AN OPEN SOURCE TOOL FOR SEMI-AUTOMATIC RHYTHMIC ANNOTATION

Fabien Gouyon, Nicolas Wack

Universitat Pompeu Fabra
Music Technology Group
fgouyon,nwack@iua.upf.es

Simon Dixon

Austrian Research Institute for
Artificial Intelligence
simon@oefai.at

ABSTRACT

We present a plugin implementation for the multi-platform WaveSurfer sound editor. Added functionalities are the semi-automatic extraction of beats at diverse levels of the metrical hierarchy as well as uploading and downloading functionalities to a music metadata database. It is built upon existing open source (GPL-licenced) audio processing tools, namely WaveSurfer, BeatRoot and CLAM, in the intent to expand the scope of those softwares. It is therefore also provided as GPL code with the explicit goal that researchers in the audio processing community can freely use and improve it.

We provide technical details of the implementation as well as practical use cases. We also motivate the use of rhythmic metadata in Music Information Retrieval scenarios.

1. INTRODUCTION

Rhythm is a fundamental musical feature. Anyone perceives rhythm while enjoying music listening. One can represent rhythm explicitly (i.e. write it down) in many ways, with diverse degrees of detail [1] and by different means, manually or automatically. For instance, a trained listener can transcribe a musical piece into score notation while listening repeatedly to it. He can also assign a single value for the basic tempo (in BPM). The level of detail in the representation depends on the purpose of annotation. That is, different applications require different representations [1].

In any case, it is clear that the task of associating such metadata to musical pieces would be eased by the use of additional software tools. For instance, a simple sound editor plotting waveform and spectrogram would be highly informative to a potential user. Also, a system that would compute automatically the desired metadata would obviously be relevant. However, in this case, subsequent human corrections are a must. Further, as it is clear that no automatic rhythm description system is perfect, nor human annotations are error-free, interactive systems are highly desirable. In such systems, either the user or an algorithm does a first rough analysis of (part of) the data, then the other uses the results of this analysis to orient its own analysis; the process can be iterated several times.

Very few beat annotation systems exist. In [2], Goto refers to a "beat-position editor." This is a manual beat annotation tool that provides waveform visualisation and, for accurate annotations, audio feedback in the form of short bursts of noise added at beat times. To our knowledge, the only publically available (and open-source) beat annotation software is BeatRoot [3]. To lower the annotation effort, an automatic beat tracking algorithm is available. Interactivity resides in that the user's corrections to the algorithm output (the beat times) are fed back as inputs to the very algorithm.

In this paper, we report on a system built upon BeatRoot as well as other open source audio processing tools, namely WaveSurfer [4] and CLAM (both part of the AGNULA GPL distribution of Linux sound software). The intent is to "take the best of several worlds", that is, group useful functionalities of those different softwares in a single application as well as expand their scope and capabilities.

We focus on a particular kind of rhythmic annotations, the metrical structure, as it has been formalised by Lerdahl and Jackendoff in the Generative Theory of Tonal Music [5]. That is, the metadata we propose to associate to musical signals are particular time points: the beats, at several metrical levels.

Annotations can be stored locally and, when correct, they can easily be uploaded to a distant repository, e.g. a structured musical metadata database such as the MTG database [6], via the SOAP protocol.

2. APPLICATIONS

The knowledge of beats at different levels of the metrical hierarchy can be useful in many applications.

In Music Information Retrieval research, metadata associated to musical data are very useful. First of all because a database of "ground truth" metadata greatly facilitates the design of automatic algorithms for audio content description. In addition, some recent work in this field includes rhythmic information as input to systems that compute other types of metadata. For instance, beats at a metrical level can be used to determine other metrical levels [7], [8], [9]. They can also be useful as audio segment boundaries for instrument classification, such as percussion [10], [11], [12]. Other examples are the use of the metrical structure for long-term segmentations and rhythmic complexity computation. However, reliable determination of such information from automatic systems is itself a challenge. It is therefore clear that in this type of research, semi-automatic systems would be desirable.

Performers' choices in tempo and expressive timing with respect to position in the metrical structure are very relevant to Musical Performance research. Software tools that ease the annotation of the whole metrical structure, and that generate tempo or timing deviation curves are clearly useful in this field [13].

Finally, other applications are the synchronisation or the sequencing of several musical excerpts, the determination of "looping points" for cut-and-paste operations, the application of tempo-synchronous audio effects (or visual animations), music identification, rhythmic expressiveness transformations.

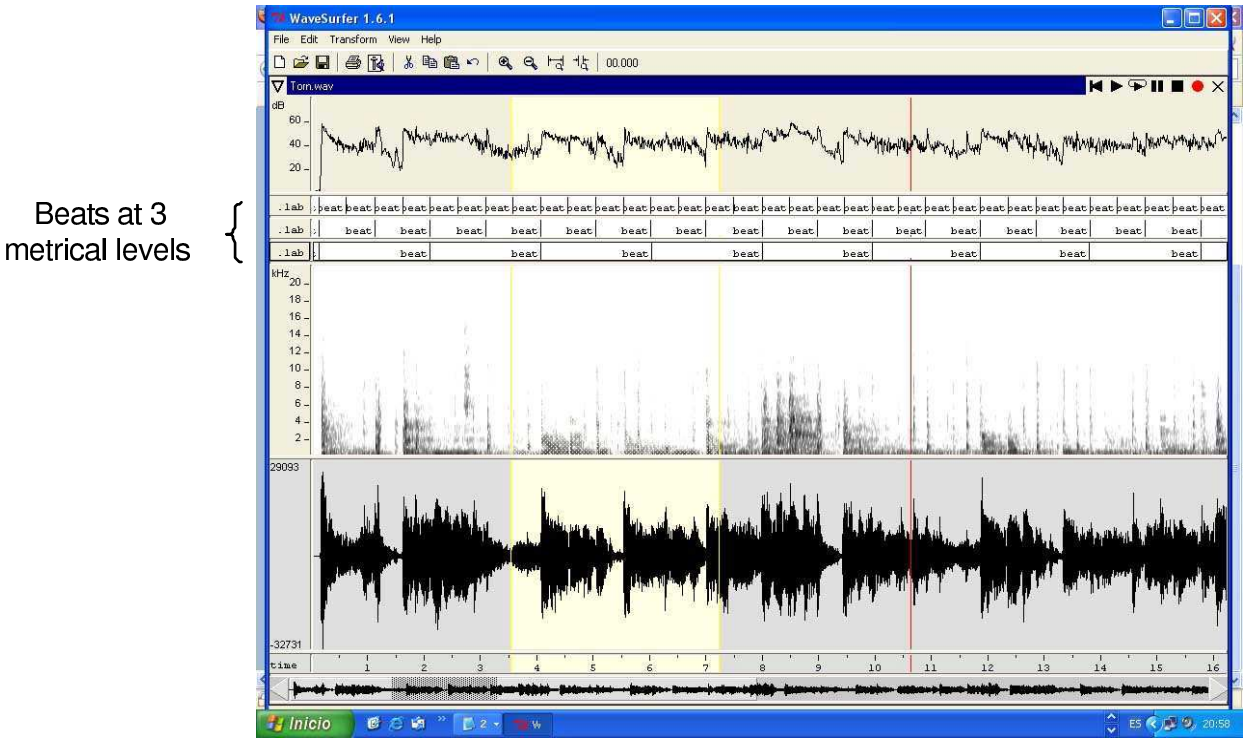

Figure 1: *Screenshot of metrical level annotations associated to common WaveSurfer functionalities (power plot, spectrogram, waveform, timeline, etc.). This annotation took less than one minute.*

3. ANNOTATING METRICAL LEVELS SEMI-AUTOMATICALLY

This section details in diverse use cases the diverse functionalities offered by the system. Some functionalities are available in WaveSurfer (common sound editing) and others have been added (beat tracking and database connections). Figure 1 gives an illustration of one possible configuration for the system and the result of the annotation of three metrical levels.

3.1. WaveSurfer functionalities

WaveSurfer [4] was initially developed as an open source software for speech research at the Department of Speech, Music and Hearing at the Royal Institute of Technology in Sweden.[1] We found diverse reasons to build a rhythmic annotation system on top of it.

First of all, WaveSurfer's typical applications are sound analysis and annotation/transcription. It therefore offers many useful functionalities such as visualisation of waveform, spectrogram, power plots, pitch contour, formant plots, etc. Panes (for transcription, data visualisation, etc.) can be dynamically added or removed, they are all time-aligned and display a running cursor while playing. Collection of panes can be saved as a configuration, that can be applied later to any other sound. This allows users to easily customize the interface. The complete sound waveform is displayed at the bottom while (optionally) a separate pane displays solely part of the waveform with additional zooming functionalities. This greatly simplifies working with large sound files. Data plots are also available, opening the way to easily visualise any relevant data, e.g. tempo curves, deviation curves. WaveSurfer has been designed in agreement with common user interface standards and it also provides intuitive keyboard shortcuts for playing, stopping, selecting regions, looping them etc. It is also possible to easily customise the look-and-feel to personal tastes.

Then, importantly, this is a multi-platform software and it handles many standard audio file formats. Last, but obviously not least, it is extensible through a simple plug-in architecture (note that it is widely used and diverse functionalities are regularly added by researchers in the audio processing community).

[1] http://www.speech.kth.se/WaveSurfer/

3.2. Diverse annotation modes

3.2.1. Manual annotations

The user can set manually the time indexes of every beat by simple left-clicks on the computer mouse. However this task is very time-consuming and error-prone. Therefore we added the possibility to specify beat times in real time while listening to the sound by simply tapping any of the keyboard's key.

Individual beats can be subsequently adjusted with the help of the computer mouse by selecting and finely shifting them.

3.2.2. Interactive annotations

The system presents several menus in addition to WaveSurfer's usual ones as found in transcription panes:

- 'Compute beats'
- 'Erase beats from cursor'

- 'Retrack beats'
- 'Play clicks on beats'
- 'Download tempo and compute beats'
- 'Upload beats to MTGDB'
- 'Download beats from MTGDB'

Those menus enable interactions between the user and the beat-tracking algorithm. The user can run the algorithm "from scratch" and check its output (i.e. finely adjust the position of each beat). Another option is to "give an orientation" to the automatic computation of beats by providing few, very reliable, information, typically a couple of beats. The user can specify those few initial beats manually, or he can select them among the output of a first run of the algorithm. This is very useful to correct phase errors[2] and confusions between metrical levels.

The possibility to keep beats up to a certain point and discard the following ones is especially useful in the case of excerpts with strong tempo changes.

In case the excerpt at hand has been downloaded from the MTG database, it might have a tempo value (in BPM) associated to it (some assets in the current repository are already characterised by an approximate value for the tempo). This metadata is clearly a very useful input to the algorithm.

As in [2] and [3], short audio signals can be added at beat times for useful audio feedback, for instance noise bursts, closed hi-hat or cowbell samples.

3.3. Annotating several metrical levels

Several metrical levels coexist in music, from low levels (small time divisions) to high levels (longer time divisions). And a beat at a high level must also be a beat at each lower level [5].

Therefore, provided that a metrical level has already been annotated, the process of specifying the beats of the next higher metrical level is simple: create a new transcription pane and copy already annotated beats (i.e. the lower level), select a few beats that determine the next level and discard the others.

When determining a low level from a higher one, the same process applies, but instead of discarding beats, the user can specify whether a duple or a triple subdivision should be performed. Alternatively, it is possible to add a few beats in between successive beats, the data can then be retracked with this newly defined metrical level.

3.4. Differences with BeatRoot

The graphical interface shows important differences with BeatRoot. For instance, it is our belief that WaveSurfer's built-in visualisation functionalities (e.g. running cursor and scrolling panes synchronized with audio playback), and its intuitive keyboard shortcuts and general look-and-feel are an enhancement of BeatRoot's interface. Also, an important point is that graphical configurations can be defined by the user. Other relevant differentiating features are the capture of keyboard messages while listening to the audio, as well as database connection facilities, multi-platform support[3] and the possibility to instantiate diverse transcription panes in order to annotate several metrical levels. However, it must be noted that BeatRoot also permits to annotate beats of MIDI data, which the system reported here does not permit.

[2] i.e. when the tempo is correctly computed but the computed beats are all a constant away from their correct positions

[3] i.e. Debian Linux and Windows XP at proceedings publishing date

4. IMPLEMENTATION

4.1. General architecture

WaveSurfer is built using Snack, a sound manipulation extension to the Tcl/Tk scripting language. Snack can be used as a scripting language, from a command prompt, WaveSurfer offers a graphical interface to it. See [4] for details. The important point here is that functionalities can be added to WaveSurfer by creating new Snack (or Tcl) commands and calling them through plugins.

In order to add beat tracking functionalities to WaveSurfer, two components are needed: an *external library* embedding those functionalities (with an appropriate C interface)[4] and a *plugin script* to call them from WaveSurfer.

The plugin script is written in Snack and logically accounts for a command that loads the library, typically: 'load C:/.WaveSurfer/1.6/plugins/libBeatSnack.dll' (in Windows). Additional commands (calling library functions) are directly accessible from WaveSurfer transcription panes, as e.g. 'Compute beats', 'Retrack', 'Erase beats from cursor', etc.

On the other hand, the external library is written in C and C++ (see Figure 2). A first part of the library creates the actual Snack function called in the plugin e.g. from the command 'Compute beats'. This part is written in C. The second part, the core beat tracking algorithm, is written in standard C++ and CLAM.

As shown in Figure 2, Interface2CLAM.h is the interface between these two parts of the library. This interface is a simple function whose input parameters are the sound samples and optionally a list of beats. Its output is a beat list (and optionally an onset list). The function definition is written in Interface2CLAM.h while its implementation is in Interface2BeatRoot.cxx.

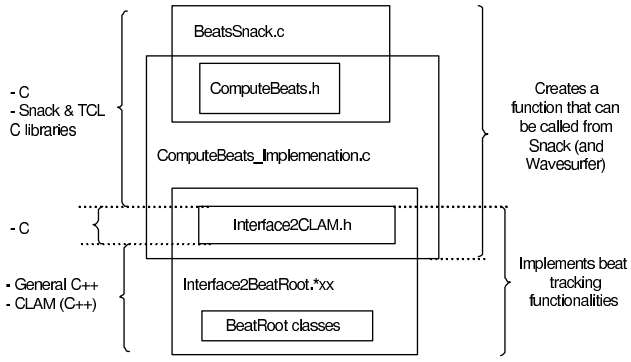

Figure 2: *External library architecture.*

4.2. Beat tracking algorithm

The automatic algorithm at the core of the system is the association of the transient detection and tempo induction algorithms already present in the CLAM library and BeatRoot's beat tracking method [3].[5] This part of BeatRoot has been ported to the CLAM framework, and in the current version, the tracking part is intended not

[4] see http://www.speech.kth.se/snack/modex.html

[5] http://www.oefai.at/~simon/beatroot/index.html

to present differences with the public version of BeatRoot (the set of default parameter may differ though).

In order to account for the diverse usage modes specified above, the input to the beat tracking algorithm can be the following:

- Case 1: The sound samples.
- Case 2: The samples and some (correct) beat times.
- Case 3: The samples and a (correct) tempo value

In all cases, outputs are beat times of the whole audio.

4.3. Database connection

In case the audio signal under annotation is part of the MTG repository (see [6]), interesting functionalities are provided.

The audio repository can be browsed from any remote web browser. It is possible to launch WaveSurfer directly from the web interface. The audio is loaded automatically together with annotations that may be available on the repository. There is a connection to the MTG database server via web services (SOAP) which allows to upload new segmentations. SOAP[6] is a lightweight protocol based on XML-RPC calls. The Service interface is described in a WSDL[7] file (a superset of XML) indicating methods calls, objects, and exceptions that will be sent across the net. As all the exchange of information is made with XML (both data and control messages), SOAP allows the interaction between programs on different platforms or in different languages running anywhere on the Internet.

5. SUMMARY

In this paper, we presented a multi-platform software for semi-automatic beat annotation of audio signals. Being open source, it can be used and modified at will. It has been implemented as a plugin for the WaveSurfer sound editor and, in addition to this software's functionality, it embeds useful functionalities from CLAM and BeatRoot. Source code and binaries are available at http://www.iua.upf.es/~fgouyon/BeatTrackingPlugin.html

Applications for this software are manifold. Particularly in the field of Music Information Retrieval.

At the time of writing, additional functionalities are being added to this software. For instance the handling of diverse audio file format (WaveSurfer can handle many standard format (WAV, AIFF, MP3, etc.), so does CLAM, however, we still did not add this functionality to the beat tracking algorithm). We are also working on the integration of the free "aubio" library developped by Paul Brossier at the Queen Mary University of London,[8] this library provides a C implementation of powerful and fast onset detection algorithms [14]. Visualization and interactive onset detection will also be a useful feature. Finally, we are working on the addition of diverse data plots such as tempo curves.

6. ACKNOWLEDGMENTS

This work reported in this paper was partially funded by the EU-FP6-IST-507142 project SIMAC (Semantic Interaction with Music Audio Contents). More information can be found at the project website http://www.semanticaudio.org. Thanks to Miguel Ramirez and David Garcia for support in dynamic library implementation within CLAM. Thanks also to Georgios Emmanouil, Günter Geiger, Pedro Cano, Jose Pedro Garcia, Vadim Tarasov, Markus Koppenberger, Paul Brossier and the SIMAC team in the MTG.

[6] http://www.w3.org/TR/soap/

[7] http://www.w3.org/TR/wsdl

[8] http://piem.homeip.net/~piem/aubio

7. REFERENCES

[1] Fabien Gouyon and Benoit Meudic, "Towards Rhythmic Content Processing of Musical Signals: Fostering Complementary Approaches," *Journal of New Music Research*, vol. 32, no. 1, 2003.

[2] Masataka Goto and Yoichi Muraoka, "Issues in evaluating beat tracking systems," in *Proc. International Joint Conference on Artificial Intelligence*, 1997.

[3] Simon Dixon, "An interactive beat tracking and visualisation system," in *Proc. International Computer Music Conference*, 2001.

[4] Kåre Sjölander and Jonas Beskow, "WaveSurfer - An open source speech tool," in *Proc. International Conference on Spoken Language Processing*, 2000.

[5] Fred Lerdahl and Ray Jackendoff, Eds., *A generative theory of tonal music*, MIT Press, Cambridge MA, 1983.

[6] Pedro Cano, Markus Koppenberger, Sira Ferradans, Alvaro Martinez, Fabien Gouyon, Vegard Sandvold, Vadim Tarasov, and Nicolas Wack, "MTG-DB: A Test Environment for Music Audio Processing," in *Proc. International Conference on Web Delivering of Music*, 2004.

[7] Masataka Goto and Yoichi Muraoka, "Real-time Beat Tracking for Drumless Audio Signals: Chord Change Detection for Musical Decisions," *Speech Communication*, vol. 27, no. 3, 1999.

[8] Fabien Gouyon and Perfecto Herrera, "A beat induction method for musical audio signals," in *Proc. WIAMIS Special session on Audio Segmentation and Digital Music*, 2003.

[9] Anssi Klapuri, "Musical meter estimation and music transcription," in *Proc. Cambridge Music Processing Colloquium*, 2003.

[10] Fabien Gouyon and Perfecto Herrera, "Exploration of techniques for automatic labeling of audio drum tracks instruments," in *Proc. MOSART Workshop on Current Research Directions in Computer Music*, 2001.

[11] Jouni Paulus and Anssi Klapuri, "Model-based Event Labeling in the Transcription of Percussive Audio Signals," in *Proc. International Conference on Digital Audio Effects*, 2003.

[12] Christian Uhle and Christian Dittmar, "Generation of Musical Scores of Percussive Unpitched Instruments from Automatically Detected Events," in *Proc. 116th AES Convention*, 2004.

[13] Simon Dixon, Werner Goebl, and Gerhard Widmer, "Real Time Tracking and Visualisation of Musical Expression," in *Proc. International Conference on Music and Artificial Intelligence*, 2002.

[14] J.P Bello, C. Duxbury, M. E. Davies, , and M. Sandler, "On the use of phase and energy for musical onset detection in the complex domain," *IEEE Signal Processing Letters*, 2004.