switch for using pulse clustering in synthesis. This requires clustering of the pulse library according to decision trees.

USE_PULSE_INTERPOLATION: Interpolate pulse library pulses according to F0 (false/true). However, this is not mandatory since pulses with approximately correct F0 are usually selected and overlap-added according to F0.

AVERAGE_N_ADJACENT_PULSES: Average adjacent pulse library pulses in order to get more smoother excitation. The number of averaged adjacent pulses is defined by this parameter, 0 sets off this feature.

ADD_NOISE_PULSELIB: Add noise to pulse library pulses according to the HNR measure (false/true). This is not mandatory since pulses with approximately correct HNR value are usually selected.

MAX_PULSES_IN_CLUSTER: Define the maximum number of pulses per cluster, if pulse clustering is used (2000).

NUMBER_OF_PULSE_CANDIDATES: Define the number of best matching pulse library candidates for each time step in the Viterbi search (200).

PULSE_ERROR_BIAS: Set bias in order to prevent the same pulse to be selected to the excitation in a row (0.3). The selection of the same library pulse multiple times in a row may cause audible artifacts.

CONCATENATION_COST: Set concatenation cost for selecting the library pulses (1.0).

TARGET_COST: Set target cost for selecting the library pulses (1.0).

PARAMETER_WEIGHTS: Set parameter weights for for selecting the library pulses. A vector consisting of the parameter weights of each parameter is defined as [LSF SRC HARM HNR GAIN F0 WAV H1H2 NAQ], e.g., [0.0, 2.0, 0.0, 2.0, 3.0, 5.0, 1.0, 1.0, 1.0].

Synthesis: Select used parameters:

Select parameters used in Synthesis. All the parameter are not required, but some parameters are required in a specific configuration. F0, Gain, and LSFs are always required and thus they are not in this list.

USE_LSFSOURCE: Use voice source spectrum (true).

USE_HNR: Use harmonic-to-noise ratio (true).

USE_HARMONICS: Use harmonics (true).

USE_H1H2: Use the magnitude difference of the first and the second harmonic (true).

USE_NAQ: Use normalized amplitude quotient (NAQ) (true).

USE_WAVEFORM: Use downsampled waveform (true).

Synthesis: Set level and band of voiced noise:

NOISE_GAIN_VOICED: The level of added noise in voiced sections (0.5).

NOISE_LOW_FREQ_LIMIT: Set low-frequency limit for the added noise (2400.0). The value is decimal number in Hz between [0,FS/2]. This prevents artifacts due to large low-frequency fluctuations.

Synthesis: Smoothing of parameters for analysis-synthesis:

Smoothing lengths of the parameter vectors/matrices for analysis-synthesis. All smoothing is set off if parameters generated by HMMs are used.

LSF_SMOOTH_LEN: LSF smoothing length (5).

LSFSOURCE_SMOOTH_LEN: Voice source spectrum smoothing length (3).

GAIN_SMOOTH_LEN: Gain smoothing length (5).

HNR_SMOOTH_LEN: Harmonic-to-noise ratio (HNR) smoothing length (15).

HARMONICS_SMOOTH_LEN: Harmonics smoothing length (5).

Synthesis: Gain related parameters:

GAIN_UNVOICED: Set gain for unvoiced part (1.0).

NORM_GAIN_SMOOTH_V_LEN: Smoothing length of the voiced part of the gain normalization algorithm (0).

NORM_GAIN_SMOOTH_UV_LEN: Smoothing length of the unvoiced part of the gain normalization algorithm (0).

GAIN_VOICED_FRAME_LENGTH: Voiced frame length of the gain normalization algorithm in milliseconds (25.0).

GAIN_UNVOICED_FRAME_LENGTH: Unvoiced frame length of the gain normalization algorithm in milliseconds (25.0).

Synthesis: Postfiltering:

POSTFILTER_METHOD: Select formant enhancement method for post-filtering. There are two methods, LSF-based and LPC-based. Select between "NONE"/"LSF"/"LPC".

POSTFILTER_COEFFICIENT: Formant enhancement strength (common for both methods) (0.5). The effect gets stronger as the value of alpha is decreased. Range: [0,1].

Synthesis: Utils:

PITCH: Modify pitch relative to the original pitch (1.0).

SPEED: Modify speech relative to the original speed (1.0).

JITTER: Add jitter to consecutive pulses (relative to pulse length) (0.0).

ADAPT_TO_PULSELIB: Adapt synthesis parameters according to pulse library parameters by normalizing the mean of the parameters. This will have an effect that the synthetic voice will sound more like the voice in the pulse library (false).

ADAPT_COEFF: Coefficient for changing the magnitude (and direction) of the adaptation of the synthesis parameters by the pulse library parameters (1.0).

USE_PULSELIB_LSF: Replace vocal tract LSFs with the closest LSFs from the pulse library, apply smoothing (false). This will have an effect that the synthetic voice will sound more like the voice in the pulse library (false).

NOISE_ROBUST_SPEECH: Create Lombard style speech by setting this to true. The quality is of the resulting speech is degraded, but the intelligibility in the presence of (low-frequency) noise is increased.

USE_HARMONIC_MODIFICATION: If single pulse technique is used, a harmonic modification procedure can be used to correctly match the low-frequency spectrum of the voice source (false).

HP_FILTER_F0: Adaptive low-pass filtering of the synthesized speech signal below current F0 value (false).

FILTER_UPDATE_INTERVAL_VT: Vocal tract filter update interval in milliseconds (0.3).

FILTER_UPDATE_INTERVAL_GL: Spectral matching filter (voice source) update interval in milliseconds (0.05).