LTFATPY: TOWARDS MAKING A WIDE RANGE OF TIME-FREQUENCY REPRESENTATIONS AVAILABLE IN PYTHON

*Clara Hollomey and Nicki Holighaus,**

Acoustics Research Institute
Austrian Academy of Sciences
Vienna, AT
`clara.hollomey@oeaw.ac.at`

ABSTRACT

LTFATPY is a software package for accessing the Large Time Frequency Analysis Toolbox (LTFAT) from Python. Dedicated to time-frequency analysis, LTFAT comprises a large number of linear transforms for Fourier, Gabor, and wavelet analysis along with their associated operators. Its filter bank module is a collection of computational routines for finite impulse response and band-limited filters, allowing for the specification of constant-Q and auditory-inspired transforms. While LTFAT has originally been written in MATLAB/GNU Octave, the recent popularity of the Python programming language in related fields, such as signal processing and machine learning, makes it desirable to have LTFAT available in Python as well. We introduce LTFATPY, describe its main features, and outline further developments.

1. INTRODUCTION

Python [1] is one of the most popular programming languages [2] and is commonly used in audio-related machine learning [3] and signal processing [4]. Popular extension packages like `scipy` [5] and `librosa` [6] allow for filter design and audio analysis. Some further packages exist, that are customarily used for the analysis and processing of audio in scientific applications, e.g. [7, 8, 9]. In addition to that, there are toolboxes originally written in other programming languages, such as the *Sound Field Synthesis Toolbox* [10] and *Essentia* [11], that are now accessible from Python.

The Large Time Frequency Analysis Toolbox (LTFAT) [12, 13] belongs to this latter class of toolboxes. Originally written in MATLAB [14] and GNU Octave [15], with a supplementary backend in C and C++, LTFAT was originally conceived as a common ground for research and education on applied harmonic analysis, with a focus on the discrete Gabor transform [16]. However, during the last 20 years, it has been extended and developed into a wide-ranging collection of flexible, invertible time-frequency representations and associated algorithms, in many cases reflecting the state-of-the-art in applied harmonic analysis and signal processing research.

Time-frequency representations often form the basis for training neural networks (e.g. [17, 18, 19]), and for extracting trainable features, e.g. mel-frequency cepstral coefficients (MFCC) (e.g [20]). Implementations of the short-time Fourier transform (STFT), the discrete wavelet transform [21], and of many other time-frequency representations exist in Python. However, they usually impose unnecessary restrictions on signal analysis, such as the reliance on fixed time-frequency sampling schemes and on orthogonal decompositions. While the former can lead to overly redundant time-frequency representations and a consequential increase in computational resources required to perform the analysis, the latter can, e.g. in the case of many wavelet transforms algorithms [22], restrict the achievable frequency resolution to a range that is impractical for audio analysis. Finally, implementations of time-frequency representations may differ in their precise implementation. These differences can still affect the comparability of the results.

Rooted in frame theory [23], LTFAT allows for the nearly arbitrary trade-off between time and frequency resolution and thus provides a flexibility in the design of invertible time-frequency representations that is hard to achieve otherwise. With its wide range of unified window and wavelet functions and principled treatment of boundary conditions, it is useful whenever signal analysis beyond the STFT and orthogonal wavelet transforms is desired.

In the following, we describe the design considerations for interfacing LTFAT with Python and outline some of its key features with regards to the analysis and the processing of audio signals. We conclude with an outlook on planned further developments of the LTFATPY package.

2. DESIGN CONSIDERATIONS

A previous version of LTFATPY [24] made use of *cython* [25] bindings to LTFATs backend to call it from Python. However, much of LTFAT's core functionality, such as the constant-Q transform [26], the auditory-inspired filter banks [27], and most of the tests and demos are part of LTFAT's MATLAB code, and were therefore not available in Python.

To facilitate access to those routines while avoiding excessive code duplication, and given that LTFAT code is fully compatible with both, MATLAB and GNU Octave, LTFATPY is based on the `oct2py` package [28], which was streamlined, adapted and extended to accommodate for LTFAT. Consequently, the space and time performance is the same as for LTFAT, except for the overhead incurred by managing Octave from Python. Importing LTFATPY from Python starts an Octave session in the background that can be managed (i.e., stopped and restarted) by the user. Whenever an LTFATPY function is called, the input arguments are converted and shared with Octave in the background via `.mat` files, which are similarly used to pass the computation results to Python. `.mat` files are handled via the `scipy` package and Octave arrays are translated to `numpy` arrays [29]. The resulting increase in computation time, as compared to Python-only routines, varies and is highly system- and setup-dependent. Even occasional acceleration can be observed in practice, in cases where Octave routines are considerably faster than their Python equivalents.

Although this approach reduces the need for writing additional code, LTFAT's sheer size of roughly 1500 functions renders its instantaneous, full conversion to a Python package infeasible in a scientifically oriented development environment. To nevertheless ensure a coherent and convergent development process that at the same time allows for the flexible addition of functionality as needed, LTFATPY is laid out in a modular fashion and such that both, single functions and whole modules can be added with little Python programming experience required, thus facilitating and encouraging its ongoing enhancement.

Additional consideration was given to maintaining a similar syntax across programming languages, as to avoid confusion, keep the switching overhead for existing LTFAT users low, and to be able to have a central documentation for the Matlab and the Python code base. A list of the differences between LTFAT and LTFATPY can be found in [30].

3. FEATURES

In the following, we detail some of LTFATPY's key features. An overview of time-frequency representations available in common Python packages is depicted in Table 1.

Table 1: *Overview of common time-frequency representations and their availability in common Python packages (Y/N...yes/no, CWT...continuous wavelet transform).*

functionality	`ltfatpy`	`librosa`	`pyfar [31]`
STFT	Y	Y	N
constant-Q	Y	Y	N
gammatone	Y	N	Y
mel	Y	MFCC	N
bark	Y	N	N
CWT	Y	N	N
reassignment	all	STFT	N

3.1. Discrete Gabor transform/STFT

In LTFAT, the STFT is commonly referred to as the discrete Gabor transform (DGT), where the time-frequency coefficients $c[m, n]$ of a signal f with length L can be obtained as

$$c[m, n] = \sum_{l=0}^{L-1} f[l]\overline{g[l-an]}e^{\frac{-2\pi iml}{M}}, \tag{1}$$

with g the window, a the hopsize, or downsampling factor in time, and M the number of frequency channels. Thus, just like for most implementations of the STFT, the time-frequency resolution of the resulting coefficients can be controlled by the ratio between a and M. A minor difference between LTFAT's `dgt` function and most other implementations is that there are no restrictions on a and M other than to be positive-valued integers, yielding more freedom in adjusting the redundancy of the time-frequency coefficients, as exemplarily depicted in Figure 1. More importantly, LTFAT offers options beyond deriving the Moore-Penrose pseudo-inverse of the STFT-matrix for arriving at its inverse. Specifically, the used windows can be designed to be tight, to have a specific length [32], or other desirable properties with regards to the overall transform-inversion system [33], e.g., decreasing both the condition number and the computational complexity as compared to the standard approach, while still ensuring its invertibility.

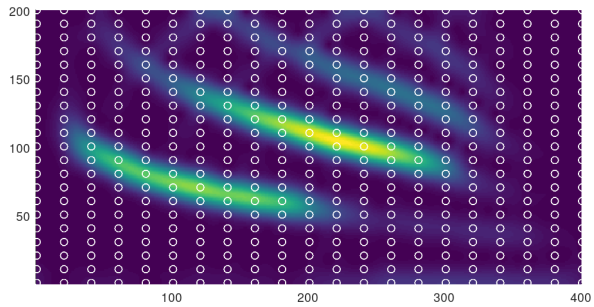

Figure 1: *The discrete Gabor transform of a bat cry, sampled with hopsize $a = 1$ and with the number of frequency channels corresponding to the length of the signal. The white circles indicate the coefficients that would be obtained for $a = 20$ and the number of frequency channels $M = 40$. Both configurations are stably invertible.*

3.2. Constant-Q transform and overcomplete wavelet filter banks

The toolbox provides two different methods for constructing overcomplete, invertible filter banks with a constant center frequency-to-bandwith ratio. The *invertible constant-Q transform* presented in [26], as well as a discretization of the continuous wavelet transform [34], as depicted in Figure 2. The methods are implemented as filter bank generators `cqtfilters` and `waveletfilters`, to be used with the `filterbank` module of LTFAT. Both filter bank types offer a high degree of flexiblity for customization, with regards to the filter prototypes, the time-frequency resolution trade-off, the spacing of center frequencies, and different uniform and non-uniform downsampling schemes.

3.3. Beyond linear and logarithmic frequency spacing

Equipped with similar configuration options as the constant-Q transforms described above, LTFAT provides the means for constructing auditory-inspired representations via its `audfilters` functionality [27], thus providing an invertible alternative to the commonly used MFCC. The featured auditory scales comprise the mel [35], bark [36], and equivalent rectangular bandwidth (ERB) [37] scale. Finally, the specification of nearly arbitrary filter banks via a warping function and its inverse, via the `warpedfilters` function [38], is supported.

3.4. Beyond linear time-frequency processing

Besides analysis and synthesis with linear time-frequency filter banks, LTFAT offers a number of nonlinear processing methods built onto the filter banks discussed in the previous sections. Here, we would like to highlight *reassignment* and *synchrosqueezing*, two methods for computing sparse time-frequency representations [39]. Synchrosqueezing is illustrated in Figure 3.

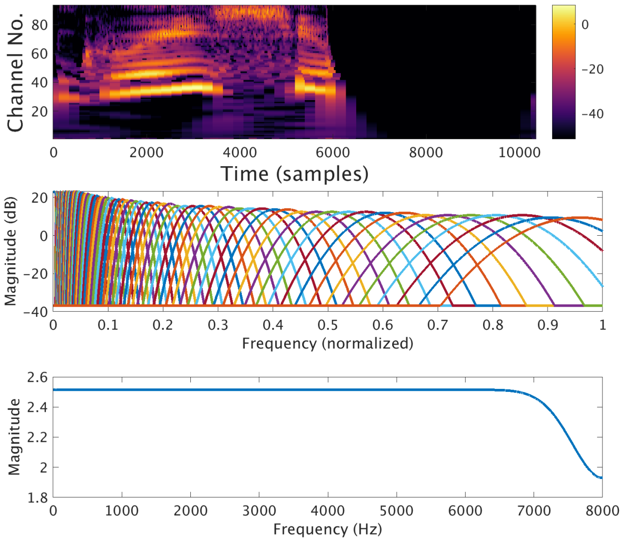

Figure 2: *Constant-Q wavelet filter bank (covering the frequency range from 40 Hz to 8 kHz with 12 bins per octave): coefficients (top) analyzing female speech (non-uniformly downsampled), the frequency responses of the individual wavelet filters (middle), and the frequency response of the overall filter bank (bottom). The filter bank was scaled for equal-energy per filter, i.e. such that each filter has the same* l_2*-norm.*

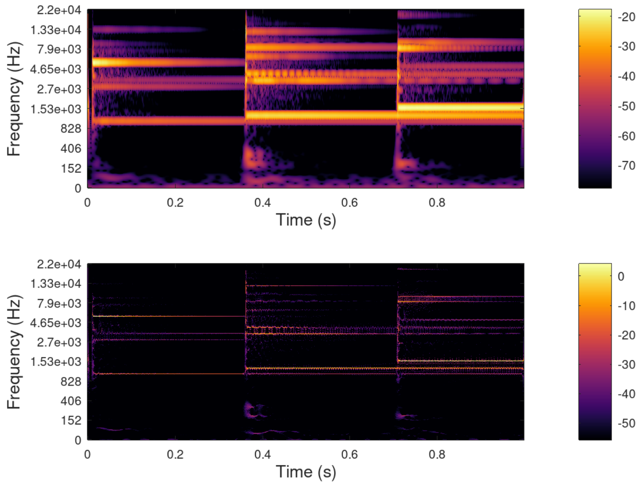

Figure 3: *An auditory filterbank using the ERB scale, representing a glockenspiel signal (top) and its synchrosqueezed version (bottom).*

4. OUTLOOK

LTFATPY is available from `github.com/ltfat/ltfatpy`, the original LTFAT written in GNU Octave can be downloaded from `github.com/ltfat/ltfat/releases`. The documentation can be found, along with the associated scientific publications, on `ltfat.org`.

LTFATPY currently covers most of LTFAT's `Gabor` and `Filterbank` module, more functionality may be added in the future. A current list of functions can be found on `https://github.com/ltfat/ltfatpy/blob/main/README.md`. Besides the future release as a Python package [40], a major planned enhancement is the addition of bindings from Python to LTFAT's C and C++ backend to enable the addition of the phase gradient heap [41] and the matching pursuit algorithm [42].

5. REFERENCES

[1] Johannes Ernesti and Peter Kaiser, "Python 3," *Rheinwerk, Bonn*, 2017.

[2] TIOBE The software quality company, "TIOBE Index for May 2024," Available at https://www.tiobe.com/tiobe-index/, accessed May 21, 2024.

[3] Eric Guizzo et al., "L3DAS21 challenge: Machine learning for 3D audio signal processing," in *2021 IEEE 31st International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2021, pp. 1–6.

[4] John C Glover, Victor Lazzarini, and Joseph Timoney, "Python for audio signal processing," 2011.

[5] Pauli Virtanen et al., "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[6] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto, "librosa: Audio and music signal analysis in python.," in *SciPy*, 2015, pp. 18–24.

[7] Theodoros Giannakopoulos, "pyaudioanalysis: An open-source python library for audio signal analysis," *PloS one*, vol. 10, no. 12, pp. e0144610, 2015.

[8] Sebastian Böck, Filip Korzeniowski, Jan Schlüter, Florian Krebs, and Gerhard Widmer, "Madmom: A new python audio and music signal processing library," in *Proceedings of the 24th ACM international conference on Multimedia*, 2016, pp. 1174–1178.

[9] Robin Scheibler, Eric Bezzam, and Ivan Dokmanić, "Pyroomacoustics: A python package for audio room simulation and array processing algorithms," in *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2018, pp. 351–355.

[10] H Wierstorf, F Winter, F Schultz, N Hahn, T Rettberg, C Hohnerlein, and S Spors, "Theory of sound field synthesis," doi:10.5281/zenodo.2589179, 2018.

[11] Dmitry Bogdanov et al., "Essentia: An audio analysis library for music information retrieval," in *Britto A, Gouyon F, Dixon S, editors. 14th Conference of the International Society for Music Information Retrieval (ISMIR); 2013 Nov 4-8; Curitiba, Brazil. p. 493-8.* International Society for Music Information Retrieval (ISMIR), 2013.

[12] Zdeněk Průša, Peter Søndergaard, Peter Balazs, and Nicki Holighaus, "LTFAT: A Matlab/Octave toolbox for sound processing," in *Proceedings of the 10th International Symposium on Computer Music Multidisciplinary Research*

(CMMR 2013), Marseille, France, October 2013, Laboratoire de Mecanique et d'Acoustique, pp. 299–314, Publications of L.M.A.

[13] Z. Průša, P. L. Søndergaard, N. Holighaus, Ch. Wiesmeyr, and P. Balazs, "The Large Time-Frequency Analysis Toolbox 2.0," in *Sound, Music, and Motion*, LNCS, pp. 419–442. Springer International Publishing, 2014.

[14] Starting Matlab, "Matlab," *The MathWorks, Natick, MA*, vol. 9, 2012.

[15] John Wesley Eaton, David Bateman, Søren Hauberg, et al., *Gnu octave*, Network thoery London, 1997.

[16] Peter Lempel Søndergaard, *Finite discrete Gabor analysis*, Ph.D. thesis, Technical University of Denmark, 2007.

[17] Zhiheng Ouyang, Hongjiang Yu, Wei-Ping Zhu, and Benoit Champagne, "A fully convolutional neural network for complex spectrogram processing in speech enhancement," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 5756–5760.

[18] Abdul Malik Badshah, Jamil Ahmad, Nasir Rahim, and Sung Wook Baik, "Speech emotion recognition from spectrograms with deep convolutional neural network," in *2017 international conference on platform technology and service (PlatCon)*. IEEE, 2017, pp. 1–5.

[19] Wentao Zhu and Mohamed Omar, "Multiscale audio spectrogram transformer for efficient audio classification," in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023, pp. 1–5.

[20] Vipin Bansal, Gaurav Pahwa, and Nirmal Kannan, "Cough classification for covid-19 based on audio mfcc features using convolutional neural networks," in *2020 IEEE international conference on computing, power and communication technologies (GUCON)*. IEEE, 2020, pp. 604–608.

[21] Gregory Lee, Ralf Gommers, Filip Waselewski, Kai Wohlfahrt, and Aaron O'Leary, "Pywavelets: A python package for wavelet analysis," *Journal of Open Source Software*, vol. 4, no. 36, pp. 1237, 2019.

[22] Limin Yu, Fei Ma, and Langford B White, "Fast wavelet transform algorithms for sonar signal analysis," *International J. of Sig. Proc. Systems*, vol. 1, no. 2, 2013.

[23] Richard J Duffin and Albert C Schaeffer, "A class of nonharmonic Fourier series," *Transactions of the American Mathematical Society*, vol. 72, no. 2, pp. 341–366, 1952.

[24] Denis Arrivault and Florent Jaillet, "ltfatpy," Available at https://hal.science/hal-04147603/, 2018.

[25] Stefan Behnel et al., "Cython: The best of both worlds," *Computing in Science & Engineering*, vol. 13, no. 2, pp. 31–39, 2010.

[26] Gino Angelo Velasco, Nicki Holighaus, Monika Dörfler, and Thomas Grill, "Constructing an invertible constant-q transform with non-stationary Gabor frames," *Proceedings of DAFX11, Paris*, vol. 33, pp. 81, 2011.

[27] Thibaud Necciari, Peter Balazs, Nicki Holighaus, and Peter L. Søndergaard, "The ERBlet transform: An auditory-based time-frequency representation with perfect reconstruction," in *Proceedings of the 38th International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2013)*, Vancouver, Canada, May 2013, IEEE, pp. 498–502.

[28] Python software foundation, "oct2py 5.6.1," Available at https://pypi.org/project/oct2py/, accessed May 21, 2024.

[29] Stefan Van Der Walt, S Chris Colbert, and Gael Varoquaux, "The numpy array: a structure for efficient numerical computation," *Computing in science & engineering*, vol. 13, no. 2, pp. 22–30, 2011.

[30] The ltfatpy developers, "LTFATPY - Readme," Available at https://github.com/ltfat/ltfatpy/blob/main/README.md, accessed Jul 09, 2024.

[31] The pyfar developers, "pyfar - python packages for acoustics research," Available at https://pyfar-gallery.readthedocs.io/en/latest/, accessed Jul 09, 2024.

[32] Peter L. Søndergaard Nathanaël Perraudin, Nicki Holighaus and Peter Balazs, "Gabor dual windows using convex optimization," in *Proceedings of SampTA*, 2013.

[33] Tobias Werther, Yonina C Eldar, and Nagesh K Subbanna, "Dual Gabor frames: theory and computational aspects," *IEEE Transactions on Signal Processing*, vol. 53, no. 11, pp. 4147–4158, 2005.

[34] Ingrid Daubechies, *Ten lectures on wavelets*, SIAM, 1992.

[35] Stanley Smith Stevens, John Volkmann, and Edwin Broomell Newman, "A scale for the measurement of the psychological magnitude pitch," *The Journal of the Acoustical Society of America*, vol. 8, no. 3, pp. 185–190, 1937.

[36] Eberhard Zwicker, "Subdivision of the audible frequency range into critical bands (frequenzgruppen)," *The Journal of the Acoustical Society of America*, vol. 33, no. 2, pp. 248–248, 1961.

[37] Brian CJ Moore and Brian R Glasberg, "Suggested formulae for calculating auditory-filter bandwidths and excitation patterns," *The journal of the acoustical society of America*, vol. 74, no. 3, pp. 750–753, 1983.

[38] Nicki Holighaus, Christoph Wiesmeyr, and Zdenek Prusa, "A class of warped filter bank frames tailored to non-linear frequency scales," *Journal of Fourier Analysis and Applications*, vol. 26, no. 1, pp. 22, 2020.

[39] Nicki Holighaus, Zdenek Průša, and Peter L. Søndergaard, "Reassignment and synchrosqueezing for general time–frequency filter banks, subsampling and processing," *Signal Processing*, vol. 125, pp. 1–8, 2016.

[40] Python software foundation, "The Python Package Index (PyPI)," Available at https://pypi.org/, accessed May 21, 2024.

[41] Zdeněk Průša, Peter Balazs, and Peter L. Søndergaard, "A Noniterative Method for Reconstruction of Phase from STFT Magnitude," *IEEE/ACM Trans. on Audio, Speech, and Lang. Process.*, vol. 25, no. 5, May 2017.

[42] Z. Průša, N. Holighaus, and P. Balazs, "Accelerating matching pursuit for multiple time-frequency dictionaries," *to appear in Proceedings of the International Conference on Digital Audio Effects 2020 (DAFx20)*, vol. preprint available: http://ltfat.github.io/notes/056/, 2020.