AMFM_decompy Documentation

Release 1

Bernardo J.B. Schmitt

September 19, 2014

CONTENTS

Contents:

CHAPTER

ONE

PYAAPT

This a ported version for Python from the YAAPT (Yet Another Algorithm for Pitch Tracking) algorithm. The original MATLAB program was written by Hongbing Hu and Stephen A. Zahorian.

The YAAPT program, designed for fundamental frequency tracking, is extremely robust for both high quality and telephone speech. The YAAPT program was created by the Speech Communication Laboratory of the state university of New York at Binghamton. The original program is available at http://www.ws.binghamton.edu/zahorian as free software. Further information about the program could be found at [ref1].

It must be noticed that, although this ported version is almost equal to the original, some few changes were made in order to make the program more "pythonic" and improve its performance. Nevertheless, the results obtained with both algorithms were similar.

1.1 Quick start

The pYAAPT basically contains the whole set of functions to extract the pitch track from a speech signal. These functions, in their turn, are independent from the pyQHM package. Therefore, pYAAPT can be used in any other speech processing application, not only in the AM-FM decomposition. Its only dependency is the pcm_utility.py (in case that is necessary to read a wav file), which is already included in the AMFM_decompy package.

USAGE:

`amfm_decompy.pYAAPT.`**`yaapt`**(*raw_signal*, *sample_frequency*[, *options*])
`amfm_decompy.pYAAPT.`**`yaapt`**(*wav_file*[, *options*])

Parameters

- **raw_signal** (*numpy array*) – speech signal.
- **sample_frequency** – sample rate of the input data in Hz.
- **options** (Must be formatted as follow: **{'option_name1' : value1, 'option_name2' : value2, ...}) – the default configuration values for all of them are the same as in the original version. A short description about them is presented in the next subitem. For more information about them, please refer to [ref1].
- **wav_file** (*string*) – string containing the path from the speech signal wav file.

Return type pitch object

OPTIONS:

- 'frame_length' - length of each analysis frame (default: 25 ms)
- 'frame_space' - spacing between analysis frames (default: 10 ms)
- 'f0_min' - minimum pitch searched (default: 60 Hz)

- 'f0_max' - maximum pitch searched (default: 400 Hz)
- 'fft_length' - FFT length (default: 8192 samples)
- 'bp_forder' - order of band-pass filter (default: 150)
- 'bp_low' - low frequency of filter passband (default: 50 Hz)
- 'bp_high' - high frequency of filter passband (default: 1500 Hz)
- 'nlfer_thresh1' - NLFER (Normalized Low Frequency Energy Ratio) boundary for voiced/unvoiced decisions (default: 0.75)
- 'nlfer_thresh2' - threshold for NLFER definitely unvoiced (default: 0.1)
- 'shc_numharms' - number of harmonics in SHC (Spectral Harmonics Correlation) calculation (default: 3)
- 'shc_window' - SHC window length (default: 40 Hz)
- 'shc_maxpeaks' - maximum number of SHC peaks to be found (default: 4)
- 'shc_pwidth' - window width in SHC peak picking (default: 50 Hz)
- 'shc_thresh1' - threshold 1 for SHC peak picking (default: 5)
- 'shc_thresh2' - threshold 2 for SHC peak picking (default: 1.25)
- 'f0_double'- pitch doubling decision threshold (default: 150 Hz)
- 'f0_half' - pitch halving decision threshold (default: 150 Hz)
- 'dp5_k1' - weight used in dynamic program (default: 11)
- 'dec_factor' - factor for signal resampling (default: 1)
- 'nccf_thresh1' - threshold for considering a peak in NCCF (Normalized Cross Correlation Function) (default: 0.25)
- 'nccf_thresh2' - threshold for terminating search in NCCF (default: 0.9)
- 'nccf_maxcands' - maximum number of candidates found (default: 3)
- 'nccf_pwidth' - window width in NCCF peak picking (default: 5)
- 'merit_boost' - boost merit (default. 0.20)
- 'merit_pivot' - merit assigned to unvoiced candidates in definitely unvoiced frames (default: 0.99)
- 'merit_extra' - merit assigned to extra candidates in reducing pitch doubling/halving errors (default: 0.4)
- 'median_value' - order of medial filter (default: 7)
- 'dp_w1' - DP (Dynamic Programming) weight factor for voiced-voiced transitions (default: 0.15)
- 'dp_w2' - DP weight factor for voiced-unvoiced or unvoiced-voiced transitions (default: 0.5)
- 'dp_w3' - DP weight factor of unvoiced-unvoiced transitions (default: 0.1)
- 'dp_w4' - Weight factor for local costs (default: 0.9)

EXAMPLES:

Example 1 - read a file data and extract the pitch track using the default configurations:

```
import amfm_decompy.pYAAPT as pYAAPT
import numpy as np
from amfm_decompy.pcm_utility import pcm2float
from scipy.io import wavfile
```

```
frequency, data = wavfile.read('path_to_sample.wav')
frequency = float(frequency)
data = pcm2float(data, dtype='f')

pitch = pYAAPT.yaapt(data, frequency)
```

Example 2 - read a wav file and extract the pitch track. The minimum pitch is set to 150 Hz, the frame length to 15 ms and the frame jump to 5 ms:

```
import amfm_decompy.pYAAPT as pYAAPT

pitch = pYAAPT.yaapt('path_to_sample.wav', **{'f0_min' : 150.0, 'frame_length' : 15.0, \
       'frame_space' : 5.0})
```

1.2 Classes

1.2.1 PitchObj Class

The PitchObj Class stores the extracted pitch and all the parameters related to it. A pitch object is necessary for the QHM algorithms. However, the pitch class structure was built in a way that it can be used by any other pitch tracker, not only the YAAPT.

USAGE:

amfm_decompy.pYAAPT.**PitchObj**(*frame_size*, *frame_jump*[, *nfft=8192*])

Parameters

- **frame_size** (*int*) – analysis frame length.
- **frame_jump** (*int*) – distance between the center of a extracting frame and the center of its adjacent neighbours.
- **nfft** (*int*) – FFT length.

Return type pitch object.

PITCH CLASS VARIABLES:

These variables not related with the YAAPT algorithm itself, but with a post-processing where the data is smoothed and halving/doubling errors corrected.

PitchObj.**PITCH_HALF**
: This variable is a flag. When its value is equal to 1, the halving detector set the half pitch values to 0. If PITCH_HALF is equal to 2, the half pitch values are multiplied by 2. For other PITCH_HALF values, the halving detector is not employed (default: 0).

PitchObj.**PITCH_HALF_SENS**
: Set the halving detector sensibility. A pitch sample is considered half valued if it is not zero and lower than:

  mean(pitch) - PITCH_HALF_SENS*std(pitch)

  (default: 2.9).

PitchObj.**PITCH_DOUBLE**
: This variable is a flag. When its value is equal to 1, the doubling detector set the double pitch values to 0. If PITCH_DOUBLE is equal to 2, the double pitch values are divided by 2. For other PITCH_DOUBLE values, the doubling detector is not employed (default: 0).

PitchObj.**PITCH_DOUBLE_SENS**
Set the doubling detector sensibility. A pitch sample is considered double valued if it is not zero and higher than:

mean(pitch) + PITCH_DOUBLE_SENS*std(pitch)

(default: 2.9).

PitchObj.**SMOOTH_FACTOR**
Determines the median filter length used to smooth the interpolated pitch values (default: 5). [1]

PitchObj.**SMOOTH**
This variable is a flag. When its value is not equal to 0, the interpolated pitch is smoothed by a median filter (default: 5). [1]

PitchObj.**PTCH_TYP**
If there are less than 2 voiced frames in the file, the PTCH_TYP value is used in the interpolation (default: 100 Hz). [1]

EXAMPLE:

Example 1 - the pitch is extracted from sample.wav and after that, recalculated with different smoothing and interpolation configurations:

```
import amfm_decompy.pYAAPT as pYAAPT

pYAAPT.PitchObj.PITCH_DOUBLE = 2       # set new values
pYAAPT.PitchObj.PITCH_HALF = 2
pYAAPT.PitchObj.SMOOTH_FACTOR = 3

pitch = pYAAPT.yaapt('path_to_sample.wav') # calculate the pitch track
```

PITCH OBJECT ATTRIBUTES:

PitchObj.**nfft**
Length in samples from the FFT used by the pitch tracker. It is set during the object's initialization.

PitchObj.**frame_size**
Length in samples from the frames used by the pitch tracker. It is set during the object's initialization.

PitchObj.**frame_jump**
Distance in samples between the center of a extracting frame and the center of its adjacent neighbours. It is set during the object's initialization.

PitchObj.**noverlap**
It's the difference between the frame size and the frame jump. Represents the number of samples that two adjacent frames share in common, i.e, how much they overlap each other. It is set during the object's initialization.

PitchObj.**mean_energy**
Signal's low frequency band mean energy. It is set by the PitchObj.set_energy method.

PitchObj.**energy**
Array that contains the low frequency band energy from each frame, normalized by PitchObj.mean_energy. It is set by the PitchObj.set_energy method.

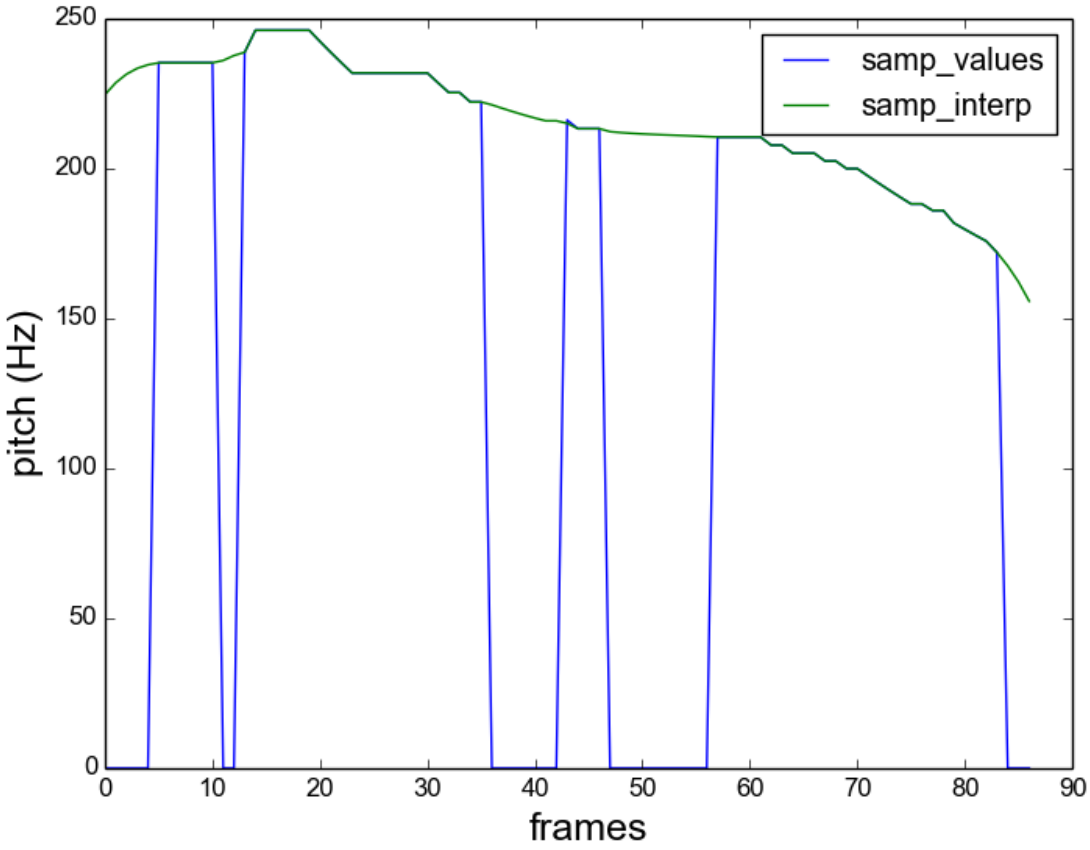
PitchObj.**vuv**
Boolean vector that indicates if each speech frame was classified as voiced (represented as 'True') or unvoiced (represented as 'False'). It is set by the PitchObj.set_energy method.

[1] don't mistake this interpolation with the one performed by the pYAAPT.upsample method. For more explanation, please refer to the pYAAPT.samp_interp and pYAAPT.values_interp attributes.

PitchObj.**frames_pos**
A numpy array that contains the samples where the center of the frames were placed during the extraction. It is set by the PitchObj.set_frame_pos method.

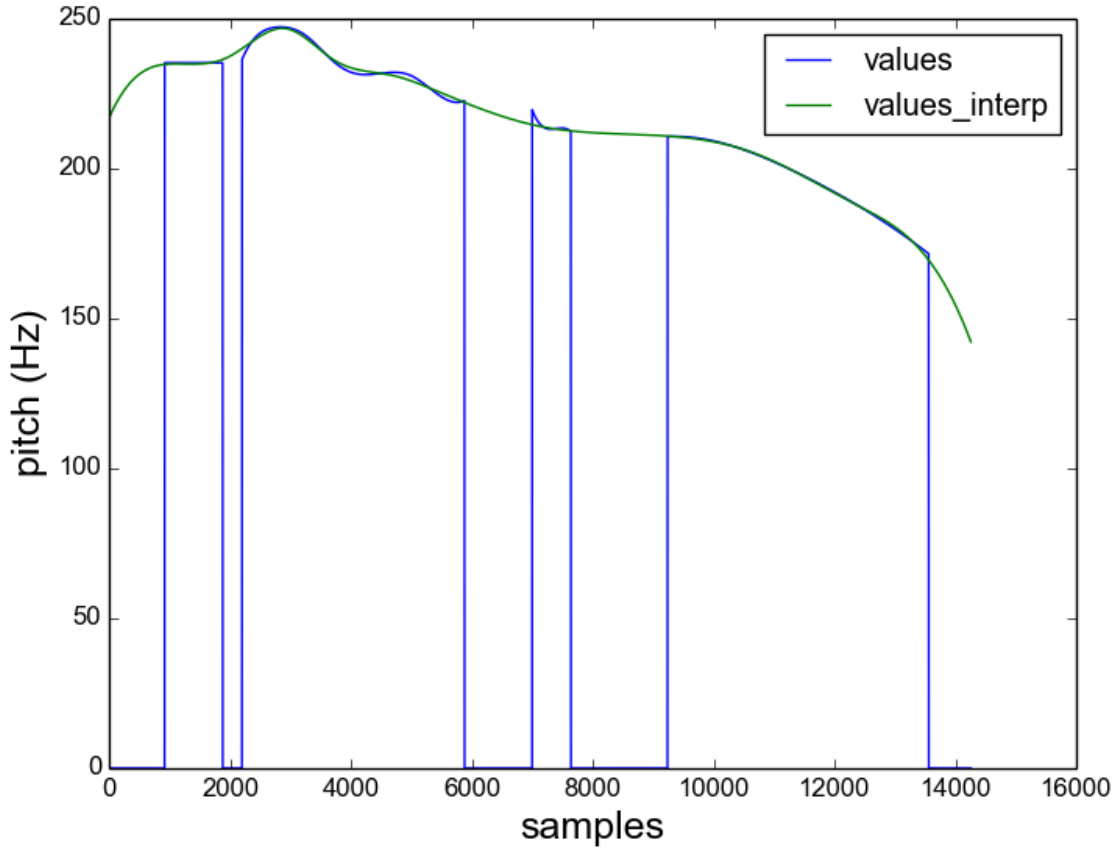
PitchObj.**nframes**
Number of frames. It is set by the PitchObj.set_frame_pos method.

PitchObj.**samp_values**
PitchObj.**samp_interp**
Both arrays contain the pitch values from each of the nframes. The only difference is that, in PitchObj.samp_interp the unvoiced segments are substituted by the interpolation from the adjacent voiced segments edges. This provides a non-zero version from the pitch track, which can be necessary for some applications. The figure below presents both arrays from the sample.wav file:



Both attributes are set by the PitchObj.set_values method.

PitchObj.**values**
PitchObj.**values_interp**
PitchObj.values and PitchObj.values_interp are the upsampled versions from PitchObj.samp_values and PitchObj.samp_interp respectively. Therefore, their length is equal to the original file length (for more information, check the PitchObj.upsample() method). The figure below presents both arrays from the sample.wav file:



Both attributes are set by the PitchObj.set_values method.

`PitchObj.`**`edges`**
: A list that contains the index where occur the transitions between unvoiced-voiced and voiced-unvoiced in PitchObj.values. It is set by the PitchObj.set_values method, which employs internally the PitchObj.edges_finder method.

PITCH OBJECT METHODS:

`PitchObj.`**`set_energy`**(*energy*, *threshold*)

Parameters

- **energy** (*numpy array*) – contains the low frequency energy for each frame.
- **threshold** – normalized threshold.

Set the normalized low frequency energy by taking the input array and dividing it by its mean value. Normalized values above the threshold are considered voiced frames, while the ones below it are unvoiced frames.

`PitchObj.`**`set_frames_pos`**(*frames_pos*)

Parameters **frames_pos** – index with the sample positions.

Set the position from the center of the extraction frames.

`PitchObj.`**`set_values`**(*samp_values*, *file_size*[, *interp_tech='spline'*])

Parameters

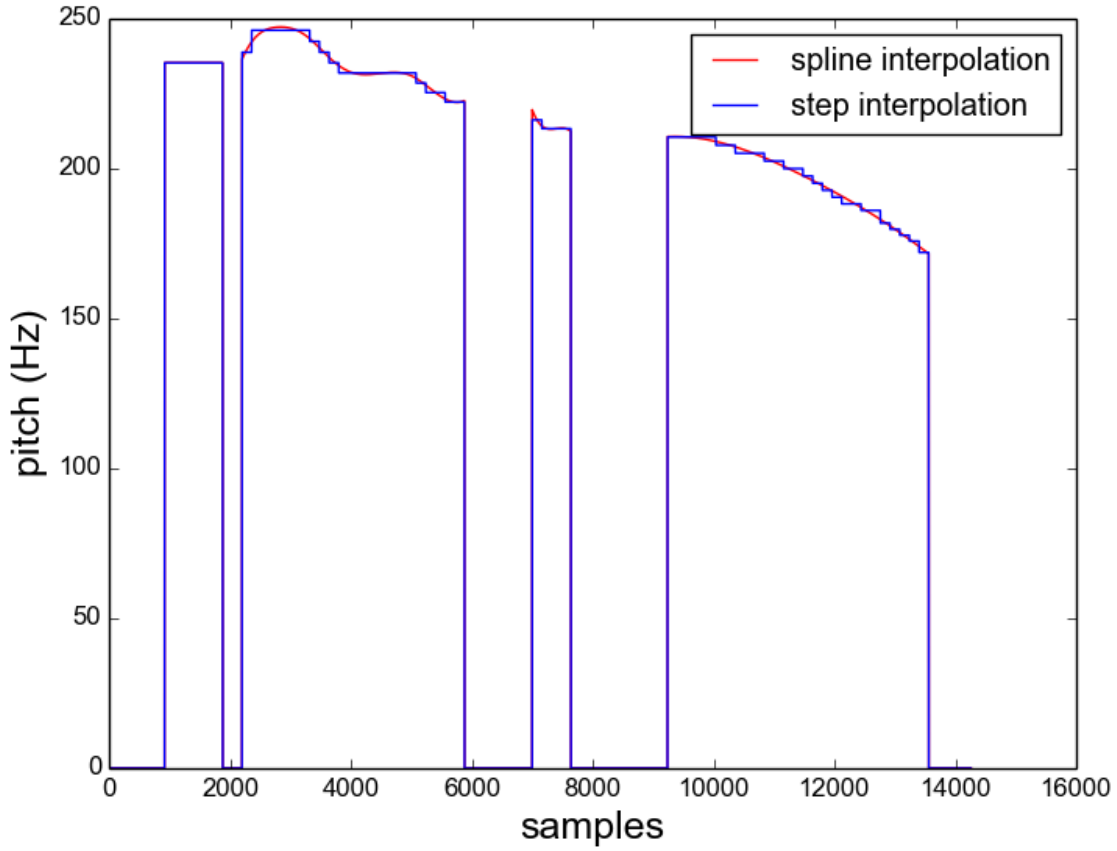
- **samp_values** (*numpy array*) – pitch value for each frame.
- **file_size** (*int*) – length of the speech signal.
- **interp_tech** (*string*) – interpolation method employed to upsample the data. Can be 'spline' (default) and 'step'.

Set the pitch values and also calculates its interpolated version (for more information, check the PitchObj.samp_values and PitchObj.samp_interp attributes). A post-process is employed then using the PitchObj class attributes. After that, both arrays are upsampled, so that the output arrays have the same length as the original speech signal. In this process, a second interpolation is necessary. The interpolation technique employed is indicated by the parameter interp_tech.

Example:

```
import amfm_decompy.pYAAPT as pYAAPT
from matplotlib import pyplot as plt

pitch = pYAAPT.yaapt('path_to_ample.wav')

plt.plot(pitch.values, label='spline interpolation', color='red')

pitch.set_values(pitch.samp_values, len(pitch.values), interp_tech='step')

plt.plot(pitch.values, label='step interpolation')

plt.xlabel('samples', fontsize=18)
plt.ylabel('pitch (Hz)', fontsize=18)
plt.legend(loc='upper right')
```

The output is presented below:



PitchObj.**edges_finder**(*values*)

Parameters **values** (*numpy array*) – contains the low frequency energy for each frame.

Return type list.

Returns the index of the samples where occur the transitions between unvoiced-voiced and voiced-unvoiced.

1.2.2 SignalObj Class

The SignalObj Class stores the speech signal and all the parameters related to it. It must be noticed that the SignalObj here is different from the pyQHM module one.

USAGE:

amfm_decompy.pYAAPT.**SignalObj**(*args*)

Parameters **args** – the input argument can be a string with the wav file path OR a tuple containing the speech signal data and its fundamental frequency in Hz.

Return type speech signal object.

SIGNAL OBJECT ATTRIBUTES:

SignalObj.**data**
Numpy array containing the speech signal data. It is set during the object's initialization.

SignalObj.**fs**
Sample frequency in Hz. It is set during the object's initialization.

SignalObj.**size**
Speech signal length. It is set during the object's initialization.

SignalObj.**filtered**
Bandpassed version from the speech data. It is set by the SignalObj.filtered_version method.

SignalObj.**new_fs**
Downsampled fundamental frequency from the speech data. It is set by the SignalObj.filtered_version method.

SIGNAL OBJECT METHODS:

SignalObj.**filtered_version**(*bp_filter*)

Parameters **bp_filter** – BandpassFilter object.

Filters the signal data by a bandpass filter.

1.2.3 BandpassFilter Class

Creates a bandpass filter necessary for the YAAPT algorithm.

USAGE:

amfm_decompy.pYAAPT.**BandpassFilter**(*fs*, *parameters*)

Parameters

- **fs** (*float*) – signal's fundamental frequency
- **parameters** (*dictionary*) – contains the parameters options from the YAAPT algorithm.

Return type bandpass filter object.

BANDPASS FILTER ATTRIBUTES:

BandpassFilter.**b**
Bandpass filter zeros coefficients. It is set during the object's initialization.

BandpassFilter.**a**
Bandpass filter poles coefficients. It is set during the object's initialization.

BandpassFilter.**dec_factor**
Decimation factor used for downsampling the data. It is set during the object's initialization.

1.3 References

[ref1] Stephen A. Zahorian, and Hongbing Hu, “A spectral/temporal method for robust fundamental frequency tracking,” J. Acosut. Soc. Am. 123(6), June 2008.