Drum rhythm retrieval based on rhythm- and sound similarity

Felix Tutzer

Master's thesis September 15, 2011
Supervisor: Hendrik Purwins
Departament of Information and Communication Technologies
Universitat Pompeu Fabra, Barcelona

Abstract

The objective of this thesis is, to develop an application, similar to a midi drum sequencer, that is capable of automatically providing a user with rhythms in the context of a user generated reference rhythm. An underlying rhythmic similarity algorithm is used to compute the distance between the reference and all the possible candidate or target rhythms that are stored as rhythmic descriptions in a database. A rhythmic description contains several descriptors to describe the rhythmic qualities of the tracks of a rhythm such as event timing, sound class, syncopation degree and metrical weight histogram. A track represents the event sequence in a rhythm, where all the events belong to the same sound class. We define the distance between a reference and a target rhythm as the averaged differences between the metrical weight histograms of the respective tracks that belong to the same sound class. The user can define how similar or dissimilar the retrieved target rhythms should be in the context of the reference rhythm by adjusting several settings in the user interface of the application.
The quality of the rhythmic similarity measure was evaluated by comparing the results of listening tests, that were done with 6 electronic music producers including 1 professional percussionist, with the results we obtained from the algorithm. The evaluation shows that the rhythmic similarity algorithm works accurate for the purpose of this project.
The system was implemented in Max/Msp and Java.

Acknowledgment

First of all I would like to thank Hendrik Purwins for his continuous support and guidance, his motivational words and his patience to deal with my doubts and uncertainties throughout the whole last year.

I would also like to thank all my colleagues and friends that I have met in the last two years in Barcelona and with whom I share great experiences and memories.

Last but not least I would like to thank my parents for their love and support.

Contents

Abstract

The objective of this thesis is, to develop an application, similar to a midi drum sequencer, that is capable of automatically providing a user with rhythms in the context of a user generated reference rhythm. An underlying rhythmic similarity algorithm is used to compute the distance between the reference and all the possible candidate or target rhythms that are stored as rhythmic descriptions in a database. A rhythmic description contains several descriptors to describe the rhythmic qualities of the tracks of a rhythm such as event timing, sound class, syncopation degree and metrical weight histogram. A track represents the event sequence in a rhythm, where all the events belong to the same sound class. We define the distance between a reference and a target rhythm as the averaged differences between the metrical weight histograms of the respective tracks that belong to the same sound class. The user can define how similar or dissimilar the retrieved target rhythms should be in the context of the reference rhythm by adjusting several settings in the user interface of the application.
The quality of the rhythmic similarity measure was evaluated by comparing the results of listening tests, that were done with 6 electronic music producers including 1 professional percussionist, with the results we obtained from the algorithm. The evaluation shows that the rhythmic similarity algorithm works accurate for the purpose of this project.
The system was implemented in Max/Msp and Java.

Chapter 1

Introduction

The underlying idea of this thesis is the development of an application, alike a typical MIDI *step sequencer*, that facilitates the experimentation with rhythms and rhythmic structures in an intuitive and constructive way. The application analyzes a user generated sample based midi rhythm and provides alterations of that rhythm that can either be similar, dissimilar or of different complexity. The user can switch between these *target rhythms* and perform them in real time to create and discover new rhythmic patterns on the fly. This feature can be of interest especially for electronic dance music producers, because it makes it possible to discover the rhythmic vicinities of sequenced drum loops, which can be a big support and advantage during trying to find interesting variations of rhythmic patterns.
One of the main objects of this thesis is the development of a robust similarity measure. This measure is needed in order to compare the rhythmic structure of the user generated source sequence to a large set of target sequences that are stored in a database. From this database, the sequences with the highest quality are loaded as midi sequences into dedicated sections in the application that can be accessed and edited in real time. The quality of a target rhythm or sequence is determined, independently of style or genre, by its similarity to the source rhythm within user defined constrains such as complexity (syncopation) or event density.
In the next two sections we first introduce the concept of a sequencer and then present similar applications and theories in a state of the art review.

1.1 Sequencer

The term sequencer appeared first in the 60s with the rise of electronic music and refers to a device, in hardware or software form, for playing back, programming and storing notes. Mechanical music machines such as the *barrel organ* can be seen as direct precursors of the sequencer, because they stored notes in a mechanical fashion

before passing them on to the sound generation. Sequencers were initially purely analog devices that would send control signals (CV) to electronic devises such as synthesizers in order to trigger notes at a certain frequency. They play patterns by using a grid with up to 16 buttons or steps, with each step being 1/16 of a measure. This limitations in resolution, that lead to very repetitive musical patterns, left a very important stylistic imprint onto early electronic music like *Berlin school* with pioneer musicians such as *Tangerine dreams* or *Michael Hoenig*, and is now one of most important stylistic devices in electronic dance music such as techno, house or electro. Sequencers with resolution limitations are also know as step sequencers, because the programming of the events and event properties such as pitch and volume happens in a step by step fashion.
In 1977, *Roland* introduced the first computer-based hardware sequencer (the MC-8), with significantly larger storage, and called it a computer music composer [1]. Since the beginning of the 80s though, sequencers are mostly associated with MIDI sequencers. Also MIDI sequencers do not store the actual sounds of a sequence, but only the necessary control data with whom several tone generators such as digital synthesizers, midi compatible analog synthesizers or samplers are triggered. The stored data in a midi sequence contains all the timing information of the single notes, their length and volume (velocity). Midi sequencer, as distinct from step sequencer, are not necessarily limited to a clearly audible fixed number of steps and are therefore also very useful to record expressive performances from any midi compatible input device such as keyboards, without losing crucial timing information. As distinct from midi sequencers, drum machines or drum computers in the 80s were analog sequencers with integrated tone generators, that would produce synthesized drum sounds. A very popular drum computer from 1980 is the *TR-808* by *Roland*. It is considered to be the most important drum synthesizer and has influenced and shaped music such as House, Hip Hop or Pop.

In the 80s *EM-U* carried on a concept (and made it affordable) that had initially been introduced by *CMIs* with the very expensive *Fairlight*: these *EM-U* drum machines would allow a user to load short sound samples and sequence them. Later on companies such as *Akai* introduced their drum machines (the MPC series) with much better sound quality and enhanced sequencing features. These drum machines (or samplers) were decisive for the rise of a variety of music genres and most importantly Hip Hop.
With the rising popularity of the MIDI protocol and the gaining popularity of the *Atari ST*, people started to write little programs to record and playback notes played by a musician. In the early 90s, the first software midi sequencer came into the market. From that point on, with the steadily increasing capabilities of home computers, midi sequencers were combined with audio editing solutions and extended in such a way, that they would represent a whole virtual music studio solution with mixing console and interfaces for the integration of effects and virtual instruments [1] [3].

Figure 1.1: The Roland TR-808 was first introduced in late 1980 and its distinctive signature sound became essential for many different genres (image from [2])

These complete audio workstation are also known as digital audio workstations or DAWs. Notable DAWs are *Steinberg Cubase*, *FL Studio*, *Ableton Live*, *Logic Pro*, *Reason* and *Reaper*. Notable hardware sequencers are the *Akai MPC series*, the *Korg SQ series*, the *Roland MC series* and the *Yamaha Q series*.

1.2 State of the art

There are two major topics of interest that we want to cover in this section. First we want to talk about commercial and non commercial applications that are able to create variations of existing rhythmic patterns. Secondly we want to talk about different approaches to measure rhythmic similarity.

1.2.1 Expressive rhythm transformation

Very early, sequenced music and especially sequenced drum patterns were accused of sounding too mechanical due to the lack of human expressivity caused by strict quantization [4]. With the rise of Hip Hop music in the 80s and the gaining popularity of drum machines such as the MPC, the so called swing or groove function was introduced. It allowed a sequencer to play a piece in straight or in swing time. Musical notes and structure are identical in both cases but the notes temporal patterns

have slight, significant differences [5]. This difference is a small temporal "shift" of the isolated notes with respect to their quantized location in the midi sequence [4]. These timing deviations do not occur at any arbitrary point in time, but are tightly linked to the metrical structure of a rhythm [6]. A particular case of groove is swing, a term that originates in Jazz music. In swinged rhythms, timing deviations happen on the metrical level of eighth notes in long-short patterns [7]. The swing ratio, the parameter that defines the strength of the swing, is a mathematical expression that refers to the duration of the first eight-note divided by the duration of the second (see Figure 1.2)

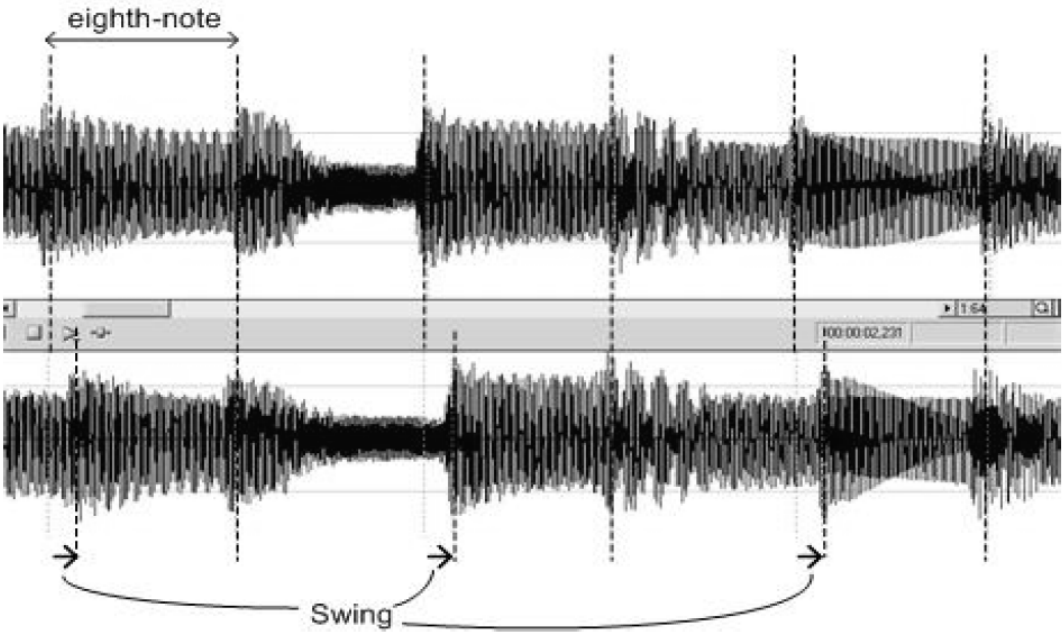

Figure 1.2: Adding swing to an audio file [6]

The interesting aspect of the groove ratio is, that it can be applied in real time. As distinct from our project, grooved notes are normally shifted to time positions outside the maximum quantization resolution. In our application a user can switch to more syncopated or grooved rhythms, but the events of these rhythms always happen on defined quantization steps. On the other hand, target rhythms in our project are extracted from (already existing) meaningful rhythms to avoid random and sometimes meaningless results that can however happen with the application of groove, that is "blind" to the rhythmic structure of the source material. So in order to escape perfectly quantized midi sequences by applying different sorts of timing deviations through swing or groove functions, a rhythm can become more lively, groovy or "human" and less quantized or mechanical sounding. Exactly as

musicians do when they interpret a score, groove can add add a certain amount of expressiveness to a sequenced composition. The most famous drum machines or samplers with swing functionality are the MPCs (starting with the MPC-60 in 1988). The first drum machine with swing functionality was the Linn LM-1 in the early 80s. Nowadays groove functions can not only be found in hardware drum machines such as the current MPC versions, but are also included in most of the modern software sequencers such as *NI Maschine* or *Reason's Matrix*. DAWs like *Ableton Live* or *Logic Pro* even provide drag and drop groove modules for midi sequences, categorized by genres (notice that different genres such as funk, jazz, hip hop or latin all have different styles of timing deviations) and with advanced functionalities. [6] propose an algorithm to apply swing to a whole audio file instead of a midi sequence.
Beat shuffler are another interesting type of rhythm transformation tools. A beat shuffler is an application that shuffles portions of an incoming audio stream while being unaware of its content. These tools normally buffer quantized slices of an incoming audio signal and repeat them at different times and in different patterns. Tools such as *Izotopes Stutter*, *Sugar Bytes Effectric* or *Live's Beat repeat* are used in studio as well as live situations. Tweaking their parameters in real time is an expressive way of manipulating audio in a rhythmic fashion.
Loopmash by *Steinberg* is an tool to blend drum loops together by juggling similar segments across looped beats. It makes use of MIR techniques in order to segment drum loops and retrieve timbre information which is then used to compare segments across different drum loops. While in our project the single sounds of a drum loop are kept and only the rhythmic structure is changed, *Loopmash* does the exact opposite and keeps the original drum pattern intact while changing the sounds of the single segments.

Another interesting approach to rhythmic transformation is the generation of percussive sequences based on the analysis of structure and style of a source audio file. [8] uses Variable Lenght Marcov Chains to generate an arbitrarily long percussive sequence that shares stylistic characteristics with the source sequence. What is interesting about this approach is, that it uses only the information of a source file to create similar variations. In our approach these variations already exist in a database and are retrieved by means of rhythmic similarity.

1.2.2 Rhythm description and rhythmic similarity

Despite onset times, intensity, pitch and duration, there seems to be a consensus, that, in order to describe a rhythm, one must also take timing, tempo and the underlying metrical structure into account [9]. However, there is no common or universal way to represent these three rhythmic concepts. For [10] metrical structure, tim-

ing and tempo can have different meanings depending on the composer, listener or performer. It is therefore important to find perceptual differences according to a listeners culture, musical background, age or sex [11, 12]. Finding a universal way to describe and represent rhythm is often difficult because of the trade-off between the level of abstraction and comprehensiveness. [10] states, that the typical Western music notation is a sufficient mean for a composer to communicate music to a performer, whereas it is insufficient when trying to capture the expressiveness of a performance. A MIDI recording of a performance is able to represent all aspects of a performance, but lacks abstraction.
In the literature, rhythmic descriptions are often represented as a set of features [10] that are extracted from either symbolic input data such as MIDI or audio signals. These feature lists involve:
Onset times Where the parsing of note onset times from symbolic data is an easy task [13, 14], it is much more difficult and ambiguous for audio signals. In the literature we can find many different onset detection algorithms, each one suitable for different source material [15].
Note durations For [16], note durations are almost equivalent to inter onset interval durations (IOIs). Starting from the difficulties of onsets detection, it is much less reliable to find the exact event durations in audio signals. [17] determine IOIs in order to detect "weak" onsets, which are onsets that differ less then a certain threshold (20-50ms) from a preceding one.
Relative amplitude The relative amplitude for MIDI data is associated with the velocity value of a single note event and contributes to the perceptual accentuation. It is also used to assign wights to the events of a rhythmic sequence [10, 18].
Instrument classes In MIDI data there is a mapping standard for percussive sounds [19]. For isolated audio samples, instrument classes can be estimated with high accuracy (if the samples are mono and clean) by means of MIR and machine learning techniques [20].
Frame features In the literature we can find systems, that do not focus on features such as onset detection or note durations, but on lower level descriptions that can be found in audio frames. [21] for example base percussion detection on the energy of the lower frequency bands in the spectrum of each frame whereas [22] look into the energy of several weighted subbands. [23] uses extended feature lists for each subband. Other approaches of frame based percussion detection focus on the energy value variation between consecutive frames [24, 25].

The process of determining a metrical level is called *pulse induction*. Pulse induction aims to detect periodic behavior in feature lists because it is "central to any form of rhythmic understanding" [10]. There are two main different approaches to pulse induction: *pulse selection* and *periodicity functions*.
Pulse selection aims to evaluate the saliency of only a restricted number of possible

periodicities [26]. [27] for example consider the first two events as the potential beats. [17] claims that the first equivalent pair of IOIs defines the pulse of a sequence.
With periodicity functions it is possible to find periodic behavior of certain frame features over time. In [28, 29] e.g. the periodicity is computed for each frequency subband and the results are integrated. The periodicity function is often computed with the Fourier transformation, the autocorrelation function but also IOI histograms. In order to retrieve useful rhythmic information from a periodicity function, usually peak-picking algorithms such as the N-point running window method are applied.
In contrast to pulse induction do *pulse tracking* models consider small timing deviations not as noise and aim to determine changes in pulse period and phase [10] assuming that the tempo is not constant.
[30] proposes a popular automatic tempo extraction method that is based on onset times and beat tracking. For our project, we limit the length of drum loops to one bar, which makes preprocessing procedures such as beat induction and tempo estimation redundant.

Rhythmic similarity measures are often applied in order to contribute to the overall similarity measure between songs [31, 32]. [33] e.g. propose a combination of *fluctuation patterns* which measure the periodicities of loudness in various frequency bands, *onset patterns* and *onset coefficients* in combination with timbre descriptors in order to compute a robust similarity measure. In [34], the similarity between two rhythmic patterns is obtained by aligning the feature lists that consist of fluctuations of loudness and brightness within one pattern by means of the dynamic time warping algorithm. [35] introduces several methods in order to compare purely symbolic rhythmic patterns:
The hamming distance is the number of places where two binary strings do not match (a binary string is a common representation of an event sequence; 1 indicate events, 0 indicate rests).
The euclidean interval vector distance measures the euclidean distance between rhythmic patterns represented as IOI vectors.
The interval-difference vector distance [36] denotes the dissimilarity between two difference-of-rhythm vectors X and Y by

$$d_{ID}(X,Y) = (\sum_{i=1}^{n} \frac{max(x_i, y_i)}{min(x_i, y_i)}) - n$$

If $T = (t_1, \ldots, t_n)$ represents the inter onset interval vector then $X = (x_1, \ldots, x_n)$ is the difference-of-rhythm vector where $x_i = t_1/t_n$.
The swap distance measures the dissimilarity between two string by computing the

amount of "work" needed in order to transform one string into the other. It is defined as the minimum amount of swaps (the interchange of a one and a zero that are adjacent to each other) needed to convert one binary string into another (of the same length).
The chronotonic distance makes use of a 2 dimensional interval representation of a rhythmic pattern, where x and y represent the time length of the interval. The discrimination information between the two density functions $f_1(x)$ and $f_2(x)$ is given by the *Kullback-Liebler divergence*, which is a well known measure of distance or separation between two density functions. Another interesting approach is to just look at the area difference K between the two functions.

Chapter 2

Theory

In this chapter we will talk about the theoretical background we need in order to develop the proposed system. Firstly we will discuss rhythm theory and introduce notions such as beat, meter and metrical salience. Based on this background we will discuss different rhythmic descriptors that are part of a rhythmic similarity measure.

2.1 Rhythm theory

Rhythm is one of the most basic ways that we interact with and understand time [37]. It can be defined as a temporal pattern of acoustic events where the events need to occur with some repetitive structure [38]. In music, these acoustic events are sounds. Some of them are perceptually more expected (salient) then others. *Beat induction* is a process that activates an internal regular (isochronous) pattern while listening to music, the *beat*, *pulse* or *tactus*. Often, the beat coincides with peoples natural motor accompanying like foot-tapping or clapping. The beat provides the temporal framework on which a composition rests and drives music forward [37]. Also, temporal positions marked by the beat do not necessarily coincide with actual sound events. For example, a song may momentarily stop and yet the beat continues to exist in our head although there is no sound. This also occurs for *syncopated* rhythms (which we will discuss a bit later in this chapter) where events might be off beat and listeners can still follow a regular pattern. *Meter* groups beats into stressed (accented) and unstressed (deemphasized) events, introducing a periodic structure on a coarser temporal level. In most cases, musical pieces evolve around this theoretical metrical structure (a bar in our case). These events tend to occur at regularly spaced intervals [39] and form as such a temporal grid that serves as a frame for rhythmical perception. Meter requires therefore the identification of repeating patterns of stressed and unstressed events to perceptually form groups of

beats. It is commonly understood in the West but clearly not a universal concept nor a phenomenon that observable in all world musics" [40]. Rhythm is a phenomenon that consists of the interplay of various temporal levels. On the most refined level the *tatum* is found, the fastest pulsation that corresponds to the duration of the greatest common divisor of all (idealized) inter-onset intervals of the piece.

[41] and [42] discuss the notion of *metrical salience* and introduce a theoretical model based on *the Generative Theory of Tonal Music (GTTM)* to quantify this concept. In this model, every sequential position within a rhythmic pattern has a specific value assigned to it regarding to its specific weight within the metrical structure. A rhythmic sequence is broken down into patterns of equal length. These patterns are recursively divided into sub-patterns of equal length. The number of subdivisions needed in order to arrive at a certain acoustical event in a pattern defines the salience of that point. The more sub-divisions needed, to arrive at a specific point, the less salient the position of this point within the metrical structure (the pattern). For [43, 40], these rules, based on the western concept of the well-formdness of meter, can however not be applied on any style of music.

2.2 Segmentation

Segmentation is a process we apply to audio signals (in our case drum loops) in order to isolate single events such as single strokes. This is important, because it allows us to treat each drum segment individually and furthermore we are able to represent a sound file in midi format, where each drum sound is represented as a single track. The segmentation process is based on an onset detection algorithm. A common way to detect onsets in an audio signal is to detect *transient regions*, which are referred to as sudden bursts of energy or changes in the spectrum of the signal. In the literature we can find a variety of different onset detection algorithms suitable for different source signals [15]. The *high frequency content (HFC)* function is suitable for percussive sounds. It produces sharp peaks during attack transients and measures the presence of high frequencies in a signal by linearly weighting the spectral power. For this project we are using the HFC implementation provided in *aubio* library [44].

Because certain drum sounds such as crash, open hi hat or ride hits can make the algorithm detect false onsets that are very close in time to the actual onsets, we need to do an onset smoothing. For detected onsets $t < t'$ that are closer together then a threshold $\theta > t - t'$, only the first onset t is considered. An audio event is then defined by the segment between two consecutive onsets. For each event, we need to determine the sound class, it belongs to. We talk about this problem in the following two sections.

2.3 Timbre descriptors for percussive sounds

Timbre is a multidimensional, subjective and context dependent property of sound. Each dimension is associated with a measurable physical property of the audio signal itself and can be extracted automatically [20]. These automatically extracted properties are called descriptors or audio features. For our project we use a smaller set set of low level timbre descriptors and the *mfccs* and *delta mfccs* in order to create a multidimensional timbre space for drum samples. The selection of descriptors is based on [20].

2.3.1 RMS

Most of the extracted features use the *Root mean square* (RMS) x_{rms}, which basically is a magnitude spectrum over time.

2.3.2 Zero crossing rate

The *zero crossing rate* (zcr) is an average measure of the times a digital waveform changes from positive to negative values and vice versa. It is a temporal descriptor, that is interesting for drum sounds, because it is associated with the noisiness of a signal, i.e. noisy sounds, such as cymbals tend to have higher zcr rates than more harmonic sounds.

$$\sqrt{\frac{1}{N}\sum_{i=1}^{N} x(i)^2}$$

2.3.3 Log-attack-time

The *log attack time* is typically computed as the base-10 logarithm of the attack time of a signal. It is common practice, to define the starting point of the attack time of as sound as the location, where the signal is below 20% of the maximum amplitude, and the endpoint (when the signal reaches its sustained part) as the position where the signal is above 80% of the maximum amplitude [45]. For an sound event $\mathbf{x}$ with RMS $x_{rms}(t)$define the *log attack time* as:

$$t_{attack} = \log 10(t_{stop} - t_{start})$$

2.3.4 Temporal centroid

The *temporal centroid* (TC) is defined as the time average over the energy envelope of the signal.

$$tc = \frac{\sum_{t=1}^{N} RMS(t)t}{\sum_{t=1}^{N} RMS(t)}$$

2.3.5 Spectral flatness

The *spectral fatness* is computed as the the ratio between the geometric and the arithmetic mean of the spectrum. It is computed separately for 4 bands ranging 250-500 Hz, 500-1KHz, 1KHz-2KHz,2KHz-4KHz. High values of spectral flatness are an indicator for noisy signals. This makes the descriptor very interesting for percussive sounds.

$$SFM(numband) = \frac{\prod_{k \epsilon numband} mX(k))^{1/k}}{\frac{1}{k}\sum_{k \epsilon numband}^{N} mX(k)}$$

2.3.6 MFCC

The *Mel-frequency Cepstrum Coefficients* (Mfcc) are a series of descriptors that are taken from speech recognition and recently they have been proven to be generally useful for instrument classification. Each step in the process of creating MFCC features is motivated by perceptual or computational considerations. The five main steps are:

- Divide signal into frames
- Obtain the amplitude spectrum
- Take the logarithm
- Convert to Mel spectrum (This scale is based on a mapping between actual frequency and perceived pitch as the human auditory system does not perceive pitch in a linear manner)
- Take the discrete cosine transform

The Mfccs are then the amplitudes of the resulting spectrum (typically 13 values).

2.4 Classification

Classification is a very important task for our project, because it allows us to find out with a certain probability, to which sound class (kick, snare, hi hat, ..) a sample

or segment belongs. As we will see later on, the sound class is of big importance for the rhythmic similarity algorithm.

Classification can be divided into two phases: the *training* process, where an algorithm learns from a set of attributes (descriptor values) how to discriminate between categories (drum sound classes). Furthermore, the algorithm generalizes rules, so it is able to categorize new incoming instances. The second step is known as *validation* and is important in order to prevent an overfitting of parameters. Evaluation is usually done in a cross-validation manner or by the supply of a new set of data-points that are not in the training data. The classification process can be done in a supervised manner, that is when the input and output data are a priori known (so kick and snare samples i.e. are labeled as such) or in an unsupervised way, when the classes are not known. In unsupervised learning such as clustering, input data is treated as random variables and expected to group into smaller sets by means of similarity [45]. For both supervised and unsupervised classification, there are a variety of different algorithms that can be applied.

For this project, we are using the *Support Vector Machine* classifier (SVM) from the *libswm* library [46]. SVM belong to the category of supervised classifiers. The learn how to separate two classes by finding the hyperplane that can separate them optimally. The complexity (polynomial order) of the hyperplane is user defined but can however cause overfitting.

2.5 Rhythm descriptors

In order to create a certain amount of abstraction when describing a rhythm (in midi or in sound format), we need to introduce descriptors to quantify specific rhythmic features that are relevant for the development of a rhythmic similarity measurement. In the following sections, we want to describe these descriptors in detail. Each one of them is computed for one single track. A track is a rhythmic event-sequence of one drum-sound within a rhythm. In Figure 2.1 we can see a rhythmic sequence in midi format that contains 3 different tracks for 3 different sounds.

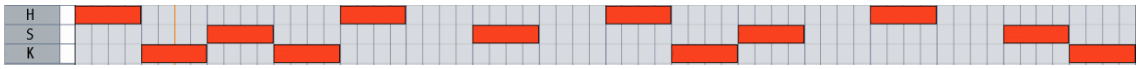

Figure 2.1: An example of a midi rhythm for three different tracks (rhythmic sequences). The tracks all represent different drum sounds (K = Kick, S = Snare and H=hi hat)

2.5.1 Onset density

Each track of a rhythm has a non-zero onset density. It is of central importance, because it allows an immediate comparison of rhythms in terms of event-count.

2.5.2 Binary sequence

In the literature we can find a variety of different rhythm notations [35]. In this chapter we want to introduce the notation of the binary sequence, that is, as we will see later on, specifically interesting for our approach to a rhythmic similarity algorithm. Each event sequence can be represented as a binary sequence of a specific length. This binary-vector is 1 at event positions and 0 elsewhere. A binary-vector can also be interpreted as a quantization of the continuous timeline of a rhythm, where the length or resolution defines the accuracy of the event positions. A higher resolution (bigger subdivion of the musical meter) is important, especially, because in expressive drumming, events often fall on loose metrical positions [5]. In Figure 2.2 we see an example of rhythmic sequences and the correspondent binary-vectors.

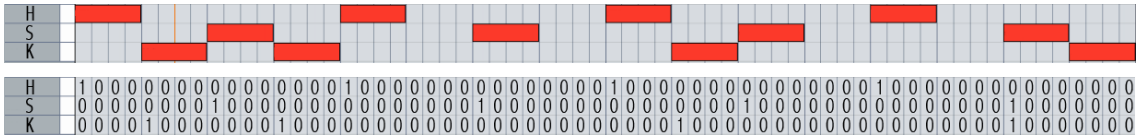

Figure 2.2: An example of a midi rhythm for three different tracks and its corresponding binary sequences

The binary-vector representation of drum sequences is normally useful, because it enables the computation of a distance between two sequences. There are many different distance measures such as the *Hamming distance*, the *Manhattan distance* or the *Swap distance* [47], that can be used to compute the distance between binary sequences. In the scope of this project though, we won't use the binary sequence as a direct mean to compute the similarity between two rhythmic sequences. Instead we use the binary sequence solely as a lightweight storage for the event times of a sequence.

2.5.3 Resolution

The resolution is the maximum length of the binary sequence that is needed, so that all events fall exactly on one metrical position. If the event position is too "loose" for the overall grid resolution, the event timing falls onto the nearest metrical position in the grid. In Figure 2.3 we can see three different sequence examples all of different resolution. We can see, that for the second track, the grid of the first track would be too wide to fully capture each event. Also, although the first track could be easily fitted into the grid of the second or the third track, the resolution of the first sequence is sufficient to represent the event sequence.

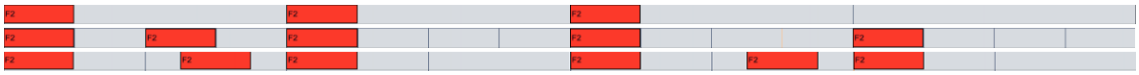

Figure 2.3: An example of three different midi tracks with 3 different resolutions. The first track is of resolution 4, the second of resolution 16 and the third track is of resolution 32

2.5.4 Syncopation degree

In the *Introduction* chapter we talked about the notion of the syncopation to describe rhythms that deviate in some extend from a regular pulse within a meter, have strong events on weak metrical positions and/or pauses on regular (normally stressed) positions within the metrical framework [48]. In [9] syncopation is thought of as being an indicator for rhythmic complexity. The *syncopation degree* is a value that quantifies the overall syncopation of a rhythmic pattern. It relates to the metrical positions of the events of a track and their *accentuation*.

An event-salience, or metric salience, of a position within a pattern refers to the structural level the position is assigned to, which is an indicator of its importance relative to other positions within a metrical unit (a bar in our case) [9]. Events in salient positions are memorized and recalled easier and attract more attention, are more expected to occur, and when they are absent, lead to the impression of rhythmic complexity or syncopation [48]. Different theoretical models of meter perception make alternative predictions about the structure and the depth of the metric hierarchy or a rhythmic pattern (see Figure 2.4).

Model A assigns each metrical level different weights, assumes therefore a different salience for each structural level [49]. *Model B* assumes different weights for events on beat level, higher then beat level and in subbeat level. All event below the tactus (dotted line) are assumed to have the same salience [50]. *Model C* is basically the opposite of *Model B*. Events above,at subbeat and at beat level have the same weight assigned to it. All events below subbeat level are hierarchically

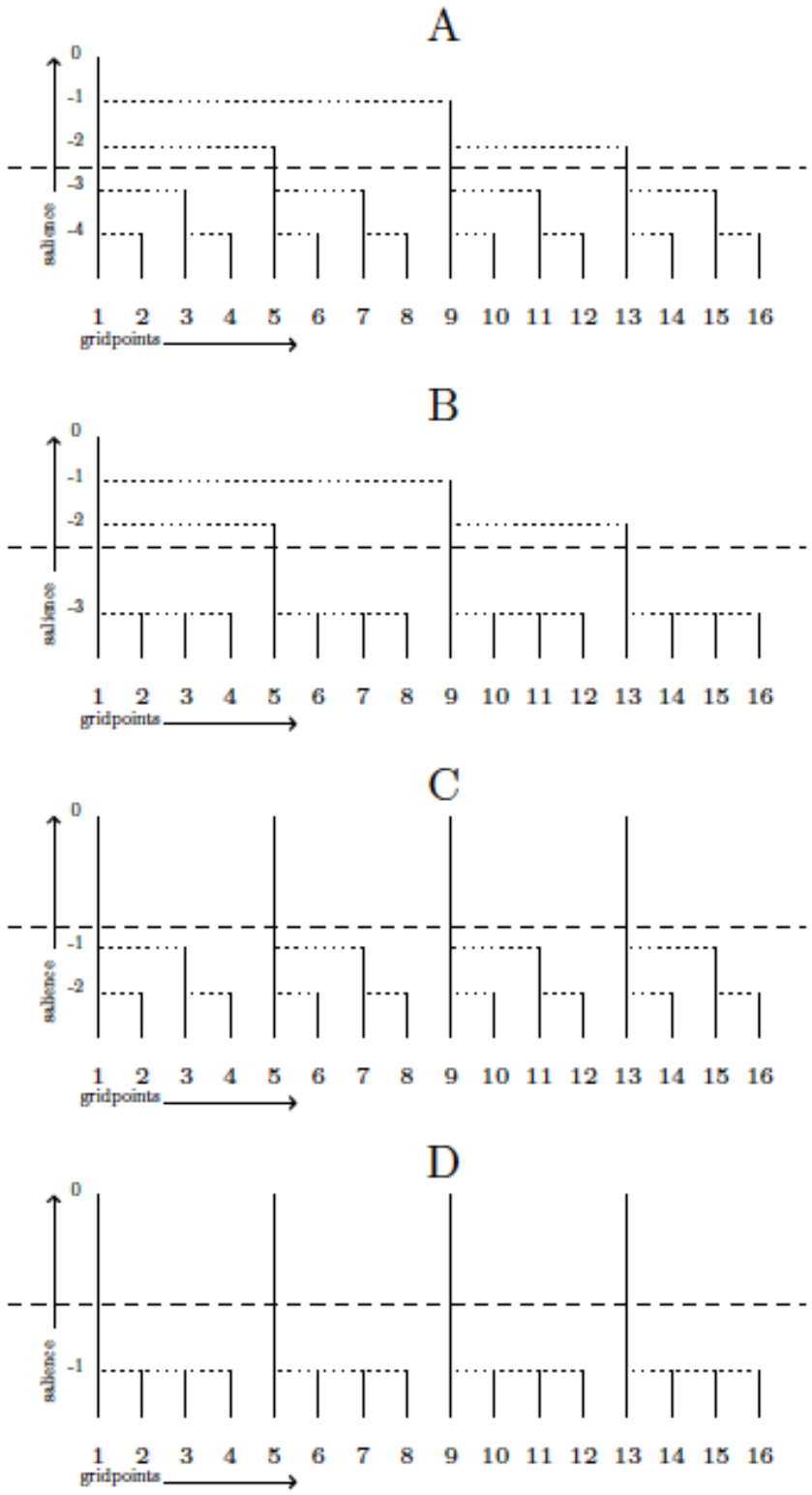

Figure 2.4: (Taken from [9]) Four models of metrical structure. Predicted significant differences in event salience between grid-points within the duration of one bar (x-axis) are expressed using an ordinal scale (y-axis). The dashed lines indicate which events lie above or below the tactus

structured as in *Model A* [51]. *Model D* assumes only two different levels. Events above and at beat level form a unit and everything below beat level forms a different unit. For [49] Model A in Figure 2.4) is the event salience profile for musicians, Model B for non-musicians. [9] suggest a improved model, that also differs between musicians and non-musicians. In Fig 2.5 we adopted the model for musicians for three example tracks.

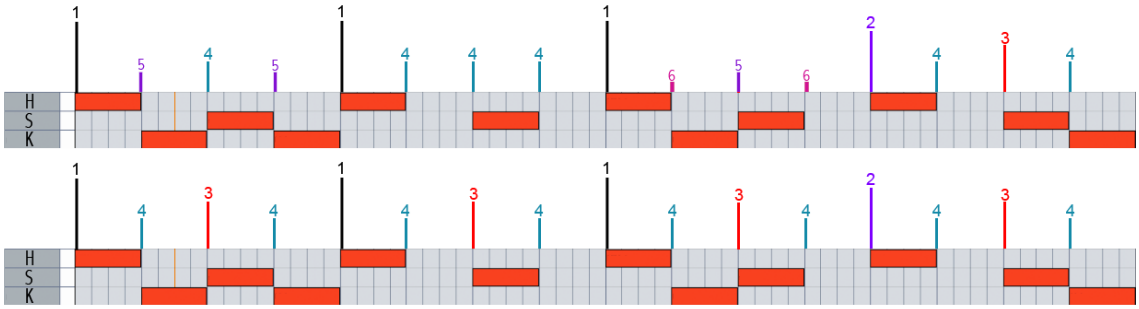

Figure 2.5: Event salience (structural levels) taken from [9] for non-musicians (top) and musicians (bottom).

For the binary-vector representation of a rhythmic sequence, we assign a weight to each metrical position. As we have seen, this weight is directly related to the structural level the position belongs to. [42] proposes a reproducible measure of rhythmic syncopation that follows the idea of metrical weights and divides rhythmic sequences in groupings of pulses. It recursively breaks a sequence down into equal subparts, and assigns to every event a weight relating to its metrical level, assuming a metric hierarchy of maximal depth. With a 4/4 time signature and the smallest note being a 16th note, this would for example result into 5 distinct levels of event salience [41]. [42] introduce the notion of a *rhythm tree* in order to describe a rhythmic sequence with its weights. In Figure 2.6 we can see a rhythm tree with an example pattern and algorithm 1 shows the computation of the syncopation degree.

We propose two modifications to the computation of the syncopation degree by [42]. Firstly, we want to consider events, that are followed by a rest of the same structural level, also as syncopated. Furthermore, we want to consider the saliency or accentuation of the specific events that are detected as syncopated. Events that are louder or of longer duration then their neighbours are perceived as more accented [49] and have therefore more influence on the syncopation degree then less accented ones. Specific patterns of musical events may be interpreted as containing different accent combinations if considered in different metrical contexts. Since in our project we are limiting ourselves to one-bar sequences, where the time signatures are always a multiple of 2 (starting from 2/4), we can ignore the last statement.

Algorithm 1 Syncopation degree [41]

Input
$\mathcal{W} = \{\omega_1, \ldots, \omega_N\}$: metrical weights
$\mathcal{B} = \{b_1, \ldots, b_N\}$: binary sequence
Output
SD syncopation degree
Algorithm
$i = 1$
$SD = 0$
for $b \in \mathcal{B}$ **do**
  if $i < N$ AND $b_i == 1$ AND $b_{i+1} == 0$ AND $\omega_i < \omega_{i+1}$ **then**
    $SD \leftarrow SD + (\omega_{i+1} - \omega_i)$
    $i \leftarrow i + 1$
  else
    if $i == N$ AND $b_N == 1$ AND $b_0 == 0$ **then**
      $SD \leftarrow SD + (\omega_0 - \omega_N)$
    else
      $i \leftarrow i + 1$
    end if
  end if
end for
return SD

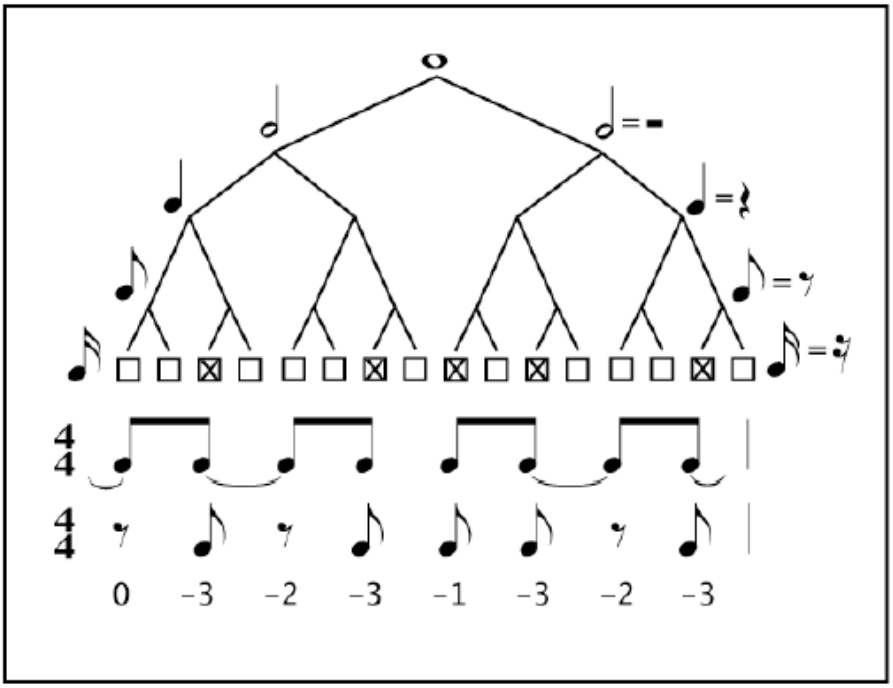

Figure 2.6: (Taken from [48]) A Longuet-Higgins and Lee rhythm tree [42] with an example rhythm and corresponding event weights. The rhythm is notated in musical notation and in *box notation*. The numbers at the bottom indicate the assigned weight, where 0 (the highest weight) is assigned to the first metrical position, -1 to the third, -2 to the second and the fourth. The off-beat eight notes receive the value -3 and so on. Syncopations occur, when a rest (of arbitrary length) is preceded by a sounding note of lower weight. The difference in weight of a rested and a sounded note is the syncopation value for each such pair. The total syncopation value for this rhythm is the sum of all these pairs. In this example $(-2 - -3) + (-2 - -3) + (0 - -3) = 5$

.

The accentuation for acoustic events after segmentation can be the *RMS* value of the single segment, the accentuation of a midi event is the *velocity* value. We multiply the syncopation weight of the single event pairs with the normalized accentuation value of the stressed note. The normalization is done by dividing the accentuation value of the relevant events with the highest accentuation value in the sequence. The sum of the normalized accentuation-weighted syncopation weights is then divided by the overall amount of detected syncopated events to make the syncopation degree independent from the amount of events. In algorithm 2 we can see the modifications for the computation of the syncopation degree.

Algorithm 2 Syncopation degree with accentuation profiles

Input
$\mathbf{w} = (w_1, \ldots, w_N)$: metrical weights
$\mathbf{b} = (b_1, \ldots, b_N)$: uniformly sampled binary onset indicator
$\mathbf{a} = (a_1, \ldots, a_N)$: normalized accentuations
Output
S syncopation degree
Algorithm
$S = 0$
$c = 0$
for $i := 0$ **to** $N - 1$ **do**
  $j \leftarrow (i \mod N) + 1$
  $j' \leftarrow (i + 1 \mod N) + 1$
  if $b_j = 1 \wedge b_{j'} = 0 < w_j \leq w_{j'}$ **then**
    $S \leftarrow S + (w_{j'} - w_j)a_j$
    $c \leftarrow c + 1$
  end if
end for
return S/c

2.5.5 Metric weight histogram

The rhythmic perception of events in an sequence does not only depend on the structural level the events are associated with but also on the structural level of the direct subsequent events or possible syncopations if the subsequent events are rests of a higher structural level. The metric weight histogram counts events based on their metrical position [9], possible subsequent events or syncopation. The order of the bins is directly related to the perceptual regularity of possible events. The first bin of the histogram is reserved for the strongest events (events in the highest structural level) whereas the last bin is reserved for the weakest events (events in the

lowest structural level with the highest syncopation degree). For the computation of the histogram, the algorithm determines the bin position for each event (see Figure 2.7) in a sequence. The histogram is modeled for 4 structural levels. Events that require a resolution or 32 are interpreted as the closest rested sixteenth note. For resolution 64 and 128, events are associated with the closest rested metrical position. If the closest metrical position is not a rest, the algorithm skips the event.

Bin nr.	Associated events
1	Events on m. level 1
2	Events on m. level 2
3	Events on m. level 3 followed by event on m. level 1 or 2
4	Events on m. level 3 followed by rest on m. level 1 or 2 (syncopation)
5	Events on m. level 4 followed by event on m. level 1 or 2
6	Events on m. level 4 followed by rest on m. level 1 or 2 (syncopation)
7	Events on m. level 4 followed by event on m. level 3
8	Events on m. level 4 followed by rest on m. level 3 (syncopation)

Figure 2.7: Metrical weight histogram. Each bin is associated with events of a rhythmic sequence depending on their metrical level, the subsequent events and their possible syncopation. The bin ranking is related to the regularity of the metrical position of an event in a 4/4 drum sequence of one bar length.

The metric weight histogram is a useful tool to compare the rhythmic structure of two sequences of one bar length. In order to determine a similarity between rhythmic structures of one bar length, we propose a weighted difference algorithm.

The histogram as a similarity measure

Histogram bin positions provide detailed information about the structure of a rhythmic sequence by clustering perceptually similar or identical events over a whole sequence on different positions. The difference between two histograms is directly related to the perceptual dissimilarity or similarity of two rhythmic sequences. This similarity measure is also tightly linked to the syncopation degree of the sequences. This means that sequences with a similar syncopation are more likely to be considered as similar (have a lower histogram difference) compared to those with very different syncopation.

The method (see algorithm 3) we propose in order to compute the difference between two histograms is optimized for sequences of one bar length with a 4/4

time signature. Two rhythmic sequences tend to have low histogram difference when they share the same or very close non-zero bins (which can be an indication that both rhythms have a similar groove or syncopation degree). The histogram difference d is the sum of all the non-zero bin differences of the target histogram to the nearest non-zero bin position in the source histogram of higher regularity. First we identify the lowest non-zero bin position l_s in the source histogram h_s and assign the value to c_r. Afterwards we search for the next non-zero bin position i in the target histogram h_s. If $h_t^{cr} = 0$, we compute the distance between c_r and i, multiply that value by 0.1 and $h_s^{c_r}$. Otherwise, if h_s is not zero in c_r and h_s is zero in i, we compute the difference between c_r and i, multiply that value by 0.1 and by h_t^i. Otherwise, if h_s and h_t are not zero in i, we dont compute any difference and update c_r with the value of i. We compute the other distances in the same way from i onwards until the end of the histogram is reached. We also search for all the non-zero bin positions h_s^j where $h_t^j = 0$ and $j > l_s$. We compute the distance from these positions to the current c_r position. If h_t has non-zero bin positions that are smaller then l_s, we compute the distances to l_s. All the computed distances are then summed up in d.

As we can see, the algorithm does not consider the event count on overlapping non-zero bin positions. In the case where two target track histograms have the identical non zero bin positions as a source track histogram but different event counts for these positions, the algorithm would calculate the same histogram differences for both target tracks. We will see in the *Methodology* chapter, that the target track selection algorithm in our system groups tracks based on their onset count and syncopation degree in order to overcome problems that would occur if the histogram difference would be our only similarity measure.

2.6 Discretization and tempo

All the previously mentioned event sequences are symbolic representations with 16 discrete steps and 4 different weighting levels as proposed by [9]. These weighting levels result from listening tests that are based on sequences with 100 BPM in tempo (which corresponds to a 600ms inter beat interval). In our application rhythmic sequences are represented as binary vectors with 128 dimensions in length (expressed in 4/4 time signature) to gain a higher accuracy for event timings. For higher tempos it gets increasingly difficult to perceptually discriminate events on very loose metrical positions from events on metrical positions on lower structural levels (on more regular positions). Tempo is defined as the rate of the pulses at a given metrical level [10]. [52] shows that pulses, that are separated by less then $0.1s$ or more then $2s$, are heard as nearly identical. The perception of even simple periodic series of events can therefore change as the playback speed changes. This implies

Algorithm 3 HD : Histogram difference

Input
h_s : source histogram
h_t : target histogram
Output
d : histogram difference
Algorithm
l_s : lowest non-zero bin in the source histogram
l_t : lowest non-zero bin in the target histogram
$i = l_s$
c_r : current regularity
while $i \leq 8$ **do**
  if $h_i^s > 0 \wedge h_i^t > 0$ **then**
    $c_r = i$
  else
    if $h_i^s > 0 \wedge h_i^t = 0$ **then**
      $d = d + 0.1 * (i - c_r) * h_i^s$
    else
      $d = d + 0.1 * (i - c_r) * h_i^t$
    end if
  end if
  $i = i + 1$
end while
if $l_t < l_s$ **then**
  $i = l_t$
  while $i < l_s$ **do**
    if $h_i^t > 0$ **then**
      $d = d + 0.1 * (l_t - i) * h_i^t$
    end if
    $i = i + 1$
  end while
end if
return d

that, in order for the metrical weighting that use to be accurate, we need to have a tempo minimum of $t_{min} = 75$BPM and a tempo maximum of $t_{max} = 150$BPM. If the tempo is slower then t_{min}, the thirty-second note events need to be represented by an extra weighting level. This is, because if a bar has 32 possible event positions at a tempo of 75BPM, the temporal interval between two adjacent positions is exactly $100ms$, which is the time limit that is needed to clearly distinguish two distinct events perceptually. On the other hand, if the tempo is faster then t_{max}, the time interval between two adjacent sixteenth positions is $100ms$. If we would choose a higher tempo such as 151BPM, we would need to remove the weighting level for sixteenth notes. In Figure 2.8, we can see how a sixteenth note event would perceptually be associated with a higher structural level if the tempo is faster then 150BPM.

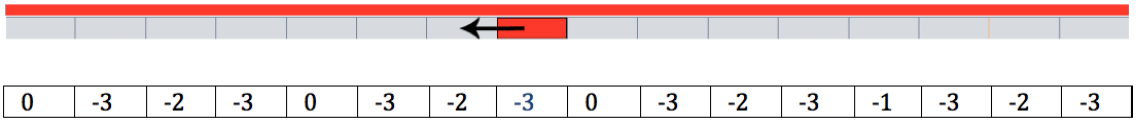

Figure 2.8: If the playback speed is > 150BPM, the event in this example (on structural level -3), can be shifted to structural level -2 because the tempo is too fast to perceptually distinguish the two levels.

As a direct result of this observation we will use 4 structural levels for a resolution of 128 in our system, if the playback tempo lies between 75BPM and 150BPM. Events on thirty-second, sixty-fourth and hundred twenty-eighth note positions are associated with the closest metrical position from figure 2.5 (if that position is not occupied by an event) and its correspondent structural level. If the playback tempo is faster then 150BPM, level 3 and 4 are merged. If the playback tempo is slower then $75BPM$, events on thirty-second note position are assigned to a new 5th level.

Chapter 3

Methodology

We want to be able to change rhythmic structures of MIDI drum loops in a meaningful way. To do so we will build a database and fill it with rhythmic descriptions that we extract from a large collection of either sound or MIDI files. Then we will use the database to find and retrieve the most meaningful rhythms in the context of a user generated reference rhythm by means of a rhythmic similarity measure.
In this chapter we want to discuss the usage and implementation of all the theoretical concepts we introduced previously in order to develop such a system.

3.1 Rhythm database

The rhythm database is a storage for rhythm descriptions which are sets of descriptors (see algorithm 3) that are computed for every single rhythm we want to store. Rhythm descriptions allow us to compare rhythmic sequences and to compute a distance between them.
The rhythms that we want to analyze can either be in sound file or in MIDI format. In order to have a uniform representation, we need to do preprocessing of the sound-files which requires the following four steps:

1. onset detection
2. onset smoothing
3. segmentation
4. classification

The segmentation process isolates the single sounds from the rest of the file. The goal of the classification is to assign a label (sound class) to each of the isolated sounds or segments. A sound class indicates whether a rhythmic sequence belongs

to a kick drum, a snare drum, a opened hi hat, a closed hi hat, a cymbal, a tom or another percussive sound.

3.1.1 Classification

We train the SVM classifier with drum samples that are labeled either as

1. Kick drum (monophonic kick drums and kick drums mixed with other percussive sounds)
2. Snare drum (including claps and rim shots)
3. Hats (including closed hi hats, opened hi hats and cymbals)
4. Toms (including bongos)

Where the sound class detection in midi loops is an easy task due to predefined note mapping, we need to limit the amount of classes we use for the classification of sound files in order to improve the accuracy of the system. To have a good classification accuracy we only consider 4 generic subclasses:

1. Kick drums (1599 training samples)
2. Snare drums (1282 training samples)
3. Hi Hats (1464 training samples)
4. Toms (1034 training samples)

For this reason, we only use and segment drum loop sound files that use drum sounds that belong to either one of the above mentioned classes.

Each labeled sound is represented by a 37 dimensional feature vector (13 MFCC coefficients, 13 delta MFCC coefficients, the attack zcr, the attack RMS, the log attack time, the temporal centroid, the spectral flatness of the decay, the zcr of the decay, the spectral skew of the decay and the kurtosis).
To compute the MFCC, the RMS, the temporal centroid, the spectral flatness, the spectral skew, the kurtosis and the zcr, we use the MIRtoolbox [53] library for Matlab. The classification algorithm is a SVM with a radial basis function and grid-search optimized parameters $\gamma = 0.125$ and $cost = 2$. For 66% cross validation, we have 88,197% correctly classified instances (see Figure 3.1).

Having trained our classifier with clean cut monophonic samples, we established a benchmark and we expect the accuracy of the classifier on segmented sound to be suboptimal.

	TP rate	FP rate	Precision	Recall	F-Measure	ROC Area	Class
	0.846	0.038	0.901	0.846	0.873	0.97	A
	0.873	0.046	0.855	0.873	0.864	0.969	B
	0.968	0.028	0.93	0.968	0.949	0.994	C
	0.82	0.044	0.816	0.82	0.818	0.968	D
Weight avg	0.882	0.038	0.882	0.882	0.882	0.976	

Figure 3.1: Results for 66% cross validation for 4 classes where A = Kick, B = Snare, C = Hat and D = Tom

3.1.2 Drum loops

A big part of the drum loops we analyze and store in our database are in MIDI format. Regular MIDI drum files are build in such way, that events are mapped to specific notes, that are reserved for specific drum sound classes. MIDI has a general percussion key mapping that distinguishes between 46 different drum sounds [19]. This makes it very easy to extract the single tracks from a MIDI drum loop and assign it to a specific sound class. For sound files we need to do all the preprocessing as described in the previous sections, which makes it more difficult and error-prone to extract the component tracks and determine the correct sound classes. In both cases we only use one bar loops. MIDI files provide information about each event, the MIDI ticks, which makes it easy to automatically isolate sections of one bar length. Sound files on the other hand need to be cut manually. The advantage of using only loops with the length of one bar is, that it makes tempo estimation and meter induction redundant.

3.1.3 Building the database

Once we know the exact onset timings and sound classes of events in a drum loop, we can start extracting the rhythm descriptors. First we map the event sequence (the events that belong to the same class) or MIDI tracks to binary vectors of resolution 128. We compute the syncopation degree and the inter event interval histogram for the three different tempo regions. We store these descriptors, the event density, resolution and sound class for each track in the database. Each track is then represented by its descriptors and by the rhythm it belongs to.
We filled the database with 914 one bar MIDI (which results into approximately 4000 single tracks) and 290 sound file drum loops (which result into approximately 900 single tracks). The database we use *Apache Derby* for Java.

3.2 Selection process

Once the user has programmed a MIDI rhythm and chosen the right drum samples for each track, the classifier determines the sound class of the sample and a selection algorithm retrieves the most qualified rhythms or tracks from the rhythm database so that the user can switch between these rhythms in real time.

A rhythmic sequence is composed of different tracks, where each track is an event sequence of a specific sound class. The selection process returns a ranked list of 19 tracks per source track that represent the set with the highest quality. The quality of the selection can be adjusted by the user and is defined by the following settings:

1. event density
2. similarity
3. syncopation
4. resolution

The **event density** defines how much the target track candidates should deviate from the event density of the source track. The **similarity** sets the perceptual similarity of the target and the source tracks. The **syncopation** setting defines the syncopation and the **resolution** the maximum resolution the target tracks can have.

In the following subsection we propose a possible implementations of the selection process. It is based on the concepts of the syncopation degree and the metric weight histogram, that are explained in the *Theory* chapter.

3.2.1 Implementation

Algorithm 4, 5 illustrate the implementation of the track selection. We define a track as a set of descriptors (sound class, metric weight histogram, syncopation degree and event count). The input parameters of algorithm 4 are these descriptors ($\mathbf{s_h}$, s_c and c) of the source track, all possible target tracks $\mathcal{T}$ from the database and the user settings. The user settings are needed in order to retrieve a first rough selection of candidate tracks from all possible database tracks $\mathcal{T}$. In algorithm 4 only those tracks from $\mathcal{T}$ are selected, that belong to the sound class s_c, that have a certain syncopation degree within the region defined in $\mathbf{D}_{sd}$, that deviate only within the region $\mathbf{B}_{dd}$ from the source track event density $\mathbf{c}$ and whose maximum resolution $tr < res$. The final selection of 19 target tracks from $\mathcal{S}$ is determined in algorithm 5. The similarity of two tracks is interpreted as a weighted difference

between the metric weight histograms (see algorithm 3). The difference to the source histogram is computed for all target tracks in $\mathcal{T}$ in algorithm 3 .$\mathcal{L}$ contains all tracks from $\mathcal{T}$ sorted by ascending histogram difference value $diff_i$. Tracks with the same difference value are finally ordered by how much their event count deviates from the source event count. $\mathcal{L}$ is divided into 10 parts of equal length where each part corresponds to one of the 10 possible similarity values the user can set. The first part of the list is the one that contains the tracks with the highest similarity to the source. The last part of the list consists of the tracks that are the most dissimilar to the source. Depending on the similarity setting, the algorithm chooses 19 tracks from one of the 10 parts of the list.

3.2.2 Rhythm collocation

The overall similarity of two rhythms depends on the pairwise similarity of the component tracks of the same sound class. It also depends on the role and importance that this specific sound class has in the overall perception of the rhythm. Rhythm is not purely a product of time location but rather a multi-dimensional attribute. Timbre, envelope of attack, fundamental frequency and amplitude all have a part to play in the perception and grouping of events [54]. Some authors see certain drum sounds and particularly kick and snare drum to be the most salient for the perception of up- and downbeats in a rhythmic sequence [55, 56]. It would exceed the scope of this thesis to evaluate the roles and qualities that each possible drum-kit element plays in a rhythmic structure. We do however give the sound classes of the bass drum, the snare drum, the hi hats and cymbals more importance for the final rhythm selection. This is, because we consider this sound classes to be generally more salient and determining for the perception of a rhythmic sequence. By computing the arithmetic mean of the similarity ranking positions (see algorithm 4) of only the components that belong to these sound classes (if they are present) for all target rhythm, we can derive a rough overall similarity to the source rhythm and select those with the lowest ranking mean.

Algorithm 4 TRACK RETRIEVAL: retrieve 19 target tracks for a source track

Input

$\mathcal{T} = (\mathcal{T}_1, \ldots, \mathcal{T}_K)$: all possible target tracks from the database

source rhythm

s_c : source track sound class

$\mathbf{s_h}$: metric weight histogram

$c \in \{1, \ldots, 128\}$: source track event density

user settings

$sd \in \{1, 2, 3, \ldots, 10\}$: degree of syncopation

$dd \in \{-10, -9, \ldots, 9, 10\}$: density deviation

$sim \in \{0, 1, \ldots, 9\}$: similarity degree

$res \in \{2, 4, 8, 16, 32, 64, 128\}$: max resolution

Output

$\mathcal{S}$ final track selection

Algorithm

$$\mathbf{B} = \begin{bmatrix} -c & -c+2 \\ -c+1 & -c+3 \\ .. & .. \\ -c+7 & -c+9 \\ -c+8 & -c+10 \end{bmatrix}$$

$$\mathbf{D} = \begin{bmatrix} 0 & 2 \\ 1 & 3 \\ .. & .. \\ 7 & 9 \\ 8 & 10 \end{bmatrix}$$

for $\mathcal{T}_i \in \mathcal{T}$ **do**
  tn : sound class of $\mathcal{T}_i$
  tc : event count of $\mathcal{T}_i$
  ts : syncopation degree of $\mathcal{T}_i$
  tr : resolution of $\mathcal{T}_i$
  if $tn = s_c$ **then**
    if $tc \geq \mathbf{B_{dd,1}} \wedge tc \leq \mathbf{B_{dd,2}}$ **then**
      if $ts \geq \mathbf{D_{sd,1}} \wedge ts \leq \mathbf{D_{sd,2}}$ **then**
        if $tr \leq res$ **then**
          add $\mathcal{T}_i$ to $\mathcal{S}$
        end if
      end if
    end if
  end if
end for
$\mathcal{S} = SELECT(\mathcal{T}, \mathbf{s_h}, sim)$
return $\mathcal{S}$

Algorithm 5 SELECT : compute the list of 19 track candidates with the best quality

Input
$\mathcal{S}$: set of selected tracks
$\mathbf{s_h}$: source track metric weight histogram
sim : similarity degree
Output
$\mathcal{S}$ sorted target track selection
Algorithm
$\mathcal{L}$: final selection of tracks
for $\mathcal{T}_i \in \mathcal{S}$ **do**
  $\mathbf{t_h}$: metric weight histogram of $\mathcal{T}_i$
  t_c : event count of $\mathcal{T}_i$
  $diff = HD(\mathbf{s_h}, \mathbf{t_h})$
  add $\mathcal{T}_i$ to $\mathcal{L}$
  sort $\mathcal{L}$ by diff asc
end for
sort $\mathcal{T}_i, i \in (k, \ldots, m)$ in $\mathcal{L}$ with $diff_i = x$ by t_c asc
l : length of $\mathcal{L}$
$s_a = l/10$: similarity divisions
$step = s_a/19$
$\mathcal{L} = \mathcal{L}(s_a sim, s_a sim + step, \ldots, s_a sim + 18step)$
return $\mathcal{S}$

Chapter 4

Similarity evaluation

We think there are two important aspects to be evaluated: a) Rhythmic similarity b) Usability and intuitiveness of the system. However, due to time constraints, we believe the focus on the first aspect is more relevant to the current development and leave the second one to future development of the prototpye. The quality of the rhythmic similarity was measured by how accuratly it estimates the similarity of several target rhythms to a source rhythm. In order to evaluate the outcome, we decided to conduct listening tests to be able to validate the resulting similarity distances. The listening tests were performed on 6 electronic music producers, including one professional percussionist, because we wanted to focus mainly on the opinions of subjects from the target group of this application.

From the rhythm database (see *Methodology*) we chose 15 kick drum tracks and 15 hi hat tracks with different degrees of event density and complexity. From these 30 tracks we selected 5 source tracks and for each source track we selected 4 target tracks in order to compile a total of 10 listening sets (1 source and respectively 2 target tracks). In figure 4.1 we can see the metric weight histograms and histogram differences between targets and sources for all 10 sets. For all the selected tracks we used the same kick drum and hi hat samples and added 2 metronome clicks at the one and at the three (according to a 4/4 time signature). Furthermore, the sets consisted only of rhythms that have the same amount of events.
For each of the 10 listening sets we compiled 1 audio file. Each audio file was composed according to the following pattern: source track → target track 1 → source track → target track 2 → 1.5 second silence → source track → target track 1 → source track → target track 2.
The 6 subjects were provided per email with the 10 listening sets and asked to subjectively determine for each set, whether one of the two target tracks sounded more similar to the source and if so, which one. The term similarity was specified

to the subjects as the closeness of two rhythms in terms of rhythmic structure and perceptual similarity.

Figure 4.2 reports the results of the listening test for all the 6 subjects. We make the following observations: in 46 out of the 60 cases the subject's choices coincide with the similarity rating by the algorithm. The 14 dissenting cases happened in the sets 2, 3, 4, 5, 8, 9 and 10 (see figure 4.1). In the sets 2, 4 and 9, the histogram difference to the source differs by only 0.1 for both targets. In 8 and 10, the histogram difference is equal for both targets. In 3, 4 and 5 respectively only one out of 6 subjects labeled a different target than the algorithm. From these observations we can clearly see, that dissenting cases tend to happen mostly in sets, where target histograms have the same distance to the source or where the histograms distances to the source differ by only 0.1.
Another interesting observation can be made in set 3 and set 5: in both cases target 2 has a higher syncopation degree then target 1, which, in these cases, leads to a higher histogram difference. We asked the two subjects (that labeled different targets then the algorithm in these two cases), if they could explain their choices. In both cases, they described the target of their choice as perceptually more congruent to the source, but only in terms of onset locations and not in terms of syncopation or grooviness. This is plausible, because a listener can easily perceive a syncopated version of drum pattern to be more similar to the original pattern then a second pattern that has the same syncopation degree as the original but slightly different event timings.
In our system we solve problems that might emerge in these cases by letting the user choose the syncopation degree of the target tracks and hence allowing only target rhythms that are similarly syncopated.

The audio files used for testing and evaluation are available for download upon request.

Set		1	2	3	3s	4 I	4s 1	4 III	4s III	*diff*
	S				1					
1	T1				2				1	0.4
	T2						1		2	1
	S				1					
2	T1	1			2					0.1
	T2	1	1		1					0.2
	S				1		2			
3	T1				3					0.4
	T2				2				1	0.8
	S				1		2			
4	T1				1		1		1	0.2
	T2	1			1		1			0.1
	S	2							1	
5	T1	1		1					1	0.2
	T2	1					1		1	0.5
	S	2							1	
6	T1						1		2	0.5
	T2	1		2						0
	S	1			2				1	
7	T1	1		1	1		1		1	0.4
	T2		1	1	1		1		1	0.5
	S	1			2				1	
8	T1	1		1	1		1		1	0.5
	T2		1	1	1		1		1	0.5
	S				1		1	1	2	
9	T1						1		4	0.3
	T2	1			1		1		2	0.2
	S				1		1	1	2	
10	T1						3		2	0.5
	T2		1				2		2	0.5

Figure 4.1: The table illustrates the 10 test sets (rows) that were used for the listening tests as histograms. The left-most column indicates the different sets from 1 to 10. The second column indicates whether a histogram belongs to a source (S), to target track 1 (T1) or to target track 2 (T2) of a test set. The columns 1 to $4sIII$ indicate the 8 bin numbers of the metric weight histogram and the last column shows the histogram differences (see algorithm 3) of each target track to its source track. Lower distance values indicate more similarity to the source then higher values.

	Subj. 1	Subj. 2	Subj. 3	Subj. 4	Subj. 5	Subj. 6	SM
Set 1	1	1	1	1	1	1	1
Set 2	1	N	1	1	1	N	1
Set 3	1	1	1	1	2	1	1
Set 4	2	2	2	2	2	1	2
Set 5	1	1	1	1	1	2	1
Set 6	2	2	2	2	2	2	2
Set 7	1	1	1	1	1	1	1
Set 8	2	N	2	N	N	N	N
Set 9	N	2	N	N	2	2	2
Set 10	2	2	1	2	N	N	N

Figure 4.2: The table reports the results of the 6 subjects for 10 listening sets. Columns represent the 6 subject choices (1 means target 1 is closer to the source, 2 means target 2 is closer and N means neither) and rows represent the listening sets. The last column shows the similarity rating (metrical weight histogram difference) as computed by the algorithm.

Chapter 5

Discussion

For the application we developed a rhythmic similarity measure mainly based on syncopation degree and metric weight histogram. Both concepts refer to the GTTM and the structural level weightings proposed by [49, 9]. One of the major drawbacks of the GTTM is, that it does not deal with event timings that deviate from strict metrical positions, although departures from strict metrical timing are "apparent in almost all styles of music" [28]. Many of the drum loops in our database do have expressive event timings. If the onset timing of events can not be exactly captured by one of the 128 metrical positions in a binary sequence, the algorithm shifts it to the nearest quantizing step. On the other hand we represent each event by one of the 8 possible bins in the metric weight histogram. Events that require a higher resolution then 16 are represented by the nearest rested bin (if possible the one with the lowest structural level). In order to overcome errors that occur along the calculation of the histogram differences between histograms whose complexities differ to a great extend from each other, we decided to let the user choose syncopation degree, onset count and resolution that possible target tracks are allowed to have. With this limitations we can guarantee that the histogram differences are not computed on tracks that differ too much in resolution, syncopation degree and event count. But at the same we limit the flexibility of the track selection algorithm in that the target tracks are not allowed too be too different from each other. [57] introduces the notion of a *rhythm space* in order to describe all possible variations of a rhythmic sequence that would still be perceived as the same rhythm though in an expressive context. Based on this theory we think that highly off beat events could be labeled as grooved (swinged) event notes, whose timing deviates to some extend from a more regular metrical position, and introduces a groove-ratio in order to better represent and compare rhythmic sequences. However, this would not solve the representation problems that arise, when sequences contain subsequent events such as fast drum rolls, that are often on very off positions but not syncopated or swinged.

Another observation we made was, that in order to improve the quality of the system, we need to drastically increase the number of target rhythms in the rhythm database. Although the single track retrieval gives quite interesting results, the full rhythm retrieval is still very unsatisfactorily, mainly because of the relatively small quantity of drum loops in the rhythm database. Furthermore it is obvious, that analyzing audio files is much more error-prone (inaccurate onset detection, wrong classifications) and almost redundant given that MIDI loops contain polyphonic and easy accessible information.
An important issue that we addressed only vaguely in this thesis, is the saliency and importance of certain sound classes for the overall perception of up- and downbeat in drum rhythms. We only consider metrical salience and accentuation when comparing single tracks of one sound class although certain sound classes such as kick and snare drum have more effect on the overall perception of rhythm than other drum types. A first attempt in order to consider this topic would be, to estimate syncopation over more sound classes and not for each sound class separately.

Chapter 6

Conclusions

Throughout this project we developed the prototype of a step sequencer for percussive one bar loops, that differs from typical sequencers in that it is capable of automatically retrieving and loading a set of target rhythms from a database, that are, within user specifiable qualities, the most similar to a user generated source rhythm. This is a novel approach to create a tool that allows the experimentation and expressive performance of digitally sequenced rhythms. The system is usable in real time (notice that this does not involve the retrieval and analysis, which is a combined task that requires approximately 3 seconds of computation time). The development of the system involved several distinct steps:

- The development of a set of rhythmic descriptors (binary vector, syncopation degree, metric weight histogram)
- The analysis of approximately 900 different drum loops including the preprocessing of the loops in sound file format
- The development of a rhythm database that holds all the necessary descriptors for each analyzed drum loop
- The development of an algorithm that is capable of measuring the similarity between rhythmic sequences
- The development of the system prototype

In order to establish a reliable rhythmic similarity measurement that is suitable for such a system, we introduced the novel concepts of the metric weight histogram as a rhythmic descriptor and of a method to compute a distance between these histograms. The reason behind our decision to research on and develop a novel approach to measure rhythmic similarity between symbolic sequences was, that to the knowledge of the author no other method exists so far, that also considers

metrical position distances and syncopation, in the literature. The method has been especially designed for the purpose of this project and is therefore not extendable to a general purpose model.
One limitation of our systems is, that the lengths of drum loops are always restricted to exactly one bar. This is, because we wanted to avoid unprecise or biased meter detection and tempo estimation in favour of a more reliable similarity measure and track retrieval. It would be a logic improvement to extend the system in such a way, that tracks can have the length of multiple bars, without changing the rhythmic similarity measure.
The target group of the application are electronic music producers. As such we plan to evaluate the system. As a results, we plan to establish a bigger user group for testing and evaluation and incorporate their input in future development of our application.

6.1 Future Work

For the rhythmic similarity algorithm, we would like to address the following topics:

1. Higher resolution to capture events with more accuracy
2. Groove-ratio to have an alternative description for off-beat events
3. Improve metric weight histogram, eventually introduce an IOI histogram for metrical positions
4. Improve overall rhythmic similarity for different sound classes and investigate salience of sound classes
5. Extend training data of the SVM classifier with more classes (for now there are only 4)

Furthermore we need to raise the number of target rhythms in the rhythm database to improve the performance of the system.

For the application there are several features we still need to incorporate or improve:

1. Multithreaded selection algorithm for better retrieval performance
2. General user interface improvements
3. Incorporate the possibility to store rhythms and define presets
4. Incorporate envelopes for samples

5. Incorporate VST or AU interfaces

We believe that the big potential of the system lies in the experimentation with rhythm in a (real time) performance situation. By now, it is implemented as a patch in Max/Msp and can only be used as a standalone version. We intent to implement the system as a *max4live* patch or as a VST/AU plugin in order to give the user the possibility to incorporate it in an extended setup.

Chapter 7

The application

In this chapter we wan to introduce the actual application prototype and describe its usage and capabilities. In figure 7.1 we can see a screenshot of the application window with hints to each feature.
The application is a typical midi step sequencer with the extra feature of loading relevant rhythms from a database and letting the user switch between them and the source rhythm in real time. The application provides different tracks, that the user can add or remove. For each track, it is possible to program a rhythmic sequence by defining the onset timings, volumes, panning and playback directions on the event grid (see figure 7.3). The grid resolution can be changed for each track separately from 4 to 128 steps per bar. For each track, the system is able to retrieve 19 relevant target tracks from the database. The selection criteria for the retrieval can be adjusted manually for each track by defining the density deviation to the source track, the similarity to the source track, the maximum resolution and the syncopation (see figure 7.2). These tracks are then loaded into the 19 remaining target track slots and can be accessed by choosing just clicking a tab from 2-20 (the system is not controllable via midi, so the use has to change rhythm tabs by mouse). Each target track can be treated exactly the same way as a source track: it is possible to change the rhythmic structure, the volume, panning and playback direction for all the events.

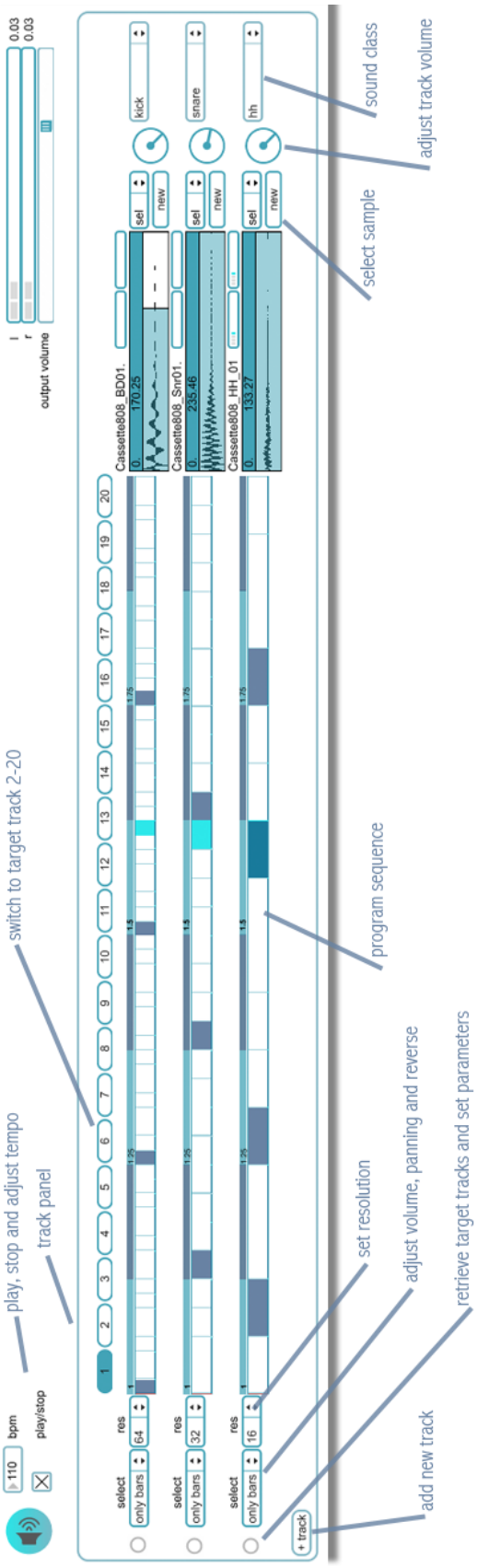

Figure 7.1: Screenshot of the application prototype

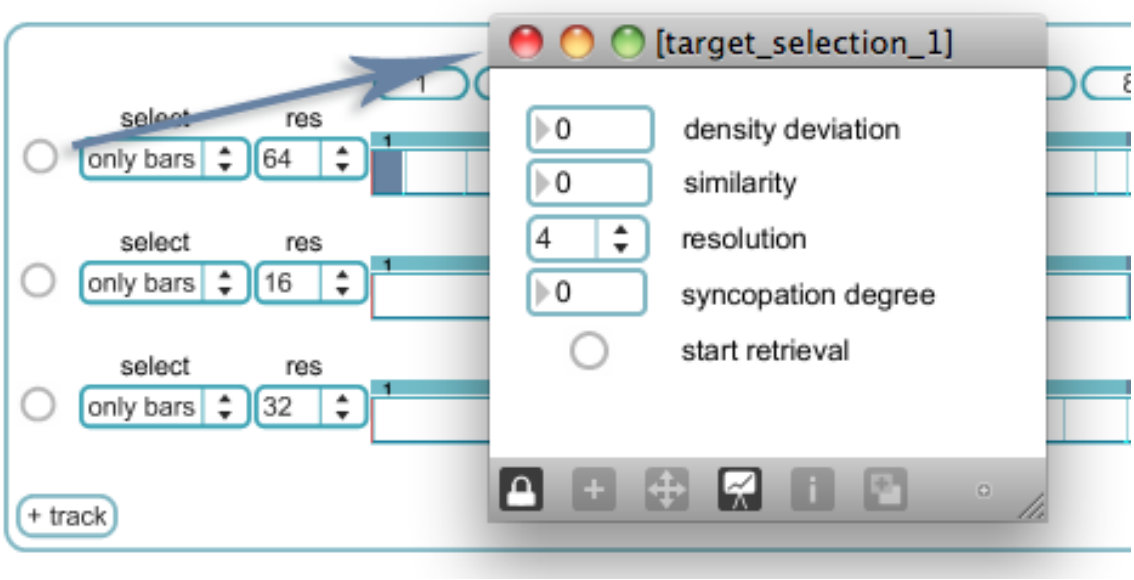

Figure 7.2: Window to define the settings for and to start the target track retrieval

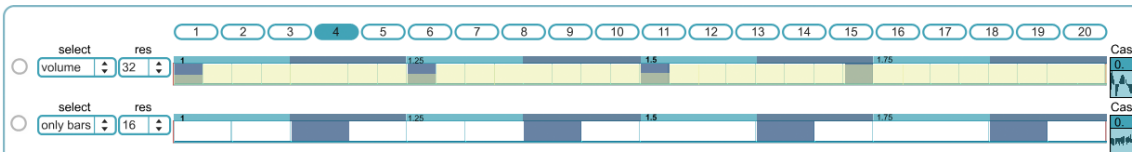

Figure 7.3: On the drop box on the left, the user can switch between the option of programming the event onsets, event volumes, event panning or event playback direction. In this case, the yellow bars on top of the event positions indicate the volume for each position in the sequence.

Bibliography

[1] R. Martin, *Sound Synthesis and Sampling*. Focal press, 2008.

[2] www.vintagesynth.com, "Roland tr-808 rhythm composer," 2011.

[3] T. Grne, *Tontechnik. 1. Auflagel*. Carl Hanser Verlag, 2006.

[4] J. A. Bilmes, *Techniques to foster drum machine expressivity*, vol. 1993, pp. 276–283. ICMA.

[5] K. A. Lindsay and P. R. Nordquist, "A technical look at swing rhythm in music," *Journal of the Acoustical Society of America*, vol. 120, p. 3005, 2006.

[6] F. Gouyon, L. Fabig, and J. Bonada, *Rhythmic expressiveness transformations of audio recordings swing modifications*, pp. 1–6. Citeseer, 2003.

[7] J. Laroche, *Estimating tempo, swing and beat locations in audio recordings*, pp. 135–138. No. October, Ieee, 2001.

[8] M. Marchini and H. Purwins, *Unsupervised Generation of Percussion Sequences from a Sound Example*. 2010.

[9] H. H. Ladinig O., "Rhythmic complexity and metric salience," in *Temporal expectations and their violations (Olivia Ladinig, 2009)*, pp. 29–47, Science Park 904 1098 XH Amsterdam: Institute for Logic, Language and Computation Universiteit van Amsterdam, 2009.

[10] F. Gouyon and S. Dixon, "A review of automatic rhythm description systems," *Computer Music Journal*, vol. 29, no. 1, pp. 34–54, 2005.

[11] C. Drake, "Reproduction of musical rhythms by children, adult musicians, and adult nonmusicians.," *Perception And Psychophysics*, vol. 53, no. 1, pp. 25–33, 1993.

[12] E. Lapidaki, "Stability of tempo perception in music listening," *Music Education Research*, vol. 2, no. 1, pp. 25–44, 2000.

[13] A. T. Cemgil, P. Desain, and B. Kappen, "Rhythm quantization for transcription," *Computer Music Journal*, vol. 24, no. 2, pp. 60–76, 2000.

[14] C. Raphael, "A hybrid graphical model for rhythmic parsing," *Artificial Intelligence*, vol. 137, no. 1-2, pp. 217–238, 2002.

[15] J. P. Bello, L. Daudet, S. Abdallah, C. Duxbury, M. Davies, and M. 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.

[16] J. C. Brown, "Determination of the meter of musical scores by autocorrelation," *Journal of the Acoustical Society of America*, vol. 94, no. 4, pp. 1953–1957, 1993.

[17] R. B. Dannenberg and B. Mont-Reynaud, *Following an Improvisation in Real Time*, pp. 241–248. International Computer Music Association, 1987.

[18] L. Smith and P. Kovesi, "A continuous time-frequency approach to representing rhythmic strata," *IN PROC 4TH INT CONF ON MUSIC PERCEPTION AND COGNITION*, pp. 197—202, 1996.

[19] M. M. A. (MMA), "General midi level 1 sound set," 2011.

[20] E. Pampalk, P. Herrera, and M. Goto, "Computational models of similarity for drum samples," *Ieee Transactions On Audio Speech And Language Processing*, vol. 16, no. 2, pp. 408–423, 2008.

[21] M. Alghoniemy and A. H. Tewfik, *Rhythm and periodicity detection in polyphonic music*, pp. 185–190. Ieee, 1999.

[22] G. Tzanetakis, G. Essl, and P. Cook, *Human Perception and Computer Extraction of Musical Beat Strength*, pp. 257–261. 2002.

[23] E. Pampalk, A. Rauber, and D. Merkl, "Content-based organization and visualization of music archives," *Proceedings of the tenth ACM international conference on Multimedia MULTIMEDIA 02*, p. 570, 2002.

[24] E. D. Scheirer, "Tempo and beat analysis of acoustic musical signals.," *Journal of the Acoustical Society of America*, vol. 103, no. 1, pp. 588–601, 1998.

[25] A. Klapuri, *Musical meter estimation and music transcription*, p. 4045. Citeseer, 2003.

[26] R. Parncutt, "A perceptual model of pulse salience and metrical accent in musical rhythms," *Music Perception*, vol. 11, no. 4, pp. 409–464, 1994.

[27] H. C. Longuet-Higgins and C. S. Lee, "The perception of musical rhythms.," *Perception*, vol. 11, no. 2, pp. 115–128, 1982.

[28] S. Dixon, E. Pampalk, and G. Widmer, "Classification of dance music by periodicity patterns," *Audio*, pp. 159–165, 2003.

[29] F. Gouyon, P. Herrera, and P. Cano, "Pulse-dependent analyises of percussive music," *Audio*, vol. 4, pp. 1–6, 2002.

[30] S. Dixon, "Automatic extraction of tempo and beat from expressive performances," *Journal of New Music Research*, vol. 30, no. 1, pp. 39–58, 2001.

[31] J. C. C. Chen and A. L. P. Chen, "Query by rhythm an approach for song retrieval in music databases," *Idea*, p. 139, 1998.

[32] J. S. Juli, "Identification of versions of the same musical composition by processing audio descriptions," *Exchange Organizational Behavior Teaching Journal*, 2011.

[33] T. Pohle, D. Schnitzer, M. Schedl, P. Knees, and G. Widmer, *On rhythm and general music similarity*, pp. 525–530. No. Ismir, 2009.

[34] J. Paulus and A. Klapuri, "Measuring the similarity of rhythmic patterns," *Signal Processing*, vol. 1, p. 44, 2002.

[35] G. T. Toussaint, *A Comparison of Rhythmic Similarity Measures*, pp. 242–245. Citeseer, 2004.

[36] E. J. Coyle and I. Shmulevich, *A system for machine recognition of music patterns*, vol. 6, pp. 3–6. INSTITUTE OF ELECTRICAL ENGINEERS INC (IEE), 1998.

[37] W. A. Setharesy, *Rhythm and Transforms*. Connecticut, USA: Springer, 2007.

[38] P. Desain and L. Windsor, *Introduction: multiple perspectives on rhythm perception and production,*, pp. xi–xvi. Routledge, 2000.

[39] S. Handel, *Listening: An Introduction to the Perception of Auditory Events*. MIT Press, 1989.

[40] M. Clayton, "Time in indian music: Rhythm, metre, and form in north indian rag performance," *North*, p. 230, 2008.

[41] F. Lerdahl and R. Jackendoff, *A Generative Theory of Tonal Music*, vol. 7. MIT Press, 1983.

[42] H. C. Longuet-Higgins and C. S. Lee, "The rhythmic interpretation of monophonic music," *Music Perception*, vol. 1, no. 4, pp. 424–441, 1984.

[43] J. M. Magill and J. L. Pressing, "Asymmetric cognitive clock structures in west african rhythms," *Music Perception*, vol. 15, no. 2, pp. 189–222, 1997.

[44] P. M. Brossier, "The aubio library at mirex 2006," *Synthesis*, 2006.

[45] E. Ayl, *Automatic detection and classification of drum kit sounds*. PhD thesis, 2006.

[46] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[47] S.-h. C. S.-seok Choi and C. Tappert, "A survey of binary similarity and distance measures," *Journal of Systemics, Cybernetics and Informatics*, vol. 8, pp. 43–48, 2010.

[48] W. T. Fitch and A. J. Rosenfeld, "Perception and production of syncopated rhythms," *Music Perception*, vol. 25, no. 1, pp. 43–58, 2007.

[49] C. Palmer and C. L. Krumhansl, "Mental representations for musical meter.," *Journal of Experimental Psychology: Human Perception and Performance*, vol. 16, no. 4, pp. 728–741, 1990.

[50] M. L. A. Jongsma, P. Desain, and H. Honing, "Rhythmic context influences the auditory evoked potentials of musicians and non-musicians.," *Biological Psychology*, vol. 66, no. 2, pp. 129–152, 2004.

[51] F. Gómez, A. Melvin, and G. T. Toussaint, "Mathematical measures of syncopation," in *In Proc. BRIDGES: Mathematical Connections in Art, Music and Science*, pp. 73–84, 2004.

[52] T. Bolton, "Rhythm," *The american journal of psychology*, vol. 6, pp. 145–238, 1894.

[53] O. Lartillot, "Introduction to mirtoolbox," *Brain*, pp. 1–142, 2007.

[54] N. Collins, "Beat induction and rhythm analysis for live audio processing : 1st year phd report," *Audio*, 2004.

[55] M. Goto, "An audio-based real-time beat tracking system for music with or without drum-sounds," *Journal of New Music Research*, vol. 30, no. 2, pp. 159–171, 2001.

[56] F. Gouyon, F. Pachet, and O. Delerue, "On the use of zero-crossing rate for an application of classification of percussive sounds," *Audio*, pp. 3–8, 2000.

[57] P. Desain and H. Honing, "The formation of rhythmic categories and metric priming," *Perception*, vol. 32, pp. 341–365, 2003.