Groove Explorer: An Intelligent Visual Interface for Drum Loop Library Navigation

Fred Bruford
Centre for Digital Music,
Queen Mary University of London
London, UK
f.w.c.bruford@qmul.ac.uk

Mathieu Barthet
Centre for Digital Music
Queen Mary University of London
London, UK
m.barthet@qmul.ac.uk

SKoT McDonald
ROLI Ltd.
London, UK
skot.mcdonald@roli.com

Mark Sandler
Centre for Digital Music
Queen Mary University of London
London, UK
m.sandler@qmul.ac.uk

ABSTRACT

Music producers nowadays rely on increasingly large libraries of loops, samples and virtual instrument sounds as part of the composition process. Intelligent interfaces are therefore useful in enabling navigation of these databases in a way that supports the production workflow. Within virtual drumming software, producers typically rely on large libraries of symbolic drum loops. Due to their large size, navigating and exploring these libraries can be a difficult process. To address this, preliminary work is presented into the *Groove Explorer*. Using Self-Organizing Maps, a large library of symbolic drum loops is automatically mapped on a 2D space according to rhythmic similarity. This space can then be explored via a Max/MSP prototype interface. Early results suggest that while the algorithm works well for smaller datasets, further development is required, particularly in the similarity metric used, to make the tool scalable to large libraries.

CCS CONCEPTS

• **Applied computing** → **Sound and music computing**; • **Information systems** → *Clustering and classification*; • **Computing methodologies** → Machine learning.

KEYWORDS

Music Information Retrieval; Search; Visualization; Self-Organizing Maps; Drum Loops

ACM Reference Format:
Fred Bruford, Mathieu Barthet, SKoT McDonald, and Mark Sandler. 2019. *Groove Explorer*: An Intelligent Visual Interface for Drum Loop Library Navigation. In *Joint Proceedings of the ACM IUI 2019 Workshops, Los Angeles, USA, March 20, 2019* , 4 pages.

1 INTRODUCTION

Since the early days of Music Information Retrieval (MIR) as a research field, one significant topic of research has been in the intelligent visual organization of music data as a means of managing and understanding the increasingly large quantities of data arising from advances in music content digitization [1]. Within digital audio workstation (DAW)-based music production environments, the searching and mining of music databases for production content is a common task. Producers frequently make use of assembled libraries of various different types of data, such as samples, loops, synthesizer presents and virtual instruments. Exploring and navigating these libraries can increasingly become a bottleneck for producers due to their large size, and often inadequate searching functions.

An example of this problem is found within the symbolic drum loop libraries used in virtual drumming software. Software emulations of drum kits are finding increasing use in music production for generating realistic, expressive drum kit parts 'in the box', without requiring recording equipment, a drum kit or a drummer. Examples of this kind of software include *SuperiorDrummer* [14], *Steven Slate Drums* [13] and *FXpansion*'s *BFD3*, upon which this work is based [3].

These pieces of software work by combining a drum synthesis engine with a pattern library to sequence the sounds into realistic, idiomatic and expressive drumming. In BFD there are roughly 2500 of these sequences, referred to as Ѐrooves; within the Core library, but thousands more in third-party addons and expansion packs. Unfortunately, this library can be very difficult to navigate; as well as being very large, it uses limited metadata and names which can be difficult to interpret.

To address this problem, we present ongoing development of the *Groove Explorer*, a prototype tool for visual exploration of this symbolic drum loop library. Using Self-Organizing Maps (SOMs) [8], the Grooves within the library are automatically mapped across a 2D space according to rhythmic similarity. A user can then explore this 2D loop space, with the goal of making the process of discovering and choosing loops quicker, easier and more enjoyable.

First, related work that inspired this project will be discussed. Then the details of the mapping algorithm will be described, along with implementation details of the prototype tool. Then, the results of an initial evaluation on the algorithm's output mapping quality will be presented. Finally, conclusions will be made considering the limitations of the algorithm and evaluation methodology, and an outline of further research will be provided.

2 RELATED WORK

A growing body of work now attempts to use MIR to address the limitations of DAW-based music production. The development of intelligent visual navigation tools for the many types of data libraries used in music production is a significant area of this research. By automatically sorting music data according to similarity, it is suggested that visualizations can make it quicker, easier and more enjoyable to find music data [12].

A particular focus has been in improving the navigation of drum sample libraries. In [7], Shier et al. apply principal component analysis (PCA) to a set of timbral features extracted from kick, snare and hi-hat samples to map the samples onto a 2D grid representing two dimensions of similarity. The same problem is tackled in [16] by Turquois et al, instead using t-distributed stochastic neighbor embedding (t-SNE), and in [2], where Fried et al. employ embedded kernelized sorting to perform the mapping. They additionally developed a tool for exploring synthesizer sounds in a similar way [2]. For sample navigation, they found that users could find samples quicker using the intelligent mapping versus traditional folder diving.

There has similarly been research specifically into the exploration of loops. In [11], authors used force-directed graphs to support exploration of the *freesound.org* library. For drum loops specifically, the concept of rhythm spaces has been proposed to explore generative maps of drum rhythms for electronic dance music [6]. The same authors have also begun to address the challenges of implementing such a system, generating a 2D map of 9 drum loops based on a number of rhythmic descriptors and multi-dimensional scaling [4].

3 ALGORITHM

The core mapping algorithm used in the *Groove Explorer* is the Self-Organizing Map (SOM). The SOM, first developed by Teuvo Kohonen [8] is a nonlinear dimensionality reduction technique, designed primarily for the visualization of high-dimensional data. Essentially a neural network, it is usually organized as a 2D grid of nodes, each with its own weight vector. SOMs differ from most other neural network architectures in that they utilize competitive learning. After the nodes have been randomly initialized, for each input vector the node with the closest matching weight vector is calculated and deemed to be the 'winner'. The winning node and surrounding neurons are then updated and moved closer to the input vector. The update rule updates the nodes by decreasing amounts at larger distances from the winner, with the winner node being changed the most.

In addition to the radius function, a learning rate reduces the amount of updating over time, ensuring the map converges over its training period. To calculate similarity between input vectors and node vectors, a Euclidean distance function is typically used, although any similarity metric can be used. Once the map has been trained, the final map can be generated of the winner node position for each input data item. As each data item is mapped onto a node, to capture the topological structure of the input data, the number of neurons must be large compared to the number of input data items.

It is possible to perform additional clustering operations on the output map to aid visualization. The U-matrix is a popular visual clustering method for the SOM that shows distances between neighboring node vectors as a heatmap [17]. Areas with high levels of similarity are clusters, and shown on the output heatmap as dark areas. The map of each data item on its respective winner node can then be overlaid on the U-matrix.

3.1 Feature design

To feed into the SOM, raw symbolic drum loops were represented as 160-point feature vectors. The input loops were all 2 bars in length, reduced to 5-part polyphony, and quantized to semiquaver time steps (16 per bar), with velocities for each hit. Each time step for each part was therefore a point on the feature vector. All of the loops were in 4/4 time. While the raw loops used 10-part polyphony, some of the parts were grouped by functional and timbral similarity, as shown in **Table 1**.

Table 1: Kit piece grouping for feature construction

Group	Drum kit parts
1	Kick
2	Snare
3	Closed hi-hat, ride bell, cymbal bell, ride bow
4	Crash cymbal, open hi-hat
5	Hi tom, mid tom, low tom

Grouping kit pieces by functionality and similarity was done partly to reduce computational intensity, but also to make the similarity function more accurate. By grouping kit pieces, rhythms in similar kit parts could be compared against each other, instead of just considering each instrument individually.

3.2 Similarity metric

The calculation of the SOM relies on a similarity metric to calculate the difference between individual Groove vectors and nodes. Ensuring this similarity metric is musically and perceptually valid is crucial to ensure the output mapping correlates with the user's perception of similarity between drum loops in the space.

While the edit or swap distance is considered by some to be the most robust metric for calculating monophonic rhythm similarity [15], recent work has indicated coincidence or Hamming distance-based metrics to correlate closer to human similarity perception in the specific case of polyphonic electronic drum loops [5, 9, 18]. The Hamming distance counts the number of corresponding positions in two rhythm vectors where the value matches. In the case of drum loops, the values at each beat position are binary, with a 1 for a hit and 0 silence for a given drum part at a given time step. It then takes an average over the vectors' length.

BFD's Grooves also have velocities for each hit, ranging between 0 for silence, and 1 for the loudest hit. To account for continuous velocity values, a Euclidean vector distance function is an appropriate technique that is analogous to a coincidence-based metric like Hamming distance in that it computes a measure of the combined distance between corresponding points in the vectors.

4 IMPLEMENTATION

The code for all feature construction and SOM computation was written in Python 2.7, and can be accessed at https://github.com/fredbru/musicsom. For training the SOM, scripts for both CPU and GPU computation were written for added flexibility, but GPU computation was used for the large maps due to increased speed. The GPU SOM code was written using CUDA via the Numba just-in-time compiler Python library [10]. Using an NVIDIA Tesla P100 GPU, a performance increase of around 10x was achieved in training an 80x60 map. It is expected that by employing a cluster of GPUs, the training process could be sped up much more as a further parallelization could be achieved.

With non-4/4 Grooves removed, the BFD Core dataset contains approximately 2300 Grooves. The Grooves are structured and tagged in terms of 12 genres and 59 'Palettes'. Palettes represent sub-groups of loop style, each containing around 20-40 similar Grooves, with names such as 'Smooth Jazz' and 'Trash Metal'. Each Groove in one Palette is therefore of the same genre.

4.1 User interface

For interactively navigating the Groove map, the prototype Groove Explorer was designed in Max/MSP, the UI of which is shown in **Figure 1**. The Max/MSP patch loads the SOM output maps as JSON files of Groove metadata and map coordinates. Each circular object represents a Groove, which is played back when clicked. The user can zoom in and pan across the map, and save Grooves in their own custom bank within BFD. It is possible to switch between coloring Grooves according to genre and according to Palette.

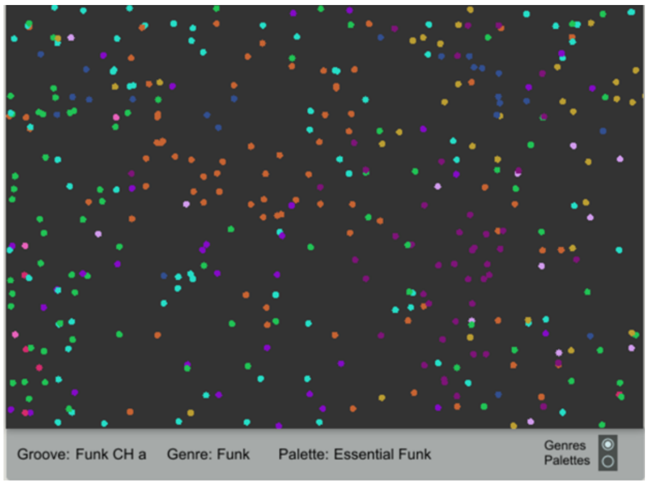

Figure 1: *Groove Explorer* Prototype UI

The Max UI communicates with BFD with a sufficiently low latency for real-time use in a DAW. To load and play back Grooves, the names of selected Grooves are sent from Max to BFD via a TCP/IP socket to a Python background script. This script then parses these messages to generate commands in the right format for a Lua interface used for BFD scripting. Important features of BFD, such as its drum sample playback engine and audio mixing features can therefore be accessed alongside the *Groove Explorer*. Additionally, Python integration opens up the future possibility of real-time updating of the SOM from the UI.

5 EVALUATION

A pilot evaluation of the mapping quality of the Groove Explorer was carried out using Palette and genre tags provided in the software as a benchmark. To assess the ability of the SOM to separate Palettes, the subgenre groupings in the dataset, a smaller map was first generated using a group of 8 Palettes, one for each major genre, for a total of 252 Grooves. To test the ability of the Groove Explorer to represent the structure of the data on a large scale, a map of the whole BFD Core Groove library was generated, with genre tags used to offer a higher-level indicator of similarity correlating with human categorization. The small SOM dimensions were 16x16 nodes and the larger 80x60. The small map was trained on the CPU of a Dell XPS-15 9550 laptop running Ubuntu 18.10, for a training time of 30 minutes, or 2000 epochs. Due to its much larger size, the larger map was trained using a NVIDIA Tesla P100 GPU, for a training time of 3 days, or 3500 epochs.

The SOM output was plotted as the winner node positions for each Groove overlaid over the U-matrix. That way as well as observing the map positions of groove, it was possible to evaluate the ability of the algorithm to perform clustering.

6 RESULTS AND DISCUSSION

In the first map, shown in **Figure 2**, the algorithm is able to separate most of the Palettes into separate parts of the map, showing that it is successfully grouping many of the Grooves according to similarity. The Pop V3, Jungle V1, Peter Erskine Rock, Heavy Metal and Funk V3 Palettes are all in their own clear groups, with fairly few outliers.

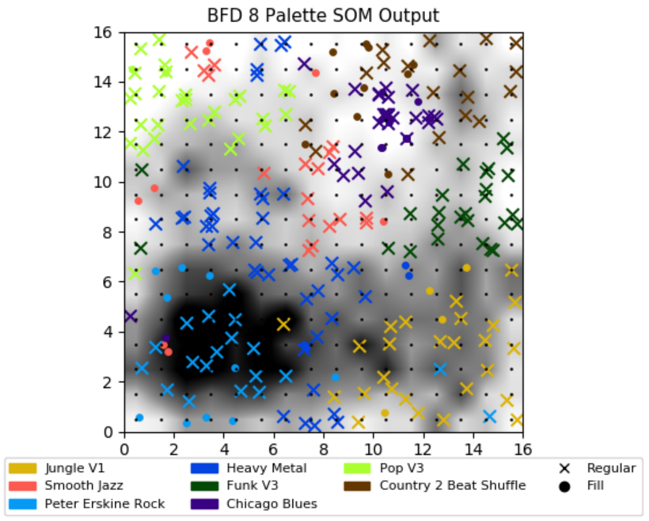

Figure 2: 8 Palette 16x16 SOM Output with U-matrix

However, some Palettes are less well separated from others 'Smooth Jazz' in particular. Similarly, the Country and Blues Palettes are plotted on top of each other, although this makes some sense as they are similar genres. While the mapping makes topological sense, the U-matrix is not able to cluster entirely successfully, other than the Rock Palettes in the bottom-left of the map. This could be the result of a high level of variation within the input features making clusters unclear.

Although the mapping corresponds quite well to the tagging in the smaller SOM, the larger map, shown in **Figure 3**, did not clearly separate genre groups, and did not show any real clusters in the U-matrix. Grooves within the same Palette are clustered together still, but Palettes themselves are not really grouped with other Palettes of the same genre. This could indicate a limitation in the similarity function, as it is unable to take account of the higher-level stylistic characteristics of drum loops that motivate human categorization. It could suggest limitations in using factory-supplied genre meta-tags as ground-truth categories for evaluation, due to the high amount of variance within genres, crossover between genres, and possible inconsistencies in tagging. For an evaluation truly representative of the practical quality of the map, a user study is required.

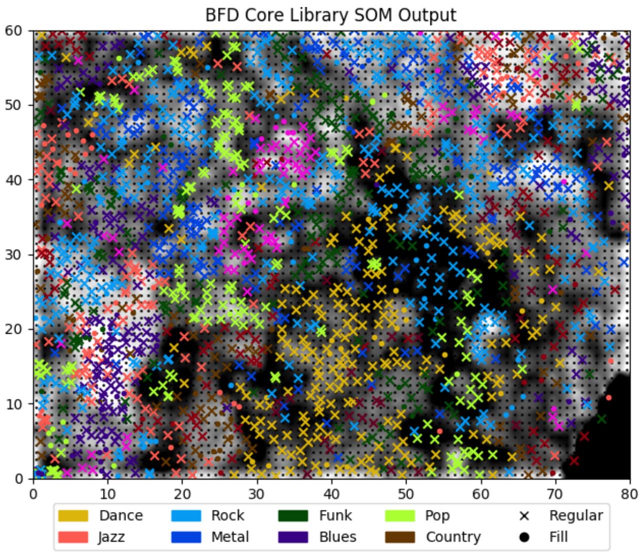

Figure 3: BFD Core Library SOM with U-matrix

7 CONCLUSIONS AND FURTHER WORK

As a work in progress, the Groove Explorer represents a positive step towards easier navigation of drum loop libraries. The current visualization approach is capable of arranging and separating loops by stylistic categories in the case of smaller data libraries. However, for datasets spanning a larger range of styles and genres, the system is not yet able to convincingly express the higher-level genre organization. This could indicate a limitation of the SOM, or limitations in the use of genre tags as a ground truth. Alternatively, the issue could be in the similarity metric; metrics based purely on rhythm similarity may be unable to pick up stylistic similarities of expressive drum loops. Further investigation is required into other similarity metrics, along with the development of alternative features for drum loop analysis. Investigation into different visualization techniques other than the SOM could also prove useful. Finally, a study should be carried out involving music producers to assess the usability of the interface and mapping.

REFERENCES

[1] Matthew Cooper, Jonathan Foote, Elias Pampalk, and George Tzanetakis. 2006. Visualization in Audio-Based Music Information Retrieval. *Computer Music Journal* 30, 2 (June 2006), 42–62.
[2] Ohad Fried, Zeyu Jin, Reid Oda, and Adam Finkelstein. 2014. AudioQuilt: 2D Arrangements of Audio Samples using Metric Learning and Kernelized Sorting.. In *NIME*. 281–286.
[3] FXpansion 2019. BFD3. Retrieved Feb 17, 2019 from https://www.fxpansion.com/products/bfd3/
[4] Daniel Gómez-Marín, Sergi Jordà, and Perfecto Hererra. 2017. Drum rhythm spaces: from global models to style-specific maps. In *CMMR*.
[5] Daniel Gómez-Marín, Sergi Jordà, and Perfecto Herrera. 2015. Pad and Sad: Two awareness-Weighted rhythmic similarity distances. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR).*
[6] Daniel Gómez-Marín, Sergi Jordà, and Perfecto Herrera. 2016. Rhythm Spaces. In *MUME*.
[7] Jordie Shier, Kirk McNally, and George Tzanetakis. 2017. Sieve: A plugin for the automatic classification and intelligent browsing of kick and snare samples. In *WIMP*.
[8] T. Kohonen. 1990. The self-organizing map. *Proc. IEEE* 78, 9 (Sept. 1990), 1464–1480.
[9] Cárthach Ó Nuanáin, Perfecto Herrera, and Sergi Jorda. 2015. Target-based rhythmic pattern generation and variation with genetic algorithms. In *Sound and Music Computing Conference*.
[10] Numba 2019. Numba: A High Performance Python Compiler. Retrieved Feb 17, 2019 from https://numba.pydata.org/
[11] Gerard Roma and Xavier Serra. 2015. Music performance by discovering community loops. In *Proceedings of the Web Audio Conference (WAC), Paris*.
[12] Markus Schedl. 2017. Intelligent User Interfaces for Social Music Discovery and Exploration of Large-scale Music Repositories. In *In Proceedings of the 2017 ACM Workshop on Theory-Informed User Modeling for Tailoring and Personalizing Interfaces*. 7–11.
[13] Steven Slate Drums 2019. Steven Slate Drums - World-Class Virtual Drum Instruments & Replacers. Retrieved Feb 17, 2019 from http://stevenslatedrums.com/
[14] Toontrack 2019. Superior Drummer 3. Retrieved Feb 17, 2019 from https://www.toontrack.com/product/superior-drummer-3/
[15] Godfried Toussaint. 2004. A Comparison of Rhythmic Similarity Measures. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*. 23.
[16] Chloé Turquois, Martin Hermant, Daniel Gómez-Marín, and Sergi Jord: 2016. Exploring the Benefits of 2D Visualizations for Drum Samples Retrieval. In *NIME*. 329–332.
[17] Juha Vesanto and Esa. Alhoniemi. 2000. Clustering of the self-organizing map. *IEEE Transactions on Neural Networks* 11, 3 (May 2000), 586–600.
[18] Richard Vogl, Matthias Leimeister, and Cárthach Ó Nuanáin et al. 2016. An Intelligent Interface for Drum Pattern Variation and Comparative Evaluation of Algorithms. *Journal of the Audio Engineering Society* 64, 7/8 (Aug. 2016), 503–513.