UNTWIST: A NEW TOOLBOX FOR AUDIO SOURCE SEPARATION

Gerard Roma, Emad M. Grais, Andrew J.R. Simpson, Iwona Sobieraj, Mark D. Plumbley
Centre for Vision, Speech an Signal Processing, University of Surrey , UK
{g.roma,grais,andrew.simpson,i.sobieraj,m.plumbley}@surrey.ac.uk

ABSTRACT

Untwist is a new open source toolbox for audio source separation. The library provides a self-contained object-oriented framework including common source separation algorithms as well as input/output functions, data management utilities and time-frequency transforms. Everything is implemented in Python, facilitating research, experimentation and prototyping across platforms. The code is available on github [1] .

1. INTRODUCTION

The availability of software for audio source separation is not on par with related fields, such as Music Information Retrieval (MIR). In order to test different approaches, a researcher must often retrieve scripts from different sources, when available. Most of the available code is implemented in Matlab. However, following many other disciplines, research on audio source separation could greatly benefit from the tools available in scientific Python. In this paper we introduce a new library focusing on audio source separation. The design is inspired in object-oriented data-flow systems, but using pure Python to facilitate research and experimentation.

2. RELATED WORK

After many years of MIR and general audio analysis research, a number of libraries for audio analysis are available [2–4, 11, 14, 15, 18, 21]. A common approach has been to implement the basic building blocks in C or C++, and provide bindings for high level languages. With the development of `NumPy`, `Cython` and specialized libraries like `Theano`, this approach does not seem necessary in many cases. Pure Python implementations are especially convenient for research, since the code for any algorithm can be consulted and modified without changing the programming language and environment. Some recent libraries like `libROSA` [15] or `MadMom` [2] are fully implemented in Python. Both include some source separation algorithms, but focus more broadly on MIR.

Libraries specifically focusing on audio source separation are scarce. Perhaps the most well-known is the *Flexible Audio Source Separation Toolkit* (`FASST`) [19]. `FASST` is based on a general mathematical formulation, which focuses on the *local gaussian model* [17]. The different options are specified via configuration. In this sense, `FASST` is modular in a mathematical sense, but from the point of view of software engineering, it is not designed to facilitate building new algorithms outside this framework. Our library focuses specifically on audio source separation, using modular, object-oriented scripting.

3. DESIGN CONCEPTS

The general design is based on Object-Oriented Programming (OOP), but leveraging the weak encapsulation philosophy in Python. All code and data are available to the researcher. The library contains mainly data objects, processing objects and models.

Our approach for data representation consists in subclassing the `ndarray` class in `NumPy`, and extend it with convenience methods for I/O. All data objects can still be indexed and operated as standard arrays.

Processor ojects are configured in the constructor, and then process incoming data according to the configured parameters. Processor objects must be serializable, and the `process` method shall not modify the instance state, so that they can be used in parallel. Models represent algorithms that may take time to train, and so a persistence mechanism is implemented for their parameters. A special consideration is needed for channel layouts since it is a central topic in audio source separation. We adapt the notion of "function rank" used in array programming [10] in order to automatically parallelize `process` methods via Python annotations. For the moment, this allows computing time-frequency transforms of multi-channel waveforms. A set of common conventions are used with respect to data layout: waveform channels are column vectors, spectrograms are 2D arrays where columns are spectral frames. Models expect data to have one observation per row, so spectrograms and spectral masks are transposed when with models.

[1] `https://github.com/IoSR-Surrey/untwist`

[2] `http://madmom.readthedocs.org/en/latest/`

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3  from untwist.data import Wave, RatioMask
4  from untwist.transforms import STFT, ISTFT
5  from untwist.factorizations import RPCA
6
7  stft = STFT()
8  istft = ISTFT()
9  rpca = RPCA(iterations = 100)
10
11 x = Wave.read("mix.wav")
12 X = stft.process(x)
13
14 # this may take some time
15 (L,S) = rpca.process(X.magnitude())
16
17 M = RatioMask(np.abs(S), np.abs(L))
18 v = istft.process(X * M)
19 v.write("vocal_estimate.wav")
20
21 # (...) calls to plotting method in X, L, S, M
```

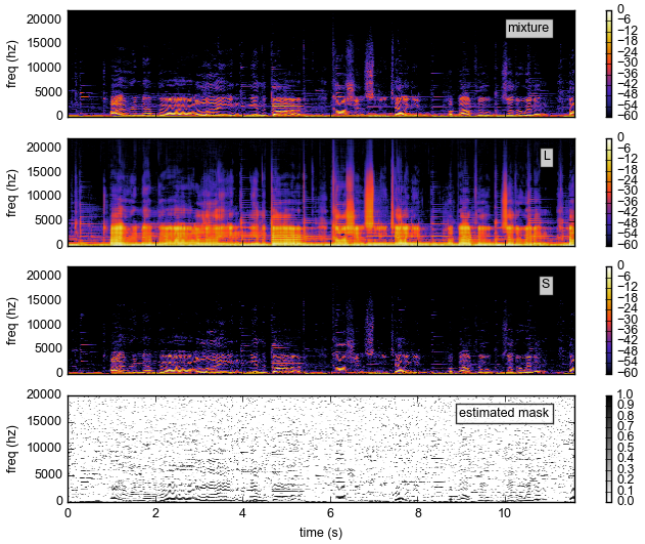

Figure 1. Code and resulting output for vocals separation using RobustPCA

4. FUNCTIONALITY

This section describes the different modules currently included in `untwist`.

4.1 I/O

Input and output functionality is implemented in data objects. Audio buffers can be read and written from/to disk. Plotting functions using `matplotlib` [9] are implemented in most objects. Audio playback is possible using he Python bindings for portaudio [1] available in `pyaudio`. A Dataset class provides basic functionality for building, indexing, loading, saving, shingling, shuffling and normalizing datasets. A specialized subclass is available for using memory-mapped files [3] , beyond available RAM.

4.2 Time-frequency transforms

Most audio source separation algorithms work on time-frequency representations, mainly the Short-Time Fourier Transform (STFT). In addition to STFT, the Quadratic ERB transform [22] is implemented, since it has been used in several separation and transcription experiments [5, 23].

4.3 Analysis

While analysis is not the goal of the library, some basic audio features can be useful for separation. For the moment a few common onset and pitch detection algorithms are available.

4.4 Factorizations

Non-negative Matrix Factorization (NMF) [12] is probably the most widely used family of algorithms for audio source separation. Our implementation is inspired by `nmflib` [4] . Many variants are accommodated under a unified interface.

[3] `http://docs.scipy.org/doc/numpy-1.10.0/reference/generated/numpy.memmap.html`

[4] `http://www.ee.columbia.edu/~grindlay/code.html`

Robust Principal Component Analysis (RPCA) is another decomposition that has been recently used for singing voice separation [7]. Our implementation is based on the Augmented Lagrange Multiplier (ALM) method [13].

4.5 Neural networks

Deep Neural Networks (DNN) are increasingly used for audio source separation [8, 16]. These algorithms allow leveraging existing data in a supervised setting. Our library includes a generic Multi-Layer Perceptron (MLP) implementation, allowing the creation of feed-forward networks with a simple specification for multiple architectures and activation functions. The implementation is based on `Theano` [20]. Training is perfrormed by a Stochastic Gradient Descent (SGD) wrapper with parameters for momentum, early-stopping and learning rate scheduling.

4.6 HPSS

Harmonic-Percussive Source Separation (HPSS) can be seen as a general application of source separation, or as a preprocessing stage. We implemented the simple and popular HPSS method by Fitzgerald [6] based on median filters.

5. EXAMPLE

The library is primarily targeted at audio source separation research. In this context, the user is expected to start writing short scripts and trying things in an interactive console. Consolidated algorithms can then be implemented in classes. Our aim is to support brevity and readability for research code. Figure 1 shows a very short example of separation using RPCA. All variables defined in lines $11 - 18$ correspond to data objects that can be inspected, plotted and played (in the case of waveforms) in an interactive console.

6. ACKNOWLEDGEMENTS

This work is supported by grants EP/L027119/1 and EP/L027119/2 from the UK Engineering and Physical Sciences Research Council (EPSRC). This work has also been partly funded from the European Union's H2020 Framework Programme (H2020-MSCA-ITN-2014) under grant agreement no. 642685 MacSeNet.

7. REFERENCES

[1] Ross Bencina and Phil Burk. Portaudio–an open source cross platform audio api. In *Proceedings of the 2001 International Computer Music Conf erence (ICMC-01)*, 2001.

[2] Dmitry Bogdanov, Nicolas Wack, Emilia Gómez, Sankalp Gulati, Perfecto Herrera, Oscar Mayor, Gerard Roma, Justin Salamon, José R Zapata, and Xavier Serra. Essentia: An audio analysis library for music information retrieval. In *Proceedings of the International Symposium on Music Information Retrieval (ISMIR 2013)*, pages 493–498, 2013.

[3] Paul M Brossier. The aubio library at mirex 2006. *MIREX 2006*, page 1, 2006.

[4] J Bullock and U Conservatoire. Libxtract: A lightweight library for audio feature extraction. *Proceedings of the International Computer Music Conference*, 43, 2007.

[5] Zhiyao Duan, Yungang Zhang, Changshui Zhang, and Zhenwei Shi. Unsupervised single-channel music source separation by average harmonic structure modeling. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(4):766–778, 2008.

[6] Derry Fitzgerald. Harmonic/percussive separation using median filtering. In *13th International Conference on Digital Audio Effects (DAFX10)*, Graz, Austria, 2010.

[7] Po-Sen Huang, Scott Deeann Chen, Paris Smaragdis, and Mark Hasegawa-Johnson. Singing-voice separation from monaural recordings using robust principal component analysis. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 57–60. IEEE, 2012.

[8] Po-Sen Huang, Minje Kim, Mark Hasegawa-Johnson, and Paris Smaragdis. Deep learning for monaural speech separation. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1562–1566. IEEE, 2014.

[9] John D Hunter et al. Matplotlib: A 2d graphics environment. *Computing in science and engineering*, 9(3):90–95, 2007.

[10] Kenneth E Iverson. Operators and functions. Technical report, IBM Thomas J. Watson Research Center, 1978.

[11] O Lartillot, P Toiviainen, and T Eerola. A matlab toolbox for music information retrieval. In *Data analysis, machine learning and applications*, pages 261–268. Springer, Berlin, Heidelberg, 2008.

[12] Daniel D Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*, pages 556–562, 2001.

[13] Zhouchen Lin, Minming Chen, and Yi Ma. The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. *arXiv preprint arXiv:1009.5055*, 2010.

[14] B Mathieu, S Essid, T Fillon, J Prado, and G Richard. YAAFE, an Easy to Use and Efficient Audio Feature Extraction Software. *Proceedings of the International Symposium on Music Information Retrieval (ISMIR)*, 2010.

[15] B McFee, C Raffel, and D Liang. librosa: Audio and music signal analysis in python. In *Proceedings of the 14th Python in Science Conference*, 2015.

[16] Arun Narayanan and DeLiang Wang. Ideal ratio mask estimation using deep neural networks for robust speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7092–7096. IEEE, 2013.

[17] Alexey Ozerov, Emmanuel Vincent, and Frédéric Bimbot. A general flexible framework for the handling of prior information in audio source separation. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(4):1118–1133, 2012.

[18] F Pedregosa and G Varoquaux. Scikit-learn: Machine learning in Python. *The Journal of Machine Learning Research*, 12:2825–2830, 2011.

[19] Yann Salaün, Emmanuel Vincent, Nancy Bertin, Nathan Souviraa-Labastie, Xabier Jaureguiberry, Dung T Tran, and Frédéric Bimbot. The flexible audio source separation toolbox version 2.0. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014.

[20] The Theano Development Team, Rami Al-Rfou, Guillaume Alain, Amjad Almahairi, Christof Angermueller, Dzmitry Bahdanau, Nicolas Ballas, Frédéric Bastien, Justin Bayer, Anatoly Belikov, et al. Theano: A python framework for fast computation of mathematical expressions. *arXiv preprint arXiv:1605.02688*, 2016.

[21] G Tzanetakis and P Cook. Marsyas: A framework for audio analysis. *Organised sound*, 2000.

[22] Emmanuel Vincent. Musical source separation using time-frequency source priors. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(1):91–98, 2006.

[23] Emmanuel Vincent, Nancy Bertin, and Roland Badeau. Adaptive harmonic spectral decomposition for multiple pitch estimation. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(3):528–537, 2010.