libf0: A PYTHON LIBRARY FOR FUNDAMENTAL FREQUENCY ESTIMATION

Sebastian Rosenzweig **Simon Schwär** **Meinard Müller**
International Audio Laboratories Erlangen, Germany
{sebastian.rosenzweig, simon.schwaer, meinard.mueller}@audiolabs-erlangen.de

Late-Breaking / Demo Session Extended Abstract, ISMIR 2022 Conference

ABSTRACT

The extraction of fundamental frequency (F0) information from music recordings is a crucial task in the field of music information retrieval. The sequence of F0-estimates over successive time frames (also called F0-trajectory) often corresponds to a melodic phrase and serves as a representation for downstream tasks such as automatic music transcription and performance analysis. A large number of algorithms and tools for F0-estimation have been proposed in the literature and implemented in various programming languages. However, these heterogeneous implementations are often not easily comparable and may vary considerably in performance and accuracy, which is problematic for reproducible research. In this contribution, we introduce libf0, a Python library of reference implementations that can conveniently be used to apply, compare, and develop F0-estimation algorithms in a reproducible way.

1. INTRODUCTION

Over the last decades, various approaches for the estimation of the fundamental frequency (F0) of an audio signal have been proposed, ranging from model-based algorithms in time- or frequency-domain to deep learning systems. Along with the scientific literature, one can find open-source implementations for many F0-estimation algorithms. For example, the following versions for the four popular algorithms YIN [1], pYIN [2], Melodia [3], and SWIPE [4] are available: Python [1] and Matlab [2] for YIN, Python [3] and Vamp-Plugin [4] for pYIN, C++ [5] and Vamp-Plugin [6] for Melodia, and Matlab and C [7] for SWIPE. This creates two challenges for reproducible research. First, for most existing implementations, the source cannot be cited or traced back. This is particularly problematic since we found that different implementations of the same algorithm can vary significantly in performance and accuracy. Second, studies that require applying several algorithms jointly, e.g., fusion experiments [5] or comparative studies, are hard to conduct due to the diversity of programming languages and sources. In this contribution, we introduce a Python package called libf0 that contains open-source implementations of the aforementioned algorithms YIN, pYIN, a salience-based approach inspired by Melodia, and SWIPE. Our toolbox offers reproducible baseline implementations for developing and comparing F0-estimation algorithms for computational analysis of music recordings. Furthermore, our toolbox contains example code for visualizing and sonifying F0-trajectories to gain an intuitive understanding of the estimation results.

[1] https://librosa.org/doc/main/generated/librosa.yin.html
[2] http://audition.ens.fr/adc/
[3] https://librosa.org/doc/main/generated/librosa.pyin.html
[4] https://code.soundsoftware.ac.uk/projects/pyin
[5] https://essentia.upf.edu/reference/std_PredominantPitchMelodia.html
[6] https://www.upf.edu/web/mtg/melodia
[7] https://github.com/kylebgorman/swipe

2. F0-ALGORITHMS

In the following, we briefly describe the algorithms included in libf0. For illustration, we use an excerpt of a singing voice recording from the publicly available multitrack choral singing dataset "Dagstuhl ChoirSet" [6] as our running example. The recording was obtained from a soprano singer using a throat microphone. In Fig. 1, estimated F0-trajectories are shown on a log-frequency spectrogram of this excerpt to give a visual reference and highlight the differences between estimates.

2.1 YIN

One of the most well-known algorithms for F0-estimation is YIN, which was first introduced in [1]. YIN is a time-domain algorithm, which produces one F0-estimate for each time frame. First, using a variant of autocorrelation, one computes a function referred to as *cumulative mean normalized difference function* (CMNDF). This function has local minima at the multiples of the expected period, whose reciprocal is the F0-estimate. Fig. 1a depicts a log-frequency spectrogram superimposed with the F0-trajectory estimated by YIN for our running example. Since the algorithm does not enforce continuity of the estimated F0-trajectories, one often obtains highly fluctuating F0-estimates (e.g., see Fig. 1a at around 6 seconds). In particular, YIN suffers from confusion of the F0 with higher harmonics (especially the octave).

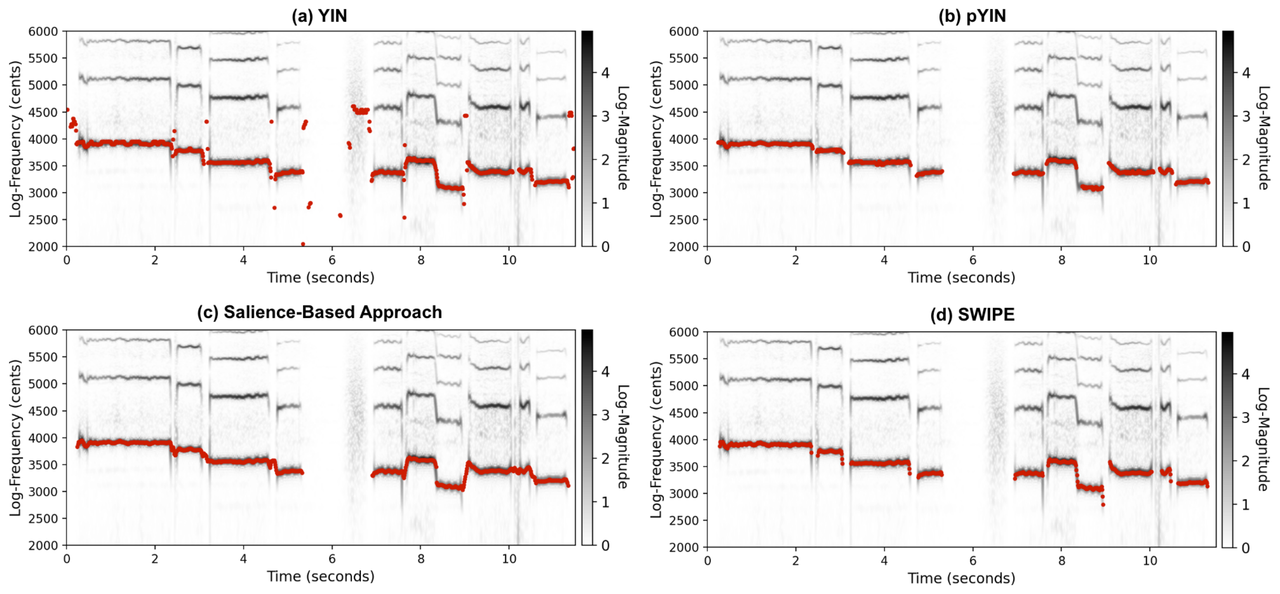

Figure 1. Illustration of different F0-estimation algorithms of libf0. The log-frequency spectrograms (with frequency values given in cents with reference frequency 55 Hz) are shown only for visualization purposes.

2.2 pYIN

Probabilistic YIN (pYIN), introduced in [2], is a modification of the previously described YIN algorithm. It increases the robustness and alleviates the continuity problems of the YIN algorithm at the cost of increased computational complexity. First, one applies YIN multiple times with different thresholds, yielding multiple F0-candidates per frame. Then, temporal smoothing is achieved by using a hidden Markov model (HMM) and Viterbi decoding. Furthermore, the HMM smoothing includes frame-wise decision of whether a frame is voiced or unvoiced (commonly referred to as *voicing detection*). Fig. 1b shows the improved pYIN-trajectory for our running example.

2.3 Salience-Based Approach

This approach is inspired by the Melodia algorithm introduced in [3]. Melodia is a frequency domain algorithm, which relies on an enhanced time–frequency representation (also called *salience representation*) of the audio signal. First, a short-time Fourier transform (STFT) is computed from the signal, which is refined using the *instantaneous frequency* (IF) (see [3] and [7, Section 8.2.1] for details). These IF-estimates are binned onto a logarithmic frequency axis and *harmonic summation* is applied to enhance the magnitude of the expected F0. To compute the F0-trajectory, our implementation uses a dynamic programming approach as described in [7, Section 8.2.1]. The salience representation for our running example along with the estimated F0-trajectory is visualized in Fig. 1c.

2.4 SWIPE

SWIPE (Sawtooth Waveform Inspired Pitch Estimator) is a frequency domain algorithm originally introduced in [4]. The algorithm works in frequency domain and relies on a correlation with multiple precomputed kernels, each representing a single F0-candidate. Each kernel is constructed such that it has local maxima at prime multiples of the F0-candidate with decreasing amplitude over frequency. The correlation-maximizing kernel yields the F0-estimate for a given frame, possibly refined using parabolic interpolation. Fig. 1d shows the F0-trajectory estimated by SWIPE for our running example. Note that SWIPE, due to its robust kernels, is able to produce smooth estimates without an additional temporal smoothing step.

3. USAGE

We included our libf0 toolbox in the Python package index PyPI, which makes it possible to install libf0 with the standard Python package manager pip. The GitHub repository[8] contains an additional a demo notebook `demo_libf0.ipynb`, which reproduces the figures in this paper and allows for exploring the algorithms and different parameter settings interactively. Beyond visualizations, the notebook provides sinusoidal sonifications of the estimated F0-trajectories for acoustical evaluation. In conjunction with publicly available datasets, e.g., as provided by the mirdata library [8], and evaluation toolboxes such as mir_eval [9], we hope that libf0 contributes to the reproducibility of MIR research on F0-estimation and subsequent downstream tasks.

4. ACKNOWLEDGEMENTS

This work was supported by the German Research Foundation (MU 2686/13-1, SCHE 280/20-1). We thank Edgar Suárez and Vojtech Pesek for helping with the implementations. The International Audio Laboratories Erlangen are a joint institution of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and Fraunhofer Institute for Integrated Circuits IIS.

[8] https://github.com/groupmm/libf0

5. REFERENCES

[1] A. de Cheveigné and H. Kawahara, "YIN, a fundamental frequency estimator for speech and music." *Journal of the Acoustical Society of America (JASA)*, vol. 111, no. 4, pp. 1917–1930, 2002.

[2] M. Mauch and S. Dixon, "pYIN: A fundamental frequency estimator using probabilistic threshold distributions," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Florence, Italy, 2014, pp. 659–663.

[3] J. Salamon and E. Gómez, "Melody extraction from polyphonic music signals using pitch contour characteristics," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 6, pp. 1759–1770, 2012.

[4] A. Camacho and J. G. Harris, "A sawtooth waveform inspired pitch estimator for speech and music," *The Journal of the Acoustical Society of America*, vol. 124, no. 3, pp. 1638–1652, 2008.

[5] S. Rosenzweig, F. Scherbaum, and M. Müller, "Reliability assessment of singing voice F0-estimates using multiple algorithms," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Toronto, Canada, 2021, pp. 261–265.

[6] S. Rosenzweig, H. Cuesta, C. Weiß, F. Scherbaum, E. Gómez, and M. Müller, "Dagstuhl ChoirSet: A multitrack dataset for MIR research on choral singing," *Transactions of the International Society for Music Information Retrieval (TISMIR)*, vol. 3, no. 1, pp. 98–110, 2020.

[7] M. Müller, *Fundamentals of Music Processing – Using Python and Jupyter Notebooks*, 2nd ed. Springer Verlag, 2021.

[8] R. M. Bittner, M. Fuentes, D. Rubinstein, A. Jansson, K. Choi, and T. Kell, "mirdata: Software for reproducible usage of datasets," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, 2019, pp. 99–106. [Online]. Available: http://archives.ismir.net/ismir2019/paper/000009.pdf

[9] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. P. W. Ellis, "MIR_EVAL: A transparent implementation of common MIR metrics," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Taipei, Taiwan, 2014, pp. 367–372.