Getting started

1  Installation  3
2  Tutorial  5
3  pytsmod  7
4  utils  11
Python Module Index  15
Index  17
PyTSM is an open-source library for Time-Scale Modification algorithms in Python 3. PyTSM contains basic TSM algorithms such as Overlap-Add (OLA), Waveform-Similarity Overlap-Add (WSOLA), Time-Domain Pitch-Synchronous Overlap-Add (TD-PSOLA), and Phase Vocoder (PV-TSM). We are also planning to add more TSM algorithms and pitch shifting algorithms.
1.1 PyPI

PyTSMod is hosted on PyPI. To install, run the following command in your Python environment:

```
$ pip install pytsmod
```

1.2 Build with Poetry

To build the package manually from the release archive or repo, Poetry is needed. After install Poetry, build wheel file with the following command:

```
$ poetry build
```

After build the package, you can install through pip:

```
$ pip install dist/NAME_OF_PACKAGE.whl
```

1.3 Requirements

To use PyTSMod, Python with version >= 3.6 and following packages are required.

- Numpy (>=1.16.0)
- Scipy (>=1.0.0)
- libROSA (>=0.8.0)
- soundfile (>=0.10.0)
2.1 Using OLA and WSOLA, and PV-TSM

OLA, WSOLA, and PV-TSM can be imported as module to be used directly in Python. To get the result easily, all you need is just two parameters, the input audio sequence \( x \) and the time stretching factor \( s \). Here’s a minimal example:

```python
import numpy as np
import pytsmod as tsm  # you can use other audio load packages.
import soundfile as sf
# you can use other audio load packages.

x, sr = sf.read('/FILEPATH/AUDIOFILE.wav')
x = x.T  # if the input is multichannel audio, it is recommended to use the shape
        # (num_channels, audio_len)
x_length = x.shape[-1]
s_fixed = 1.3  # stretch the audio signal 1.3x times.
s_ap = np.array([[0, x_length / 2, x_length], [0, x_length, x_length * 1.5]])  # double the first half of the audio only and preserve the other half.

x_s_fixed = tsm.wsola(x, s_fixed)
x_s_ap = tsm.wsola(x, s_ap)
```

2.1.1 Time stretching factor \( s \)

Time stretching factor \( s \) can either be a constant value (alpha) or an 2 x n array of anchor points which contains the sample points of the input signal in the first row and the sample points of the output signal in the second row.

2.2 Using TD-PSOLA

When using TD-PSOLA, the estimated pitch information of the source you want to modify is needed. Also, you should know the hop size and frame length of the pitch tracking algorithm you used. Here’s a minimal example:
import numpy as np
import pytsmod as tsm
import crepe  # you can use other pitch tracking algorithms.
import soundfile as sf  # you can use other audio load packages.

x, sr = sf.read('/FILEPATH/AUDIOFILE.wav')

_, f0_crepe, _, _ = crepe.predict(x, sr, viterbi=True, step_size=10)

x_double_stretched = tsm.tdpsola(x, sr, f0_crepe, alpha=2, p_hop_size=441, p_win_size=1470)  # hop_size and frame_length for CREPE step_size=10 with sr=44100
x_3keyup = tsm.tdpsola(x, sr, f0_crepe, beta=pow(2, 3/12), p_hop_size=441, p_win_size=1470)

x_3keydown = tsm.tdpsola(x, sr, f0_crepe, target_f0=f0_crepe * pow(2, -3/12), p_hop_size=441, p_win_size=1470)

### 2.2.1 Time stretching factor alpha

In this version, TD-PSOLA only supports the fixed time stretching factor alpha.

### 2.2.2 Pitch shifting factor beta and target_f0

You can modify pitch of the audio sequence in two ways. The first one is beta, which is the fixed pitch shifting factor. The other one is target_f0, which supports target pitch sequence you want to convert. You cannot use both of the parameters.

### 2.3 Command-Line Interface

From version 0.3.0, this package includes a command-line tool named `tmod`, which can create the result file easily from a shell. To generate the WSOLA result of `input.wav` with stretching factor 1.3 and save to `output.wav`, please run:

```bash
$ tmod wsola input.wav output.wav 1.3 # ola, wsola, pv, pv_int are available.
```

Currently, OLA, WSOLA, and Phase Vocoder(PV) are supported. TD-PSOLA is excluded due to the difficulty of sending extracted pitch data to TD-PSOLA. Also, non-linear TSM is not supported in command-line.

For more information, use `-h` or `--help` command to see the detailed usage of `tmod`. 
Modify length of the audio sequence using OLA algorithm. WSOLA with zero tolerance is working same as OLA.

**Parameters**

- `x` [numpy.ndarray [shape=(channel, num_samples) or (num_samples)]] the input audio sequence to modify.
- `s` [number > 0 [scalar] or numpy.ndarray [shape=(2, num_points)]] the time stretching factor. Either a constant value (alpha) or an 2 x n array of anchor points which contains the sample points of the input signal in the first row and the sample points of the output signal in the second row.
- `win_type` [str] type of the window function. hann and sin are available.
- `win_size` [int > 0 [scalar]] size of the window function.
- `syn_hop_size` [int > 0 [scalar]] hop size of the synthesis window. Usually half of the window size.

**Returns**

- `y` [numpy.ndarray [shape=(channel, num_samples) or (num_samples)]] the modified output audio sequence.

Modify length of the audio sequence using WSOLA algorithm.

**Parameters**

- `x` [numpy.ndarray [shape=(channel, num_samples) or (num_samples)]] the input audio sequence to modify.
- `s` [number > 0 [scalar] or numpy.ndarray [shape=(2, num_points)]] the time stretching factor. Either a constant value (alpha) or an 2 x n array of anchor points which contains the sample points of the input signal in the first row and the sample points of the output signal in the second row.

- `tolerance` [int > 0 [scalar]] tolerance value in WSOLA.
PyTSMod, Release 0.3.3

win_type [str] type of the window function. hann and sin are available.

win_size [int > 0 [scalar]] size of the window function.

syn_hop_size [int > 0 [scalar]] hop size of the synthesis window. Usually half of the window size.

tolerance [int >= 0 [scalar]] number of samples the window positions in the input signal may be shifted to avoid phase discontinuities when overlap-adding them to form the output signal (given in samples).

Returns

y [numpy.ndarray [shape=(channel, num_samples) or (num_samples)]] the modified output audio sequence.

pytsmod.phase_vocoder(x, s, win_type='sin', win_size=2048, syn_hop_size=512, zero_pad=0, restore_energy=False, fft_shift=False, phase_lock=False)

Modify length of the audio sequence using Phase Vocoder algorithm.

Parameters

x [numpy.ndarray [shape=(channel, num_samples) or (num_samples)]] the input audio sequence to modify.

s [number > 0 [scalar] or numpy.ndarray [shape=(2, num_points)]] the time stretching factor. Either a constant value (alpha) or an 2 x n array of anchor points which contains the sample points of the input signal in the first row and the sample points of the output signal in the second row.

win_type [str] type of the window function for the STFT. hann and sin are available.

win_size [int > 0 [scalar]] size of the window function.

syn_hop_size [int > 0 [scalar]] hop size of the synthesis window. Usually half of the window size.

zero_pad [int > 0 [scalar]] the size of the zero pad in the window function.

restore_energy [bool] tries to reserve potential energy loss.

fft_shift [bool] apply circular shift to STFT and ISTFT.

phase_lock [bool] apply phase locking.

Returns

y [numpy.ndarray [shape=(channel, num_samples) or (num_samples)]] the modified output audio sequence.

pytsmod.phase_vocoder_int(x, s, win_type='hann', win_size=2048, syn_hop_size=512, zero_pad=None, restore_energy=False, fft_shift=True)

Modify length of the audio sequence using Phase Vocoder algorithm. Works specially well for integer stretching.

Parameters

x [numpy.ndarray [shape=(channel, num_samples) or (num_samples)]] the input audio sequence to modify.

alpha [int > 0 [scalar]] the time stretching factor. Only a integer value greater than 0 is allowed.

win_type [str] type of the window function for the STFT. hann and sin are available.

win_size [int > 0 [scalar]] size of the window function.

syn_hop_size [int > 0 [scalar]] hop size of the synthesis window. Usually half of the window size.
zero_pad [int > 0 [scalar]] the size of the zero pad in the window function.

restore_energy [bool] tries to reserve potential energy loss.

fft_shift [bool] apply circular shift to STFT and ISTFT.

Returns

y [numpy.ndarray [shape=(channel, num_samples) or (num_samples)]] the modified output audio sequence.

pytsmod.hptsm(x, s, hp_kernel_size=31, hp_power=2.0, hp_mask=False, hp_margin=1.0,
      pv_win_type='hann', pv_win_size=2048, pv_syn_hop_size=512, pv_zero_pad=0,
      pv_restore_energy=False, pv_fft_shift=False, pv_phase_lock=True,
      ola_win_type='hann', ola_win_size=256, ola_syn_hop_size=128)

Modify length of the audio sequence using both Phase Vocoder and OLA. Apply Phase Vocoder to harmonic signal, and apply OLA to percussive signal. For HPSS, median filter based algorithm is used.

Parameters

x [numpy.ndarray [shape=(channel, num_samples) or (num_samples)]] the input audio sequence to modify.

s [number > 0 [scalar] or numpy.ndarray [shape=(2, num_points)]] the time stretching factor. Either a constant value (alpha) or an 2 x n array of anchor points which contains the sample points of the input signal in the first row and the sample points of the output signal in the second row.

hp_ [parameters for HPSS.]

pv_ [parameters for phase vocoder.]

ola_ [parameters for OLA.]

Returns

y [numpy.ndarray [shape=(channel, num_samples) or (num_samples)]] the modified output audio sequence.

pytsmod.tdpsola(x, sr, src_f0=None, tgt_f0=None, alpha=1, beta=None, win_type='hann', p_hop_size=441,
       p_win_size=1470)

Modify length and pitch of the audio sequence using TD-PSOLA algorithm.

Parameters

x [numpy.ndarray [shape=(channel, num_samples) or (num_samples)]] the input audio sequence to modify.

sr [int > 0 [scalar]] sample rate of the input audio sequence.

src_f0 [numpy.ndarray [shape=(channel, num_freqs) or (num_freqs)]] the fundamental frequency contour of the input audio sequence.

tgt_f0 [numpy.ndarray [shape=(channel, num_freqs) or (num_freqs)]] the target fundamental frequency contour you want to modify the input audio sequence. Should not be used with beta.

alpha [number > 0 [scalar]] time stretching factor.

beta [number > 0 [scalar]] the pitch shifting factor. should not be used with target_f0.

win_type [str] type of the window function. hann and sin are available.

p_hop_size [int > 0 [scalar]] the hop size of src_f0 (in samples).

p_win_size [int > 0 [scalar]] the window size of pitch tracking algorithm you used. (in samples).
Returns

$y$ [numpy.ndarray \[shape=(channel, num_samples) or (num_samples)]] the modified output audio sequence.
pytsmod.utils.win(win_type='hann', win_size=4096, zero_pad=0)
Generate diverse type of window function

Parameters

- **win_type** [str] the type of window function. Currently, Hann and Sin are supported.
- **win_size** [int > 0 [scalar]] the size of window function. It doesn’t contains the length of zero padding.
- **zero_pad** [int > 0 [scalar]] the total length of zero-pad. Zeros are equally distributed for both left and right of the window.

Returns

- **win** [numpy.ndarray[shape=(win_size)]] the window function generated.

pytsmod.utils.stft(x, ana_hop=2048, win_type='hann', win_size=4096, zero_pad=0, sr=44100, 
fft_shift=0, time_frequency_out=False)
Short-Time Fourier Transform (STFT) for the audio signal. This function is used for phase vocoder.

Parameters

- **x** [numpy.ndarray[shape=(num_samples)]] the input audio sequence. Should be a single channel.
- **ana_hop** [int > 0 [scalar] or numpy.ndarray[shape=(num_frames)]] either a analysis hop size (scalar) or analyze window positions (array).
- **win_type** [str] type of the window function for the STFT. hann and sin are available.
- **win_size** [int > 0 [scalar]] size of the window function.
- **zero_pad** [int > 0 [scalar]] the size of the zero pad in the window function.
- **sr** [int > 0 [scalar]] the sample rate of the audio sequence.
- **fft_shift** [bool] apply circular shift to STFT.
- **time_frequency_out** [bool] returns time and frequency axis indices in (spec, t, f).
Returns

- `spec` [numpy.ndarray [shape=(win_size // 2 + 1, num_frames)]] the STFT result of the input audio sequence.
- `t` [numpy.ndarray [shape=num_frames]] timestamp of the output result.
- `f` [numpy.ndarray [shape=win_size // 2 + 1]] frequency value for each frequency bin of the output result.

```python
pytsmod.utils.istft(spec, syn_hop=2048, win_type='hann', win_size=4096, zero_pad=0, num_iter=1, original_length=-1, fft_shift=False, restore_energy=False)
```
Inverse Short-Time Fourier Transform to recover the audio signal from the spectrogram. This function is used for phase vocoder.

Parameters

- `X` [numpy.ndarray [shape=(num_bins, num_frames)]] the input audio complex spectrogram.
- `syn_hop` [int > 0 [scalar]] the hop size of the synthesis window.
- `win_type` [str] type of the window function for the ISTFT. hann and sin are available.
- `win_size` [int > 0 [scalar]] size of the window function.
- `zero_pad` [int > 0 [scalar]] the size of the zero pad in the window function.
- `num_iter` [int > 0 [scalar]] the number of iterations the algorithm should perform to adapt the phase.
- `original_length` [int > 0 [scalar]] original length of the audio signal.
- `fft_shift` [bool] apply circular shift to ISTFT.
- `restore_energy` [bool] tries to reserve potential energy loss.

Returns

- `y` [numpy.ndarray [shape=(original_length)]] the output audio sequence.

```python
pytsmod.utils.lsee_mstft(X, syn_hop, win_type, win_size, zero_pad, fft_shift, restore_energy)
```
Least Squares Error Estimation from the MSTFT (Modified STFT). Griffin-Lim procedure to estimate the audio signal from the modified STFT.

Parameters

- `X` [numpy.ndarray [shape=(num_bins, num_frames)]] the input audio complex spectrogram.
- `syn_hop` [int > 0 [scalar]] the hop size of the synthesis window.
- `win_type` [str] type of the window function for the ISTFT. hann and sin are available.
- `win_size` [int > 0 [scalar]] size of the window function.
- `zero_pad` [int > 0 [scalar]] the size of the zero pad in the window function.
- `fft_shift` [bool] apply circular shift to ISTFT.
- `restore_energy` [bool] tries to reserve potential energy loss.

Returns

- `x` [numpy.ndarray [shape=num_samples]] the output audio sequence through LSEE_MSTFT

```python
pytsmod.utils._validate_audio(audio)
```
validate the input audio and modify the order of channels.

Parameters
audio [numpy.ndarray [shape=(channel, num_samples) or (num_samples) or (num_samples, channel)]] the input audio sequence to validate.

Returns

audio [numpy.ndarray [shape=(channel, num_samples)]] the validated output audio sequence.

pytsmod.utils._validate_scale_factor (audio, s)
Validate the scale factor s and convert the fixed scale factor to anchor points.

Parameters

audio [numpy.ndarray [shape=(num_channels, num_samples) or (num_samples) or (num_samples, num_channels)]] the input audio sequence.
s [number > 0 [scalar] or numpy.ndarray [shape=(2, num_points) or (num_points, 2)]] the time stretching factor. Either a constant value (alpha) or an (2 x n) (or (n x 2)) array of anchor points which contains the sample points of the input signal in the first row and the sample points of the output signal in the second row.

Returns

anc_points [numpy.ndarray [shape=(2, num_points)]] anchor points which contains the sample points of the input signal in the first row and the sample points of the output signal in the second row.

pytsmod.utils._validate_f0 (audio, f0)
Validate the input f0 is suitable for input audio.

Parameters

audio [numpy.ndarray [shape=(num_channels, num_samples) or (num_samples) or (num_samples, num_channels)]] the input audio sequence.
f0 [numpy.ndarray [shape=(num_channels, num_pitches) or (num_pitches) or (num_pitches, num_channels)]] the f0 sequence that used for TD-PSOLA. If f0 is 1D array, the f0 of all audio channels are regarded as the same f0.

Returns

f0 [numpy.ndarray [shape=(num_channels, num_freqs)]] the f0 sequence that used for TD-PSOLA.
p
pytsmod, 7
pytsmod.utils, 11
Index

Symbols

_validate_audio() (in module pytsmod.utils), 12
_validate_f0() (in module pytsmod.utils), 13
_validate_scale_factor() (in module pytsmod.utils), 13

H
hptsm() (in module pytsmod), 9

I
istft() (in module pytsmod.utils), 12

L
lsee_mstft() (in module pytsmod.utils), 12

O
ola() (in module pytsmod), 7

P
phase_vocoder() (in module pytsmod), 8
phase_vocoder_int() (in module pytsmod), 8
pytsmod (module), 7
pytsmod.utils (module), 11

S
stft() (in module pytsmod.utils), 11

T
tdpsola() (in module pytsmod), 9

W
win() (in module pytsmod.utils), 11
wsola() (in module pytsmod), 7