Continuous State Modeling for Statistical Spectral Synthesis

Release 0.1

Tim-Tarek Grund

Dec 04, 2022

CONTENTS:

CHAPTER

ONE

USAGE

1.1 Installation

The code for Continuous State Modeling for Statistical Spectral Synthesis can be downloaded from Github.

The analysis functions require the SMS-Tools to be installed.

It solely makes use of the samples from the TU-Note Violin Sample Library and requires the directory structure to be left as is.

1.2 Analysis

The results of the analysis part are contained in *../CSM4SS/extracted_parameters*. If you like to refine these, you can follow this guide.

The github repository contains the python notebook 0_sample_preparation.ipynb. You can use it to prepare and analyze the TU-Note Violin samples. Make sure to change the following folder paths to the correct directories:

- `path_frequency` should link to the *../TU-Note_Violin/File_Lists/list_Single.txt* file.
- `path_annotations` should link to *../TU-Note_Violin/Segments/SingleSounds/SampLib_DPA_*. It is important to leave the path ending with the underscore, as a counting variable will be added to the path later in order to access the single sound items.
- `path_soundfile_96k` should link to *../TU-Note_Violin/WAV/SingleSounds/BuK/SampLib_BuK_*. Again, let the path end with an underscore.

You can leave the other folder paths unchanged.

You can use the function `CSM_functions.batch_convert_to_44100()`. While the analysis functions in this library can deal with other sampling rates, the standalone GUI version of the SMS-Tools throws an error when trying to read in files with a higher sampling rate. The GUI can help with testing out the parameters for the analysis functions.

Next you can extract the relevant meta-information on the sound files. Informations like the dynamics and frequencies are contained in the *list_single.txt* file, which you can load using `CSM_functions.read_list_single_TU_VSL()`. Using the `CSM_functions.read_frequency_TU_VSL()` function, you can extract the frequency from the object. Likewise, you can extract the sustain segmentation time stamps from the *SampLib_DPA_XX* files using the `CSM_functions.read_annotations_TU_VSL()` function.

Finally, you can analyze and write the parameters into files using the `CSM_functions.write_parameters()` function. Caution: Calling this function will overwrite the existing files contained in *../CSM4SS/extracted_parameters*. You can batch extract parameters by using a list of integers (maximum: from 01 to 336). However, not all values work for all samples. You can refine them by reusing the function with different parameters to overwrite the current files.

1.3 Synthesis

To synthesize sound items using Markovian spectral synthesis you can use the 1_sample_creation.ipynb.

Make sure to change the following folder paths to the correct directories: - path_frequency should link to the *../TU-Note_Violin/File_Lists/list_Single.txt* file. - path_annotations should link to *../TU-Note_Violin/Segments/SingleSounds/SampLib_DPA_*. It is important to leave the path ending with the underscore.

Afterwards, create the list_single object by calling CSM_functions.read_list_single_TU_VSL(). Finally, you can use CSM_functions.markov_scaled_normal_synthesis() and CSM_functions.markov_skew_normal_synthesis() to create samples. By using a list for the *list_indices* argument you can batch create samples following the name structure for frequency and dynamics from the TU-Note Violin Sample library.

CHAPTER

TWO

CONTENTS

2.1 API

CSM_functions

2.1.1 CSM_functions

Functions

batch_convert_to_44100(file_indices, ...)	Convert sound items to 44.1kHz.
interpolation(trajectory, dist_supportpoints, sr)	Linear interpolate between support points of parameter trajectory.
markov_scaled_normal_synthesis(list_indices, ...)	Synthesize violin sound using the scaled normal Markovian synthesis.
markov_skew_normal_synthesis(list_indices, ...)	Synthesize violin sound using the skew normal Markovian synthesis.
plot_FFT(index, path_in, path_annotations, ...)	Plot and optionally save the Fourier transform of the original or the synthesized soundfiles.
plot_spectrogram(index, path_in, ...[, save])	Plot and optionally save the spectrogram of the original or the synthesized soundfiles.
plot_waveform(index, path_in, ...[, save])	Plot and optionally save the waveform of the original or the synthesized soundfiles.
read_annotations_TU_VSL(path, index)	Read the annotation file from the TU-Note Violin Sample Library and return the start and stop points of the sustain part of a sound item in samples.
read_frequency_TU_VSL(list_single, index)	Return the frequency for sound item from the list_single object.
read_list_single_TU_VSL(path)	Load and return the list_single.txt file from path.
reject_outliers(data[, m])	Remove outliers beyond an absolute distance m from data and return result.
skew_normal_distribution_henze(mu, sigma, ...)	Draw new values from the skew normal distribution.
write_parameters(path_soundfile, ...[, verbose])	Call harmonic model function from sms-tools and write extracted parameters into files.

2.2 Documentation

2.2.1 List of functions

CSM_functions.batch_convert_to_44100(*file_indices*, *path_in*, *path_out*)

Convert sound items to 44.1kHz.

Parameters

- file_indices (ndarray) – List of sound item indices, which are to be converted to 44.1kHz.
- path_in (string) – Input path, where sound items are located.
- path_out (string) – Output path, where converted sound items should be placed.

CSM_functions.interpolation(*trajectory*, *dist_supportpoints*, *sr*)

Linear interpolate between support points of parameter trajectory.

Parameters

- trajectory (ndarray) – Parameter trajectory made of support points.
- dist_supportpoints (int) – Distance between support points in samples.
- sr (int) – Sampling rate.

Returns

Interpolated parameter trajectory of length len(trajectory) x dist_supportpoints.

Return type

ndarray

CSM_functions.markov_scaled_normal_synthesis(*list_indices*, *list_single*, *path_amp*, *path_vol*, *path_output*, *alpha_f0=0.001*, *alpha_amp=0.001*, *sigma_f0=0.004*, *sigma_amp=0.02*, *num_samples=600*, *len_support_points=512*, *sr=44100*, *num_partials=80*)

Synthesize violin sound using the scaled normal Markovian synthesis.

Synthesis starts with a given value and draws following values from a normal distribution, where the location parameter is scaled between a target value and the preceding value.

Parameters

- list_indices (list) – List of indices of sound items to be recreated.
- list_single (ndarray) – List_single object (created by read_list_single_TU_VSL()) given as input.
- path_amp (string) – Path to the partial amplitudes extracted by write_parameters().
- path_vol (string) – Path to the overall loudness extracted by write_parameters().
- path_output (string) – Path to where synthesized sound items should be saved.
- alpha_f0 (float) – Scaling the influence of target value or preceding value on location parameter for frequency trajectory.
- alpha_amp (float) – Scaling the influence of target value or preceding value on location parameter for amplitude trajectory.
- sigma_f0 (float) – Standard deviation for the frequency trajectory.
- sigma_amp (float) – Standard deviation for the amplitude trajectory.

- num_samples (int) – Number of sample points. Needs to be greater than or equal to 1.
- len_support_points (int) – Length of support points between sample points.
- sr (int) – Sampling rate. Can be anything, but shoulde be 44100 in order to be analyzed with sms-tools.
- num_partials (int) – Number of partials to be created for the output sound.

CSM_functions.markov_skew_normal_synthesis(*list_indices*, *list_single*, *path_amp*, *path_vol*, *path_output*, *gamma_f0=1*, *gamma_amp=0.8*, *sigma_f0=0.008*, *sigma_amp=0.03*, *num_samples=600*, *len_support_points=512*, *sr=44100*, *num_partials=80*)

Synthesize violin sound using the skew normal Markovian synthesis.

Synthesis starts with a given value and draws following values from a skew normal distribution, where the skew parameter is calculated by the distance between target value and preceding value multiplied by a scaling factor gamma. Based on the formula for skew normal distribution by Henze. Norbert Henze, "A probabilistic representation of the 'skew-normal' distribution," Scandinavian Journal of Statistics, vol. 13, no. 4, pp. 271–275, 1986.

Parameters

- list_indices (list) – List of indices of sound items to be recreated.
- list_single (ndarray) – List_single object (created by read_list_single_TU_VSL()) given as input.
- path_amp (string) – Path to the partial amplitudes extracted by write_parameters().
- path_vol (string) – Path to the overall loudness extracted by write_parameters().
- path_output (string) – Path to where synthesized sound items should be saved.
- gamma_f0 (float) – Scaling the influence of target value or preceding value on skewness parameter for frequency trajectory.
- alpha_amp (float) – Scaling the influence of target value or preceding value on skewness parameter for amplitude trajectory.
- sigma_f0 (float) – Standard deviation for the frequency trajectory.
- sigma_amp (float) – Standard deviation for the amplitude trajectory.
- num_samples (int) – Number of sample points. Needs to be greater than or equal to 1.
- len_support_points (int) – Length of support points between sample points.
- sr (int) – Sampling rate. Can be anything, but shoulde be 44100 in order to be analyzed with sms-tools.
- num_partials (int) – Number of partials to be created for the output sound.

CSM_functions.plot_FFT(*index*, *path_in*, *path_annotations*, *path_save*, *mode*, *dyn*, *save=False*)

Plot and optionally save the Fourier transform of the original or the synthesized soundfiles.

Currently expects files to start with the index of the sound item followed by an underscore, and to have only one file per sound item. Plotting of original FFT uses only sustain part, but only when using mode = "original".

Parameters

- index (int) – Index of sound item.
- path_in (string) – Path to the soundfiles (Original, or skew or scaled synthesized).

- path_annotations (string) – Path to the annotation files for the attack, decay, sustain and release segmentation.
- path_save (string) – Path to where output figures should be saved.
- mode (string) – Origin of sound file. Options are “original”, “skew” and “scaled”.
- dyn (string) – Simply adds the dynamic marker (pp, mp, mf, ff) to the file name.
- save (bool) – Saves resulting plot at location specified by path_save.

CSM_functions.plot_spectrogram(*index*, *path_in*, *path_annotations*, *path_save*, *mode*, *dyn*, *save=False*)

Plot and optionally save the spectrogram of the original or the synthesized soundfiles.

Basically just a wrapper for the librosa.display.specshow() function. Currently expects files to start with the index of the sound item followed by an underscore, and to have only one file per sound item. Currently needs path to the 44.1k original sound items instead of the 96k original sound items. Plotting of original FFT uses only sustain part, but only when using mode = “original”.

Parameters

- index (int) – Index of sound item.
- path_in (string) – Path to the soundfiles (Original, or skew or scaled synthesized).
- path_annotations (string) – Path to the annotation files for the attack, decay, sustain and release segmentation.
- path_save (string) – Path to where output figures should be saved.
- mode (string) – Origin of sound file. Options are “original”, “skew” and “scaled”.
- dyn (string) –
- save (bool) –

CSM_functions.plot_waveform(*index*, *path_in*, *path_annotations*, *path_save*, *mode*, *dyn*, *save=False*)

Plot and optionally save the waveform of the original or the synthesized soundfiles.

Currently expects files to start with the index of the sound item followed by an underscore, and to have only one file per sound item. Plotting of original waveforms shows full sample and sustain part, but only when using mode = “original”.

Parameters

- index (int) – Index of sound item.
- path_in (string) – Path to the soundfiles (Original, or skew or scaled synthesized).
- path_annotations (string) – Path to the annotation files for the attack, decay, sustain and release segmentation.
- path_save (string) – Path to where output figures should be saved.
- mode (string) – Origin of sound file. Options are “original”, “skew” and “scaled”.
- dyn (string) – Simply adds the dynamic marker (pp, mp, mf, ff) to the file name.
- save (bool) – Saves resulting plot at location specified by path_save.

CSM_functions.read_annotations_TU_VSL(*path*, *index*)

Read the annotation file from the TU-Note Violin Sample Library and return the start and stop points of the sustain part of a sound item in samples.

Parameters

- path (string) – folder path to annotation file from TU-Note Violin Sample Library.

- index (int) – index of sound item from TU-Note Violin sample library.

Return float start_sec
Start point of sustain part in samples.

Return float stop_sec
End point of sustain part in samples.

CSM_functions.read_frequency_TU_VSL(*list_single*, *index*)

Return the frequency for sound item from the list_single object.

Parameters

- list_single (ndarray) – List_single object (created by read_list_single_TU_VSL()) given as input.
- index (int) – Index of sound item from TU-Note Violin sample library

Returns
Frequency of sound item

Return type
float

CSM_functions.read_list_single_TU_VSL(*path*)

Load and return the list_single.txt file from path.

The list_single.txt file contains information string, position, MIDI number, ISO pitch notation, dynamic and frequency in Hz.

Parameters
path (string) – Folder path to the list_single.txt file from TU-Note Violin Sample Library.

Returns
contents of list_single as ndarray.

Return type
ndarray

CSM_functions.reject_outliers(*data*, *m=1.1*)

Remove outliers beyond an absolute distance m from data and return result.

Bannier, B. [benjamin-bannier]. (2013, May 15). Something important when dealing with outliers is that one should try to use estimators as robust as possible. The mean of. [Comment on the online forum post Is there a numpy builtin to reject outliers from a list]. Stack Overflow. https://stackoverflow.com/a/16562028 (accessed: 04.12.2022)

Parameters

- data (ndarray) – The data from which outliers need to be removed.
- m (float) – Distance m, all values outside of median(data) +/- m will be removed.

Returns
Data with (possibly) removed values.

Return type
ndarray

CSM_functions.skew_normal_distribution_henze(*mu*, *sigma*, *theta*, *size*)

Draw new values from the skew normal distribution.

Implements an algorithm to based on a process outlined in Norbert Henze, "A probabilistic representation of the 'skew-normal' distribution," Scandinavian Journal of Statistics, vol. 13, no. 4, pp. 271–275, 1986.

Parameters

- mu (float) – location parameter.
- sigma (float) – scale parameter.
- theta (float) – shape, or skewness parameter.
- size (int) – length of return ndarray.

Returns
array of values which have the skew noral distribution.

Return type
ndarray

CSM_functions.write_parameters(*path_soundfile*, *path_parameters*, *frequency_f0*, *index*, *start_samp*, *stop_samp*, *window*, *M*, *N*, *t*, *minSineDur*, *nH*, *minf0*, *maxf0*, *f0et*, *harmDevSlope*, *Ns*, *H*, *verbose=False*)

Call harmonic model function from sms-tools and write extracted parameters into files.

Wrapper function for the sms-tools by Serra. Writes partial amplitude mean, partial amplitude standard deviation, partial frequency standard deviation and mean amplitude of sound item into files. Xavier Serra, "Spectral modeling synthesis tools," Available at https://www.upf.edu/web/mtg/sms-tools, accessed 29.03.2022, 2013.

Parameters

- path_soundfile (string) – input folder path, where sound item files are located.
- path_parameters (string) – output folder path for mean amplitude of sample, mean partial amplitudes, and standard deviations of partial amplituds and frequencies.
- frequency_f0 (float) – fundamental frequency of sound item.
- index (int) – index of sound item.
- start_samp (int) – Beginning index of segmentation of audio data.
- stop_samp (int) – Ending index of segmentation of audio data.
- window (string) – window type, refer to sms-tools.
- M (int) – window length, refer to sms-tools.
- N (int) – FFT size (must be power of two, must be larger than or equal to M).
- t (float) – spectral peak threshold.
- minSineDur (float) – min duration of sinusoidal tracks.
- nH (int) – number of Harmonics.
- minf0 (float) – min f0 frequency.
- maxf0 (float) – max f0 frequency.
- f0et (float) – max error in f0 detection.
- harmDevSlope (float) – max deviation of harmonic tracks.
- Ns (int) – size of fft used in synthesis, not used here.
- H (int) – Hop size, must be 1/4 of Ns, if used for sinusoidal synthesis.
- verbose (bool) – if true, prints information on sound item.

PYTHON MODULE INDEX

C

INDEX