An Interactive Software Instrument for Real-time Rhythmic Concatenative Synthesis

Cárthach Ó Nuanáin
Music Technology Group
Universitat Pompeu Fabra
Barcelona, Spain
carthach.onuanain@upf.edu

Sergi Jordà
Music Technology Group
Universitat Pompeu Fabra
Barcelona, Spain
sergi.jorda@upf.edu

Perfecto Herrera
Music Technology Group
Universitat Pompeu Fabra
Barcelona, Spain
perfecto.herrera@upf.edu

ABSTRACT

In this paper we describe an approach for generating and visualising new rhythmic patterns from existing audio in real-time using concatenative synthesis. We introduce a graph-based model enabling novel visualisation and manipulation of new patterns that mimics the rhythmic and timbral character of an existing target *seed* pattern using a separate database of *palette* sounds. Our approach is described, reporting on those features that may be useful in describing units of sound related to rhythm and how they might then be projected into two-dimensional space for visualisation using reduction techniques and clustering. We conclude the paper with our qualitative appraisal of using the interface and outline scope for future work.

Author Keywords

NIME, MIR, software instruments, concatenative synthesis, visualisation

ACM Classification

H.5.5 [Information Interfaces and Presentation] Sound and Music Computing, H.5.2 [Information Interfaces and Presentation] User Interfaces, J.5 [Computer Applications] Arts and Humanities

1. INTRODUCTION

Sampling of existing audio is a cornerstone of electronic music composition and production. In rhythm-centric electronic music styles it is common practice to combine synthesis of drum sounds with samples of existing drum sounds or even complete loops such as breakbeats. Its origins lie in the hip-hop movement of the 1980s, where the typical arsenal of a producer was an Akai style MPC sampler and a drum machine such as the iconic Roland TR-808. As dance music emerged in the late 1980s and 1990s, sampling became more and more sophisticated, as exhibited by complex, intricate and frantic styles such as drum and bass and IDM (Intelligent Dance Music).

Concatenative synthesis is a sample-based form of sound synthesis, whereby existing snippets of sound are stitched together to form new ones according to some set of rules. It is closely related to granular synthesis, but differs in the order of scale of the length of the sounds that are used. Granular synthesis typically operates on the microscale with "grains" of lengths 20-200 ms [13], whereas concatenative synthesis makes use of samples of unit lengths more musically related, such as a note or a phrase. It is an inherently Music Information Retrieval (MIR) geared approach such that feature extraction of acoustic and musical descriptors (such as sprectral, energy and timbral features) are essential for analysing and sorting existing sounds then concatenating them to create new ones according to some predefined strategy.

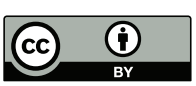
Concatenative synthesis, with its ability to redefine and adapt existing samples in a more automatic, assistive and intelligent manner, then seems ideally suited for these dance music applications of the sampling aesthetic. We propose and describe a graph-based visualisation and interaction system to generate sequences of rhythmic content that models the sonic character of an existing rhythmic loop based on its timbral and spectral profile. The system strives to break away from traditional interfaces such as the step sequencer, drum machines and linear timelines. The software itself is implemented as a multi-touch capable VST plugin and is intended to be playful and useful in exploring sound while easily integrated into existing musical applications and workflows.

Let us now turn to some state of the art examples relating to concatenative synthesis, or to use its alternative portmanteau, *musaicing*. We will examine them in turn and identify their key strengths and features in terms of musical operation and interaction but also raise some shortcomings as we see them.

2. EXISTING WORK

Concatenative synthesis has already seen many successful applications in the areas of speech synthesis [9], but one of the most concrete first applications in the domain of music has been the *CataRT* system for the Max/MSP environment by Schwarz [16]. Its defining characteristic is the 2D search space where the user selects which features are assigned to each axis and uses the mouse to explore the playback of sounds that have been previously organised as an "amorphous" collection. It is especially suited for sound design where a collection of sound files can be stochastically rearranged and made indecipherable from their original character.

Related to CataRT is the *EarGram* system available for the Pure Data environment [3]. It adds clustering strategies and additional novel visualisation possibilities such as the "InfiniteMode" which allows some clever mashups by means of a self-similarity matrix. Crucially it has a tempo source that facilities simple triggering and sequencing according to a chosen BPM and metre. Frisson et al. also report on their *AudioGarden* application for sound designers which also explores different ways of presenting concatenated sounds in



NIME'16, July 11-15, 2016, Griffith University, Brisbane, Australia.
.

2D space [7]. In their "Disc" mode for example, sounds are placed according to their radius/angle around a virtual circle. The logarithm of the duration of the sound file is mapped to the radius while the angle corresponds to some component of the timbre.

Contrasting with these academic, experimental prototypes, a commercial product suitable for the end producer/musician is LoopMash [1] by Steinberg, originally released as a software plugin but since expanded to mobile devices. Visually the operation of the instrument is quite intuitive. A target loop resides as a master track on the top of the screen with other loops underneath it. Mashups are generated by mixing portions of the other tracks depending on their similarity to the master.

While LoopMash is close to our aims in terms of providing a concatenative model that mimics a target, the interface is restrictive in a number of ways. Most fundamentally it does not offer the user any conceptual map or visual representation of the relationship between the units of sound. User selection and interaction is restricted to setting similarity "gain" levels and individual loudness levels of up to seven tracks. Conversely, the functionality offered in CataRT [2] and EarGram is more exploratory-oriented and don't offer much in the way of loop targeting capabilities or indeed integration with existing audio systems. Our system aims to combine these novel interface and interaction paradigms with intuitive operation and ease of integration.

3. SYSTEM DESCRIPTION

In this section we will describe our implementation of the system, beginning with an explanation of the musical analysis stages of onset detection, segmentation and feature extraction. This is followed by an examination of the interactive user interface and the the pattern generation process. Figure 1 gives a diagrammatic overview of these important stages, which can be briefly summarised as:

1. Sound Input
2. Onset Detection & Segmentation
3. Audio Feature Extraction
4. Storage & Data Representation
5. Similarity Computation & Pattern Generation
6. Real-time Audio Output

The system is developed in C++ using the JUCE framework [3], the Essentia musical analysis library [5] and the OpenCV computer vision library [6] (for matrix operations).

3.1 Sound Analysis

The first stage in building a concatenative music system generally involves gathering a database of sounds to select from during the synthesis procedure. This database can be manually assembled but in many musical cases the starting point is some user-provided audio that may range in length from individual notes to phrases to complete audio tracks.

The two inputs to the system are the *Sound Palette* and the *Seed Sound*. The sound palette refers to the pool of sound files we want to use as the sample library for generating our new sounds. The seed sound refers to the short loop that we wish to use as the similarity target for generating those sounds. The final output sound is a short (one to two bar) loop of concatenated audio that is rendered in real-time to the audio host.

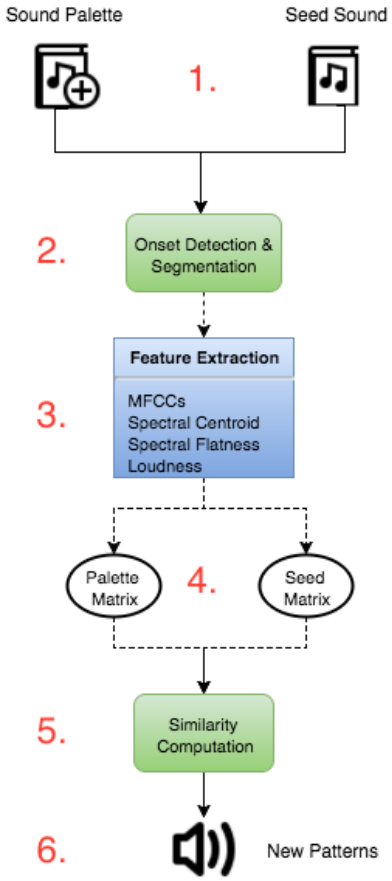


Figure 1: Block Diagram of Functionality

3.1.1 *Onset Detection and Segmentation*

In cases where the sounds destined for the sound palette exceed note or unit length, the audio needs to be separated into its constituent units using onset detection and segmentation.

Onset detection is a large topic of continuous study, and we would encourage the reader to examine the excellent review of methods summarised in [1]. Currently, with some tuning of the parameters, Sebastien Böck's Superflux algorithm represents one of the best performing state of the art detection methods [4]. For our purposes we have experienced good results with the standard onset detector available in Essentia, which uses two methods based on analysing signal spectra from frame to frame (at a rate of around 11 ms). The first method involves estimating the high-frequency content in each frame while the second method involves estimating the differences of phase and magnitude between each frames.

The onset detection process produces a list of onset times for each audio file, which we use to segment into new audio files corresponding to unit sounds for our concatenative database.

3.1.2 *Feature Extraction*

In music information retrieval systems, the task of deciding which features are used to represent musical and acoustic properties is a crucial one. It is a trade off in choosing

[1] steinberg.net/en/products/mobile_apps/loopmash.html
[2] The author does briefly suggest how loop-based concatenative synthesis might be achieved in another article [15] proposing the use of loop libraries and acoustic descriptors.
[3] http://www.juce.com

the richest set of features capable of describing the signal succinctly at the expense of storage and computational complexity. When dealing with musical signals specifically, there are a number of standard features that correspond roughly to certain perceptual ideals. We briefly describe our chosen features here, but for a more thorough treatment of feature selection with relation to percussion the reader should consult [8], [14] and [17].

Our first feature is the loudness of the signal, which is is implemented in Essentia according to Steven's Power Law, namely the energy of the signal raised to the power of 0.67 [5]. This is purported to be a more perceptually effective measure for human ears. Next we extract is the spectral centroid, which is defined as the weighted mean of the spectral bins extracted using the Fourier Transform. Each bin is weighted by its magnitude (Equation 1).

$$centroid = \frac{\sum_{n=0}^{N-1} k(n) f(n)}{\sum_{n=0}^{N-1} f(n)} \quad (1)$$

Perceptually speaking, the spectral centroid relates mostly to the impression of the brightness of a signal. In terms of percussive sounds, one would expect the energy of a kick drum to be more concentrated in the lower end of the spectrum and hence a lower centroid than that from a snare or crash cymbal.

Another useful single-valued spectral feature is the *spectral flatness*. It is defined as the geometric mean of the spectrum divided by the arithmetic mean of the spectrum (Equation 2). A spectral flatness value of 1.0 means the energy spectrum is flat whereas a value of 0.0 would suggest spikes in the spectrum indicating harmonic (with a specific frequency) tones. The value intuitively implies a discrimination between noisy or inharmonic signals and harmonic or more tonal signals. Kick drums sounds (especially those generated electronically) often comprise quite a discernible centre frequency whereas snares and cymbals are increasingly broadband in spectral energy.

$$flatness = \frac{\sqrt[N]{\prod_{n=0}^{N-1} F(n)}}{\frac{\sum_{n=1}^{N-1} F(n)}{N}} \quad (2)$$

Our final feature is the vector quality known as MFCCs (Mel Frequency Cepstrum Coefficients). MFCCs can be considered as a compact approximation of the spectra for the purposes of conveniently estimating the "timbre" of a signal. It has been applied extensively in speech processing, genre detection [18] and instrument identification [11]. MFCC computation, as outlined in [10], is basically achieved by computing the spectrum, mapping result into the more perceptually relevant Mel scale, taking the log then applying the Discrete Cosine Transform (Equation 3).

$$MFCC(x) = DCT(\log Mel(FFT(x)) \quad (3)$$

It is difficult to interpret exactly what each of the MFCC components mean, but the first component is generally regarded as encapsulating as the energy. Since we are already extracting the loudness using another measure we have discarded this component in our system.

3.1.3 Data Representation and Seed Computation

Further on in the paper we will describe a bit more on how the seed or target audio signal is actually received from the VST host, but in terms of analysis on that seed signal, the process is the same as before: onset detection and segmentation followed by feature extraction.

The resulting feature vectors are stored in two matrices: the *palette* matrix and the *target* matrix. The palette matrix stores the feature vectors of each unit of sound extracted from the sound palette and similarly the target matrix stores feature vectors of units of sound extracted from the seed loop.

3.2 Pattern Synthesis and User Interaction

This section details the visible, aural and interactive elements of the system as they pertain to the user. Figure 2 gives a glimpse of the user interface in a typical pattern generation scenario.

3.2.1 Workflow

The layout of the interface was the result of a number of iterations of testing with users who, while praising the novelty and sonic value of the instrument, sometimes expressed difficulty understanding the operation of the system. One of the main challenges faced was how best to present to the user the general workflow in a simple and concise manner. It was decided to represent the flow of the various operations of the software emphatically by using a simple set of icons and arrows, as is visible in Figure 2 - A.

The icons indicate the four main logical operations that the user is likely to do, and opens up related dialogs, namely:

- The Palette Dialog - as indicated by the folder icon
- The Seed Dialog - as indicated by the jack cable icon
- The Sonic Dialog - as indicated by the square feature space icon
- The Output Dialog - as indicated by the speaker icon

3.2.2 Sound Palette

The user loads a selection of audio files or folders containing audio files which are analysed to create the sound palette as has previously been discussed. Next, dimensionality reduction is performed on each feature vector of the units in the sound palette using Principal Component Analysis (PCA). Two PCA components are retained and scaled to the visible area of the interface to serve as coordinates for placing a circular representation of the sound in two-dimensional space. These visual representations, along with their associated audio content we term *sound objects* and are clearly visible in main timbre space window in the figure.

3.2.3 Seed Input

Seed audio is captured and analysed by recording directly from the input audio of the track on which the instrument resides in the audio host. Using the real-time tempo and bar/beat information provided by the host, the recorder will wait until the next bar starts to begin capture, and will only capture complete bars of audio. This audio is analysed as before but with one exception. Since the goal of the instrument is to integrate with an existing session and generate looped material, we make the assumption that the incoming audio is quantised and matches the tempo of the session. Thus onset detection is not performed on the seed input; rather, segmentation takes place at the points in time determined by the grid size (lower left of the screen).

An important aspect to note: since the instrument is fundamentally real-time in its operation, we need to be careful about performing potentially time consuming operations as feature extraction when the audio system is running. As Ross Bencina so aptly puts it: "time waits for nothing" [2]. Thus we perform the audio recording stage and feature extraction process on separate threads so the main audio

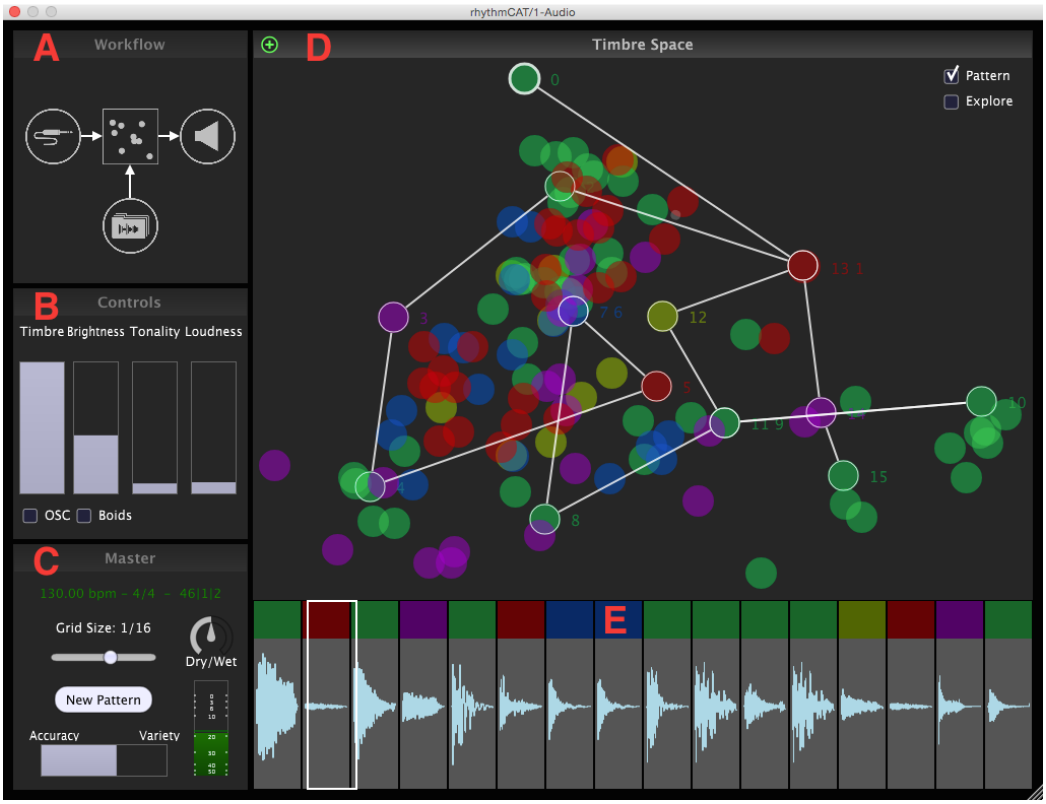


Figure 2: Main User Interface with Onset Graph

playback thread is uninterrupted. This is separate to yet another thread that handles elements of the UI.

3.2.4 *Sonic Parameters*

Clicking on the square sonic icon in the centre of the workflow component opens up the set of sliders shown in Figure 2 (B) that allows us to adjust the weights of the features in the system. Adjusting these weightings have effects in terms of the pattern generation process but also in the visualisation. Presenting their technical names (centroid, flatness and MFCCs) would be confusing for the general user, thus we have relabelled them with what we consider their most descriptive subjective labelling. With the pattern generation process, these weights directly affect the features when performing similarity computation and unit selection, as we will see in the next section. Depending on the source and target material, different combinations of feature weightings produce noticeably different results, informally we have experienced good results using MFCCs alone for example, as well combinations of the flatness and centroid. In terms of visualisation, when the weights are changed, dimensionality reduction is re-initiated and hence positioning of the sound objects in the timbre space changes. Hence manipulating these parameters can help disperse and rearrange the sound objects for clearer interaction and exploration by the user in addition to affecting the pattern generation process.

3.2.5 *Pattern Generation*

Once the palette and seed matrices have been populated, a similarity matrix between the palette and seed matrix is created. Using the feature weightings from the parameter sliders a sorted matrix of weighted Euclidean distances between each onset in the target matrix and each unit sound in the palette matrix is computed, as given by equation 4.

$$SIM\_MATRIX = SORT\_ASC\left(\sqrt{\sum_{i=1}^{n} w_i(a_i - b_i)^2}\right) \tag{4}$$

When the user hits the "New Pattern" button (Figure 2 - C), a new linked list of objects we term *sound connections* is formed, representing a traversal through connected sound objects in the timbre space. The length of the linked list is determined by the grid size specified by the user, thus if the user specifies a grid size of 1/16 for example, a one bar sequence of 16th notes will be generated. The exact procedure whereby we generate a list is detailed in Algorithm 1. The variance parameter affects the threshold of similarity by which onsets are chosen. With 0 variance the most similar sequence is always returned. This variance parameter is adjustable from the Accuracy/Variety slider in the lower left corner of the instrument (Figure 2 - C).

Algorithm 1 Onset Generation Algorithm

```
procedure GET–ONSET –LIST
    for n in GridSize do
        R = Random Number between 0 and Variance
        I = Index from Row R of Similarity Matrix
        S = New SoundConnection
        S->SoundUnit = SoundUnit(I)
        Add S to LinkedList
    end for
return LinkedList
end procedure
```

In the main timbre space interface (Figure 2 - D), a visual graph is generated in the timbre space by traversing the linked list and drawing line edges connecting each sound object pointed to by the sound connection in the linked list. In

this case a loop of 16 onsets have been generated, with the onset numbers indicated beside the associated sound object for each onset in the sequence. The user is free to manipulate these sound connections to mutate these patterns by touching or clicking on the sound connection and dragging to another sound object. Multiple sound connections assigned to an individual sound object can be group selected by slowly double tapping then dragging.

On the audio side, every time there is a new beat, the linked list is traversed and if a sound connection's onset number matches the current beat the corresponding sound unit is played back. One addition that occurred after some user experiments with the prototype is the linear waveform representation of the newly generated sequence (Figure 2 - E). Users felt the combination of the 2D interface with the traditional waveform representation made the sequences easier to navigate as well as being able to manipulate the internal arrangement of sequence itself once generated.

4. CONCLUSIONS AND FUTURE WORK

In this paper we introduced a real-time system for generating and interacting with rhythmic patterns using concatenative synthesis. We described how the system can mimic the rhythmic and sonic character of a seed sound using an unrelated palette of sound samples with a variable range of accuracy and variety. A graph-based model was introduced as a way to present the structure and relation of sounds in the generated pattern for the user to manipulate these patterns in a novel manner.

The instrument has undergone a number of development iterations based on feedback and testing with internal users, including one live performance for an external audience for a project review. Currently we are developing a formalised framework for testing the instrument with external users such as composers and producers. In this survey we aim to evaluate quantitative aspects of the system relating to the fundamental ideas presented in this paper. Firstly, we will gather ratings of generated patterns in terms of their similarity to the target and "suitability" of the generated patterns in the wider context of the composition being work on (i.e. "does it fit"), as we had previously carried out in a similar project using symbolic patterns [12]. Secondly, we will examine the suitability of the audio features we have chosen, their combinations and their impact. We have chosen a small, but in our view, effective set of features for encapsulating rhythmic sounds. However there are many more that warrant investigation and experimentation. Fortunately our design is flexible and scalable in such a way that such features can be easily incorporated.

5. LINKS

Binary software is available to download here:
`https://github.com/carthach/rhythmCAT`

A video demonstration can be viewed here:
`https://youtu.be/Jl1bV4STAmw`

6. ACKNOWLEDGMENTS

This research has been partially supported by the EU-funded GiantSteps project (FP7-ICT-2013-10 Grant agreement nr 610591).[4]

7. REFERENCES

[1] J. Bello and L. Daudet. A tutorial on onset detection in music signals. *IEEE Transactions on Audio, Speech, and Language Processing*, 13(5):1035–1047, 2005.

[2] R. Bencina. Real-time audio programming 101: time waits for nothing, 2011.

[3] G. Bernardes, C. Guedes, and B. Pennycook. EarGram : An Application for Interactive Exploration of Concatenative Sound Synthesis in Pure Data. *From Sounds to Music and Emotions*, pages 110–129, 2013.

[4] S. Böck and G. Widmer. Maximum filter vibrato suppression for onset detection. *Proc. of the 16th Int. Conference on Digital Audio Effects*, pages 1–7, 2013.

[5] D. Bogdanov, N. Wack, E. Gómez, S. Gulati, P. Herrera, O. Mayor, G. Roma, J. Salamon, J. Zapata, and X. Serra. ESSENTIA: an Audio Analysis Library for Music Information Retrieval. *International Society for Music Information Retrieval Conference (ISMIR 2013)*, pages 493–498, 2013.

[6] G. Bradski. The OpenCV Library. *Dr Dobbs Journal of Software Tools*, 25:120–125, 2000.

[7] C. Frisson, C. Picard, and D. Tardieu. Audiogarden : Towards a Usable Tool for Composite Audio Creation. *QPSR of the numediart research program*, 3(2):33–36, 2010.

[8] P. Herrera, A. Dehamel, and F. Gouyon. Automatic labeling of unpitched percussion sounds. *Audio Engineering Society Convention*, 2003.

[9] A. Hunt and A. Black. Unit selection in a concatenative speech synthesis system using a large speech database. *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, 1:373–376, 1996.

[10] B. Logan. Mel Frequency Cepstral Coefficients for Music Modeling. *International Symposium on Music Information Retrieval*, 28:11p., 2000.

[11] R. Loughran, J. Walker, M. O'Neill, and M. O'Farrell. The Use of Mel-frequency Cepstral Coefficients in Musical Instrument Identification. *Proceedings of the International Computer Music Conference*, pages 42–43, 2004.

[12] C. Ó. Nuanáin, P. Herrera, and S. Jorda. Target-Based Rhythmic Pattern Generation and Variation with Genetic Algorithms. In *Sound and Music Computing Conference 2015*, Maynooth, Ireland.

[13] C. Roads. *Microsound*. 2004.

[14] P. Roy, F. Pachet, and S. Krakowski. Analytical Features for the classification of Percussive Sounds: the case of the pandeiro. In *10th Int. Conference on Digital Audio Effects (DAFx-07)*, pages 1–8, 2007.

[15] D. Schwarz. The Caterpillar System for Data-Driven Concateantive Sound Synthesis. In *Proceedings of the 6th Int. Conference on Digital Audio Effects (DAFx-03)*, pages 1–6, 2003.

[16] D. Schwarz, B. Grégory, V. Bruno, and B. Sam. Real-time corpus-based concatenative synthesis with catart. *Proc. of the 9th Int. Conference on Digital Audio Effects (DAFx-06)*, (September):1–7, 2006.

[17] A. Tindale, A. Kapur, G. Tzanetakis, and I. Fujinaga. Retrieval of percussion gestures using timbre classification techniques. *Proceedings of the International Conference on Music Information Retrieval*, pages 541–545, 2004.

[18] G. Tzanetakis, G. Essl, and P. Cook. Automatic musical genre classification of audio signals. In *Proceedings of the Second International Symposium on Music Information Retrieval*, page 6 total pages, 2001.

[4] http://www.giantsteps-project.eu