GRAPH-BASED AUDIO LOOPING AND GRANULATION

Gerard Roma , Pierre Alexandre Tremblay and Owen Green

CeReNeM
University of Huddersfield
Huddersfield, UK
n.surname@hud.ac.uk

ABSTRACT

In this paper we describe similarity graphs computed from time-frequency analysis as a guide for audio playback, with the aim of extending the content of fixed recordings in creative applications. We explain the creation of the graph from the distance between spectral frames, as well as several features computed from the graph, such as methods for onset detection, beat detection, and cluster analysis. Several playback algorithms can be devised based on conditional pruning of the graph using these methods. We describe examples for looping, granulation, and automatic montage.

1. INTRODUCTION

Short sound recordings, around the length of words and sentences, are used in the creative stages of many forms of audio and music production. Extending the these snippets in time has many applications, such as creating different musical gestures, or simulation of realistic sound textures and sound effects in cinema or video games. The general idea of extending sounds in time is almost as old as recording technology and can be traced back to tape loops and early approaches to granulation [1].

In the general case, the possibilities are not limited to a fixed objective such as the synthesis of a known sound, but very often emerge from the qualities of the sample at hand. Thus, automatic audio analysis can be used to leverage the inner structure, textures or objects in a given recording in an interactive setting. In popular platforms such as digital audio workstations and audio editing suites, it has become common to conflate the reading of a sound file with a time-frequency analysis that enables playback capabilities such as time scale modification.

One particular possibility of time-frequency analysis is computing similarity between frames. This has been extensively exploited by concatenative synthesis, either guided by some target sequence [2] or interactive exploration of a corpus of grains or short sounds [3]. On the other hand, content-based structural analysis has been used most often in the context of music information retrieval (MIR). In this paper, we explore using content-based structural analysis of audio for facilitating different playback algorithms that can be used to extend the content of a recorded sound in time. In particular, we propose using networks of similarity between different points in the spectrogram. We describe three algorithms: one for automatic looping, one for granular synthesis and one for automatic montage, and provide implementations for the Max and SuperCollider environments.

In the next section we review existing work related to the proposed approach. In Section 3, we detail the analysis framework. Section 4 describes the playback algorithms and their implementation. Some examples of using the proposed algorithms are presented in Section 5. We then conclude and outline future directions.

2. RELATED WORK

The idea of concatenating time-frequency frames based on similarity has been extensively used under the framework of corpus-based concatenative synthesis (CBCS) [4]. In CBCS the framework usually considers the audio material as an (ideally large) database of units with a focus on specifying the resulting sound. Here our focus is in the opposite direction: we aim to obtain different ways of extending short snippets interactively, with an interest in existing gestures and textures, using common paradigms such as looping and granulation. Several works have studied these tasks in a similar ways, based on time-frequency analysis and similarity graphs.

The idea of automating the creation of music loops from audio was proposed in [5], where an algorithm that found repetitive sections was presented. This problem is similar to other MIR tasks, such as finding the chorus in a pop song. In practice, however, loops are often devised to create new rhythmic structures even if the original audio does not contain a repetition. A user interface for automatic and semi-automatic loop editing was proposed more recently, using a very basic analysis algorithm [6]. We propose a more detailed algorithm that can be used interactively, addressing both the issue of seamless concatenation points and the possibility of leveraging existing repetitive content.

The proposed algorithms for granulation and montage are related to existing work on texture synthesis, particularly algorithms based on CBCS. Several algorithms were evaluated in [7]. Among these, the Montage Synthesis (MS) algorithm, is perhaps the closest to our approach, although its focus is concatenating larger segments for realistic texture synthesis and audio inpainting [8]. Another algorithm for inpainting was presented in [9], based on pruning the similarity graph. Our algorithms are similarly based on graph pruning but, instead of inpainting missing audio fragments with realistic reproductions, our focus is on interactive control of real-time playback. With respect to prior work, a particularly novel aspect of our approach is the use of spectral clustering of the graph to identify regions of similar sounds. This allows random navigation of the similarity graph to provide some variation, while retaining some stability in the qualities of the resulting texture.

Our approach is also similar in spirit to the algorithm in [10], in that it allows extending the stationary part of sounds, although here we use a time-frequency concatenative approach instead of convolution with noise.

3. ANALYSIS

The proposed framework is based on time-frequency analysis using the short-time Fourier transform (STFT). We assume a preliminary analysis step resulting in a static data structure, which is used during real-time playback. From the STFT frames we extract a lower dimensional perceptual representation that is used to construct a self-similarity matrix (SSM). From this matrix we can extract some basic functions, such as an onset detection function and the beat spectrum. The SSM is then thresholded into a recurrence plot, which is interpreted as the adjacency matrix of the similarity graph.

3.1. Feature extraction

The STFT of the signal is given by

$$X(m,k) = \sum_{n=0}^{N-1} x(n+mH)w(n)e^{-j\frac{2\pi kn}{N}} \quad (1)$$

where n is the sample index, m is the spectral frame index, H is the hop size, k is the frequency bin index, and w is a window function, such as the Hann window. In order to compute the similarity between spectral frames we need a low-dimensional representation that relates to human perception. While many descriptors have been used in the concatenative synthesis literature to encode different perceptual features we are interested in a general representation that can be used to assess the similarity between audio spectra regardless of their content. We use the Mel filterbank, which is one of the most widely used representations for audio:

$$M(m,f) = \sum_{f=0}^{F-1} M_{fb}(k,f)|X(m,k)|, \quad (2)$$

where M_{fb} is a matrix of F triangular filters scaled along the Mel frequency scale. The number of filters can be tuned to the required resolution (along with the size of the FFT window) in the analysis stage, depending on the sound. Figure 1 shows the Mel spectrogram of a drum loop which is used throughout this section.

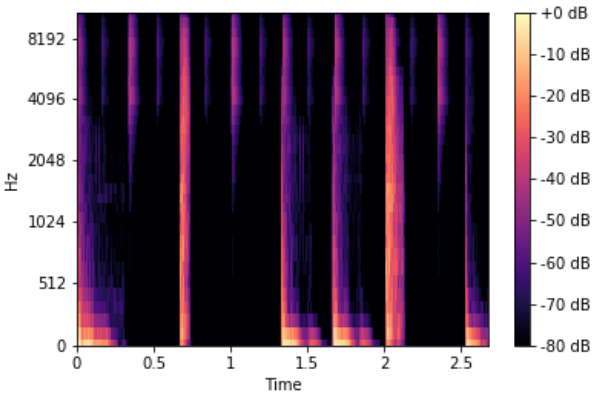

Figure 1: *Mel spectrogram of a drum loop.*

3.2. Self-similarity matrix

In order to compute the similarity between two frames, M_i and M_j, in the Mel spectrogram, we use the Jensen-Shannon (JS) distance:

$$D_{JS}(M_i, M_j) = (\frac{1}{2}D_{KL}(\hat{M}_i||\hat{M}_k) + \frac{1}{2}D_{KL}(\hat{M}_j||\hat{M}_k))^{\frac{1}{2}}, \quad (3)$$

where

$$\hat{M}_k = \frac{1}{2}(\hat{M}_i + \hat{M}_j), \quad (4)$$

$$\hat{M}_x = \frac{M_x}{\sum M_x}, \quad (5)$$

and D_{KL} is the Kullback-Leibler (KL) divergence:

$$D_{KL}(P||Q) = \sum P(x)log(P(x)/Q(x)) \quad (6)$$

The KL divergence is frequently used for audio features. In particular, it was found to perform well in early concatenative synthesis experiments [11]. The JS distance provides a symmetric version with all the properties of a metric, and is conveniently bounded between 0 and 1 [12]. In initial experiments, we found this distance resulted in similar visual patterns as the cosine distance used in [13]. Like the cosine distance, it is normalised with respect to the magnitude of each frame (here in order to represent a probability distribution). At the same time, following subjective assessment, we found it would give better perceptual results when used with a threshold to allow concatenation of frames from different locations with low distance.

The similarity between two spectral frames is then computed as:

$$SSM(i,j) = 1 - D_{JS}(M_i, M_j). \quad (7)$$

Eq. 7 defines a self-similarity matrix that shows some patterns in the sound [13]. An example is shown in Figure 2.

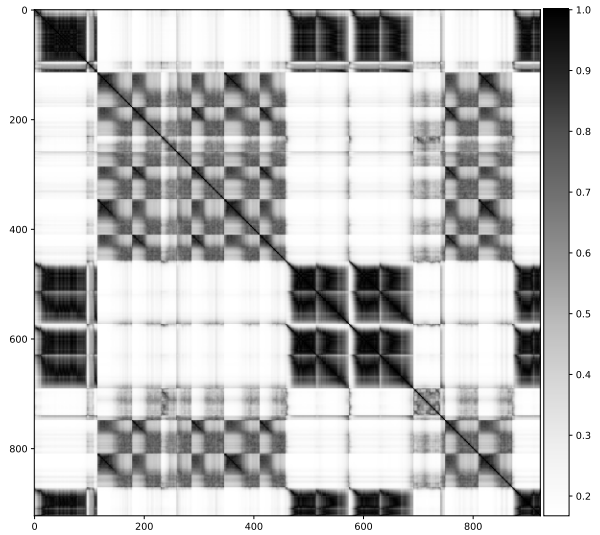

Figure 2: *Self-similarity matrix.*

3.3. Useful measures from the SSM

The main diagonal of the SSM is simply the similarity of each frame to itself. The superdiagonal represents the similarity of each frame to the next. Thus, it can be used to obtain an onset detection function $ODF(i) = 1 - SSM(i, i+1)$. The advantage of using the same distance measure throughout is that is that it is consistent with the rest of the framework and the threshold used for the similarity, which we use as concatenation cost. As such, onsets will normally signal a disruption in the connections corresponding to successive frames. However, since the distance is normalised, it can be noisy at low amplitude values. This is common in other onset detection functions [14]. We post-process the ODF by removing a median-filtered version and clipping it below zero. We also remove peaks that are closer than 100ms to a previous peak. Figure 3 shows an example of the post-processed ODF.

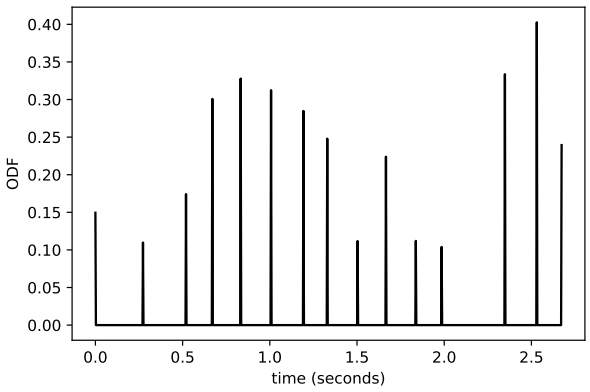

Figure 3: *Onset detection function from the SSM diagonal.*

Another useful measure that can be obtained from the SSM is the beat spectrum, defined as the sum of diagonals [13] (Figure 4):

$$BS(l) = \frac{1}{I-l} \sum_{i=0}^{I-l-1} SSM(i, i+l), \tag{8}$$

where I is the total number of frames. The sample at $BS(l)$ is the average similarity between frames separated by lag l. Thus, peaks represent periodicities in the original audio, which can be used to automatically find loop points that capture existing rhythms. In practice, the prominence of peaks depends on whether the audio is repetitive or not. For rhythmic audio, the peaks are very clear and are often found at durations that are multiples of the same basic beat. For short recordings we pick the earliest of the k highest peaks of the first half of the beat spectrum (typically with a value of $k = 3$), in order to obtain a small quantisation beat.

3.4. Recurrence plot

Finally, the SSM is thresholded to obtain the recurrence plot (RP) (Figure 5):

$$RP(i,j) = \left\{ \begin{array}{ll} 1, & \text{if } SSM(i,j) \geq \epsilon \\ 0, & \text{otherwise} \end{array} \right\}, \tag{9}$$

where ϵ is a threshold parameter. The dots in RP define combinations of spectral frames with high similarity. The recurrence

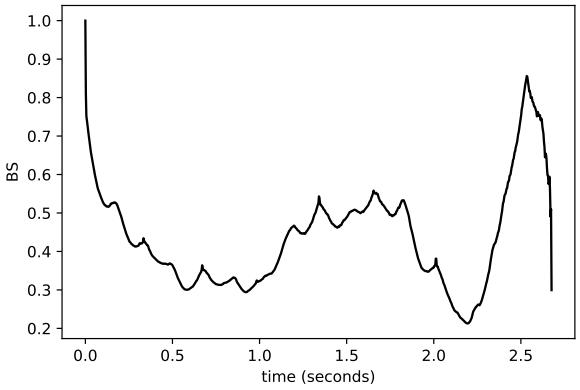

Figure 4: *Beat spectrum.*

plot can also be seen as the adjacency matrix of a similarity graph, which we also use to define potential transitions for playback. A circular graph layout such as in Figure 6 provides a more intuitive representation of the graph as a transition network (here a high threshold was used to generate fewer links for illustration purposes). Nodes in the graph represent spectral frames. The circular layout corresponds to the original order (thus, here, implying a general loop). The rest of the links can be thought as 'wormholes', or shortcuts, which provide potential alternative playback paths. The graph could also be defined through a nearest neighbours algorithm. However, here the parameter ϵ can be seen as a constant bound for the transition cost and is also a useful tradeoff: a higher value will create a sparser RP with fewer transitions. A low threshold will allow more transitions but with higher potential for perceived discontinuity.

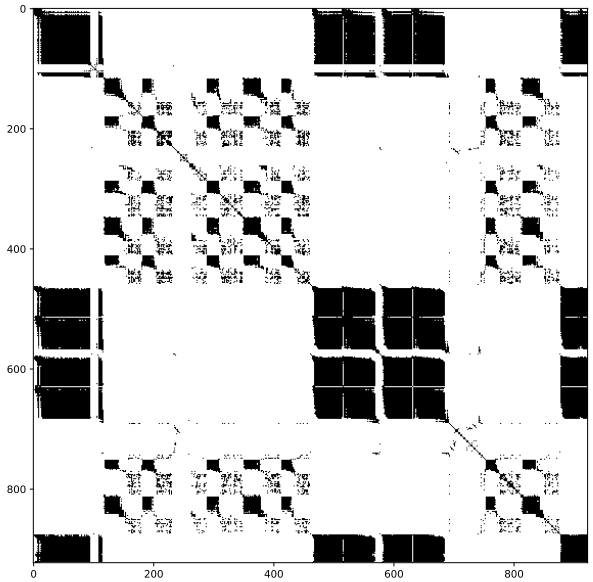

Figure 5: *Recurrence plot.*

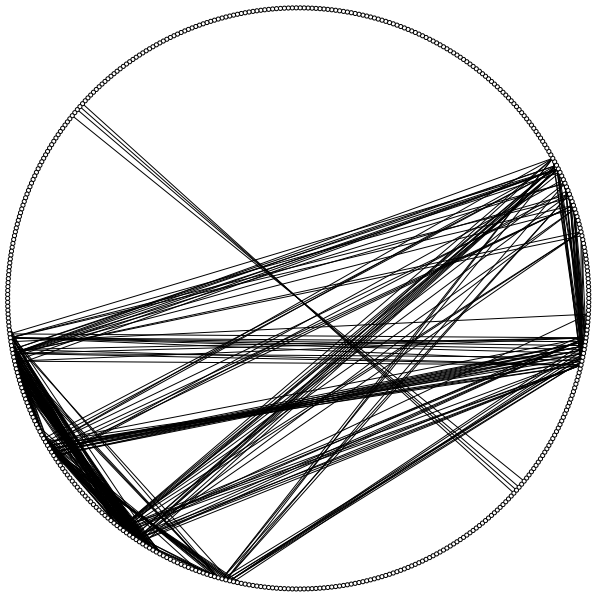

Figure 6: *Circular view of the graph as a transition network.*

3.5. Spectral clustering

The graph obtained in the previous section can also be used to cluster the frames via spectral clustering [15]. In order to preserve the information about similarities, we use a weighted version of the adjacency matrix RP:

$$\hat{RP}(i,j) = RP(i,j) \odot SSM(i,j), \tag{10}$$

where $\odot$ is the element-wise product.

The weights in $\hat{RP}$ are the similarities of each frame with the others above the given threshold. The degree of frame i is the sum of the weights,

$$d_i = \sum_{j=0}^{L-1} \hat{RP}(i,j), \tag{11}$$

The symmetric normalized laplacian is then

$$L_{sym} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}, \tag{12}$$

where D is the diagonal matrix formed with the degrees of the frames $d_0 ... d_{I-1}$ along the diagonal. We compute the first l eigenvectors and eigenvalues of L_{sym} (where l is the maximum number of clusters). The number of clusters c can be specified manually as a parameter, or automatically estimated by the *eigengap* method [15], by looking at the largest gap between consecutive eigenvalues up to l. The clusters are then obtained by running the k-means algorithm for the first c eigenvectors (here used as features for each of the spectral frames and row-normalised) [16].

4. GRAPH-BASED AUDIO PLAYBACK

As described in the previous sections, in this paper we propose using the similarity graph as a general mechanism for audio playback. The general principle is to move a playback head over spectral frames, selected by traversing the graph, which are then synthesised via the inverse STFT. Depending on the sound and the chosen threshold, there may still be many links. Several algorithms are possible based on different ways of pruning the graph. We provide three examples focusing on looping, granulation and automatic montage. It is worth noting that pruning can be simply implemented by removing rows and columns in $\hat{RP}$. Given its simplicity, this process could eventually be presented as a user interface.

The three algorithms are implemented as externals for the Max and SuperCollider languages, using the Fluid Corpus Manipulation Toolbox[17, 18]. The implementation can be obtained from github.[1]

All three algorithms include an initial analysis step in which the Mel spectrogram and similarity graph are computed. The graph is then pruned using different strategies for each object. After the analysis, the playback guided by the graph can be controlled in real time. We now describe each of the algorithms in more detail.

4.1. Looping

Looping is used in many musical genres, often based on repetition and rhythm. The choice of a looping region may be influenced by existing periodicities in the audio, although it is also possible that there are none, or that new rhythms are created by the loop itself. The choice of the looping region can thus be seen as a dialogue between the user and the audio content. We propose to represent this dialogue as a search process: the user makes an initial query of loop start and end points, and the system proposes an alternative set of start and en points based on the audio content.

4.1.1. Analysis

In the analysis, phase, $\hat{RP}$ is computed using a threshold parameter.[2] A quantised mode is provided as an option that will use the earliest detected peak in the beat spectrum to remove links that are not multiples of the detected beat. Since this may end up with a very small number of links, all multiples of the beat that start in a frame labelled as an onset are added to $\hat{RP}$ in the quantised mode. Onsets are detected as peaks in the ODF defined in Section 3.3. We then fit a KD-Tree index [19] to the 2D points in the upper triangle of the symmetric matrix $\hat{RP}$ (Figure 5).

4.1.2. Playback

During playback the play head generally follows the original order of frames, and jumps at the loop positions. The user can specify a query point in real time. Every query point (l_{s0}, l_{e0}) is represented in the same 2D space as the indexed transitions, so the system returns the nearest neighbour (l_{s1}, l_{e1}). This can be useful for simplifying the selection of loop points for beginners, or for more sophisticated loop-based playback by modulating the query point.

4.2. Granulation

Granular synthesis is often used to create textures and tones, often using some random variation of parameters. Here, we are con-

[1] `https://github.com/flucoma/graph_loop_grain`

[2] Note that in all the implemented objects, the threshold parameter is defined over distance instead of similarity

strained to the STFT framework with respect to some of the grain parameters, but some variation is obtained by random navigation of the graph.

4.2.1. Analysis

For the granulation algorithm, the analysis phase includes spectral clustering of the graph as described in Section 3.5. The graph is then pruned to remove links between different clusters. The number of clusters thus controls the definition of the sound: if the number of clusters is small, the resulting texture will have higher variation. The number of clusters can be specified as a parameter, or an automatic value can be used via the *eigengap* method. In practice this method can end up returning too few clusters if the graph is fully connected, so we use the double of the value returned by the *eigengap* method as the choice for the automatic mode. In our implementation the maximum number of clusters is empirically set to 50, and clusters smaller than 10 frames are merged into the cluster with the nearest centroid.

In addition to clustering, we remove all links to onset frames (again using the ODF derived from the SSM) to promote continuity and avoid stuttering effects commonly found in granulation. Since the analysis is based on overlapping windows, and also because audio around onsets is often loud and spans a broad frequency range, we remove the links to the 2 frames before and after a frame labelled as an onset.

4.2.2. Playback

The main parameters for controlling the granulation are the starting point, which determines the selected cluster, and the threshold. Both are selected in real-time and thus $\hat{RP}$ is computed dynamically from the SSM. The navigation is generally driven by a sorted list of nearest neighbours with respect to the current frame and threshold. This is controlled by two parameters: 'randomness' and 'forgetfulness'. Randomness (0 to 1) defines the number of nearest neighbours as a fraction of the allowed transitions from the current frame. A value of 0 will always select the nearest neighbour. This will create a totally deterministic behaviour, which in general ends in a cycle of frames in the network. Choosing the nearest neighbour is always the smoothest transition. Thus, higher values of randomness will increase variation at the cost of more artefacts in the transitions. Forgetfulness specifies a duration during which already visited transitions are removed from the available transitions. In general, this parameter will encourage exploration of the cluster. When randomness is 0, forgetfulness determines the length of the deterministic loop. In order to reduce the artefacts from the concatenation of random frames, we added the option of synthesising the phase from the magnitude, using the real-time phase gradient heap integration (RTPGHI) algorithm [20].

4.3. Automatic montage

We describe a final algorithm based on a user-specified minimum duration parameter, similar to the MS algorithm in [8].

4.3.1. Analysis

For this algorithm no pruning is used in the analysis phase, which provides more freedom for experimenting with the content of the sound. The algorithm focuses on preserving the shapes of existing sound objects in the recording, so the playback head is allowed to cross onsets and cluster boundaries.

4.3.2. Playback

In real time, the playback head follows the original sequence of frames for a given duration, specified by a minimum duration parameter. After this it jumps randomly to a similar point in the graph. The threshold parameter controls the number of candidate points. As with the granulation algorithm, a 'forgetfulness' parameter controls the predictability of the walk by blacklisting visited transitions for a given duration, but here the effect is not so immediately noticeable. A minimum distance parameter is used to remove transitions to nearby frames, which may create freezing artefacts.

5. EXAMPLES

The different algorithms based on the similarity graph offer a variety of ways to extend sound recordings in time. We now describe some examples created using the implemented objects. The audio files for these and further examples, can be found in the companion website for this paper[3].

In the case of looping, we noted that by implementing the operation in the STFT domain using overlap-add synthesis, and on the basis of similarity between the overlapped frames, there are generally no issues with wave discontinuity for musical audio. In addition, by finding areas with similar spectrum at different points in time, the algorithm tends to find natural-sounding phrases. An example is shown in Figure 7. Here, a rough guess was made for a loop at 10% and 20% of the duration of an excerpt of a syncopated drums performance, marked in yellow. This object generates the candidate loop points in the analysis phase and then it can be queried at any time during playback. In this case, the object replied with the points marked in green: a similarity is found in the onsets of two bass drum sounds, which determines a short bass drum and snare phrase. It is worth noting that this was found without using the quantisation option described in Sec. 4.1.1.

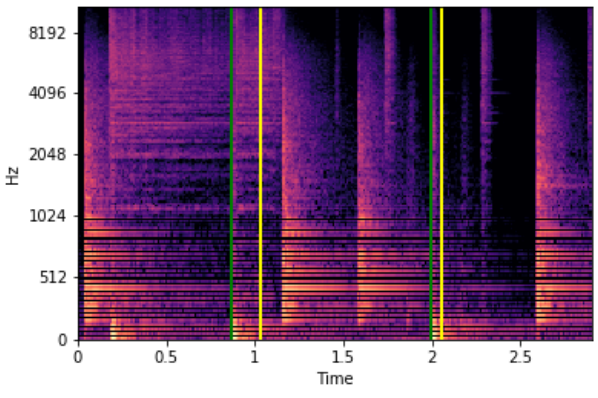

Figure 7: *Looping example.*

With respect to the granulation object, we noted it shares the characteristic sound and artefacts of audio granulation techniques.

[3] https://www.flucoma.org/DAFX-2021/

These were considerably reduced by using high overlap factors (e.g. 4 or higher), and, when using tonal source material, using the RTPGHI phase synthesis. For noisy sources, using phase synthesis will still tend to produce periodicities, so for realistic results it is better to turn it off. With respect to existing granulation techniques, this algorithm offers several particularities. One is the possibility of creating deterministic loops, which can be used to design new sounds. Another one is the ability to 'wander' by using the similarity network. This allows obtaining slowly varying textures. Finally, unlike traditional granulators, this algorithm can create stable textures while making use of different parts of the sound. The stability of the sound is controlled by the number of clusters in the analysis stage. A small number will produce clusters with different pitches and timbres, and wandering behaviour, while a large number will produce small clusters and stable textures and tones. An example is shown in Figure 8. A recording of a music box melody was used as source material with a large (around 100ms) grain duration. The clustering matched parts of the sound with similar spectra (the selected cluster is highlighted in white), and the algorithm produced a stable tone by concatenating similar frames through the random walk and synthesizing the phase (Figure 9).

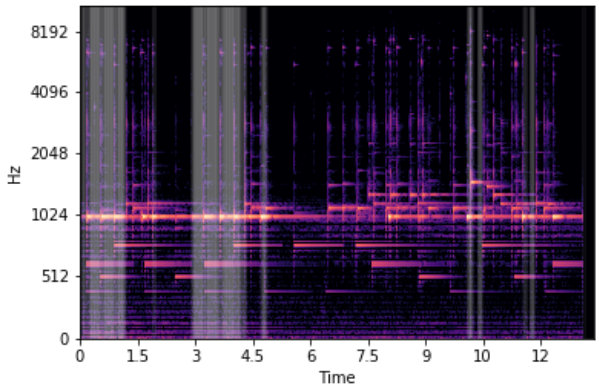

Figure 8: *Granulation source material (recording of music box)*

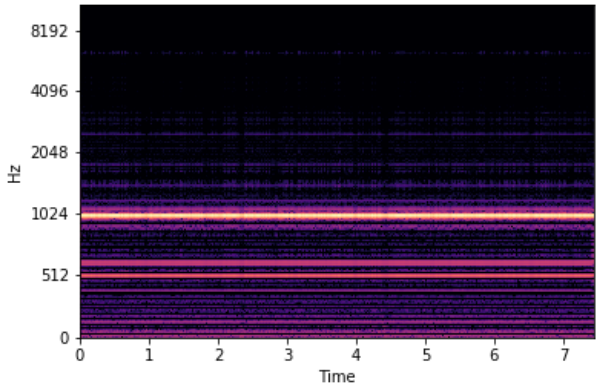

Figure 9: *Granulation example*

Trying the montage object, we found it could be useful both for articulating gestures found in recordings for music and sound design, and for synthesizing more realistic textures, particularly resulting from aggregation of random events. Figures 10 and 11 show an example where a short recording of an applause is extended in time, again using 100ms windows.

In all objects, low values (around 0.2) of the threshold (defined over distance) provided a good compromise for a sufficient number of seamless transitions. This also depends on the choice of the number of Mel bands. A high threshold value will result in more noticeable transition artefacts, while a low value can result in freezing effects created by a repetition of short sequences. In the montage algorithm, this can prevented by the minimum distance parameter, while in the granulation algorithm, low values can still be used to produce artificial sounds.

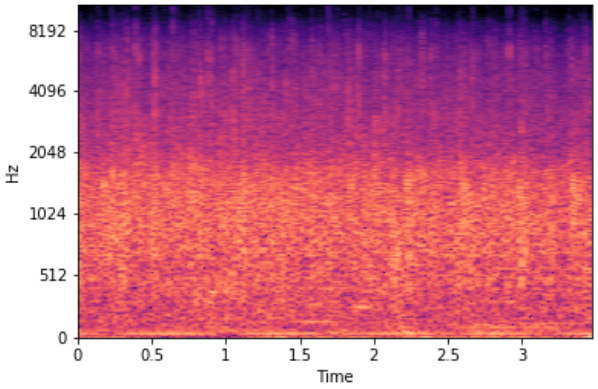

Figure 10: *Applause recording fragment*

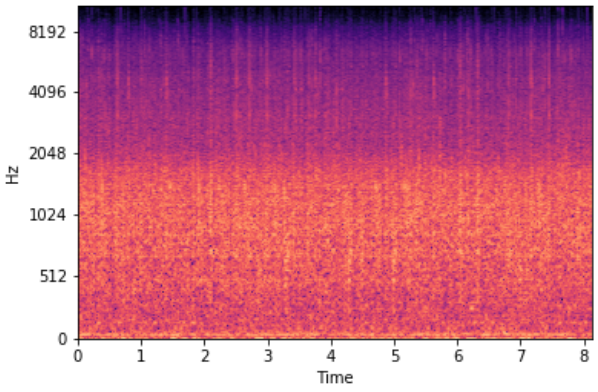

Figure 11: *Texture generated using the montage object with applause*

6. CONCLUSIONS AND FUTURE WORK

Alternative playback mechanisms for short audio excerpts are generally useful for music and sound design. In this paper, we have

explored audio similarity graphs as a guide for devising different algorithms for synthesis and playback in the time-frequency domain. This is an intuitive model that can be used to produce novel interfaces for a variety of tasks. We have shown algorithms for automatic granulation, looping and montage. For the first case, synthesising the phase from the real-time concatenated sequence of magnitude frames has proven useful for creating continuous pitched sounds based on existing material. Noisy textures can also be synthesised with both the granulation and the montage algorithm. When using longer sequences of the original sound (in the looping and montage algorithms), the graph helps creating seamless transitions, while leveraging existing patterns and gestures in the source audio.

One limitation of the proposed framework is the cost of the JS distance. In this paper, our focus is on extending short snippets, but for longer recordings computing the full SSM using the JS distance is unfeasible. This could be improved by computing an initial guess of the graph with a fast nearest neighbours algorithm as in [9]. In general, the effect of the distance measure, along with other aspects such as the number of clusters in the granulation algorithm, could benefit from formal evaluation through listening tests.

Many other algorithms could be devised on the basis of different strategies for pruning the similarity graph, as well as biasing random or deterministic navigation. In future work, we plan to investigate more open user interfaces that allow musicians and sound designers to develop their own playback sequences and algorithms in musical creative coding environments.

7. ACKNOWLEDGMENTS

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement no. 725899).

8. REFERENCES

[1] Curtis Roads, *Microsound*, MIT press, 2004.

[2] Aymeric Zils and François Pachet, "Musical Mosaicing," in *Proceedings of the 2001 Conference on Digital Audio Effects (DaFx)*, 2001.

[3] Diemo Schwarz, Grégory Beller, Bruno Verbrugghe, and Sam Britton, "Real-Time Corpus-Based Concatenative Synthesis with CataRT," in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, 2006.

[4] Diemo Schwarz, "Corpus-based concatenative synthesis," *IEEE signal processing magazine*, vol. 24, no. 2, pp. 92–104, 2007.

[5] Bee Suan Ong and Sebastian Streich, "Music loop extraction from digital audio signals," in *2008 IEEE International Conference on Multimedia and Expo*. IEEE, 2008, pp. 681–684.

[6] Zhengshan Shi and Gautham J Mysore, "Loopmaker: Automatic creation of music loops from pre-recorded music," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 2018, pp. 1–6.

[7] Diemo Schwarz, Axel Roebel, Chunghsin Yeh, and Amaury LaBurthe, "Concatenative sound texture synthesis methods and evaluation," in *19th International Conference on Digital Audio Effects (DAFx-16)*. Brno University of Technology, Faculty of Electrical Engineering and Communication, 2016, pp. 217–224.

[8] Seán O'Leary and Axel Röbel, "A montage approach to sound texture synthesis," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 6, pp. 1094–1105, 2016.

[9] Nathanael Perraudin, Nicki Holighaus, Piotr Majdak, and Peter Balazs, "Inpainting of long audio segments with similarity graphs," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 6, pp. 1083–1094, 2018.

[10] Vesa Välimäki, Jussi Rämö, and Fabián Esqueda, "Creating endless sounds," in *Proc. 21st Int. Conf. Digital Audio Effects (DAFx-18), Aveiro, Portugal*, 2018, pp. 32–39.

[11] Esther Klabbers and Raymond Veldhuis, "On the reduction of concatenation artefacts in diphone synthesis," in *Fifth International Conference on Spoken Language Processing*, 1998.

[12] Jianhua Lin, "Divergence measures based on the shannon entropy," *IEEE Transactions on Information theory*, vol. 37, no. 1, pp. 145–151, 1991.

[13] Jonathan Foote and Shingo Uchihashi, "The beat spectrum: A new approach to rhythm analysis," in *Proceedings of the IEEE International Conference on Multimedia and Expo, 2001.*, 2001.

[14] Juan Pablo Bello, Laurent Daudet, Samer Abdallah, Chris Duxbury, Mike Davies, and Mark B Sandler, "A tutorial on onset detection in music signals," *IEEE Transactions on speech and audio processing*, vol. 13, no. 5, pp. 1035–1047, 2005.

[15] Ulrike Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.

[16] Andrew Ng, Michael Jordan, and Yair Weiss, "On spectral clustering: Analysis and an algorithm," *Advances in neural information processing systems*, vol. 14, pp. 849–856, 2001.

[17] Pierre Alexandre Tremblay, Owen Green, Gerard Roma, and Alex Harker, "From collections to corpora: Exploring sounds through fluid decomposition," in *Proceedings of the International Conference on Computer Music (ICMC)*, 2019.

[18] Pierre Alexandre Tremblay, Gerard Roma, and Owen Green, "Digging it: Programmatic data mining as musicking," in *Proceedings of the International Conference on Computer Music (ICMC)*, 2021.

[19] Jon Louis Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[20] Zdeněk Průša and Peter L. Søndergaard, "Real-Time Spectrogram Inversion Using Phase Gradient Heap Integration," in *Proceedings of the 2016 International Conference on Digital Audio Effects (DAFx-16)*, 2016, pp. 17–21.