Loop-aware Audio Recording for the Web

Gerard Roma
Georgia Institute of Technology
Atlanta, GA, USA
gerard.roma@gatech.edu

Anna Xambó
Georgia Institute of Technology
Atlanta, GA, USA
anna.xambo@gatech.edu

Jason Freeman
Georgia Institute of Technology
Atlanta, GA, USA
jason.freeman@gatech.edu

ABSTRACT

Music loops are audio recordings used as basic building blocks in many types of music. The use of pre-recorded loops facilitates engagement into music creation to users regardless of their background in music theory. Using online loop databases also affords simple collaboration and exchange. Hence, music loops are particularly attractive for web audio applications. However, traditional musical audio recording typically relies on complex DAW software. Recording loops usually requires consideration of musical meter and tempo, and withstanding metronome sounds.

In this paper, we propose *loop-aware audio recording* as a use case for web audio technologies. Our approach supports hands-free, low-stress recording of music loops in web-enabled devices. The system is able to detect repetitions in an incoming audio stream. Based on this information, it segments and ranks the repeated fragments, presenting the list to the user. We provide an example implementation, and evaluate the use of the different MIR libraries available in the web audio platform for the proposed task.

1. INTRODUCTION

Repetitive structures are ubiquitous in music. Repetition is a particular trait of many electronic and computer music genres, where humans can offload the task of repeating patterns to the machine. Since the appearance of digital audio samplers, audio loops have been a prominent device for music creation. Audio loops can be simply digital audio fragments that are repeated, although often they contain some sort of variation that creates a rhythm sensation. Loops are sometimes extracted from existing music recordings to make new compositions, simplifying the music creation process regardless of the author's background in conventional music theory. In this sense, the use of audio loops has played an important role in the democratization of music production attributed to digital technology.

As basic collaborative building blocks, audio loops are particularly relevant to the emerging space of web-based music creation. They are shared in online general audio databases such as Freesound[1] [1], or specific loop databases like Looperman.[2] Loops are also commercially distributed in online shops like Beatport.[3]

The work presented in this paper originates in the context of Earsketch,[4] an online Digital Audio Workstation (DAW) developed to enhance computer science learning [5]. In Earsketch, users create musical compositions through programing, using a database of audio loops. The loops in the database are originally recorded at different tempos, but a time-stretching module is used to allow importing any of them in the same composition while ensuring a consistent tempo. This system has been used by more than 100,000 Internet users, including students from 200 schools, to create more than 80,000 compositions, providing a good example of how audio loops can help facilitating musical creativity to a wide audience.

While indexing and retrieval of loop databases has been addressed in the music information retrieval (MIR) [12] and web audio [11] communities, the creation of loops is very often considered from the point of view of synthesis and computer-based tools, and little attention is paid to interfaces for capturing loops from the acoustic environment. In this paper, we propose *loop-aware audio recording* as a use case for collaborative audio. As an illustration, one could envision a musician exploring ideas with a musical instrument or other sound-producing device. In order to share an idea as a loop, one would have to record and then edit the recording to select the loop points. Alternatively, recording into a DAW could involve selecting the appropriate meter and tempo, and then using a metronome as a cue and record some number of bars. This latter approach is the one currently used by Earsketch. In both cases, the creative exploration would have to be interrupted in order to attend more tedious tasks. Also, in many cases using a metronome is out of question, as tempo variations are important for many music styles.[5] A metronome cue also requires an audio setup that prevents its sound bleeding into the microphone (e.g. headphones and potentially a directional microphone) which a regular web user may not have. Our proposed solution is

[1] `http://freesound.org`
[2] `http://looperman.com`
[3] `http://beatport.com`
[4] `http://earsketch.gatech.edu`
[5] As an illustration of the popular sentiment against the metronome, consider the page about it at the online quote compendium wikiquote, which contains more than 40 quotes against it and only 5 in favor: `https://en.wikiquote.org/wiki/Metronome`

Web Audio Conference WAC-2017, August 21–23, 2017, London, UK.

to enhance the recorder to automatically segment and select repeated fragments. Our system is implemented using web audio standards, allowing its use in mobile phones and tablets. In addition to improving the process of capturing loops with musical instruments, our system can also be used to record repetitive sounds in the environment, such as everyday sounds or serendipitous music encounters.

2. DETECTING REPETITION

Self-similarity matrices (SSM) of audio spectrum frame sequences are used often in MIR to identify patterns, particularly repetition. Early on in the history of MIR, Foote and Uchihashi introduced the Beat Spectrum [2] as the average of the SSM diagonals. Peaks in the beat spectrum represent repeated audio sequences.

Similar ideas based on SSMs have been used since then to analyze music structure [8, 7]. For example, methods based on SSMs have been used for chorus detection and music thumbnailing [3, 6]. The idea was also applied to detection of loops in music for creative applications [9], leading to indexing of loop databases [12]. However, samples extracted from music recordings (especially where regular rhythms are used) can be used as loops in new music regardless of whether they were originally repeated. Here, instead of seeking repetition in existing music recordings, we use repetition as a cue for improving the usability of the recorder, considering that repetition is often used for consolidating musical ideas.

Our system does not need to deal with the complexity of music structure, since it is aimed at providing the building blocks of the creative process. On the other hand we are interested in interactive web-based applications, with the possibility of mobile-based capture. At the same time, we aim at detecting repetition in a wide variety of signals that may be captured with web-enabled devices, including music and environmental sound. For these reasons, our system is based on simple and cost-effective methods based on the original beat spectrum. While similar ideas were used in [11] for indexing loops in Freesound, in this paper we present a purely client-side implementation based on web audio, and include loop start detection, aimed at loop recording.

3. LOOP-AWARE RECORDING

The proposed workflow starts by analyzing incoming audio using a short-time Fourier transform (STFT). Spectral frames are summarized into features and the SSM is then computed using cosine distance between all the feature vectors of the sequence (Figure 1a). The beat spectrum is obtained as the average of the diagonals of the SSM (Figure 1c). Peaks in this spectrum indicate repetitive periods in the input signal. For multiple repetitions of the same sequence, several peaks will be found (e.g. if it is repeated four times there will be a peak for the length of the repeated sequence and another one for the length of twice the sequence). As the diagonals get shorter, fluctuation increases and the average is less reliable, often resulting in spurious peaks towards the end. Since only periods of up to half the length of the recording make sense, we only consider peaks in the first half of the beat spectrum. We select the highest peak as the best candidate for segmenting the audio recording.

The next problem is finding the start of the loop. This can be determined by analyzing the diagonal corresponding to the selected peak (Figure 1e). However, in the presence of background sound, the diagonal can become significantly noisier due to general similarities between audio frames. We found that, in this case, they exhibit a characteristic flat region after a sharp increase when audio starts repeating (Figure 1f). After smoothing the signal, we detect the first of such *plateau* regions when the second derivative is considerably close to zero and the diagonal is above its median for a certain minimum amount of frames. We experimentally set this threshold to 200 ms.

Once the starting point and loop duration have been determined, the frame sequence is split into loop candidates. The number of loops is determined by the number of peaks in the beat spectrum that have a harmonic relationship with the first, given some error threshold. The next step is selecting the best one. We see this as a ranking problem, since the user may always have some specific criterion. While rhythmic accuracy features could be of interest depending on the musical focus, we adopt a more general strategy by aggregating the frames of each candidate and computing similarities between them. The list is then sorted according to the distance to the centroid, so that the most "typical" candidate is recommended first.

4. EXPERIMENTS

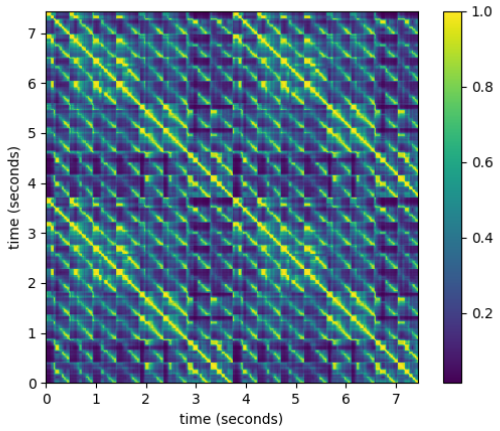
We devised two experiments in order to determine the viability of the proposed approach. In the first experiment, we compared general spectrum descriptors from two feature extraction libraries available for web audio applications. In the second experiment, we evaluated our approach for determining the starting point of the loop. Both experiments were developed as NodeJS programs and executed on a laptop computer for convenience.

4.1 Dataset

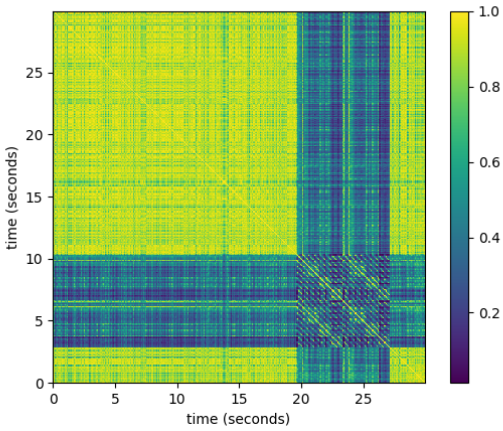
For both experiments, we collected a dataset by sampling the loop library available in Earsketch. The library is composed of two main collections, contributed by two authors, totaling around 4,000 loops. Each collection is divided in sample packs with a common style, which contain multiple instruments each. We randomly chose two loops from each pack, omitting those that started with silence (since our method for loop start detection relies on acoustic features of the signal). This resulted in a dataset of 217 loops including many different instruments (e.g. drum set sounds, synths, bass, guitar, piano...) and sound effects. The files were later processed for each of the experiments.

4.2 Loop period estimation

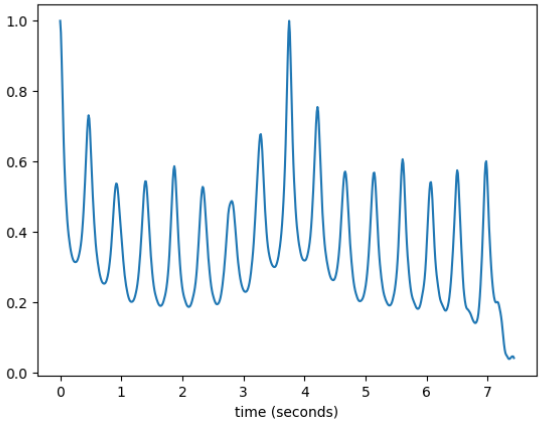
In the first experiment, we evaluated the use of the beat spectrum for estimating repetitive periods. We processed the dataset sounds by simply concatenating the samples of each loop to produce an identical repetition. Most of the excerpts contain two bars; in some cases the second bar is an exact repetition, but in some others it may be a variation, or part of a longer phrase. There are also cases where shorter patterns are repeated along the two bars. Thus, in addition to the original duration, we decided to accept half and a quarter of it as valid loop points. We consider the problem as a classification task, where the loop period found for a given file is considered correct if the distance to the ground truth is less than 20 ms. We use accuracy, as the fraction of correctly classified loops, to evaluate the algorithm.



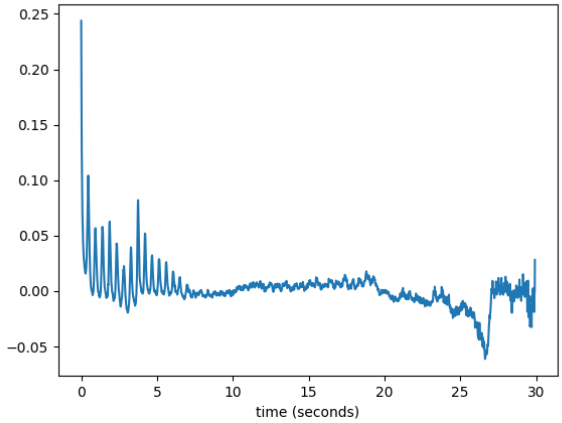
(a) Self-similarity matrix (loop).



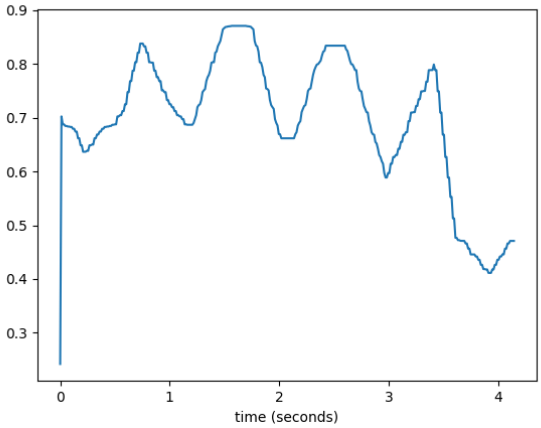
(b) Self-similarity matrix (loop over background noise).



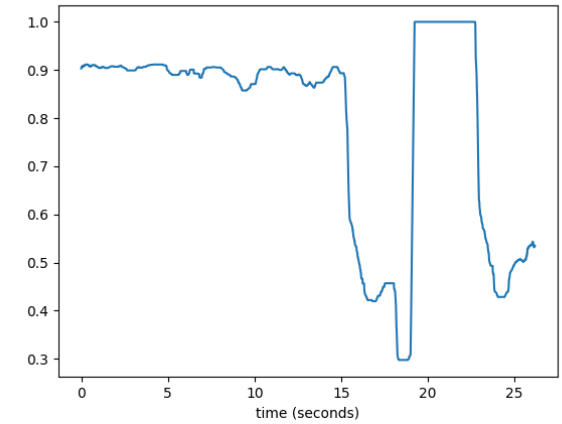
(c) Beat spectrum (loop)



(d) Beat spectrum (loop over background noise, de-trended).



(e) Matrix diagonal for maximum peak (loop).



(f) Matrix diagonal for maximum peak (loop over background noise).

Figure 1: Steps in the analysis of a bass loop, by itself and embedded in background noise.

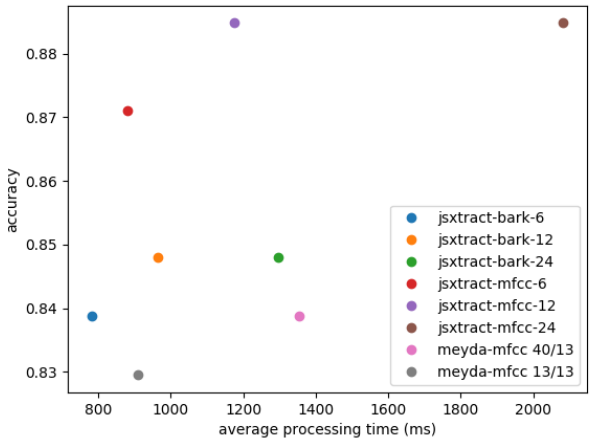


Figure 2: Accuracy vs average processing time on a core i5 laptop for different features.

Beyond accuracy, we were also interested in computation time. A trade-off exists between the cost of computing features from the spectrum and cost induced by the resulting dimensionality. For large feature dimension, the SSM takes very long to compute, so it is crucial to reduce the amount of features.

Within this framework, we compared Mel-Frequency Cepstral Coefficients (MFCC) and bark bands as generic spectral features, using two feature extraction libraries: JS-Xtract [4] and Meyda [10]. Neither of the two libraries include an STFT implementation for offline processing, since they are mainly designed to work for real-time analysis with Web Audio API analyzer nodes or similarly non-overlapping buffers, so we adapted them to work with a custom STFT implementation.[6] We measured the time for obtaining the estimate including feature extraction and SSM calculation, but factoring out the time to compute the magnitude spectrum. For JS-Xtract we tried several sizes of bark bands and MFCC features. In the case of MFCC the number of filters and cepstral coefficients was the same. Meyda does not implement bark bands as a feature, and does not allow changing the number of coefficients for MFCC, so we used the standard 13 coefficients from 40 bands and also 13 coefficients from 13 bands. Figure 2 shows the result of the experiment.

Perhaps surprisingly, we found that just a few bands/coefficients suffice to obtain a high accuracy, while saving some processing time. In this sense, using 6 MFCC coefficients would seem a good compromise. Obviously, the cost of the SSM grows quadratically with the length of the sequence, but using this setting, a recording of 30 seconds was processed in around 3 seconds (durations for the duplicated recordings ranged from 7 to 30 seconds, averaging around 15 seconds). Hence it should be possible to use this setting in an interactive system. A real-time implementation would also be possible using a fixed time window (implying a maximum loop length). Meyda MFCCs were in general faster but less accurate than JS-Xtract's. We also found the latter to be more flexible.

[6] https://github.com/g-roma/stft.js

4.3 Starting point estimation

We conducted a second experiment in order to analyze how to find the starting point. Since we are interested in capturing loops in uncontrolled recording conditions, our problem is significantly different from that of existing MIR works, where repetitive structures are sought in audio with a stable polyphonic timbre. On the other hand, determining the start of the loop in total silence is trivial.

In order to simulate a challenging recording condition, we mixed the files from the previous experiment with environmental sound from the DCASE2016 challenge dataset.[7] The loop was added at a random point in the background. This tends to add a trend to the beat spectrum, so we subtracted a moving median filter to remove it. We then computed the SSM and beat spectrum, and estimated the starting point from the diagonal corresponding to the maximum peak. While conventional onset detectors could be used with low noise conditions, the idea of using the diagonal is that non-repetitive sounds from the same target source should not trigger the start of a loop. We compared two approaches based on the diagonal by computing, for each file, the difference between estimated and the ground truth position in seconds. The first approach consists simply thresholding the smoothed diagonal so that 25% of the data is above the threshold. This approach has been used for musical audio [6], where the acoustic conditions are generally more controlled. In the presence of noise it approximates the start of the loop but with mixed results. The second approach was outlined in Section 3. With respect to the previous experiment, we found in initial trials that using MFCC features (which are known to be sensitive to noise) resulted in much worse estimation, so we used 24 bark band features from JS-Xtract.

Figure 3 shows box plots for the distribution of the difference between ground truth and estimated loop start. Clearly, the proposed method gives better results than simple thresholding: although there are many loops that are not detected correctly, most of the points are around a median value of 40 ms. A more detailed plot of the distribution of errors for our approach is shown in Figure 4. While listening to the results, it became clear that loops that had been mixed with indoor background noise were correctly localized, while loops with louder outdoor noise (especially loops containing noisy sounds such as drums) produced the worst errors. We tried both approaches in an initial prototype (Section 5) and found that in both cases for indoor usage the noise level allows a good identification of the loop start in most cases.

5. IMPLEMENTATION

We implemented the ideas presented in this paper in a JavaScript library and an initial demo prototype. The library is composed by a few classes for computation of SSMs and beat spectrum, visualization and recording of loops, as well as utility functions for smoothing, peak finding and so on. The code heavily relies on the *scijs*[8] collection of JavaScript packages that implements basic linear algebra operations and n-dimensional arrays in the style of the NumPy library for Python.

The prototype allows trying different settings and visualizing the different signals involved. The application relies

[7] http://www.cs.tut.fi/sgn/arg/dcase2016/

[8] https://github.com/scijs

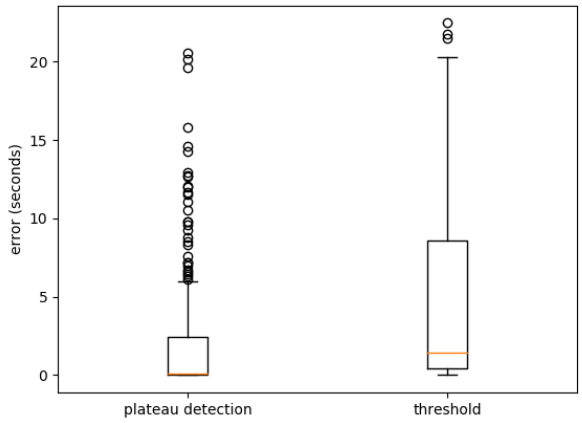


Figure 3: Box plots for loop start detection using step detection (*plateau*) vs thresholding.

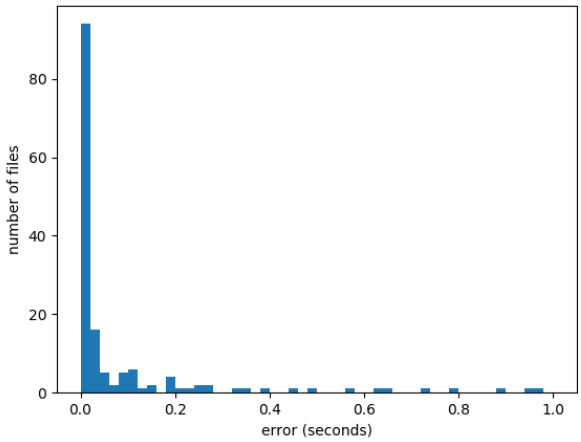


Figure 4: Histogram of deviations from the reference starting point for the *plateau* method.

on the *MediaDevices* interface, which at the time of this writing is still not implemented in all major browsers, but works in Firefox and Chrome. Incoming audio is recorded using Recorder.js,[9] and the analysis is performed at the end of the recording using the STFT, SSM and BeatSpectrum modules. Although web audio analyzer nodes do not provide a reliable callback interval, the process could be improved by using a *ScriptProcessorNode* and performing the analysis in real time. After loop period identification and loop start estimation, the recorded audio is segmented at zero-crossings near the start and end of the estimated loops, and a list of segments is presented to the user. While a user evaluation is out of the scope of this paper, our initial impressions were very positive with respect to the viability of the proposed use case. The system works best for loud and clear sounds and sequences with rich frequency content, such as repetitive speech or musical instruments. We plan to improve the initial demo to allow maintaining a personal collection of loops.
The current implementation can be obtained from GitHub: `https://github.com/g-roma/loopiness`.

6. CONCLUSIONS AND FUTURE WORK

In this paper we have proposed a novel application of loop detection: loop-aware audio recording. We hope that the proposed application will contribute to the development of collaborative music making based on web technologies, by allowing casual generation of music building blocks. We have shown that the basic signal processing techniques that allow the identification of loops can be implemented in interactive web audio applications. At the same time, we have identified a novel challenge specific to the proposed application, which is not usually addressed in MIR: the identification of repetitive structures embedded in noisy backgrounds. While further work is required for ubiquitous loop-aware recording, we have presented an implementation that can be used in indoor settings with typically low noise levels. As a next step, we plan to conduct a user study, including an evaluation of different approaches for ordering the segmented loops. We will also integrate this functionality in Earsketch in order to improve the potential for collaboratively growing the collection of audio loops.

[9] `https://github.com/mattdiamond/Recorderjs`

7. ACKNOWLEDGEMENTS

The EarSketch project receives funding from the National Science Foundation (CNS #1138469, DRL #1417835, DUE #1504293, and DRL #1612644), the Scott Hudgens Family Foundation, the Arthur M. Blank Family Foundation, and the Google Inc. Fund of Tides Foundation.

8. REFERENCES

[1] F. Font and X. Serra. Tempo estimation for music loops and a simple confidence measure. In *Proceedings of the 17th International Society for Music Information Retrieval (ISMIR 2016)*, 2016.

[2] J. Foote and S. Uchihashi. The beat spectrum: A new approach to rhythm analysis. In *2001 EEE International Conference on Multimedia and Expo (ICME 2001)*, 2001.

[3] M. Goto. A chorus-section detecting method for musical audio signals. In *Proceedings of the 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing ICASSP'03)*, 2003.

[4] N. Jillings, J. Bullock, and R. Stables. Js-xtract: A realtime audio feature extraction library for the web. In *Proceedings of the 17th International Society for Music Information Retrieval (ISMIR 2016)*, 2016.

[5] B. Magerko, J. Freeman, T. Mcklin, M. Reilly, E. Livingston, S. Mccoid, and A. Crews-Brown. Earsketch: A STEAM-based approach for underrepresented populations in high school computer science education. *ACM Transactions on Computing Education (TOCE)*, 16(4):14:1–14:25, Sept. 2016.

[6] M. Müller, P. Grosche, and N. Jiang. A segment-based fitness measure for capturing repetitive structures of music recordings. In *Proceedings of the 12th International Society for Music Information Retrieval (ISMIR 2011)*, pages 615–620, 2011.

[7] O. Nieto. *Discovering structure in music: Automatic approaches and perceptual evaluations*. PhD thesis, New York University, 2015.

[8] B. S. Ong. *Structural analysis and segmentation of music signals*. PhD thesis, Universitat Pompeu Fabra, 2006.

[9] B. S. Ong and S. Streich. Music loop extraction from digital audio signals. In *Proceedings of the 2008 IEEE International Conference on Multimedia and Expo (ICME 2008)*, 2008.

[10] H. Rawlinson, N. Segal, and J. Fiala. Meyda: An audio feature extraction library for the Web Audio API. In *Proceedings of the 1st Web Audio Conference (WAC)*, 2015.

[11] G. Roma and X. Serra. Music performance by discovering community loops. In *Proceedings of the 1st Web Audio Conference (WAC)*, Paris, France, 2015.

[12] S. Streich and B. S. Ong. A music loop explorer system. In *Proceedings of the 34th International Computer Music Conference (ICMC2008)*, 2008.