MIR IN MATLAB (II): A TOOLBOX FOR MUSICAL FEATURE EXTRACTION FROM AUDIO

Olivier Lartillot, Petri Toiviainen
University of Jyväskylä
PL 35(M), 40014, Finland

ABSTRACT

We present the *MIRtoolbox*, an integrated set of functions written in Matlab, dedicated to the extraction of musical features from audio files. The design is based on a modular framework: the different algorithms are decomposed into stages, formalized using a minimal set of elementary mechanisms, and integrating different variants proposed by alternative approaches – including new strategies we have developed –, that users can select and parametrize.

This paper offers an overview of the set of features, related, among others, to timbre, tonality, rhythm or form, that can be extracted with the *MIRtoolbox*. One particular analysis is provided as an example. The toolbox also includes functions for statistical analysis, segmentation and clustering. Particular attention has been paid to the design of a syntax that offers both simplicity of use and transparent adaptiveness to a multiplicity of possible input types. Each feature extraction method can accept as argument an audio file, or any preliminary result from intermediary stages of the chain of operations. Also the same syntax can be used for analyses of single audio files, batches of files, series of audio segments, multi-channel signals, etc. For that purpose, the data and methods of the toolbox are organised in an object-oriented architecture.

1 MOTIVATION AND APPROACH

MIRtoolbox is a *Matlab* toolbox dedicated to the extraction of musically-related features from audio recordings. It has been designed in particular with the objective of enabling the computation of a large range of features from databases of audio files, that can be subjected to statistical analyses.

Few softwares have been proposed in this area. One particularity of our own approach relies in the use of the *Matlab* computing environment, which offers good visualisation capabilities and gives access to a large variety of other toolboxes. In particular, the *MIRtoolbox* makes use of functions available in public-domain toolboxes such as the *Auditory Toolbox* [6], *NetLab* [5] and *SOMtoolbox* [10]. Other toolboxes, such as the *Statistics toolbox* or the *Neural Network toolbox* from MathWorks, can be directly used for further analyses of the features extracted by *MIRtoolbox* without having to export the data from one software to another.

Such computational framework, because of its general objectives, could be useful to the research community in Music Information Retrieval (MIR), but also for educational purposes. For that reason, particular attention has been paid concerning the ease of use of the toolbox. In particular, complex analytic processes can be designed using a very simple syntax, whose expressive power comes from the use of an object-oriented paradigm.

The different musical features extracted from the audio files are highly interdependent: in particular, as can be seen in figure 1, some features are based on the same initial computations. In order to improve the computational efficiency, it is important to avoid redundant computations of these common components. Each of these intermediary components, and the final musical features, are therefore considered as *building blocks* that can been freely articulated one with each other. Besides, in keeping with the objective of optimal ease of use of the toolbox, each building block has been conceived in a way that it can adapt to the type of input data. For instance, the computation of the MFCCs can be based on the waveform of the initial audio signal, or on the intermediary representations such as spectrum, or mel-scale spectrum (see Fig. 1). Similarly, autocorrelation is computed for different range of delays depending on the type of input data (audio waveform, envelope, spectrum). This decomposition of all feature extraction algorithms into a common set of building blocks has the advantage of offering a synthetic overview of the different approaches studied in this domain of research.

2 FEATURE EXTRACTION

2.1 Feature overview

Figure 1 shows an overview of the main features implemented in the toolbox. All the different processes start from the audio signal (on the left) and form a chain of operations proceeding to right. Each musical feature is related to one of the musical dimensions traditionally defined in music theory. Boldface characters highlight features related to pitch and tonality. Bold italics indicate features related to rhythm. Simple italics highlight a large set of features that can be associated to timbre and dynamics. Among them, all the operators in grey italics can be

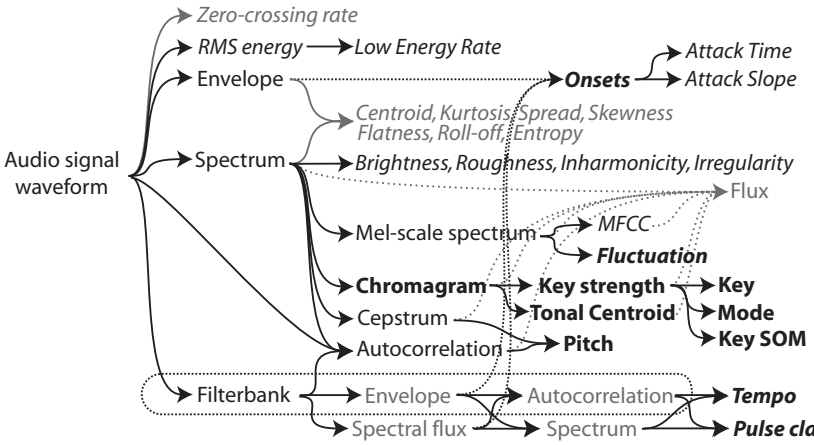

Figure 1. *Overview of the musical features that can be extracted with* MIRtoolbox.

in fact applied to many different representations. [1]

More elaborate tools have also been implemented that can carry out higher-level analyses and transformations. In particular, audio files can be automatically segmented into a series of homogeneous sections, through the estimation of temporal discontinuities along diverse alternative features such as timbre in particular [2, 3].

2.2 Example: Rhythm analysis

The estimation of rhythmic pulsation, for instance, can be based on the modelling of auditory perception of sound and music [8] (circled in Figure 1), as described in figure 2. The audio signal is first decomposed into auditory channels using a bank of filters. Diverse types of filterbanks have been implemented, and the number of channels can be specified, as can be seen in the syntax shown below at code line (1). The envelope of each channel is then extracted (2). As pulsation is generally related to increase of energy only, the envelopes are differentiated, half-wave rectified, before being finally summed together again (3). This gives a precise description of the variation of energy produced by each note event from the different auditory channels.

After this onset detection, the periodicity is estimated through autocorrelation (5) [2] . However, if the tempo varies throughout the piece, an autocorrelation of the whole sequence will not show clear periodicities. In such cases it is better to compute the autocorrelation for a frame decomposition (4) [3] . This yields a periodogram that highlights the different periodicities, as shown in figure 2. In order to focus on the periodicities that are more perceptible, the periodogram is filtered using a resonance curve [7] (5), after which the best tempos are estimated through peak picking (6), and the results are converted into beats per minute (7).

[1] For more details about the feature extraction algorithms, see [3].

[2] For the sake of clarity, several options in the following functions have been omitted.

[3] This shows an example of MIRtoolbox function that can adapt to the type of input (audio waveform, envelope, etc.). Here, the frame size is 3 seconds and the hop factor .1.

Due to the difficulty of choosing among the possible multiples of the tempo, several candidates (three for instance) may be selected for each frame, and a histogram of all the candidates for all the frames, called periodicity histogram, can be drawn (8).

```
fb = mirfilterbank(a,20)                    (1)
e = mirenvelope(fb,
'Diff','Halfwave','Center')                 (2)
s = mirsum(e,'Center')                      (3)
fr = mirframe(s,3,.1)                       (4)
ac = mirautocor(fr,'Resonance')             (5)
p = mirpeaks(ac,'Total',1)                  (6)
t = mirtempo(p)                             (7)
h = mirhisto(t)                             (8)
```

The whole process can be executed in one single line by calling directly the mirtempo function with the audio input as argument:

```
mirtempo(a,'Frame')                         (9)
```

2.3 Data analysis

The toolbox includes diverse tools for data analysis, such as a peak extractor, and functions that compute histograms, entropy, zero-crossing rates, irregularity or various statistical descriptors (centroid, spread, skewness, kurtosis, flatness) on data of various types, such as spectrum, envelope or histogram.

The `mirpeaks` functions can accept any data returned by any other function of the MIRtoolbox and can adapt to the different kinds of data of any number of dimensions. In the graphical representation of the results, the peaks are automatically located on the corresponding curves (for 1D data) or bit-map images (for 2D data). We have designed a new strategy of peak selection, based on a notion

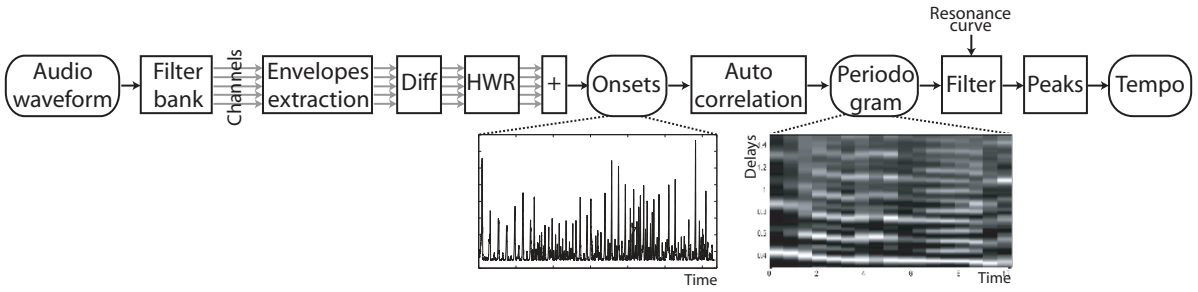

Figure 2. *Successive steps for the estimation of tempo illustrated with the analysis of an audio excerpt. In the periodogram, high autocorrelation values are represented by bright nuances.*

of *contrast*, discarding peaks that are not sufficiently contrastive (based on a certain threshold) with the neighbouring peaks. This adaptive filtering strategy hence adapts to the local particularities of the curves. Its articulation with other more conventional thresholding strategies leads to an efficient peak picking module that can be applied throughout the *MIRtoolbox*.

Supervised classification of musical samples can also be performed, using techniques such as K-Nearest Neighbours or Gaussian Mixture Models. One possible application is the classification of audio recordings into musical genres. The results of feature extraction processes can be stored as text files of various format, such as the ARFF format that can be exported in the Weka machine learning environment [11].

3 DESIGN OF THE TOOLBOX

3.1 Data encapsulation

All the data returned by the functions in the toolbox are encapsulated into typed objects. The default display method associated to all these objects is a graphical display of the corresponding curves. In this way, when the display of the values of a given analysis is requested, what is printed is not a listing of long vectors or matrices, but rather a correctly formatted graphical representation.

The actual data matrices associated to those data can be obtained by calling a method called `mirgetdata`, which constructs the simplest possible data structure associated to the data.

3.2 Adaptive syntax

As explained previously, the diverse functions of the toolbox can accept alternative input. The name of a particular audio file (either in wav or au format) can be directly specified as input:

```
mirspectrum('myfile')                    (10)
```

Alternatively, the audio file can be first loaded using the `miraudio` function, which can perform diverse operations such as resampling, automated trimming of the silence at the beginning and/or at the end of the sequence, extraction of a given subsequence, centering, normalization with respect to RMS energy, etc.

```
a = miraudio('myfile','Sampling',11025,
             'Trim','Extract',2,3,
             'Center','Normal')          (11)
mirspectrum(a)                           (12)
```

Batch analyses of audio files can be carried out by simply replacing the name of the audio file by the keyword `'Folder'`.

```
mirspectrum('Folder')                    (13)
```

Any feature extraction can be based on the result of a previous computation. For instance, the autocorrelation of a spectrum curve can be computed as follows:

```
s = mirspectrum(a)                       (14)
as = mirautocor(s)                       (15)
```

And product of curves can be performed easily:

```
mirautocor(a)*mirautocor(s)              (16)
```

In this particular example, the waveform autocorrelation `mirautocor(a)` is automatically converted to frequency domain in order to be combined with the spectrum autocorrelation `mirautocor(s)`.

3.3 Memory optimization

The flexibility of the syntax requires a complex data representation that can handle alternative configurations (frame and/or channels decompositions, segmentation, batch analysis) [3]. This data structure could in theory become very extensive in terms of memory usage, especially if entire folders of audio files are loaded into the memory in one go. To overcome this, we have designed new methods allowing a better management of memory without deterioration of the syntactical simplicity and power. Audio files are loaded one after the other in the memory, and if necessary, long audio files are also divided into a series of successive blocks of frames that are loaded one after the other. We plan to further optimise the computational efficiency of the toolbox by proposing the possibility of distributing the computational loads among a network of computers, with the help of the *Distributed Computing Toolbox* and *Engine* made available by Matlab.

3.4 Software Development Kit

The different feature extraction algorithms will be progressively refined and new features will be added in future versions of the *MIRtoolbox*. Users are encouraged to write their own functions, using the building blocks offered by the current version. A set of meta-functions have been designed that enable the writing of additional algorithms using very simple function templates. As the meta-functions take care of all the complex management of the data structure and methods, the development of new algorithms can concentrate simply on the purely mathematical and DSP considerations. This may result in a computational environment where large-scale MIR systems could be developed, articulated one with each other, and compared.

4 RELATED WORKS

Few softwares have been proposed in this area. Marsyas [9], is a framework for prototyping and experimentation with computer audition applications. The architecture is based on dataflow programming, where computation is expressed as a network of processing nodes and components. Users can build their own dataflow network using a scripting language at run-time. The advantage of an object-oriented paradigm, as used in the MIRtoolbox, is the possibility of a drastic reduction of the syntax complexity [4]. Rather than a ready-to-use set of applications, Marsyas is a framework for building applications, provided with a dozen of features restricted to low-level dimensions. Batch of files can be processed as long as the sampling rate remains constant, as Marsyas does not, contrary to the MIRtoolbox, perform automatic sampling conversion (except between 44100Hz and 22050Hz).

jAudio [4] is another framework for feature extraction written in Java. One particularity of this approach is the use of a GUI for the feature selection, and the design of a mechanism for automated dependency handling to prevent duplicate calculations.

MIRtoolbox includes most of the features (or variant of them) available in the aforementioned frameworks, plus a diverse collection of other lower and higher-level features related to musical dimensions such as timbre, key, rhythm, etc. The simplicity and power of its syntax enables easy combinations of the various operators and feature extractors. Another specific advantage of MIRtoolbox concerns its extended visualisation capabilities, benefiting from the richness of Matlab computing environment.

5 AVAILABILITY OF THE *MIRTOOLBOX*

Following our first Matlab toolbox, called *MIDItoolbox* [1], dedicated to the analysis of symbolic representations of music, the *MIRtoolbox* is offered for free to the research community. [5]

[4] For instance, the patching example proposed in Marsyas User Manual, which required 20 lines in Marsyas Scripting Language, can be implemented in four lines using the MIRtoolbox syntax.

6 ACKNOWLEDGMENTS

This work has been supported by the European Commission (NEST project "Tuning the Brain for Music", code 028570). The development of the toolbox has benefitted from productive collaborations with the other partners of the project, in particular Tuomas Eerola, Jose Fornari, Marco Fabiani, and students of our department.

7 REFERENCES

[1] Eerola, T. and P. Toiviainen. "MIR in Matlab: The Midi Toolbox", *Proceedings of 5th International Conference on Music Information Retrieval*, 22–27, 2004.

[2] Foote, J. and M. Cooper. "Media Segmentation using Self-Similarity Decomposition", *Proceedings of SPIE Storage and Retrieval for Multimedia Databases*, 5021, 167–175, 2003.

[3] Lartillot, O. and P. Toiviainen. "A Matlab Toolbox for Musical Feature Extraction from Audio", *Proceedings of the International Conference on Digital Audio Effects*, 2007.

[4] McEnnis, D., C. McKay, I. Fujinaga, and P. Depalle. 2005. "jAudio: A feature extraction library", *Proceedings of the International Symposium on Music Information Retrieval*, 600–3, 2005.

[5] Nabney, I. "NETLAB: Algorithms for pattern recognition", *Springer Advances In Pattern Recognition Series*, 2002.

[6] Slaney, M. *Auditory Toolbox Version 2*, Technical Report. Interval Research Corporation, 1998-010, 1998.

[7] Toiviainen, P. and J.S. Snyder. "Tapping to Bach: Resonance-based modeling of pulse", *Music Perception*, 21-1, 43–80, 2003.

[8] Tzanetakis, G. and P. Cook. "Multifeature Audio Segmentation for Browsing and Annotation", *Proceedings of the 1999 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 1999.

[9] Tzanetakis, G. and P. Cook. "MARSYAS: A Framework for Audio Analysis", *Organized Sound*, 4-3, 2000.

[10] Vesanto, J. "Self-Organizing Map in Matlab: the SOM Toolbox", *Proceedings of the Matlab DSP Conference*, 35–40, 1999.

[11] Witten, I. H. and E. Frank. *Data Mining: Practical machine learning tools and techniques*, 2nd Edition, Morgan Kaufmann, San Francisco, 2005.

[5] It can be downloaded from the following URL: `http://users.jyu.fi/~lartillo/mirtoolbox`