Sound Analyser: A Plug-In For Real-Time Audio Analysis In Live Performances And Installations

Adam M. Stark
Independent
London, UK
hello@adamstark.co.uk

ABSTRACT

Real-time audio analysis has great potential for being used to create musically responsive applications in live performances. There have been many examples of such use, including sound-responsive visualisations, adaptive audio effects and machine musicianship. However, at present, using audio analysis algorithms in live performance requires either some detailed knowledge about the algorithms themselves, or programming – or both. Those wishing to use audio analysis in live performances may not have either of these as their strengths. Rather, they may instead wish to focus upon systems that *respond* to audio analysis data, such as visual projections or sound generators.

In response, this paper introduces the *Sound Analyser* – an audio plug-in allowing users to a) select a custom set of audio analyses to be performed in real-time and b) send that information via OSC so that it can easily be used by other systems to develop responsive applications for live performances and installations. A description of the system architecture and audio analysis algorithms implemented in the plug-in is presented before moving on to two case studies where the plug-in has been used in the field with artists.

Keywords

audio analysis, live performance, VST, Audio Unit, plug-in

1. INTRODUCTION

The last 20 years have seen a large amount of interest in the field of audio analysis. This has led to the development of algorithms capable of onset detection [1], beat tracking [8], pitch detection [7], downbeat detection [9], chord recognition [6] and many other forms of musical audio analysis.

A large driver of research in audio analysis to date has been *offline* Music Information Retrieval (MIR) tasks on databases of music, such as in [2]. However, there are also many applications for audio analysis in real-time contexts – including in the live performance of music and sound based installations. We can point to examples of audio analysis being used to produce responsive visualisations [15], playing a role in augmented instruments [11], the many examples of machine musicianship [12] and audio effect parameters augmented using audio analysis [13].

NIME'14, June 30 – July 03, 2014, Goldsmiths, University of London, UK.

Figure 1: The *Sound Analyser* plug-in interface with integrated plotting. The user can set the destination IP address and port number that audio analysis data will be sent to.

1.1 Existing Software for Audio Analysis

There are several pieces of standalone software for performing audio analysis. An example is the excellent Sonic Visualiser [5] software. However, while this software is very versatile, the algorithms are implemented in an offline fashion (even if they are causal), and it is not possible to send the data elsewhere in real-time for use in performance applications.

At present, those wishing to use audio analysis in a live performance context are faced with two options. The first is to write their own audio analysis software (e.g. in C++, Java or Python) or to interface with an audio analysis library, such as *LibXtract* [4]. This may be an excellent choice for the project in question, if the audio analysis is very specific to the application or if the desired end result is some standalone software. However, it does require a degree of literacy with the specifics of audio analysis algorithms (and programming!) that may not be the interest or strength of those wishing to incorporate real-time audio analysis into their system.

Another option is to use one or more third party objects, such as the *Zsa.Descriptors* presented in [10], within a programming environment such as Max. Again, this may be a wholly appropriate choice, but it still means programming a new tool from scratch, requires some knowledge of Max and assumes that the audio to be analysed was already going to be received by Max, and not some other music software such

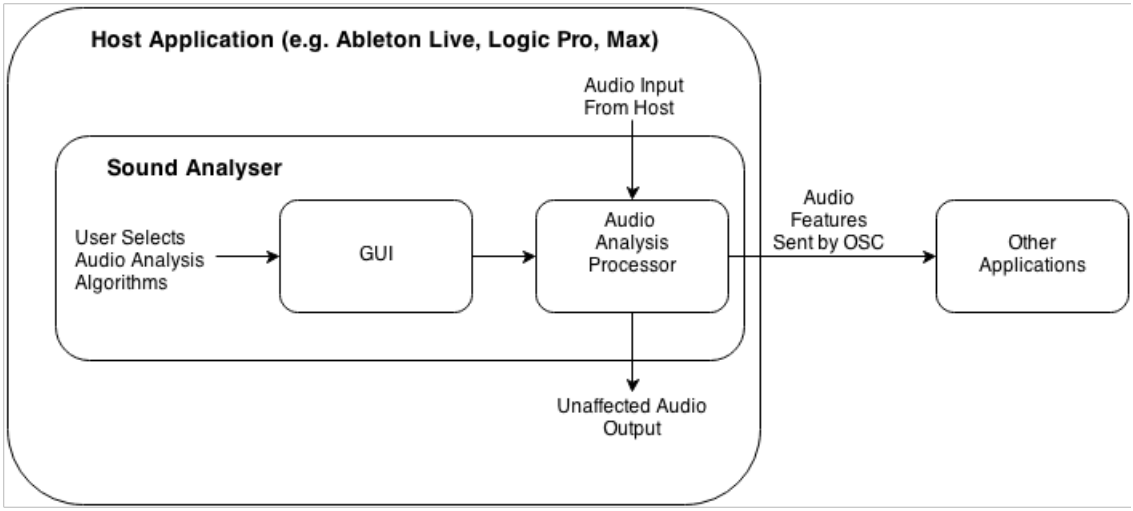

Figure 2: An overview of the operation of the Sound Analyser when it is loaded into a host application. The audio remains unaffected and once the user has selected one or more audio analysis algorithms, the resulting features are sent outside of the host via OSC to other applications.

as Ableton Live, or Logic Pro.

1.2 Sound Analyser

In order to allow non-experts to use real-time audio analysis, this paper introduces *Sound Analyser* - a plug-in for using audio analysis in live performances. It allows a user to select a custom set of real-time audio analysis algorithms using the audio plug-in interface, and sends the result of the processing via Open Sound Control (OSC) [16] to other software. This approach is quick to set up for the user, flexible in terms of the audio analysis algorithms performed and requires minimal knowledge about audio analysis algorithms and programming from the user.

The rest of this paper is structured as follows. We proceed in Section 2 with an overview of the system, including an overview of the forms of audio analysis that are presently implemented in the Sound Analyser. In Section 3 we briefly discuss the software implementation of the plug-in before proceeding to a discussion of the advantages of using an audio plug-in for audio analysis in Section 4. Finally, we turn to two case studies of where the plug-in has been used with artists in real-time contexts in Section 5.

2. SYSTEM DESCRIPTION

The Sound Analyser plug-in is intended to be used with host software that supports either the VST or Audio Unit frameworks. Whilst being an audio plug-in, it has the added functionality of being able to send audio analysis data over a network to other software. In this section we give a brief overview of the system and the audio analysis algorithms implemented in the current version.

2.1 Basic Operation

Once loaded, the plug-in operates as follows:

1. The user selects one or more forms of audio analysis to be performed in real-time, as well as the destination port and IP address for OSC messages.

2. The selected audio analysis algorithms are performed on the audio provided by the host application, sending the computed analysis data over a network via OSC. This approach is outlined in Figure 2.

2.2 Modular Selection of Audio Analysis Algorithms

Each instance of the plug-in allows the user to select a custom set of audio analyses that run in parallel. As different audio analysis algorithms can be added to or removed from the plug-in as the user requires, selecting features for a given live performance application can be done in an on the fly 'browsing' manner.

For algorithms that require the calculation of the Fourier transform, this is computed only once and shared between all analysis algorithms to reduce the computation overload.

2.3 Audio Analysis 'Audition' through Integrated Plotting

The plug-in has an integrated real-time plotting system (see Figure 1), allowing users to 'audition' different forms of audio analysis before choosing the right one for the application in question. The plotting system can display both one-dimensional time domain signals and vector signals, where N samples are presented per frame, as with the FFT magnitude spectrum. For each audio analysis algorithm selected by the user, options are presented to *send* information via OSC, or to *plot*. If neither is selected for a given audio analysis algorithm, the processing for that algorithm is not performed to reduce unnecessary computation.

2.4 Audio Analysis Algorithms

We now present an overview of the audio analysis algorithms implemented in the Sound Analyser at the time of writing.

2.4.1 Common Time Domain Features

The plug-in includes a number of time domain features, including Root Mean Square (RMS), Peak Energy and the Zero Crossing Rate.

2.4.2 Common Frequency Domain Features

The frequency domain features implemented in the plug-in include the Spectral Centroid, Spectral Flatness and Spectral Crest.

2.4.3 Onset Detection Functions

A number of onset detection functions (see [1] for explanations) are available including Energy Difference, Spec-

tral Flux, High Frequency Content Detection Function and Complex Spectral Difference.

2.4.4 Pitch Detection

The pitch detection algorithm implemented in the plug-in is based on the *YIN* pitch detector found in [7].

2.4.5 Spectra

The plug-in can compute a number of spectra and these can also be sent via OSC. Presently, the user is able to calculate the FFT Magnitude Spectrum, Mel-frequency representations and the Constant-Q Transform [3].

2.4.6 Chord Recognition

The plug-in includes an implementation of the chord recognition algorithm presented in [14]. This allows both chords and the chromagram features used to detect them to be sent over OSC.

3. SOFTWARE IMPLEMENTATION

The Sound Analyser plug-in is implemented in C++, using the *JUCE* framework, and is available in both VST and Audio Unit formats. The code for the plug-in is open source, available under a GPL license[1].

4. DISCUSSION

We have presented a real-time audio analysis plug-in capable of sending analysis data via OSC. We will now discuss some of the advantages of approaching real-time audio analysis in this way.

4.1 Potential for Use in Multiple Hosts

A key advantage of the Sound Analyser plug-in is that it can easily be used in any software that can host VST or Audio Unit plug-ins, including major software sequencers such as Ableton Live and Logic Pro as well as programming environments like Max. This allows the extraction of multiple real-time audio features without implementing entirely new systems.

4.2 Audio Analysis Without Programming

It is also important to highlight the 'no programming' approach to performing audio analysis – the GUI allows the user to select the required algorithms and so can do in minutes what may take at least several hours when writing custom software. Of course, some programming may be needed on the application side of whatever the audio analysis is intended to be used for, but the lack of any need to program the audio analysis stage both saves time and allows non-experts to use audio analysis in applications.

4.3 Multi-track Audio Analysis

It is possible to run multiple instances of the plug-in on different 'tracks', allowing multi-track audio analysis. Furthermore, each instance can use a different set of audio analysis algorithms to process the audio on that track. This would allow, for example, audio analysis of a live band where several different audio streams were available for the different instruments.

In order to distinguish the same form of audio analysis produced by different instances of the Sound Analyser on different tracks, each analyser has an instance ID that can be changed, and this is prepended to the OSC message to allow each track to be uniquely identified. This can be seen in Figure 1 where the ID has been set to 'viola'. This would render the RMS address pattern as */viola/rms*.

[1] `http://www.adamstark.co.uk/sound-analyser`

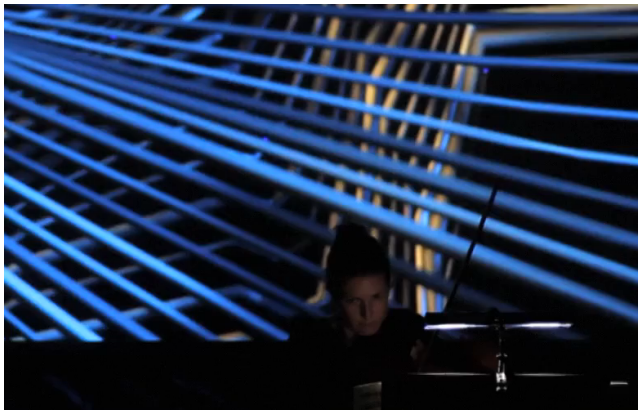

Figure 3: Odile Auboin performs Ligeti's *Sonata for Solo Viola* in 2012 with accompanying visuals informed by audio analysis from the Sound Analyser plug-in

4.4 Audio Pre-processing

In some hosts, users can load other plug-ins before the Sound Analyser, allowing them to pre-process the audio signal (e.g. by using an equaliser to remove some frequency components) so that the parts of the audio signal that are of interest can be emphasised.

4.5 Potential for Distributed Processing

The plug-in combined with OSC output allows the audio analysis to be computed on one computer, with the use of that data to occur on another computer. This may be desirable as some audio analysis algorithms can be computationally expensive.

4.6 Storage of Audio Analysis Parameters With the Host Application

Audio plug-in host applications automatically store plug-in parameters and this allows the choice of audio analysis algorithms and the parameters associated with each to be automatically stored with the host application and recalled when the host is reloaded.

5. CASE STUDIES

5.1 Case Study 1: Live Visuals for a Classical Music Concert

The Sound Analyser plug-in originates from an audio-visual project to develop responsive visualisations to classical music performances, led by artists Davide Quayola and the duo Abstract Birds (Natan Sinigaglia and Pedro Mari). In 2012, the author worked with these artists to develop a system for responsive visuals during a performance of György Ligeti's *Sonata for Solo Viola* by Odile Auboin of the Ensemble Intercontemporain. The piece was performed during the Nemo Festival in Paris, in December 2012.

The artists had prepared a number of visual themes using the programming environment *vvvv*. Using a prototype of the Sound Analyser plug-in, the author helped the visual artists to map a number of audio analysis parameters to control aspects of visual output. These included RMS energy, pitch detection, the FFT magnitude spectrum and onset detection functions.

Each visual theme required a different set of audio analyses, and we experimented with either presenting all audio analysis data at once, or switching between different sets of audio analysis algorithms by switching analyses on and off

Figure 4: Musician Imogen Heap during the filming of the *Me The Machine* music video. In the photo, information from her musical data gloves was used to draw shapes with their magnitude determined by audio features from the Sound Analyser plug-in

using the plug-in. A still from the performance can be seen in Figure 3.

5.2 Case Study 2: Sound Responsive Visuals for a Music Video

In December 2013, the author worked with a team of artists, directors, engineers and others on the production of musician Imogen Heap's music video for the song *Me The Machine*. The song itself had been written and recorded using musical data gloves, and so the intention was for the gloves to feature significantly in the video. This involved the information from the data gloves being used to augment visual projections created by a number of artists.

During the development of the video, it became clear that it would aid the overall effect of the visual projections if they were able to respond to the music directly also. Ahead of the filming, the author worked with visual artist Roman Miletitch to make his visual elements, implemented using the library *Cinder*, responsive to musical aspects of the song.

Using the Sound Analyser plug-in, the author and the visual artist were able to quickly audition different forms of audio analysis, inspecting the effect they produced on the visualisations. In the end, it was found that very simple relationships between signal energy information and the magnitude of shapes was the most effective mapping – but the process of discovery was simple and allowed the audition of several other forms of analysis including several onset detection functions and the spectral centroid. A still from the video can be seen in Figure 4.

6. CONCLUSION

This paper has presented *Sound Analyser*, an audio plug-in for performing audio analysis in real-time and sending the result via OSC to other applications. We have argued that this approach to real-time audio analysis is a simple way for non-audio analysis experts to use such information to create audio-responsive applications in live performances.

In future work, the audio plug-in will be expanded to be capable of many other forms of audio analysis including beat tracking, downbeat detection, key detection and multi-pitch detection.

7. ACKNOWLEDGEMENTS

The author would like to thank Davide Quayola, Natan Sinigaglia, Imogen Heap and Roman Miletitch for their time, feedback and advice which has been a huge contribution to the research.

8. REFERENCES

[1] Juan P. Bello, Laurent Daudet, Samer Abdallah, Chris Duxbury, Mike Davies, and Mark B. Sandler. A tutorial on onset detection in music signals. *IEEE Trans. on Audio, Speech and Language Processing*, 13(5):1035–1047, September 2005.

[2] Dmitry Bogdanov, Nicolas Wack, Emilia Gómez, Sankalp Gulati, Perfecto Herrera, Oscar Mayor, Gerard Roma, Justin Salamon, José Zapata, and Xavier Serra. Essentia: An audio analysis library for music information retrieval. In *Proc. ISMIR*, pages 493–498, 2013.

[3] Judith C. Brown. Calculation of a constant-Q spectral transform. *Journal of the Acoustical Society of America*, 89:425–434, January 1991.

[4] Jamie Bullock. LibXtract: A lightweight library for audio feature extraction. In *Proc. of International Computer Music Conference*, 2007.

[5] Chris Cannam, Christian Landone, Mark Sandler, and Juan Pablo Bello. The sonic visualiser: A visualisation platform for semantic descriptors from musical signals. In *Proc. ISMIR*, 2006.

[6] Taemin Cho, Ron J. Weiss, and Juan P. Bello. Exploring common variations in state of the art chord recognition systems. In *Proc. of Sound and Music Computing Conference*, 2010.

[7] Alain de Cheveigne and Hideki Kawahara. YIN, a fundamental frequency estimator for speech and music. *Journal of the Acoustical Society of America*, 111(4):1917 – 1930, April 2002.

[8] Norberto Degara, Enrique Argones Rúa, Antonio Pena, Soledad Torres-Guijarro, Matthew E.P. Davies, and Mark D. Plumbley. Reliability-informed beat tracking of musical audio signals. *IEEE Trans. on Audio, Speech and Language Processing*, 2012.

[9] Jason A. Hockman, Matthew E.P. Davies, and Ichiro Fujinaga. One in the jungle: Downbeat detection in hardcore, jungle and drum and bass. In *Proc. of 13th International Society for Music Information Retrieval Conference (ISMIR 2012)*, pages 169–174, 2012.

[10] Mikhail Malt and Emmanuel Jourdan. Zsa.descriptors: a library for real-time descriptors analysis. In *Proc. of Sound and Music Computing Conference*, 2008.

[11] Loïc Reboursière, Christian Frisson, Otso Lähdeoja, John Anderson Mills III, Cécile Picard, and Todor Todoroff. Multimodal guitar: A toolbox for augmented guitar performances. In *Proc. New Interfaces for Musical Expression (NIME 2010)*, pages 415–418, Syndey, Australia, 2010.

[12] Robert Rowe. *Machine Musicianship*. MIT Press, 2001.

[13] Adam M. Stark, Matthew E.P. Davies, and Mark D. Plumbley. Real-time beat-synchronous audio effects. In *Proc. New Interfaces for Musical Expression*, 2007.

[14] Adam M. Stark and Mark D. Plumbley. Real-time chord recognition for live performance. In *Proc. of International Computer Music Conference*, 2009.

[15] Robyn Taylor, Pierre Boulanger, and Daniel Torres. Real-time music visualization using responsive imagery. In *Proc. of the 8th International Conference on Virtual Reality*, pages 62–69, April 26-30 2006.

[16] Matthew Wright, Adrian Freed, and Ali Momeni. OpenSound Control: State of the art 2003. In *Proc. New Interfaces for Musical Expression*, pages 153–159, 2003.