Matti Pesonen

Sonically Augmented Table and Rhythmic Interaction

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 28.2.2011

Thesis supervisor:

Prof. Vesa Välimäki

Thesis instructors:

Cumhur Erkut, D.Sc. (Tech.)

Antti Jylhä, M.Sc. (Tech.)

Aalto University
School of
Electrical
Engineering

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Matti Pesonen

Title: Sonically Augmented Table and Rhythmic Interaction

Date: 28.2.2011 Language: English Number of pages:6+58

Department of Signal Processing and Acoustics

Professorship: Audio Signal Processing Code: S-89

Supervisor: Prof. Vesa Välimäki

Instructors: Cumhur Erkut, D.Sc. (Tech.), Antti Jylhä, M.Sc. (Tech.)

The objective of the research is to study the different musical properties present in musical rhythmic user input, and to implement an interactive real-time system capable of retrieving, analyzing and reacting to the input with a rhythmic output of its own. The main user-interface of the system is a table top, which is sonically augmented using drum samples. The system simulates a situation of two percussion-instrument players' collaborative improvisation.

The implementation is created mainly in Max/MSP programming environment with a few additional objects coded in Javascript. The system consists of a laptop, a contact microphone, an audio-interface and a table or an equivalent surface where the microphone is connected.

The thesis starts by summarising the basic cognitive principles of the human rhythm perception, and then proceeds into the computational subtasks related to the problem. After a review of previous research on the tasks, a new interactive system consisting of previously presented and completely new algorithms are presented. The novel algorithms include most notably a context-dependent rhythm quantisation algorithm, based on rarely noticed principles of music cognition.

Finally, higher musical interaction modes are introduced, namely 'accompaniment' mode, 'solo' mode and 'call-and-response' mode. All of these simulate different social behaviour modes, which are typical in collaborative improvisation situations.

The thesis focuses on describing the implementation. For instance, user tests and subjective analyses are not included. The design is not intended for professional use, but functions mainly as a proof-of-concept for included algorithms. The main point has been on bringing up the problems related to computational musical interaction, and test different algorithms in practice.

Keywords: Acoustic signal processing, audio systems, music, psychology, interactive systems

AALTO-YLIOPISTO
SÄHKÖTEKNIIKAN KORKEAKOULU

DIPLOMITYÖN
TIIVISTELMÄ

Tekijä: Matti Pesonen

Työn nimi: Äänitehostettu pöytä ja rytminen vuorovaikutus

Päivämäärä: 28.2.2011 Kieli: Englanti Sivumäärä:6+58

Signaalinkäsittelytekniikan ja akustiikan laitos

Professuuri: Äänenkäsittelytekniikka Koodi: S-89

Valvoja: Prof. Vesa Välimäki

Ohjaajat: TkT Cumhur Erkut, DI Antti Jylhä

Tutkimuksen tavoitteena on tutkia rytmidatan musiikillisia ominaisuuksia ja toteuttaa vuorovaikutteinen, reaaliajassa toimiva järjestelmä, joka analysoi rytmidataa ja reagoi siihen. Järjestelmän pääasiallisena käyttöliittymänä toimii tavallinen pöytä, joka on äänitehostettu ääninäytteiden avulla. Järjestelmä mallintaa tilannetta, jossa kaksi perkussiosoittajaa improvisoivat yhdessä.

Käytännön toteutus on tehty suurimmaksi osaksi Max/MSP -ohjelmointiympäristössä, johon on lisätty joitakin Javascript -ohjelmointikielellä ohjelmoituja objekteja. Fyysisesti järjestelmä koostuu kannettavasta tietokoneesta, kontaktimikrofonista, ulkoisesta äänikäyttöliittymästä ja pöydästä tai vastaavasta pinnasta johon mikrofoni kiinnitetään.

Työ alkaa ihmisen kuulojärjestelmän rytmisen puolen perusominaisuuksien tarkastelulla, jota seuraa siihen liittyvien laskennallisten alitehtävien tarkastelu. Laskennallisiin algoritmeihin liittyvän aikaisemman tutkimuksen käsittelyn jälkeen esitellään uusi vuorovaikutteinen järjestelmä, joka koostuu osin aikaisemmin käytetyistä, osin uusista algoritmeista. Uusista merkittävin on kontekstiriippuvaista rytmikvantisaatiota tekevä algoritmi, joka perustuu harvoin huomioiduille musiikkikognitiivisille periaatteille.

Lopuksi esitellään toteutuksen korkeamman tason vuorovaikutteiset toimintalat, jotka ovat 'säestystila', 'soolotila' ja 'kysymys-ja-vastaustila'. Kaikki toimintatilat simuloivat erilaisia sosiaalisia toimintatiloja, joita tyypillisessä vuorovaikutteisessa improvisaatiotilanteessa on.

Työ keskittyy kuvailemaan toteutuksen yksityiskohdat. Esimerkiksi käyttäjäkokeita tai subjektiivisia analyyseja ei tutkimuksessa ole. Olennaisimpana on pidetty niiden ongelmien esiintuomista, joita laskennalliseen musiikilliseen vuorovaikutukseen liittyy ja testata erilaisia algoritmeja käytännössä.

Avainsanat: Akustinen signaalinkäsittely, audiojärjestelmät, musiikki, psykologia, vuorovaikutteiset järjestelmät

Preface

This thesis was carried out in the Department of Signal Processing and Acoustics of Aalto University School of Electrical Engineering in 2010–2011. The research was funded by the Academy of Finland as a part of the project *Synthesis, Control, and Hierarchical Event Modeling Algorithms for Sonic Interaction Design* (Schema-SID) (project number 120583).

Contents

1 Introduction

The objective of the research is to study the different musical properties present in musical rhythmic user input and based on this, implement a real-time system capable of retrieving, analysing and reacting to the input with a rhythmic output of its own. This results in an interactive system where the user affects the system and moreover, the user is reacting to the musical output and ideas from the system. The special case of musical rhythmic interaction of this study relates on a more general level to the broader research field of sonic interaction design, which concentrates on the use of non-speech sound as the main means for conveying information, meaning and aesthetic/emotional qualities in interactive contexts [1].

The thesis approaches rhythmic interaction from the perspective of a special case, musical table drumming. A focal point in creating this type of interaction system is to understand and build working computational models for rhythmic perception. From a computational point of view, such simple tasks for a human listener as for example tapping and counting along a piece of music, are not trivially simple but need complex processing including onset detection from an acoustic signal, beat detection, using different musical cues to decide on the metric structure, and so on. Extracting different rhythmical parameters, such as tempo, meter and accents from a continuous stream of audio input has proven to be fairly challenging and unresolved issues remain up to date.

Even though the application at hand is looking on the musical side of things, the research can also provide useful information on several non-musical contexts. For example, the presented gesture detection algorithms could simply be used as such in order to create a new user-interface or to enhance an existing one. More wide-ranging effects are imaginable: even in non-musical applications, the rhythmic structures of music emerge. For instance, in [2], a cutting system was sonified by producing sonic output on an up beat rhythm compared to the cutting rate, thus realising in the process simple rhythmic cognition. Similar non-musical tasks can easily be thought of, which necessitate an understanding of rhythmic properties of sound. The musical processing of human brain is not isolated from other auditory processing; it is an essential and tightly integrated part of the whole. For example, [3] shows that there exists a connection between basic psychoacoustical source separation and higher-level music perception. The importance of understanding how the musical processing functions is momentous, opening many possibilities for human-computer

interaction.

1.1 Table drumming

Table drumming as an input mode presents an easy access to rhythm, removing a multitude of variables such as controlling the pitch of an instrument from the mix, but still maintaining important features intact. Even though the act of hitting a table is conceivably one of the simplest (and most popular) ways of producing sound, combining these seemingly simple sounds in temporally different structures provides unlimited possibilities. Adding dynamic variation such as accents and timbral changes with multiple sounds (bass/treble hit) unleashes even a bigger wealth of musical potential.

In [4], maintaining an upward path of increasing complexity was pointed out to be an important point in keeping people invested in learning the intricacies of an interface, whereas too much complexity was said to hinder the entry of beginning users. As a musical interface, the table is very simple but still provides almost as rich expressive power as typically used percussive instruments. For instance, when roughly categorised, the output of a djembe drum typically consists of only three different basic sounds [5]. Also in the contemporary drum kit, the main parts for producing beats are few: a cymbal for time-keeping and bass and snare drums for metric accentuation [6]. Musically the main importance of percussive instruments lies in the temporal structures of their parts, and fairly little emphasis is on their exact timbre or pitch.

As argued above, the table as an interface does not inherently set limits on temporal aspects of input, and is thereby fairly unlimited for producing rhythmic input. Many Internet videos demonstrating expert skills on table drumming prove the same point – they clearly show that exploring the capabilities of this seemingly simple instrument is fruitful enough to keep vast amount of people interested for extensive periods of time. For the non-limitedness it is also arguable that the musical skills learned on a table top will also transfer as proficiency on other musical instruments in a same way that general rhythm reading tasks of music pedagogy have proven to be helpful regardless of main instrument [7]. In fact, looking at the gestures of manual drum playing and table drumming seem to be very closely related with few main variations in playing position, hand velocity and hand position, so it is

conceivable that the learned motor skills would facilitate playing those instruments also.

1.2 Sonically augmented table

Proceeding onwards from the points brought out above, the main shortcoming for a table top used as a musical instrument is probably its limitedness regarding timbre – the sound palette it provides is confined to very short, impulsive sounds, which all are very alike. With the aid of modern digital signal processing and electroacoustic tools, the original sound of an object can be detached and replaced with a completely unrelated electroacoustic substitute. In other words, using concepts adopted in the field of sonic interaction design, the objects can now easily be *schizophonic* (as originally defined in [8]). This can be achieved by coupling the objects with sensors suitable for distinguishing different gestures and catching their important parameters with sufficient accuracy, followed by corresponding output.

In the case of a table top, a single microphone signal should be enough for distinguishing different gestures from each other and also getting some category independent parameters such as intensity out of them. The processed microphone input can be used to control sound output after the classification and parametrisation. This process of combining a new sound to existing objects to produce an enhanced soundscape can be used to augment the physical sound of a table to make the interactive scenario more engaging.

A simple sonically augmented table itself does not produce a very rich scenario for interaction, and equivalent system called Billaboop has been already presented in [9]. Billaboop is capable of learning up to three input sounds and output a corresponding sound sample with a low, maximum of 12 millisecond throughput latency. Contrary to this, the implemented system in case will create an improvisational musical output by analysing the musical content of input sounds as a whole. This requires more complex processing, including combining the single discrete input sounds to musical wholes and retrieving their important properties. A system capable of similar analysis and musical interaction has been implemented earlier in [10], where output is produced by an anthropomorphic, mechanical percussionist playing an acoustic drum. Weinberg and colleagues have also later presented a robotic marimba player with additional capability of analysing and producing melodies (as opposed to only

rhythms) in [11] and [12]. In this research the attention is directed towards the interactive mechanisms between two rhythm instrument players and collaborative rhythmic improvisation. The question is, what kind of improvisational and social interactions are present in a collaborative interactive scenario between human players, and how can these be modeled computationally in a system working in real-time.

1.3 Research scope and thesis structure

The main objective of the thesis is to study different possible mechanisms of creating enticing and communicative rhythmic output computationally using rhythmic input of a human user, and for this reason the thesis operates mainly on musical level. Sound synthesis in the strict sense of the word is ruled out along with timbre and dynamics of sound, and more importance is given on temporal parameters of rhythmic input and output.

The primary contribution of the thesis is on implementing first of all a system, which is capable of retrieving timing information (microrhythmics) from the rhythmic input (presented in detail Chapter 4). This is achieved by combining the novel, automatically scaling rhythmic quantisation algorithm and an onset detection algorithm, enabling the extraction of such information. Secondly, the system works in real-time, which is if not unique but at least rare among comparable improvisational systems (e.g. [13]). The capability to retrieve and analyse musical timing information from audio signals opens many future possibilities, including systems which are able to react to and imitate different human-like phrasing styles.

The structure of the thesis is as follows. Chapter 2 presents the theoretical background of human music perception and previously presented algorithms for the subtasks included in the implementation. Chapter 3 starts by an overview of the implemented system and then explains the sonic augmentation of the table. Chapter 4 presents the algorithms which analyse the rhythmic user input in different ways. Chapter 5 concentrates on musical improvisation and the more social behaviour modelled in the system. Chapter 6 contains final discussion and propositions for future work.

2 Rhythmic perception and related computational algorithms

This chapter presents the background of the study, starting with basic musical and psychoacoustical concepts related to the thesis and continuing on to a review of research on subjects related to different functional parts of the system.

2.1 Rhythmic perception

This section will present the basic musical concepts and their connection to the human rhythmic perception. These concepts include pulse, tempo, meter, categorisation and timing. As the objective is to build a computational system capable of "listening" to the user reacting in a human-like manner, understanding the psychoacoustic and music cognitive mechanisms of human brain is vital.

2.1.1 Beat, pulse

Beat and pulse are used synonymously to refer to regularly recurring articulations in the flow of musical time. Also the word *tactus* is used interchangeably with the two. The sense of pulse arises through the listenerâs cognitive and kinaesthetic response to the rhythmic organisation of the musical surface; in other words through the temporal perception of the onsets of musical events or "starts of notes". A clear sense of pulse is a necessary condition for musical metre, as it forms the temporal anchor for higher levels of metric structure (measures or bars marked by downbeats) as well as smaller levels of metric subdivision. [14]

The pulse is easily perceived and defined in a situation exemplified in Figure 1, where a set of impulses follow each other in a steady pace, forming a simple rhythm. In this case, the pulse can be trivially defined as the temporal relation between successive impulse onsets. In musical situations, however, used rhythms are normally more complex with a multitude of different durations, and the pulse retrieval becomes more complex. In these situations, retrieving the correct pulse is not the only question, but also the phase of the rhythm has to be judged – that means deciding on which beats coincide with the beat and which fall in between of them. The complexity of the situation is increased when many instrument parts are intertwined

with each other, and a decision has to be made whether or not their onsets represent the same metrical instant or not – small temporal discrepancies are inevitable when the rhythms are performed by humans. The issue will be treated in more detail later.

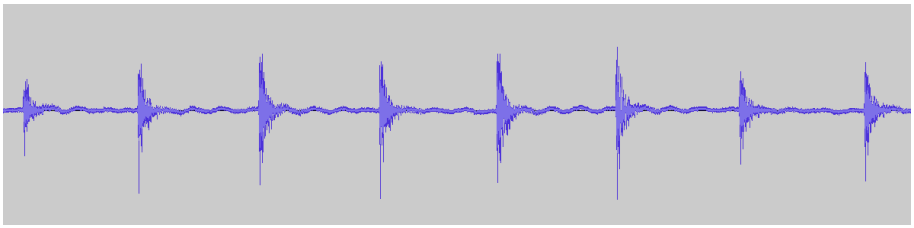


Figure 1: An audio waveform of a steady pulse. Time on the x-axis of the figure, amplitude on the y-axis.

2.1.2 Tempo

Tempo is the speed at which a piece of music is performed and marks the rate of perceived metrical beats present in it. Tempo is typically presented in BPM, beats per minute. Human perception seems to have a strong preference towards tempos ranging from 60 to 150 beats per minute where pulse is felt the strongest [15]. This may follow from the temporal limits of perception: only when a series of regular articulations in a range of 100 ms to 2 seconds are heard, does a sense of pulse arise [15]. As note durations the intervals correspond to a semiquaver at a tempo of 150 and a whole note at the tempo of 120. The range is historically argued to have a relationship to average resting heartbeat rates of adults [14].

2.1.3 Meter

Meter is the grouping of beats into a recurring pattern (measure, bar) defined by accentuation of the pattern [16]. Meter can be described as the most important property of rhythm because it functions as a framework to which all of the perceived rhythms are compared to. The connection between rhythm and meter is two-fold: in cognitive sense rhythms are perceived in comparison to the meter [17], but respectively the meter is also constructed from perceived onsets of heard rhythms [18]. In short, meter can be described to be the sense of continuity and expectation rising from the regularity of musical events, while rhythm is the mere perception of temporal placement of them [15].

Lerdahl and Jackendoff [18] describe the cognitive process of deriving meter from a rhythm with two sets of rules. The first set functions in a bottom-up fashion, building the metrical structure hierarchically upwards starting from the smallest time-duration present in music, *tatum*. Every metrical beat on a hierarchically higher level consists of two or three beats on a level below it, creating structures with binary/ternary divisions at every level. The possible metrical structures and their phase-shifted versions are then synchronised with the tatum of the music at hand, creating a tree of possible metrical interpretations. This tree is then pruned down to one remaining metrical structure based on different heuristic rules of music functioning in top-down direction. These heuristic rules are based on such musical cues and accents as the length of notes, changes in harmony or dynamic stress, and so on. After evaluation of these cues the most prominent hypothesis is picked out as the perceived meter.

Figure 2 presents three different phase-shifts for the same 4/4 metrical structure with only binary divisions between every hierarchical level of onsets, starting from the 16th note level and proceeding upwards to the measure level. The up-most metrical structure is the most likely to be heard as the meter, because there are more musical cues to support it as the meter. For example, the longer notes fall on metrically important parts, as do the lower bass notes. These kind of top-down cues are arbitrary, style dependent and learned through personal listening history. Different styles of music can use totally opposing musical rules: for example in archetypical rock music there is a strong emphasis on downbeats, whereas classical reggae styles stress the upbeat.

For this research, the most interesting property of Lerdahl and Jackendoff's explanation of meter is that every musical onset must be synchronised with the finitely precise meter which defines only discrete locations in time. The point will be taken up later on especially in the context of rhythmic quantisation.

The theory of Lerdahl and Jackendoff rules out uneven meters such as 7/8 by allowing only regular divisions at every level including the beat level. Nevertheless, uneven meters are not very uncommon in music, especially in different traditional folk music styles, jazz or progressive rock. This shortcoming could be easily attended by allowing uncommon meters, which only adds the number of possible hypotheses of meter, or computationally speaking expands the search tree of meters. However, in the scope of this research the shortcoming is accepted, as the perception of com-

plex meters will not be significant especially in the initial phases of implementing a computational equivalent to rhythm and meter perception. The most common meters presented are assumed to be even.

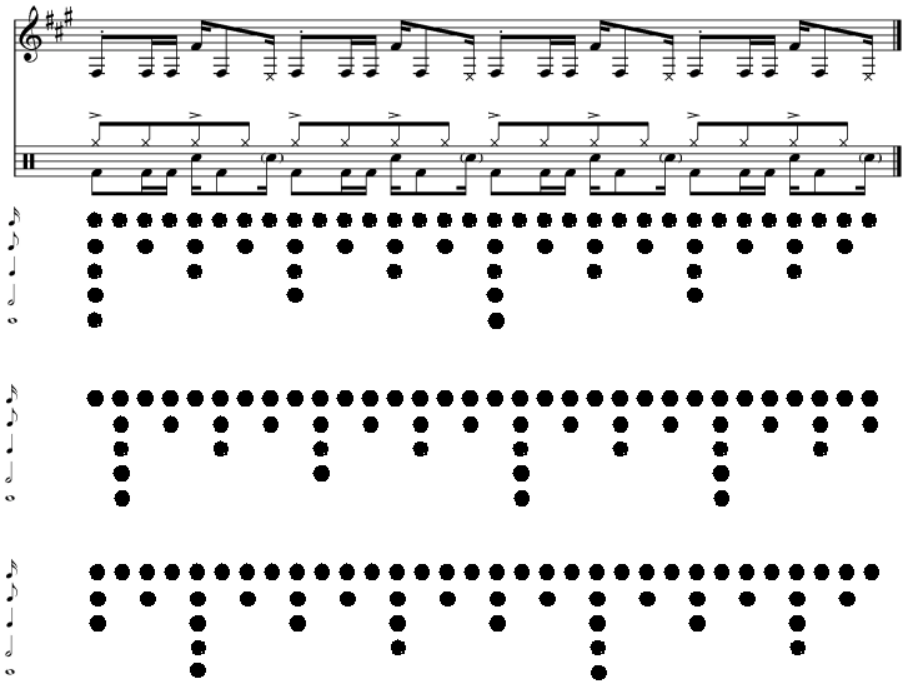


Figure 2: Figure of metrical hierarchy using Lerdahl and Jackendoff's notation.

Top-down knowledge in computational meter detection

If the perceptual structuring of meter is accepted to function in similar manner as presented in the perceptional theory of Lerdahl and Jackendoff, it could be reasoned that including analogous top-down information about music would be necessary part of computational systems executing tasks related to metrical perception. Nevertheless, current algorithms rarely seem to do so to great extent.

The need has been noticed by some researchers. Regarding beat-trackers, Hainsworth [19] points out that even style-independent beat trackers will likely have to include decision making using style-specific musical cues in the short term for performance improvements. This also suggests that such tasks as meter-related beat tracking (not only tempo tracking) need a priori knowledge about music and might not be solved using purely acoustic cues.

Goto [20] notes the importance of top-down musical knowledge for rhythmic perception on his paper presenting a music scene description system in following words:

"The estimation of the hierarchical beat structure, especially the measure (bar line) level, requires the use of musical knowledge about drum patterns and chord changes; on the other hand, drum patterns and chord changes are difficult to estimate without referring to the beat structure of the beat level (quarter note level)." The computational algorithm he presents in the paper accordingly assumes a 4/4 measure and adjusts the metrical phase by analysing drum beat patterns and positions of chord changes via spectral analysis.

2.1.4 Categorisation and timing

In the tradition of western classical music it is acknowledged that the same piece of music can be performed in several ways, which diverge from each other not only dynamically or timbrally (e.g. using different instrumentation) but on an acute level also temporally. For instance, a piece of music written in so-called straight-time performed with swinging triplet-feel is clearly the same piece of music even though temporally it contains rather large deviations from the original performance.

In Figure 3 a spectrogram of short excerpt of jazz music is presented. The figure shows that on a millisecond level, the instruments which seem to be on the same beat on perceptual ("heard") level, physically thinking are actually not. Figure 4 shows the same excerpt in music notation, transcribed both to a more "physical" 256th note precision and perceptual "heard" or rounded precision. In a group setting, players of different instruments many times choose individual positions around the overall beat: a jazz soloist can be even a 16th note behind the beat of drums and bass, and still be heard to be playing at the same metric position [21] (see also [22], [23]).

The discrepancy between the physical and perceptual impression could be explained with simply reaching the temporal limits of human auditory system. However, findings in [24], [25], [26] point out to the contrary: values of 5 to 20 milliseconds are reported as the JND (just noticeable difference) in rhythmic sequences listened by briefly trained non-musicians. Elsewhere Rammsayer [27] shows professional musicians reaching playing precisions of up to 3 milliseconds. Thus, it is arguable the human perception seems to be doing this kind of quantisation of temporal values for other reasons.

Temperley [28] presents that heard rhythms are quantised to simplified, archetypal

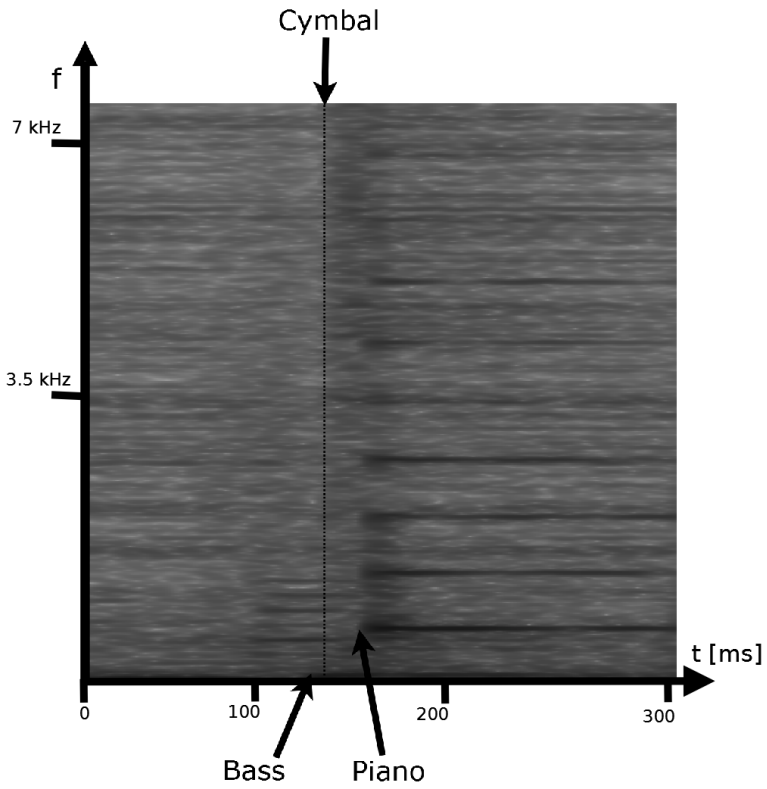


Figure 3: Spectrogram of a short excerpt of jazz music, where the small timing deviations of different instruments are visible. Frequency is on the y-axis of the figure, time on x-axis, and the darkness of the figure represents the energy present at that point.

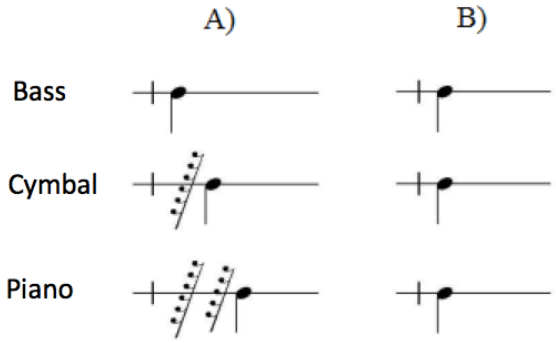


Figure 4: The previous excerpt represented with music notation with transcription precision up to a 256th note.

versions in order to achieve more efficient coding by removing excess information. Small timing differences are heard, but are only represented in long-term memory on a larger, general level (”syncopated”, ”laid-back” etc.). The quantisation is performed by locking the unideally placed temporal events to the ideal isochronous, equally spaced metric grid with finite precision.

Honing [29] refers to the action of quantisation as *categorisation*. The tests presented in the paper show that categorisation groups infinitely multiform relative durations of onsets to much fewer simple categories, where preference is on small integer relations of the durations such as 1–2–1 or 1–3–2. Figure 5 shows the situation: from the upper part of the figure, which represents the performance level of a rhythm, it can be seen that the onsets do not fall exactly on the same metrical points where they are perceived at to be in the lower part of the figure. Theoretically, the division of metrical levels could be continued until exact congruence between the two would be achieved, but Honing's tests proved this hypothesis incorrect.

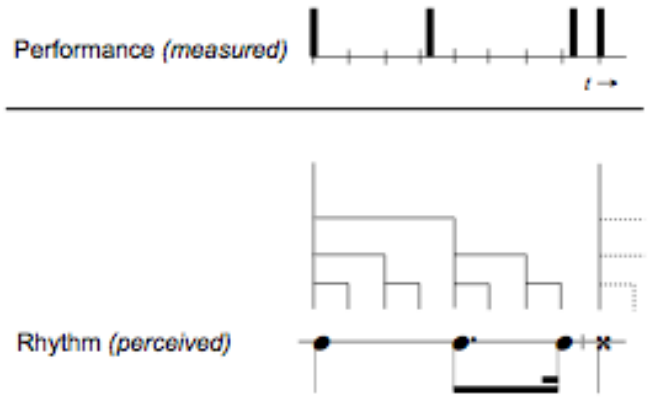


Figure 5: Example of categorisation. Figure originally from [29].

Categorisation not only explains how differently performed rhythms can be perceived as the same archetype, but also how these performances can be decoded. As rhythms are perceived in two ways, the difference between the two – quantised rhythm and performed rhythm – forms the perceived performative part of the rhythm, also referred as the *timing*. The non-strict timing of rhythms is fundamental for almost all types of music, giving it an individual sensation of liveliness, feel and flow. In contrast, the ideal and mechanical timing of sequencers or drum machines seem to many times miss this feel, even though "mechanical" is a rhythmic feel in its own.

The effect of musical context on quantisation

An interesting phenomenon related to rhythmic quantisation is that the surrounding musical context seems to affect to the resolution at which the quantisation is performed. Slower rhythms played in a physical sense 16th note after every beat will likely be heard as being on the beat (with a laid back feel), whereas faster 16th

note passages will be related to the meter on a more precise level. The number of notes per bar might be the deciding factor.

A subjective test with a simple sequencer software may be executed to illustrate this. Figure 6 shows a situation where in the A-part of the figure a rhythmically sparse rhythm is played with metronome pulses marking the on-beats. The perceptual impression is that of part B of the figure: the onsets seem to fall rather on the first and third offbeat of the measure than on the fourth 16th notes. By adding a 16th note passage after the notes as illustrated in part C, the physically identical notes of part A are now perceived as written.

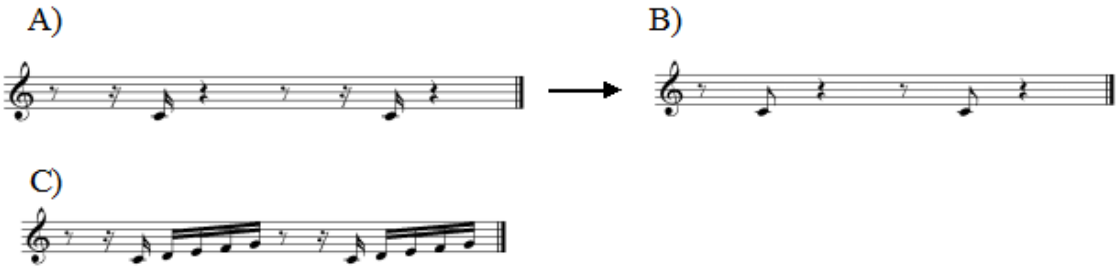


Figure 6: The effect of context on rhythm perception.

Importance of timing on musical feel

Small discrepancies are not to be seen as only human *error's*. In [22] and [21], it has been shown that individual musicians choose their position regarding to the overall beat of the music and play in that position consistently in order to maintain a certain feel. It is also illustrated in [23] that discrepancies of this type add to the "feel" of music. The analytical results of [21] also point out to the direction that the chosen overall feel is dependent of the style of music played – for instance, timing in rock music is different than in jazz.

2.2 Previous computational algorithms related to rhythm perception

The functions that the human rhythmic perception execute are diverse. This section looks into previous research carried out on implementing different computational

algorithms realising different sub-functions of it.

2.2.1 Onset detection

Onset detection in music signals refers to the action of picking up the time-instants which mark the beginning of played tones. There exists many different computational algorithms for doing this automatically. Bello and others [30] distinguish the main classes and present the principles of various onset detection algorithms based on analysis of the signal's behaviour in time-domain or frequency-domain, or after more complex processing. Five onset detection principles are compared which look at high-frequency content, spectral difference, phase deviation, wavelet regularity modulus or negative log-likelihood of the signal. The performance of each algorithm varies greatly when different test excerpts are used, varying from pitched non-percussive to unpitched percussive signals. At the highest, present algorithms can reach correct detection rates of about 98 percent when clear input signals are used – the rates decrease significantly using mixed, polyphonic signals [30], [31]. Another revision on different onset detection algorithms and their performance is presented in [31] by Dixon.

Puckette and others [32] have presented bonk~, a Max/MSP [33] and Pd [34] tool designed to detect the onsets of percussive instruments in real-time from an audio signal input. Bonk~ object does a bounded-Q filterbank analysis of the incoming sound and detects sharp relative changes without accompanying large changes in the overall power. This makes the tool very suitable for detecting percussive sounds exclusively, since the sounds of ringing instruments usually do not elicit such rapidly changing spectra. The filters of the object's internal down-sampled FIR filterbank are spaced logarithmically in frequency and can easily be parametrised in the latest versions of the object. The main advantages of bonk~ are the speed of the algorithm (5.8 ms using 256 sample windows at 44.1kHz sampling frequency[32] + processing delays) and its robustness. The object also provides access to a template matching function capable of distinguishing different input sound types.

An important part of most onset detection algorithms is an envelope follower, which detects the general dynamic variation of a signal, i.e. its *envelope*. Practically envelope followers are implemented by first full-wave or half-wave rectifying the signal and then filtering it, in effect smoothing the final result.

2.2.2 Beat tracking

Beat tracking algorithms aim to estimate a set of metrical beat times from an audio signal, which would match those given by a trained human musician [19]. Beat tracking assigns every onset a position on the metric grid on a beat level – a onset can be on a beat, upbeat or anywhere on the discrete metric grid with dual/tertiary division.

In [19], a summarising review of beat tracking approaches is presented. Methods are divided into six categories: rule-based, autocorrelative, oscillating filters, histogramming, multiple agent and probabilistic methods. Some of the methods are non-causal or only usable on symbolic (MIDI) data, and thus are not suitable for real-time beat tracking. Beat-tracking methods which work on an audio signal need to include an onset detection function.

Applications of beat-tracking vary from synchronising human and computer input to a full rhythmic transcription of input signal. B-Keeper [35] is a beat tracking algorithm system which can direct an Ableton Live tempo track to match the input of a live drummer, and is capable of detecting gradual tempo changes within a song. The algorithm measures input accuracy with Gaussian quantification between measured onset and expected onset. The phase locking of the algorithm is performed beforehand manually by deciding how many onsets are used as a count-in for each piece and which is the initial tempo.

The beat-tracking algorithm of Davies and others [36] works on audio signals and is capable of locking in to phase of given audio. This is done basically on a rule-based decision, in which most metrical beats correspond with note onsets and phase changing is rare. Internally the algorithm is using an autocorrelative method for beat tracking. The algorithm is causal and usable in real-time.

Goto and Muraoka [38] have presented a real-time beat tracking system which reportedly identified the beat with correct beat-level phase on 27 of 30 songs tested. The system is limited for 4/4 meter songs only and functions by first detecting the onsets of bass and snare drums of the drum set and then applying top-down musical information in detecting the metrical phase of the beat, such as the bass drum hits fall on strong beats and the snare drum hits on weak beats of music. This is very typically the case in substantial amount of western contemporary popular music, where the most common drum pattern is the one presented in Figure 7 or a variant of it. In

the pattern the second and fourth beats are accentuated with snare drum hits and first and third beats of the measure are marked with bass drum hits. The hihat or ride cymbal provides a consistent 8th or 4th note pulse (8ths in the figure). It is true that by using the most probable positions of snare and bass drum hits as top-down clues on beat detection, it is possible to differentiate between first and second beat or third and fourth beats of a measure, but with only this knowledge the distinction between first and third and second and fourth beats are still ambivalent. That is, the beat phase on a 4/4 measure level remains still unknown. Probably for these reasons, the authors of [38] have left the measure-level beat detection untouched in their implementation and operate only on lower beat-level.

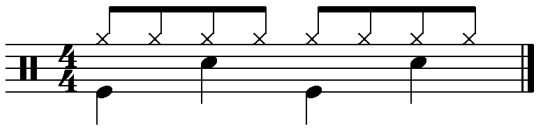


Figure 7: One of the most common 4/4 drum patterns in western popular music.

2.2.3 Meter detection

Even though the concept of beat tracking may also include the induction of meter from music, there seems not to be sufficiently robust algorithms for doing this yet. The rather complex causal algorithm (requirement for real-time implementation) of Klapuri et. al [37] reached only a 48 percent correct detection rate on a measure level of beat detection. Most beat-tracking systems work on the level of tatum, and *tactus* (beat period) and do not attempt meter detection or phase-locking. As mentioned earlier, for a system to be capable of doing this, more research should be carried out especially on including top-down musical knowledge to the system.

Having a working meter detection system at hand would open up many possibilities for such musical interaction systems as the one presented during this work. With correct detection of meter in rhythmical user input it would be possible to create a system which would follow the rhythms of the user, and consequently be more transparently reactive to what the user is doing, whereas without meter detection the consensus between the two can only be assumed and in practice left for the user. This apparently necessitates more understanding of such musical concepts from the user, thus raising the bar higher in advance.

A novel meter detection algorithm is presented in the next chapter, even though it could not be included in the final implementation for the reason of unpredictable timing of the used Max/MSP environment. The initial version of the algorithm does meter detection between 4/4 and 3/4 meters and is able to lock the beat into correct phase with user input. The algorithm is based on top-down musical cues.

2.2.4 Rhythm quantisation algorithms

A robust and working rhythm quantisation algorithm is an important part of creating an interactive system capable of listening and reacting to the user input. Rhythms performed by humans are rarely exactly precise as shown previously, for motoric reasons and more importantly for aesthetic reasons of creating a rich interpretation of certain piece.

Desain and others [39] present a rhythm quantisation algorithm, which implements human behaviour based on listening tests on professional musicians. On a computational level, the algorithm uses a vector quantisation method, where a set of consecutive onsets falling into a certain time-frame form a vector, and based on likelihood measurements, a corresponding "allowed" vector or rhythm is then output. No extensive testing was taken on with the algorithm, but it is said that on a limited test corpus it works in a human-like manner on the rhythm quantisation task.

Cemgil and Kappen [40] have created a tempo tracking and rhytm quantisation model using sequential Monte Carlo integration (particle filtering) techniques. The model is reported to work "correctly" with a short clave rhythm and a piano arrangement of the Beatles song Yesterday. The model works on symbolic information, for example data collected from a MIDI device.

A new, automatically scaling rhythm quantisation algorithm is implemented and presented in the next chapter. The algorithm works on symbolic information in real-time and is used for extracting performative timing information from user input.

3 Sonic augmentation

The three following chapters describe the design and implementation of the interactive sonically augmented table in detail, divided correspondingly in three parts. Chapter 3 treats the low-level sonic augmentation of the table, Chapter 4 the algorithms related to the higher-level rhythmic analysis of user input, and Chapter 5 the basis of interactive rhythmic output and the social musical behaviour modelled in the process. This chapter starts with a general overview of the whole system, and then progresses into a more detailed treatment of the algorithms used in the sonic augmentation, including the decisions made during the design process.

3.1 General structure

The structure of the system is represented in Figure 8. The system gets rhythmic input from the user playing a table top, giving an input audio signal. This audio signal goes through different onset detection algorithms, which give out exact time points of the impacts on the table. The frequential features of the audio signal are analysed also, producing gesture classification. In other words, analysing the audio input only, the mechanical-physical way of hitting the table with different hand configurations can be detected. For example, hitting the table with knuckles produces a sound, which differs from the sound of hitting the table with fingertips.

The combination of the gesture classification and onset detection can be used to trigger selected audio samples in real-time synchrony with the rhythmic input, thus producing an experience of a schizophonic table. Onset times are also used as a basis for tracking the tempo of the input, which will affect the global tempo of the system.

For producing a metrically solid interpretation of the absolute onset times, rhythmic quantisation is needed. This output with gesture classification information is used for analysing the rhythmic features of the input, such as metric accentuation and used note values. When the absolute onset times and metrically quantised onset times are known, the difference of these can be used in further analysis of the timing information (microrhytmical features) present in the input. This includes for instance the swing ratio of the eighth notes and relation to the beat (on beat, behind the beat, before the beat).

The parsed rhythmic and timing (microrhythmic) features of the user input can finally be used in the rhythmic interaction model. The model will output an improvisational rhythmic output, which creates either variably stochastic/imitational rhythmic solo or an accompaniment part based on a database of rhythmic patterns. This closes the interactive loop and provides a rich scenario for interaction, especially as the output is capable of introducing totally new improvisational ideas for the user to build upon.

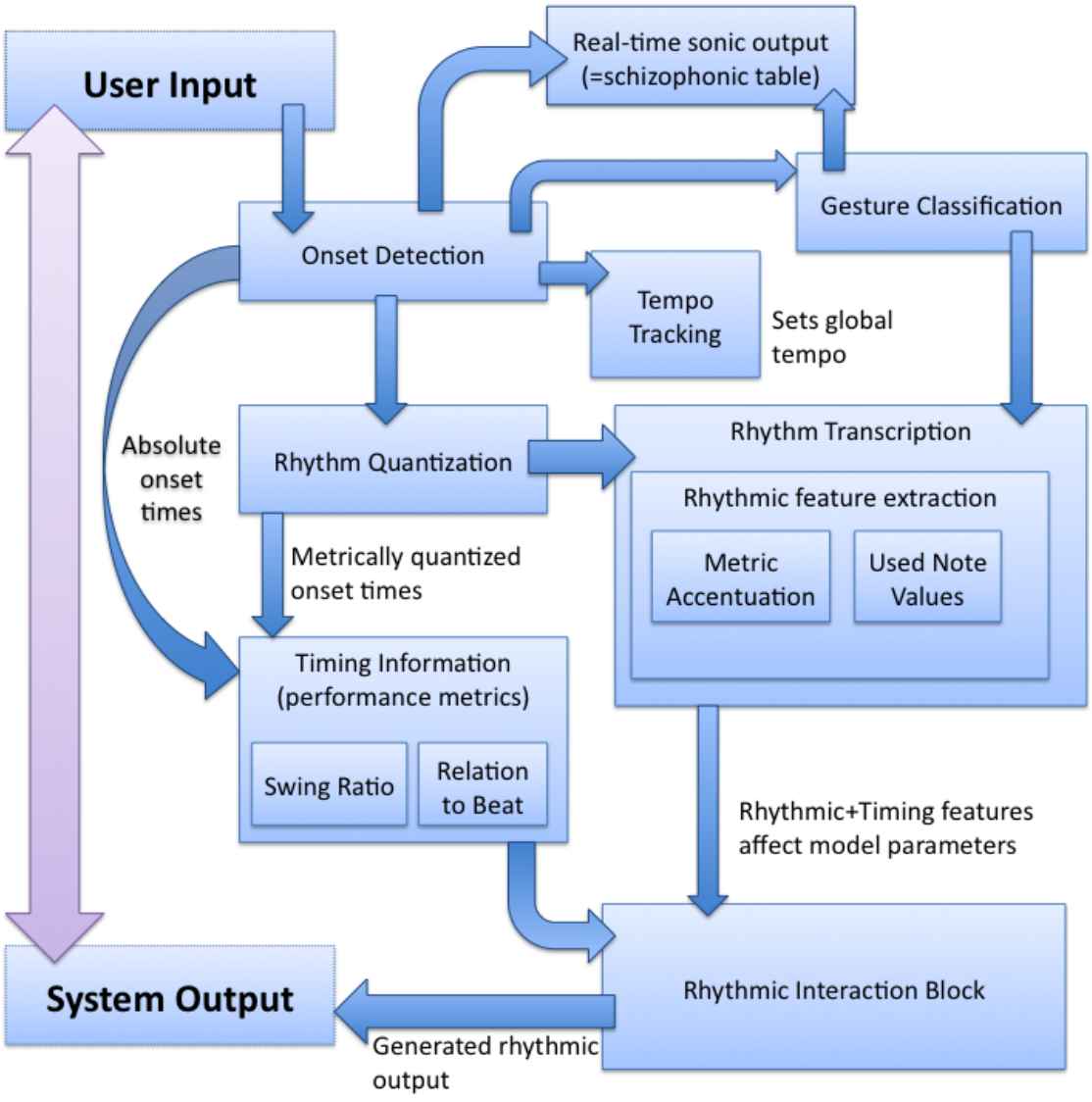


Figure 8: Functional parts of the system.

3.2 System setup

Physically the system consists of a contact microphone attached with adhesive wax to a table top. The microphone is connected to a Motu 896mk3 external audio interface, which provides phantom-power for the microphone, pre-amplifies the signal and executes analog-to-digital conversion. The audio signal is processed with a Macbook Pro laptop computer attached to the audio interface via a firewire cable. The main programming environment is Max/MSP software version 5. Some objects of the main Max/MSP patch are implemented with Javascript, as array processing, for-loops and many other functions are difficult if not impossible implement using only default Max/MSP objects.

3.3 Onset detection with bonk~

In the previous chapter, bonk~[32], a percussive instrument onset detection object for Max/MSP and Pd was introduced. During initial tests its performance especially latency-wise was proven to be sufficient with correctly chosen parameters. Because of the high rate of correct onset detections/false detections and small latencies achieved using bonk~, there seemed be no need to implement a new algorithm for detecting the onsets of rhythmic user input.

Bonk also includes a template matching function, but because the initial tests with it showed already with two different sounds a high rate of errors, external parts were to be implemented. Table 1 shows correct/false detections for a test rhythm consisting of 57 onsets, 27 of which were bass taps and 30 treble taps. As can be seen, the algorithm reached only 75.4% total accuracy, which is not suitable for practical purposes.

Table 1: Initial gesture detection rates with template matching function of bonk~. The test rhythm consisted of 57 total impacts. 27 of them were bass taps and 30 treble taps.

Gesture	Correct detection rate	False detection rate
Bass tap	48.1 %	0.0 %
Treble tap	100.0%	51.85 %
Total	**75.4 %**	**24.6 %**

3.4 Gesture classification

This section will show the gestures the system detects, how they are produced and how they were chosen. Also analyses of the spectral contents of the gestures are presented.

3.4.1 Choosing the gestures

In the primary design phases of the implementation a decision was made to keep the number of possible gestures in the system low. For one, referring to the introduction, it is musically arguable that already three different sounds is a working basis for a multitude of musical possibilities (djembe, drum kit). On the other hand, minimal gesture set keeps the system easily accessible for beginners. The final motivation was to keep the gesture detection algorithm performance high and algorithm design simple.

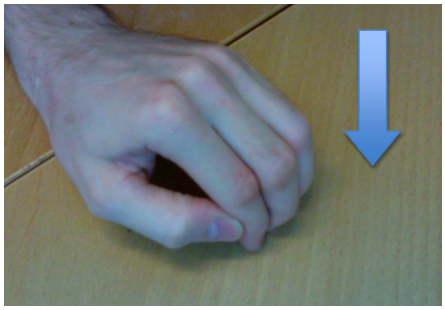
After initial analysis of different table drumming techniques and possibilities, three different gestures were chosen to be detected by the system, namely a *bass tap*, a *treble tap*, and a *sweep gesture* (see Figure 3.4.1). Bass tap (Fig. 9(a)) is produced by tapping the table top with a "fleshy part" of the hand, such as fingers or the palm of the hand. The treble tap gesture differs from the bass tap in that the table should be hit with a harder part of the hand such as the fingernails (Fig. 9(b)) or the knuckles (Fig. 9(c)). The third gesture, sweep (Fig. 9(d)), is produced by brushing the table top with for example the tips of the fingers.

3.4.2 Spectral analysis of the gesture set

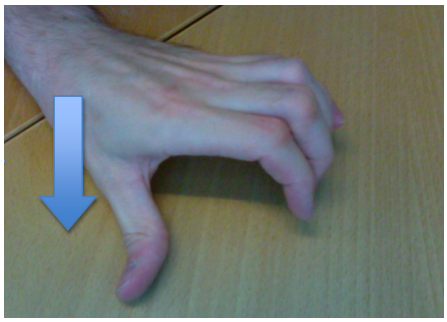
In Figure 10, a spectrogram of a sound sample of a bass tap gesture (Fig. 9(a)) recorded with a contact microphone is presented. As can be seen from the figure, the sound is very impulsive with very short attack and decay times, even though some resonances on the lowest frequencies are present throughout and even after the observed interval. The spectrum of the attack has a peak at the lowest frequencies, which falls rapidly towards about 1.5 kHz. Higher peaks can be found above the frequency of 10 kHz.

Figure 11 is a spectrogram of a treble tap gesture (Figs. 9(b), 9(c)) recorded with

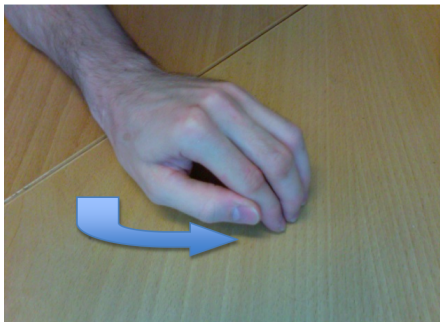
(a) Bass tap gesture with fingertips



(b) Treble tap gesture with fingernails



(c) Treble tap gesture with the knuckle of thumb



(d) Sweep gesture with fingertips

Figure 9: The figure presents the techniques of the three different gestures detected by the system. The gestures are a bass tap(a), a treble tap (b), (c), and a sweep gesture (d)

a contact microphone. It can be seen from the figure that in this manner the sound has slightly more energy in the frequency band of 1.5 kHz through 10 kHz than the bass tap. The treble tap gesture is impulsive in the same manner as is the bass tap.

The third gesture, sweep (Fig. 9(d)), is represented with a spectrogram in Figure 12. Unlike the two gestures above, this gesture is non-impulsive, and continues to have energy throughout the whole motion of sweeping the table. The spectrum seems also noise-like, with atypically high amount of energy in the highest frequencies over 10 kHz.

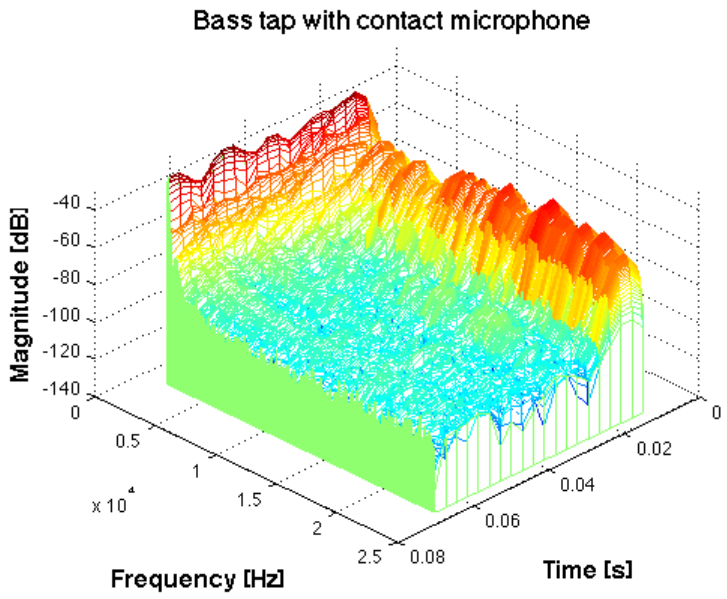


Figure 10: Spectrogram of a bass tap gesture recorded with a contact microphone.

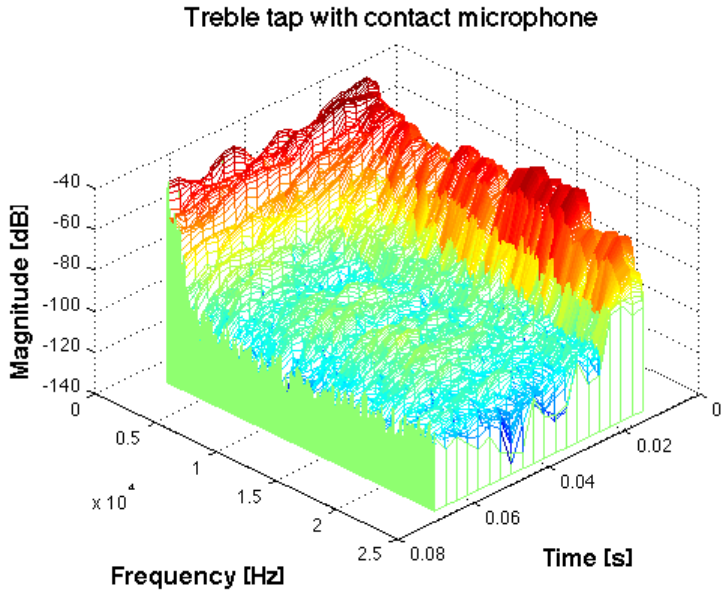


Figure 11: Spectrogram of a treble tap gesture recorded with a contact microphone.

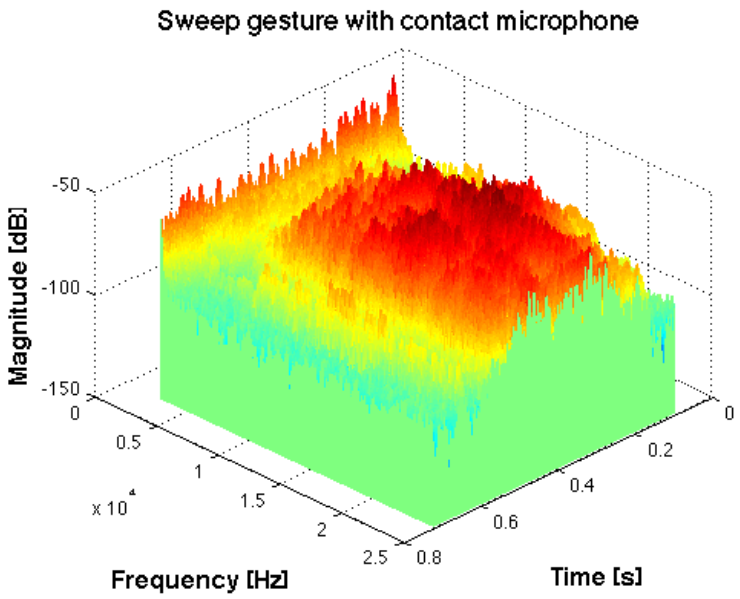


Figure 12: Spectrogram of a sweep gesture recorded with a contact microphone.

Differences of the gestures

The differences of the gestures can be summed up with the following table:

Table 2: Differences of the gestures.

Gesture	High-frequency content	Impulsive	Continuous
Bass tap	No	Yes	No
Treble tap	Yes	Yes	No
Sweep	Yes	No	Yes

From Table 2 it is clearly visible that the bass tap and treble tap could be detected for instance with a system consisting of impulsive onset detector (e.g. bonk~) and a spectral analyser. For the sweep sound, the sound has to have high-frequency content, but impulsive sounds must be ruled out – the detection of this gesture can be achieved with a combination of a high-pass filter and an envelope follower.

3.5 Implemented gesture classification algorithms

As previously stated, the bonk~ Max/MSP object, which is used in the system for onset detection also includes a pattern-matching functionality. The matching algorithm works by comparing the energies of the filterbank frequency bands weighed by their "clarity" to existing spectral templates, and choosing the best-matching template. The tool is capable of learning new patterns via supervised learning, and the templates can be saved to a separate file for later use. Even though all this sounds promising, in practice the pattern-matching functionality of bonk~proved to be making too much errors already on a minimal set of two different gestures (a bass hit and a treble hit of a table, see Table 1), so alternative techniques were to be found. This section describes the implemented gesture classification algorithm included in the system.

3.5.1 Tap-detection: averaged spectral templates

Because bonk~ already inherently does a filterbank analysis of the input signal on detected onsets, a solution was built based on this in order to avoid additional computational load. The approach is in general very similar to the one found in bonk~, and uses supervised learning to get spectral templates for two differing gestures.

For example, the bass gesture can be taught to the system by giving 10 bass taps, which all have some spectral variation in them – percussive sounds are not normally very uniform spectrum-wise. The filterbank energies of the different taps are accumulated and divided by the number of given taps to provide a mean value for each filter, which form an average spectral template of the gesture. The same procedure is repeated for the other tap gesture in order to get also its average spectral template. An example visualisation of the templates in Max/MSP is presented in Figure 13.

3.5.2 Choosing the least similar filterband pair

After this, among the 8 logarithmically spaced filterbands of the filterbank (filter number set to 8 in bonk~), a filterband pair which has the most relative energy change in them is taken to be the basis for comparison outside the learning phase. This is performed by calculating the energy ratios ($R_b[x][y]$, $R_t[x][y]$) of the previously averaged filterbank templates of given input gesture (T_b, T_t) between every

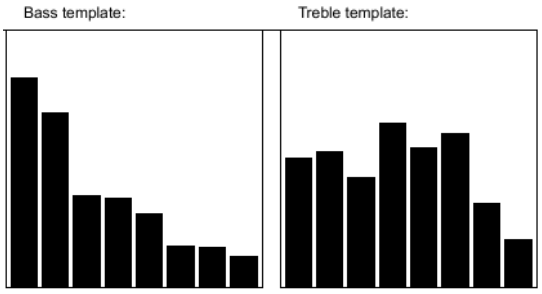


Figure 13: Example bass tap and treble tap spectral templates in Max/MSP. Every bar represents certain frequency-band, from lower to higher in left to right direction. The height of the bar indicates the average energy found in the band. The bass tap template is situated at the left and the treble tap template at the right.

unique filterbank pair:

$$R_b[x][y] = \frac{T_b[x]}{T_b[y]} \quad | \quad (x = [0, 7], y < x) \tag{1}$$

$$R_t[x][y] = \frac{T_t[x]}{T_t[y]} \quad | \quad (x = [0, 7], y < x) \tag{2}$$

The variables x and y in Equations 1 and 2 are the indices of the filterbands. Because of the uniqueness condition, the ratio matrices ($R_b[x][y]$, $R_t[x][y]$) are upper-diagonal, excluding the values on the diagonal: relation a/b equals the inverse of b/a, so only one of them is needed, and relating a filter energy to itself always results in 1.

The relative change $\Delta R[x][y])$ between the two gestures is calculated with

$$\Delta R[x][y] = \frac{R_b[x][y]}{R_t[x][y]} \quad | \quad (x = [0, 7], y < x) \tag{3}$$

The ratio matrices ($R_b[x][y]$, $R_t[x][y]$) are taken from the calculation of Eqs. 1 and 2. The maximum of this difference matrix ($\Delta R[x][y]$) gives the filterbank indices used for comparison after the learning phase. The energies of the pair of filters pointed by the maxima are taken from both the bass gesture and the treble gesture template (T_b, T_t), and an average value is calculated. This value functions as a limit on deciding which gesture was input, when the input sound's energies in the two filterbands are compared. This happens automatically triggered by bonk~.

The approach is simple and uses only a small part of the available spectral information, but it seems to be robust enough to provide good accuracy between the hits (results in Table 3, discussed later). A good feature is also its simplicity, which makes it easy to implement and rapid to calculate. In a real-time system such as this, speed is a limiting factor when different algorithms are compared. A more complex algorithm with better off-line performance might have to be ruled out for the reason of too slow operation.

3.5.3 Sweep gesture detection

The non-impulsive sweep gesture cannot be detected with bonk~which is suited only for the detection of percussive sounds. Luckily basic Max/MSP objects can be used to build a detection algorithm for the sweep. The biquad~object is a two-pole two-zero filter (for basic digital filter theory, see [41]). Figure 14 shows its magnitude response on logarithmic scale to an uniform white noise input.

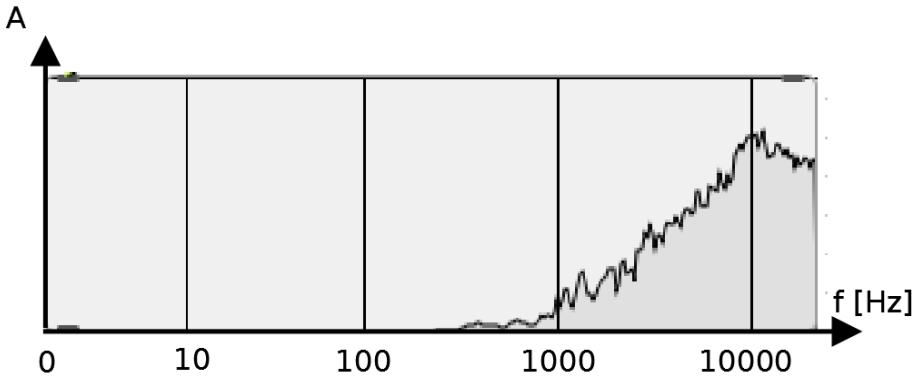


Figure 14: Magnitude spectrum of a white noise signal filtered with a high-pass biquad filter. X-axis represents logarithmic frequency, y-axis magnitude.

The filtercoeff~object can be used to calculate the multiplicator values of the filter parameters. As can be predicted from Figure 12, using very high cutoff frequency, for example 10000 Hertz, works also in practice well for the sweep gesture. By directing the raw input audio signal through the filter, using an envelope follower built with abs~and rampsmooth~objects (full-wave rectification, low-pass filtering), and finally calculating the time this signal has been above a certain threshold value, the continuous high-frequency energy of the sweep can be detected. The most important parameter for the detection is the time spent above the threshold: too small value will create false detections with impulsive sounds also, and too large value will be perceived as a clear delay between the gesture start and sound output.

Similar high-frequency filtering was used in an user-interface system, *Scratch Input* presented in [42], where it proved to be robust and working from long distances also. Scratch Input includes a detection algorithm for multiple gestures, which is based on their amplitude profiles. The gesture set of the system at hand could be extended with similar, more sophisticated sweep profile detection.

3.6 Sonic augmentation

For every detected gesture, the system will output different drum samples, sampled from either a contemporary drum kit or conga drums dependent of the users choice. The drum sounds were recorded by the author, but the conga sounds were picked from a public domain sample library [43].

Sampling (see [44]) as a drum sound synthesising technique creates at best very realistic results, as proved for example by the software EZDrummer [45]. The live-like quality of drum sounds in the software is achieved by sampling each drum sound with multitude of different velocities. Through this, the output has enough variance and avoids the noticeable repetitiveness present in many drum machines using a smaller sample bank.

Sampling is especially suited for percussion instrument sound synthesis, for one because of the nature of playing them – the sound of a percussion instrument is very rarely modified after initial excitation. On the contrary, instruments which require constant control on produced sound, namely string and brass instruments, rarely sound very realistic using solely sampling techniques. Another reason to favor sampled sounds is that in order to create equally convincing drum sounds with real synthesis techniques (sampling is not always regarded as one, i.e. [44]) requires at the moment excessive computational power to be run in real-time. For example, [46] reports 1–16 seconds of processing per one second of audio.

To avoid the above mentioned problem of repetitiveness of output sound (identically repeated samples may become unexiting and even annoying after a short while), every sound has 3–4 variants. The choice between these is made with a random-number generator.

The bonk~ object does provide an intensity measurement, but it showed too much inconsistency between similar hits to be used in the system. With the intensity

mapping, the same gesture motions could be used again for many times to create even more gestures (i.e. hard bass tap, quiet bass tap, hard sweep etc.), but it would again adhere with the ease-of-use kept in mind during the design process.

3.7 Latency and detection rates

Running the whole patch with all implemented algorithms, the latency from an input gesture to an output sound was measured to be 35 milliseconds. Perceptually, using percussion sounds this was found to be an acceptable latency, and the delay was not very noticeable or annoying. Detection rates with three gestures and a rhythm input consisting of 28 bass taps, 16 treble taps and 20 sweeps are presented in the Table 3. Even though the detection rate is perfect, it has to be noted that the user at question (the author) has taught the system the exact gestures they are going to use, and has extensive practice using them. Brief tests on unexperienced users who are using non-personalised gesture detection templates, do errors on a remarkably higher occasion. However, the case is comparable to the initial template matching results done with bonk~, presented above in Table 1, and it proves notable improvement. The only variable is the microphone setup – in the initial tests normal condenser microphones were used.

Table 3: Gesture detection rates with implemented algorithms. A rhythm consisting of 28 bass taps, 16 treble taps and 20 sweeps was used.

Gesture	Correct detection rate	False detection rate
Bass tap	100.0 %	0.0 %
Treble tap	100.0%	0.0 %
Sweep gesture	100.0 %	0.0 %
Total	**100.0 %**	**0.0 %**

4 Analysis of rhythmical properties of user input

This chapter describes the details of different algorithms, which extract higher-level rhythmical information from the user input. This includes tempo tracking, rhythm quantisation, timing information, metrical accents and used note values of the input. Finally a meter detection algorithm is proposed, although it is not included in the final implementation for the reason of synchronisation problems in the used Max/MSP environment.

4.1 Tempo tracking

The tempo detection algorithm functions as follows: a distinct number of previous inter-onset-intervals (IOIs), meaning time between consecutive onsets, are stored into an array. The array values are transformed from milliseconds to corresponding tempo (in BPM, beats per minute) with

$$tempo_{\text{bpm}} = 60 \left(\frac{1000}{IOI_{\text{ms}}} \right) \tag{4}$$

The variable $tempo_{bpm}$ is the resulting tempo in BPM, and IOI_{ms} the interval between two onsets. The tempo values produced with this transformation are accumulated into a discrete tempo histogram. After this, the histogram is time-averaged by convolving the array values with a Gaussian function $N(j)$ [19], giving a smoothed histogram:

$$h = \sum_i o_i \star N(j) \tag{5}$$

The variable h in the equation is the smoothed histogram, j the index of the window function and i the index of the unsmoothed histogram. Operator $\star$ denotes the convolution function.

The advantage of this procedure compared to a unsmoothed tempo histogram function (as for instance in [47]), is that it will allow to regard closely matching tempo values as the same intended tempo. Additionally, it can produce intermediate tempos compared to the initial histogram. Simple histogramming tempo detection is

prone to making "quantising error" type of mismatch between tempos, proportionate to the histogram bin size.

As stated in Chapter 2, human tempo perception has a strong preference towards tempos varying from 60 to 150 beats per minute. The algorithm addresses this by giving an emphasis to tempos in this range, in a sense a resonance. This is achieved by scaling the smoothed histogram $h(j)$

$$t(i) = h(j) * (1 + \rho)0.5 \left(1 - cos\left(\frac{2\pi i}{n-1}\right)\right) \quad |\{i = [0, ..., n-1]\}, \tag{6}$$

that is, with a Hanning window function (see [41]) in the range of 1 to ρ. Here $t[i]$ is the final tempo array, ρ is the resonance factor, $n-1$ the resonance array length (depends on tempo step size) and i the index number. The maximum is then picked out from $t(i)$ as the most likely beat period. The Figure 15 presents a Max/MSP visualisation of $t(i)$ calculated in this manner.

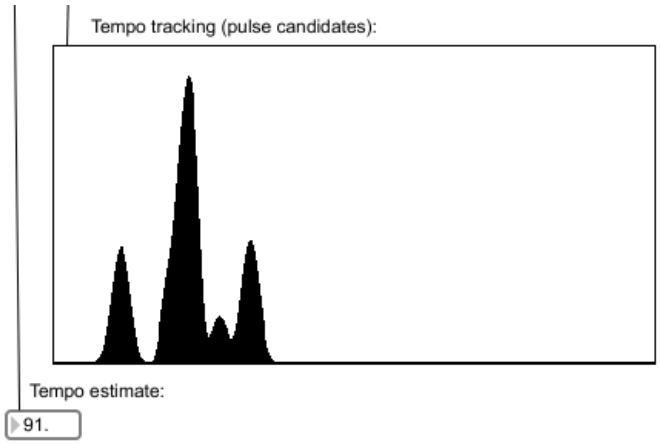


Figure 15: A real-time visualisation of the tempo tracking algorithm in Max/MSP. X-axis presents tempos from low to high, y-axis is the magnitude related to the number of intervals in each bin, convoluted with a Hanning window function for averaging the results and improving robustness.

4.2 Rhythm quantisation algorithm

Rhythm quantisation refers to the action of relating the onsets of the input signal which are temporally situated on an infinitely precise, analog grid to a more abstract, rough presentation on a discrete metrical grid with binary/ternary divisions between metrical levels. When looking at the input signal on a millisecond-level precision,

the onsets rarely fall precisely on the beat, but can still be clearly heard to represent them (see Chapter 2).

With previously discussed onset detection mechanism the absolute onset times of the user input can be picked out and related to the Max/MSP 'transport' object, which operates as a MIDI timing mechanism of different patches. Every note will then be represented on a tempo-relative 'bar/beat/ticks' -scale, where the native tick resolution of Max/MSP is 480 ticks. That means that the temporal resolution of Max/MSP regarding 'transport' related actions is tempo-relative, where every beat is divided into 480 parts forming a discrete grid.

4.2.1 Prequantisation

The implemented rhythm quantisation algorithm starts by mechanically rounding the values to the closest $480/9 = 53.33$ marker on a beat level (prequantisation). The division of the beat to nine parts is chosen in order to minimise 8th note triplet onset quantisation error. On the other hand this leads to some additional quantisation error on dual rhythms such as eighth notes or 16th notes, but this is seen as a functional compromise to detect 8th note triplets correctly. Many times rhythm quantisation in computational music systems is done only in this manner by choosing a robust enough discrete metrical grid and then quantising the rhythms to the closest point at the grid (e.g. the robot-marimba player of [12]).

4.2.2 Ideal rhythm templates

After mechanically rounding onset times the algorithm proceeds through the collected rhythmic user input data in quarter note spans, looking at pre-selected time window at a time (for example 4 bars). For each quarter note span (beat), the number of onsets inside the span in the prequantised data is calculated. The number of onsets will be used as a guidance for choosing which ideal onset time template will be used for comparison for further quantisation on the only prequantised onset times. The number of onsets will in effect provide a musical cue on what level of precision the rhythm quantisation should be performed. This simulates the context-dependent behaviour of the human auditory system regarding quantisation, previously presented in Chapter 2 and Fig. 6.

The following tables 4, 5, 6, 7 represent the different quarter note span templates.

Table 4: 1 note templates of the rhythm quantisation algorithm.

1 note templates		
Template number	Template rhythm	Explanation
1		On this beat
2		Rest + next onbeat
3		Offbeat

Table 5: 2 note templates of the rhythm quantisation algorithm.

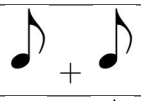
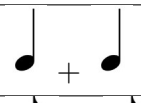
2 note templates		
Template number	Template rhythm	Explanation
1		Eighth note pair
2		Offbeat + next onbeat
3		Onbeat + next onbeat
4		16th note pair on onbeat
5		16th note pair on offbeat

Table 6: 3 note templates of the rhythm quantisation algorithm.

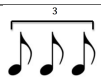
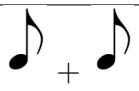
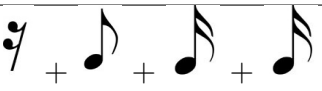
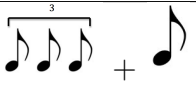
3 note templates		
Template number	Template rhythm	Explanation
1		8th note triplet
2		8th note pair
3		8th note + 16th note pair on offbeat
4		16th note rest + 8th note + 16th note pair (last on next onbeat)
5		16th note pair on off beat + next onbeat

Table 7: 4 note templates of the rhythm quantisation algorithm.

4 note templates		
Template number	Template rhythm	Explanation
1		16th note quartet
2		8th note triplet + next onbeat

As it can be seen, some of the templates are longer than the used quarter note span, and contain beats on the next beat/span. The reason for having these templates is to allow notes very close to the next beat to move there on the grid, as they are in some situations more likely to be slightly "rushed" notes than extremely delayed, "laid-back" notes. The templates containing the prescribed beat-shifts will be used only if there does not exist a beat on the next half span, in order to avoid duplicate rhythm values.

4.2.3 Error calculation

The error between the real onsets and ideal template values is calculated as a sum of differences between the absolute values of each onset time pair:

$$\Delta e_{T_i} = \sum_{n=0}^{N-1} |x[n] - T_i[n]| \tag{7}$$

In this equation, $x[n]$ is the vector containing the real onset times, $T_i[n]$ is the vector containing the ideal onset times with i different possibilities, Δe the resulting error and N the number of onsets found in the span. After the calculation of errors Δe for each template, the one with the smallest Δe is chosen to be the perceived sequence in current beat span.

Figure 16 shows the connection between onset counts and choosing the ideally spaced templates to be matched with each beat. The downbeat times are in relation to the point of Max/MSP 'transport' object timing, shown by dotted vertical lines in the figure. The first beat has two detected onsets, and the first half of the next beat does not have a detected onset, so all the five 2 note templates will be used for Δe error calculation using Equation 7. The template with the smallest error, two eighth notes will be chosen as the quantised representation of the first beat. Second beat of the figure has only one onset, so the 1 note onset templates will be compared to the prequantised onset timing during the span, resulting finally in upbeat eighth note quantisation.

4.2.4 Rhythmic precision

The templates (tables 4,5,6 and 7) have been created from the starting point of allowing only up to 16th note rhythms in one quarter note span. This is arbitrary, but for most cases sufficient constraint. The number of onsets inside every note span affects the templates used, and also indirectly the quantisation precision: for example, two-note templates use only 8th note precision. This manifests the cognitive effect of context-related categorisation presented previously in Figure 6 in Chapter 2.

Because the algorithm functions only in quarter note spans, rhythms which are created by dividing metrical levels higher than this will not be quantised correctly.

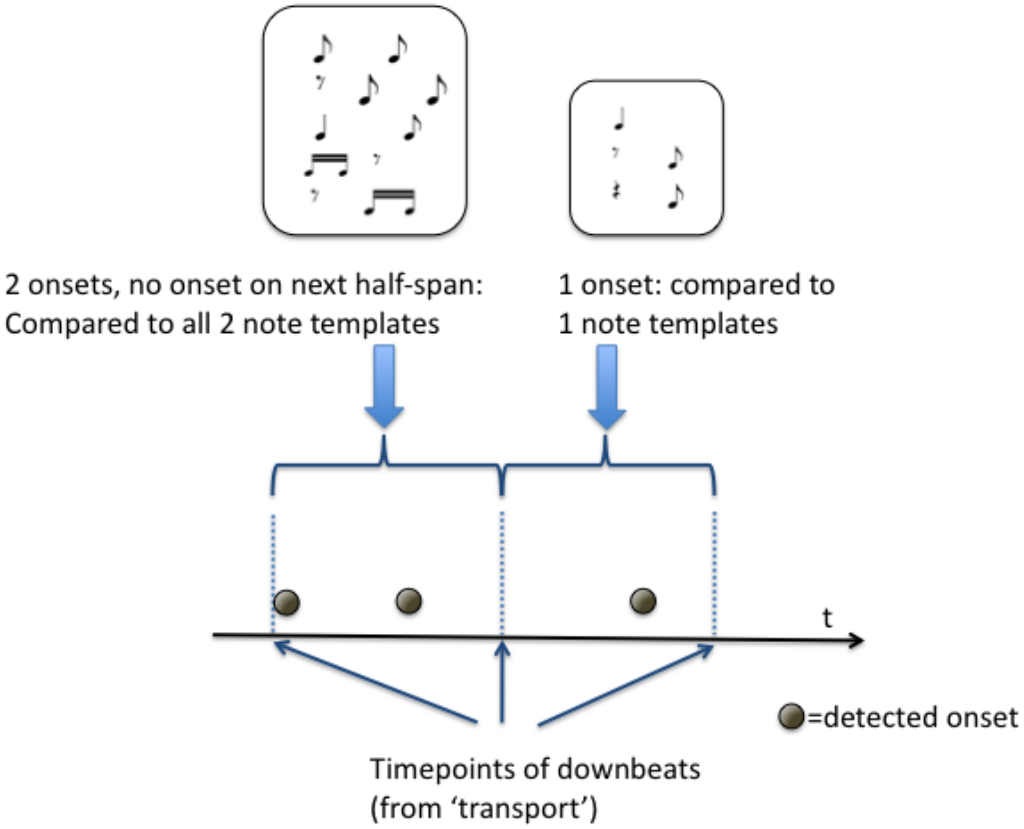


Figure 16: The connection between detected onset numbers and referred ideal templates.

For instance quarter-note triplets, which divide a span of a half note into three equal durations, will most likely be quantised as a combination of an eighth-note pair and an upbeat eighth-note. This follows from the design decision that only rhythms in the templates are allowed by the algorithm.

4.2.5 Allowed rhythms

Some rhythms have been ruled out by their infrequent usage in typical rhythms. For instance, the three note rhythm 16th note, 8th note, 16th note is not grammatical in the system and thus will not be detected by system, but quantised to one of the existing templates. This improves performance on typical rhythms which will not be falsely detected as very rare rhythms, and exemplifies the preference of simple categories as pointed out in [29].

4.2.6 Future improvements

The algorithm is only meant to act as a proof-of-concept regarding context-dependent quantisation, and has still some deficiencies. Further improvements should include higher metrical spans and divisions, and a probabilistic measure for different templates could be added for making the decision when comparing different template errors Δe. With the probabilistic measure also unusual templates could be included in the system, while still retaining high detection rates with the usual templates: the usual templates would have higher likelihoods than the uncommon ones. The probabilities could be automatically counted from a music corpus using machine learning techniques.

4.2.7 Rhythm transcription

The final result of quantising the rhythms is in effect a transcription of the input rhythm. Music transcription is the process of analysing a music signal so as to write down the parameters of the sounds that occur in it, with traditional musical notation or any symbolic representation which gives sufficient information for performing the piece using the available musical instruments [48]. The rhythm transcription can be used to imitate and emulate ideas of the user and do easily further transformations on them. Without quantisation, in practise it would be only possible to do precise imitation with all micro-rhythmical discrepancies induced by the user and the latencies of the system.

4.3 Timing information (performance metrics)

After quantising, it is possible to calculate different micro-temporal deviations regarding the timing of the notes by comparing them to the unquantised input times collected into another 'coll' object in Max/MSP. In the system the timing information is calculated for every detected 'onbeat eighth-notes' template by Equation 12 (presented later in Chapter 5). With swung eighths, the quantisation functions only as a detection mechanism where an eighth-note was played.

Further development may include playing before or after the beat, which could be calculated with

$$r_i = q_i - u_i \tag{8}$$

where q_i is the temporal position of an input note and u_i the unquantised position of the same note, and r_i the resulting phase relation to the beat. The precision in which this calculation would be done (for every 4th, 8th or smaller metrical position) can be variable, but going to too small metrical division to count position of every individual note is psychoacoustically questionable: very fast passages are more likely detected only as number of notes per beat [3].

4.4 Statistical analysis of the input

Functionally the system performs the statistical analysis of the input rhythm in the same functional block and object written in Javascript as the previous tasks. The algorithms are presented here, but their intended musical use will not be discussed in detail until the next chapter.

4.4.1 Metrical accents

The statistical measure of metrical accents divides every 4/4 bar of input into eight beats: four onbeats and four offbeats. If after quantisation an onset is detected to start on these metrically important points, an array of metrical onsets is accumulated by one. This is calculated with

$$\alpha_n = \frac{\sum_1^w (\oslash I_n)}{wN} \quad | \quad (n = [0, 7]) \tag{9}$$

where α is the probability of a rhythm at a metrical point n (number of 8th note of each bar), w the window length used for statistical analysis in bars, I_n the input rhythm, and N the number of metrical points in the bar (here selected as 8 in a 4/4 bar). The $\oslash$ operator stands for checking whether there exists onsets on the metrical point n.

For example, if on the first beat of a measure a 2 note template number 4 is detected (two 16th notes on onbeat), the onbeat of the according metrical point is incremented by one. The accumulated array is divided by the window length (number of bars

analysed at current moment), which results in a likelihood percentage of a metric accent in the rhythmic input on a certain point of the bar. A visualisation of the algorithm is presented with certain input rhythm in the left part of Figure 17.

4.4.2 Used note values

During quantisation, the used note values are counted, accumulated and averaged in similar manner as was shown previously with metrical accents. If for example the 3 note template number 1 (three 8th note triplets) is found at a certain point, the 8th note triplet array will be incremented by three. This value is averaged by the number of beats in the window used for statistical analysis of the rhythm. In addition, to avoid bias of smaller note values the averaged values are multiplied by the length of the note values in relation to the one quarter note beat. This means multiplying 8th notes by 1/2, 8th note triplets by 1/3 and 16th notes by 1/4. With this correction, with window length of 4 bars, an input rhythm consisting solely of quarter notes will provide the same likelihood value as input rhythm of exclusive 16th notes.

The previous operation is formalised mathematically as

$$\kappa_n = \beta(\sum_{i=1}^{w}(\ominus I_i)) \tag{10}$$

Operator $\ominus$ stands for counting the number of certain note value n (for example quarter note) in input rhythm I bar i. β is the note value length weighting coefficient, and w is the window length in bars. A visualisation of the statistics provided by the algorithm with certain input rhythm is presented in the right part of Figure 17.

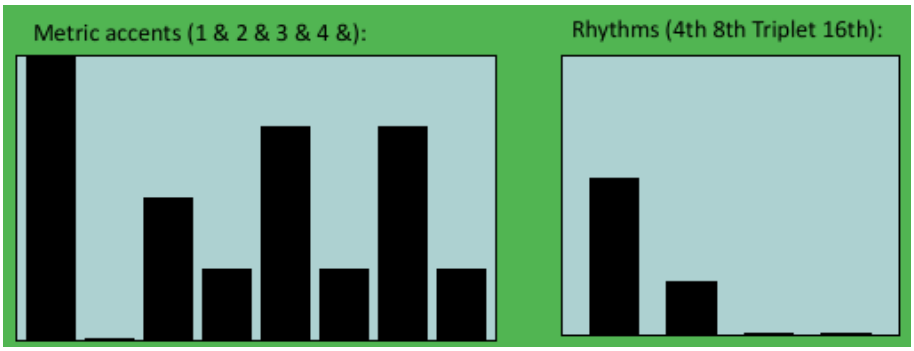


Figure 17: Statistical rhythm information visualised in Max/MSP. The left box presents metric accent probabilities for every 8th beat of a 4/4 bar, and the right presents weighted occurences of used note values.

4.5 Formulated meter and beat phase detection algorithm

An attempt to make a real-time algorithm capable of distinguishing between two types of time signatures – 4/4 and 3/4 – and their phases was made during the project. Unfortunately the signalling between different objects and reacting to these signals did not prove to be precise enough in the Max/MSP programming environment, and the meter detection algorithm and related interactive functionalities had to be left out from the final implementation. Nevertheless, the basic framework of this algorithm which uses musical top-down knowledge of typical beat patterns in meter and phase detection is presented for future use.

4.5.1 Beat phase detection in 4/4 meter

Figure 18 represents a situation where a 4/4 rhythm pattern, relative to the common drum pattern shown earlier in Fig. 7, is the rhythm the algorithm is looking for. The algorithm will check not only for the exact rhythm, but also its phase shifted versions, because the system clock (in this Max/MSP implementation represented by the 'transport' object) may not be precisely in synchrony with the user input: the user might want to start a rhythm in an arbitrary location and expect the system to respond to their rhythm. Using 8th note precision, there will be four permutations of the rhythm, since the 4/4 pattern repeats itself after the first two beats.

The phase-shifted versions of rhythm can be matched to the above templates by calculating the number of hits in the quantised rhythmic input of the user for each

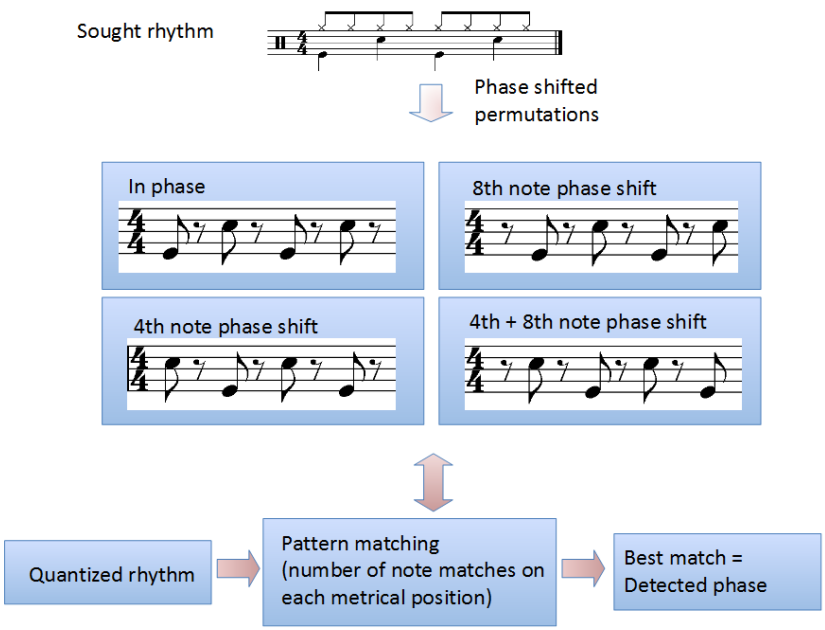


Figure 18: Example of the meter detection algorithm for a basic 4/4 rhythm.

template with

$$\rho_i = \sum_{n=1}^{N} (T_i[n] \otimes I[n]) \tag{11}$$

In the equation, T_i is the matched rhythm with phase-shift index i, I is the rhythm of the user, $\otimes$ a matching operator which returns 1 when T_i and I have the same detected gesture, n the time index (in the example rhythm, 8th beat index in the bar) and ρ_i the count of correct hits. The template T_i with the highest ρ_i will be output as the detected phase.

Phase information can be used for synchronising the system clock to user input with varying responsiveness/robustness, by skipping to the predicted next beat in the user input. Usable value has not been tested, but an immediate skip after one bar of phase-shifted user rhythm will probably lead to too frequent skips.

The matching operator $\otimes$ of Eq. 11 can be modified to return different values for matches on different indices to give more weight on certain hits, for instance in this rhythm the treble hits on the second and fourth of the bar. This will improve the algorithm performance in different variations of sought rhythm, which can be also detected in the same manner.

4.5.2 Meter shift from 4/4 to 3/4 meter

The same procedure for detecting the phase of a metrically correct input rhythm can be used for meter detecting also. Figure 19 presents the phase-shifted variants of a typical 3/4 rhythm in 4/4 meter. For a 3/4 rhythm there exists 6 unique permutations shown in the figure. The variants will be added to the phase-detection Equation 11 as new templates T_i. The one with the smallest ρ_i will give now also the phase for these 3/4 for rhythms. If the smallest one pertains to 3/4 meter, the system time signature is changed (natively possible in 'transport' of Max/MSP).

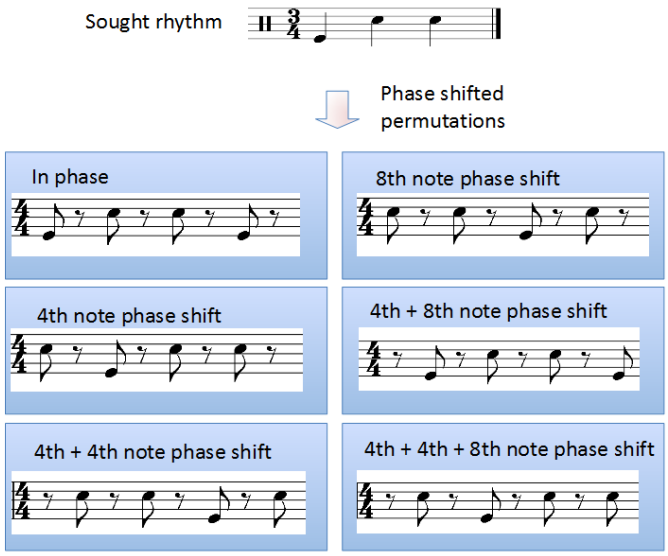


Figure 19: Example of the meter detection algorithm for a basic 3/4 rhythm.

4.5.3 The style-depence of musical top-down cues

A vital point is that the previous algorithm uses top-down knowledge cues extracted from western popular contemporary music, and can be predicted to function only on such rhythmical input, which function in the same musical framework. These kind of cues are pre-learnt from music by time and are not the same universally and independent of style. For a listener accustomed to western music, this is quickly noted by listening for example Senegalese Mbalax music, where the attempts to follow the beat using a set of musical cues learnt from western music do not apply, and easily result in a tangled perception of the meter. In the design of automatic meter

detection algorithms the consequence is that when musical top-down knowledge is used, the intended style should be taken into account also; even though some musical phenomena can be explained through universal psychoacoustical principles, the higher the lever of abstraction we move on in music, the less universal the different musical principles or "rules" are.

5 Interactive rhythmic output

This chapter describes the functions of the interactive rhythmic output model block of the system. Before this previously presented, comparable interactive musical systems are reviewed.

5.1 Previous interactive musical systems

In [13], Thom presented an interactive musical companion (BoB), capable of trading melodic 4 bar solo phrases with the user. Prior to using the system interactively in the call-and-response mode, the system is trained off-line with unsupervised machine learning techniques. The learning-phase enables the system to react to the melodic user input in a stylistically coherent way. In the thesis this training was performed off-line with jazz solo transcriptions of Charlie Parker and Stephane Grapelli, but also the possibility to do this online with a 10-minute training period with user input was proposed. The analysis included parsing the pitch content, interval magnitudes and melodic direction bar by bar. The text does not address rhythm quantisation or tempo-tracking of the input rhythms, but addresses the responsibility of synchronisation to the user. Nevertheless, the two previous problems are pointed out to be the most important ones to solve in order to create truly interactive, listening improvisational companions.

Franklin [49] has created similar interactive system, Computer Human Interacting Music Entity (CHIME), capable of trading 4 bars of jazz solo melodies with the user. The melodies are produced by an artificial neural network, which is initialised with supervised learning techniques to produce jazz melodies in two phases. The output will depend on the harmonic content of the background: different melodies will be created on different chords.

Weinberg et al. have created an interactive robotic percussionist, which is treated in detail in [10], [51] and [50]. The system works in six different interactional modes, which can be divided into call-and-response modes, accompaniment modes and collaborative improvisation modes. The robotic percussionist couples an analysis of rhythmic input picked up from the surface of a pow-wow drum with a microphone to a mechanic output with a mallet to the same drum. The analysis is realised in the Max/MSP real-time programming environment. The analysed rhythmic properties

include beat detection, rhythmic density, and rhythmic stability. The interaction is based in part imitating and stochastically transforming the user input, and in part in using existing rhythms represented with MIDI files.

Later on, Weinberg and his colleagues [12] have created a robotic marimba player which continues on expanding the musical interaction to melodic and harmonic domain, as opposed to the strictly rhythmic system treated above. The marimba player interacts in three modes: in the first it plays arpeggiated pre-set chord sequences which fit the melodic input of the user playing a MIDI keyboard, in the second it creates a beat-matched choreographic improvisation and in the third it provides pre-set stylistically coherent phrases based on the analysis of the user input.

5.2 Collective improvisation

The implemented system functions by analysing the rhythmical input of the user and providing a rhythmical output of its own, which is connected to the user input by statistical measures and differing amounts of imitation. As the rhythms can be freely invented – the only limitation being the inherent 4/4 metric structure – the interactive action between the system and the user is best described in musical terms as *collective improvisation*. Improvisation refers to the act of creating music while it is being performed [52], and catches the spontaneous nature of the use-scenario where not only the system reacts to the user input, but presumably the user will also start exploring the musical ideas provided by the system.

Scientific research and various explanations of the essence of improvisation concentrate in western literature mostly on different melodic and harmonic tools (see for example [52], [7], [53], [54] and [55]). This is likely due to the fact that most studies are in close connection to jazz music, where improvisation is above all the requirement of soloists playing melodic/harmonic instruments such as trumpet, piano or saxophone. Current improvisational computational systems do seem to also concentrate on melodic/harmonic improvisation of jazz as shown above: [13], [12], [49] all function primarily in the framework of jazz. Rhythmic tools for improvisation and solely rhythmic improvisation is a rather untouched subject even in music-pedagogical literature.

However, the typical behaviour of collective improvisation in for instance a jazz-band setting can be extended to solely rhythmic improvisation also. The most common

situation is that one individual in the group, the *soloist*, is providing the majority of improvised material whereas the others provide a predetermined *accompaniment* [52]. The predetermined accompaniment may be a set of chord changes (harmony) with larger metrical structure of the song, measure-level meter and basic rhythms used, the micro-rhythmical feel and so on, in differing amounts of variation and freedom of choice by accompanying players. In the case of drumming, the accompanist typically plays repeating, known time-keeping patterns, which fortify the feeling of meter and metric accents [56]. The role between accompanist and soloist may well change during a piece [52].

A special case of collaborative improvisation in jazz is so called *call and response* technique, where the responsibility of soloing is moved back and forth between two players in short passages ([56], [13]). During this *trading* of typically 2, 4 or 8 bar pieces, the non-improvising player listens closely to the improvising player without playing anything themself. The following part, response to the call is normally in close relation to the ideas presented before, building on and varying them. The improvisational musical companion (BoB) by Thom [13] and Computer Human Interacting Music Entity (CHIME) by Franklin [49] are both built to function in this manner exclusively.

5.3 Simulated interaction modes

Based on the previous points, the collective improvisation of the system is produced by simulating the following musical interaction modes:

1. **Accompaniment:** the system provides a clear time-keeping pattern.
2. **Solo:** the system takes a more improvisational role with less importance on time-keeping.
3. **Trading bars (call-and-response):** the system is reacting to previous bars of the rhythmic user input.

In the accompaniment mode the system will provide a clear time-keeping pattern based on a database of different metrically coherent beats. In addition, larger metrical parts like every 4th or 8th measure can be emphasised with a marking pattern, a "fill". In the solo mode the system will provide a more improvisational rhythm,

based on fill patterns and statistical variation of these. The call-and-response mode is a form of collaborative soloing, where the system will create rhythms based on previous user input. The connection is imitational and statistical. Computational details of the modes are given in following sections.

5.4 'Accompaniment' mode

In the accompaniment mode, the role of the system is to keep time, while the user is soloing, in other words creating more improvisational rhythms with less connection to the meter and more surprising rhythms. The accompanying patterns are picked from a database of different bar-length beats. Also, to mark metrical turning points, a fill is created from another collection of bar length rhythms. The fills are typically more dense rhythmically than the accompanying patterns. Technically the database of rhythms is a text file containing bar-length rhythms, which is loaded into a 'text' object in the main Max/MSP and accessed line-by-line when needed.

The relation between time keeping patterns and fills can be adjusted, which simulates a personal style of playing with either more emphasis on using technical fills more frequently, or sticking to a steady beat for longer periods of time. The same adjustment can be done on how many times the same time-keeping pattern is repeated after the marking fills – similarly, a human drummer might have preference on changing the pattern to a new one earlier or later on.

5.5 'Solo' mode

Contrary to the accompaniment mode, the solo mode will take the responsibility of creating the improvisational rhythms when the user is moving to more accompanying role. The solo patterns are created either from picking up a fill-pattern from the database of previously entered beats, or making up a new one based on statistical information of the user input. When the solos are based on statistical information, the metrical countering points of the user rhythm are more probably picked out, which results in a more intertwined rhythmical output as a whole. As the user is probably sticking to certain metrical points when accompanying, imitating the possibly repetitive user rhythms are avoided.

5.6 'Trading bars' mode

In the trading bars mode, the responsibility of soloing is moved back and forth between the user and the system. In every other bar, the user is soloing, and in every other the system outputs a rhythm, responding to the rhythmical call of the user. The rhythmical character of the system can be modified between a completely imitational or completely statistical output with a user-interface slider.

5.6.1 Imitational solo output

As described in Chapter 4, the system is collecting and quantising the input rhythm of the user. This can be used as a basis for providing improvised solos in the system as such by adjusting the time for which the rhythmic input is collected and then played back. Using the imitation as a basis is especially useful in the call-and-response mode, because the user is providing an interesting rhythm all the time, whereas in the accompaniment mode the rhythm may be repeating itself for long periods of time. On the other hand, in call-and-response mode the user is expecting for the system to respond in a way connected to the user input: while testing the first versions of the mode, it became clear that if the connection is too abstract, the sense of interacting with the system disappears.

5.6.2 Statistical output

The statistical output mode uses the statistical metrics collected during the rhythm quantisation as described in Chapter 4. The metrics describe the bar-level metric accentuation patterns present in the user input and the distribution of used note values. With this information, it is possible to create generative rhythms, which will not be actually imitated from the user rhythm, but still have more connection to it than an absolutely random rhythm.

In purely statistical mode, the music output algorithm creates rhythms which have the same kind of statistical distribution regarding previous metrics. If the user repeats metrically similar patterns, the output will correspondingly sound metrically similar. The idea of collecting these metrics is inspired by the findings in [21], where three solos of different music styles were analysed from recordings. The results showed that in different styles different metrical positions were seen important to

accentuate or avoid; this creates a recognisable feeling for the solo. In this light, using these statistical measures as a basis, a part of the rhythmic feeling of the user input should be noted also in the system output.

5.7 Detecting mode changes

The shift between previously described modes is performed based purely on the rhythmical input given to the system. There is no need to modify any values with the graphical user-interface once the system has been started, enabling continuous rhythmic interaction between the user and the system. Only the input rhythm controls the system, alongside with some randomness.

Figure 20 presents the possible shifts between different modes. When the system is started, and no rhythmic input is available, trading bars mode is chosen. It will be also entered whenever silence occurs for at least a whole bar in the input. If however the user keeps playing for four consecutive bars, trading bars mode is exited and based on the statistical analysis of the input rhythm, either solo or accompaniment mode is entered. The analysis concentrates on the rhythmic density of the input: if the user is playing a dense rhythm, it is interpreted as a solo part and accompaniment mode is chosen, conversely if the rhythm is sparse, it is interpreted as an accompanying part and solo mode is entered. Equal type of two-mode switching was also presented in [50], where softness/sparseness of the input triggered a so called leader mode and loudness/denseness a follower mode. As noted, a bar of silence exits the modes and trading bars mode is entered.

5.8 Timing information

Using the information provided with the rhythm quantisation algorithm presented in Chapter 4, and the rhythmically unquantised input, it is possible to extract phrasing-related timing information from the user input. For example, the user could be intentionally playing certain notes ahead of the beat or after the beat, or in different rhythmic timing schemes. However, currently only the timing information regarding the eighth notes are analysed and used in the rhythmic output of the system as a proof-of-concept – more wide-ranging effects are left to be implemented in the future.

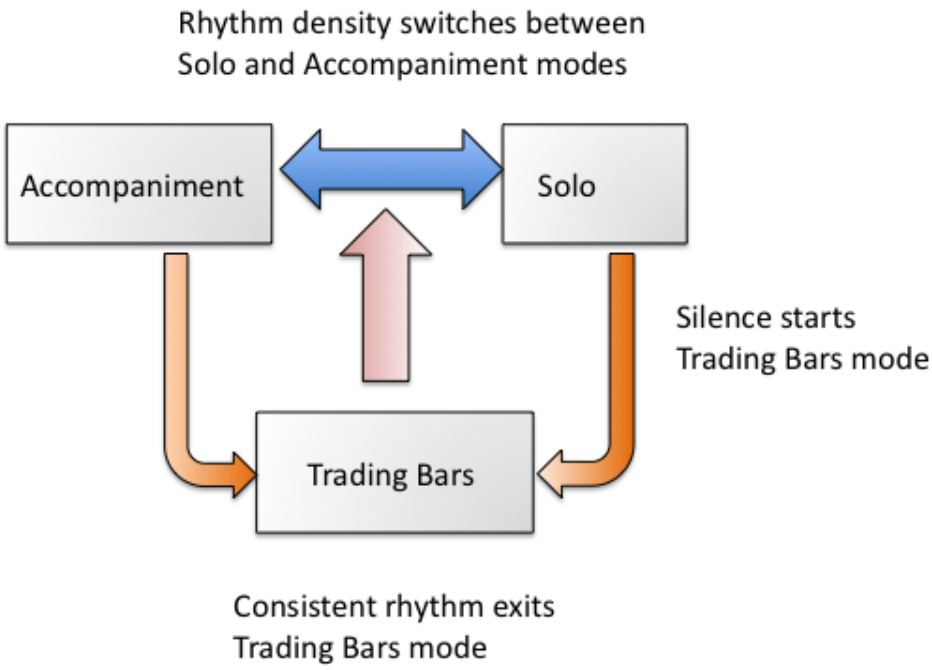


Figure 20: Switching between improvisational modes.

The most common beat interval in western music is probably the quarter note, and the most used rhythm in solos one level below this, the eighth note. In much of jazz and blues music, consecutive eighth notes are played with a "swing" feel, in other words the eighth notes falling on the beat are elongated and the following off-beat eighth notes shortened accordingly which results in a more triplet-division kind of rhythmic feel. The relation between the eighth notes can be described with a single number, which gives the length of the notes falling on the beat compared to the length of off-beat eighth notes:

$$s = \frac{|\alpha|}{|\beta|} \tag{12}$$

where $|\alpha|$ is the length of the onbeat note, $|\beta|$ is the length of the offbeat note and s the resulting swing ratio.

Typically the swing-feel of creating long-short patterns of eighth notes is described simply by creating a triplet-feel. However, Friberg and Sundström provided contradictory evidence by measuring the swing-ratios empirically from spectrograms of classical jazz performances in [22]. The results showed that at slow tempi, the swing ratio was as high as 3.5:1, whereas at fast tempi it reached equal length 1:1.

Interesting result was that different players favour similar swing ratios independent of song, and position their notes very consistently on the same phase of the beat.

It should be noted that the swing affects only the one level below the beat level, tactus [55]. For instance, in a 4/4 song with a quarter note tactus sixteenth note or eighth-note triplet passages are played in a relatively precise, mechanical feel, where the notes fall close to their "theoretical" starting points.

In the implemented system, the swing ratio of eighth-notes in user input is calculated every time the rhythm quantisation block detects one. The value is averaged over 5 pairs to avoid too rapid changes, and passed once per bar to the rhytmic output module, which will create an identical swing ratio for output eighth notes, imitating the feel of the user. The findings of [22] report similar behaviour, where the soloist and accompanying player were found out to be playing with congruent, if not identical rhythmic feel.

6 Conclusion and future work

This section gives an overall summary of the work and discusses the potential for future research. The evaluation of the system is based on subjective assesment of the author only, and more thorough user tests are to be performed later.

6.1 Results

In the thesis, a sonically augmented table with an interactive capability to produce collaborative rhythmic improvisation with the user was presented. It was shown that it is challenging to computationally solve similar problems that the human auditory system solves with ease. A review of previous research and existing algorithms on the subject was presented, and some new algoritms were also introduced. The algorithmic side tackled especially the problem of rhythm quantisation, for which a novel, automatically scalable algorithm was presented.

Improved rhythm quantisation algorithms open up possibilities in extracting timing information from musical audio: in the future, a style-dependent timing of notes, in other words temporal phrasing can be included in music generating algorithms. For instance, a sequencer can be set to phrase according to the style of certain song (rock, jazz, etc.), or even in the styles of individual musicians. The problem of traditional sequencers with temporal precision of microseconds has been that they "lack feel", or "sound like a machine". Adding human-like performance-related temporal, dynamic and timbral variation in their output, this can be avoided (see [45]).

The system is capable of producing moderately interesting rhythmic output and create an illusion of interacting with another musician. The interaction is clear especially when the system is producing imitative output. In initial tests with output connected to the input purely statistically, the sense of musical interaction was not achieved: it felt almost as if the output had been totally random.

Many of the included algorithms could be improved later on. For example, with free, improvisational input the tempo tracking algorithm starts making a great number of errors, and the user is urged to turn the tracking completely off. On the other hand, even though the gesture classification is not fool-proof, it seems to be working with very few missed detections with a little practice. An especially nice detail

is the systems capability of following the rhythmic feel of the user input, when swung eighth notes are played. This supports also directing future research into the direction of creating capabilities of performing temporal analysis of rhythmic input in even more detail.

6.2 Future improvements and research directions

Practically all of the algorithms which were included in the system for the execution of some small subtask of rhythmic perception, seem to be still making significantly more errors than an assumed naive human subject executing the same task. This shows that there still is room for improvement, especially on methods which are functioning causally and are computationally efficient enough to be included in a real-time system.

The effect of improving these algorithms would not only improve musical interaction systems such as the one introduced in the thesis, but could also be used in other domains. For example, in music sequencer tools – which in practice at the moment require synchronising recorded audio input with a click track (especially if different MIDI tools are used) – a working meter detection algorithm could be used to turn the situation around so that the sequencer would automatically synchronise itself with the user input. This would also facilitate audio editing considerably. The main point in improving these kind of algorithms seems to be including task-specific top-down information: in the musical domain, incorporating style-dependent musical cues seems a fruitful future direction. This also shows that the higher the level of abstraction we are on, the more complex, multidisciplinary and non-generalisable the problems become.

On a higher level, combining these algorithms into functional wholes (in a similar way that was done in the current implementation in only one possible way), new interactive applications could be created. For instance, the play-along CDs used in jazz improvisation pedagogy could be replaced with systems which would adapt themselves to the user regarding tempo, phrasing and harmony. Already with the current implementation, many alternative use-scenarios could be come up with minimal modifications: for example, the system could be used in teaching different rhythms by including an algorithm doing comparison between the quantised user input and the taught rhythm. The usefulness of a sonically augmented table is de-

batable, but once again, the emphasis in the thesis has mostly been on pointing out what kind of computational algorithms a system of this complexity needs, and what are the most crucial future directions to improve upon.

Even though the application at hand works in the musical domain, it is arguable that the findings are generalisable to a certain extent for non-musical rhythmic interaction tasks also. The human perception does not make a clear distinction between musical and non-musical sounds, as shown for example in [3] – on the contrary the musical processing seems to build upon very basic psychoacoustical properties. It can be subjectively noted how the sounds of repetitive motions easily elicit a perception of beat, tempo and sometimes even meter. Thus, it is no surprise that the researchers of sonic interaction design are currently interested in augmenting these kind of tasks and using computational algorithms which simulate this side of human rhythmic perception (see [2]). However, the discussion of this higher-level connection in-depth is outside the scope of the thesis and left for future takes on these subjects.

References

[1] "Sonic Interaction Design," 3.8.2010. [Online] Available: `http://www.cost-sid.org/`. [Accessed 3.8.2010].

[2] D. Rocchesso, P. Polotti and S. Delle Monache, "Designing Continuous Sonic Interaction," *International Journal of Design*, vol. 3, no. 3, pp. 13–25, 2009. Available: `http://www.ijdesign.org/ojs/index.php/IJDesign/article/view/620/271`.

[3] A. Bregman, *Auditory Scene Analysis.* Cambridge, Massachusetts: The MIT Press, 1994 [1990].

[4] T. Blaine and S. Fels, "Contexts of Collaborative Musical Experience," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, Montreal, Canada, 2003, pp. 129–134.

[5] E. Charry, "Jembe," *Oxford Music Online,* 4.8.2010. [Online] Available: `www.oxfordmusiconline.com`. [Accessed 4.8.2010].

[6] J. B. Robinson, "Drum kit," *Oxford Music Online,* 4.8.2010. [Online] Available: `www.oxfordmusiconline.com`. [Accessed 4.8.2010].

[7] P. Berliner. *Thinking in Jazz: The Infinite Art of Improvisation.* Chicago: The University of Chicago Press, 1994.

[8] R. Murray Schafer, *New Soundscape.* BMI Canada, 1969.

[9] A. Hazan, "Performing Expressive Rhythms with Billaboop Voice-Driven Drum Generator," in *Proceedings of the International Conference on Digital Audio Effects (DaFX â05)*, Madrid, Spain, 2005, pp. 1–4.

[10] G. Weinberg and S. Driscoll, "Robot-Human Interaction with an Antropomorphic Percussionist," in *Proceedings of the Conference on Human Factors in Computing Systems (CHI 2006)*, Montreal, Canada, 2006, pp. 1229–1232.

[11] G. Weinberg, A. Raman and T. Mallikarjuna, "Interactive Jamming with Shimon: A Social Robotic Musician," in *Proceedings of the 4th ACM/IEEE International Conference on Human Robot Interaction,* La Jolla, USA, 2009, pp. 233–234.

[12] G. Hoffman and G. Weinberg, "Gesture-Based Human-Robot Jazz Improvisation," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2010)*, Anchorage, USA, 2010, pp. 582–587.

[13] B. Thom, "BoB: An Improvisational Music Companion," PhD thesis, Carnegie Mellon University, Pittsburgh, USA, 2001. [Online] Available: `http://www.cs.cmu.edu/~bthom/PAPERS/thesis.pdf.zip`. [Accessed 8.9.2010].

[14] J. London, "Pulse," *Oxford Music Online,* 24.8.2010. [Online] Available: `www.oxfordmusiconline.com`. [Accessed 24.8.2010].

[15] J. London, "Rhythm, I: Fundamental Concepts & Terminology," *Oxford Music Online,* 5.8.2010. [Online] Available: `www.oxfordmusiconline.com`. [Accessed 5.8.2010].

[16] B. Kernfeld, "Metre," *Oxford Music Online,* 5.8.2010. [Online] Available: `www.oxfordmusiconline.com`. [Accessed 5.8.2010].

[17] P. Desain and H. Honing, "The Formation of Rhythmic Categories and Priming," *Perception*, vol. 32, 2003, pp. 341–365.

[18] F. Lerdahl, and R. Jackendoff, *A Generative Theory of Tonal Music.* Cambridge, Massachusetts: MIT Press, 1996 [1983].

[19] S. Hainsworth, "Beat Tracking and Musical Metre Analysis," pp. 101–130 in *Signal Processing Methods for Music Transcription*, A. Klapuri and M. Davy, Eds. New York, Springer, 2006.

[20] M. Goto, "Music Scene Description," pp. 327–360 in *Signal Processing Methods for Music Transcription*, A. Klapuri and M. Davy, Eds. New York, Springer, 2006.

[21] M. Pesonen, "Svengi ja groove rytmimusiikissa: rytmiikkaan ja hienorytmiikkaan keskittyvä vertaileva musiikkianalyysi," MA thesis, University of Helsinki, Department of Art Studies, Musicology, 2009. [Online] Available: `https://oa.doria.fi/handle/10024/63580`. [Accessed 7.9.2010].

[22] A. Friberg and A. Sundström, "Swing Ratios and Ensemble Timing in Jazz Performance: Evidence for a Common Rhythmic Pattern," *Music Perception*, vol. 19, No. 3, Spring 2002, pp. 333–349.

[23] J. A. Prögler, "Searching for Swing: Participatory Discrepancies in the Jazz Rhythm Section," *Ethnomusicology*, vol. 39, no. 1, 1995, pp. 21–54.

[24] C. Drake and M. Botte, "Tempo Sensitivity in Auditory Sequences: Evidence for Multiple-look Model," *Perception & Psychophysics*, vol. 54, no. 3, 1993, pp. 277–286.

[25] B. Wright and J. Mossbridge., "Specificity of Learning in an Auditory Temporal Order Task," *Journal of Acoustical Society of America*, vol. 107, 2000, pp. 2882 (A).

[26] B. Wright, M. Fitzgerald and J. Mossbridge, "Specificity of Learning in an Auditory Asynchrony-Detection Task," *Journal of Acoustical Society of America*, vol. 109, 2001, pp. 2289 (A).

[27] T. Rammsayer and E. Altenmüller, "Temporal Information Processing in Musicians and Nonmusicians," *Music Perception*, vol. 24, no. 1, September 2006, pp. 37–48.

[28] D. Temperley, *The Cognition of Basic Musical Structures.* Cambridge, Massachusetts: MIT Press, 2001.

[29] H. Honing, "Structure and Interpretation of Rhythm and Timing," *Tijdschrift voor Musiktheorie*, year 7, no. 3, 2002, pp. 227–232.

[30] J. Bello, L. Daudet, S. Abdallah, C. Duxbury, M. Davies and M. Sandler, "A Tutorial on Onset Detection in Music Signals," *IEEE Transactions on Speech and Audio Processing*, vol. 13, no. 5, 2005, pp. 1035–1047.

[31] S. Dixon, "Onset Detection Revisited," in *Proceedings of the 9th International Conference on Digital Audio Effects (DAFx-06)*, Montreal, Canada, 2006, pp. 1–6.

[32] M. Puckette, T. Apel and D. Zicarelli, "Real-time Audio Analysis Tools for Pd and MSP," in *Proceedings of the International Computer Music Conference.* San Francisco: International Computer Music Association, 1998, pp. 109-112.

[33] "Max," 3.8.2010. [Online] Available: `http://cycling74.com/products/maxmspjitter/`. [Accessed 3.8.2010].

[34] M. Puckette, "Pure Data: Another Integrated Computer Music Environment," in *Proceedings of the International Computer Music Conference (ICMC '96)*, Hong Kong, 1996, pp. 37–41.

[35] A. Robertson and M. Plumbley, "B-Keeper: A Beat-Tracker for Live Performance," in *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME07)*, New York, USA, 2007, pp. 234–237.

[36] M. Davies, P. Brossier and M. Plumbley, "Beat Tracking Towards Automatic Musical Accompaniment," in *Proceedings of the 118th AES Convention,* Barcelona, Spain, May 2005, paper number 6408.

[37] A. Klapuri, A. Eronen and J. Astola, "Analysis of the Meter of Acoustic Musical Signals," *IEEE Transactions of Audio, Speech, and Language Processing,* vol. 14, no. 1, 2006, pp. 1–14.

[38] M. Goto and Y. Muraoka, "A Beat Tracking System for Acoustic Signals of Music," in *Proceedings of the ACM International Conference on Multimedia*, San Francisco, USA, 1994, pp. 365–372.

[39] P. Desain, R. Aarts, T. Cemgil, B. Kappen, H. van Thienen and P. Trilsbeek, "Robust Time-Quantization for Music, from Performance to Score," in *Proceedings of the 106th AES convention*, Munich, Germany, May 1999, paper number 4905.

[40] A. T. Cemgil and H. J. Kappen, "Monte Carlo Methods for Tempo Tracking and Rhythm Quantization," *Journal of Artificial Intelligence Research*, vol.18, 2003, pp. 45–81.

[41] S. K. Mitra, *Digital Signal Processing: A Computer-Based Approach.* 3rd Edition. New York: McGraw Hill, 2006

[42] C. Harrison and S. Hudson, "Scratch Input: Creating Large, Inexpensive, Unpowered and Mobile Finger Input Surfaces," in *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST '08)*, 2008, pp. 205–208.

[43] "Drum Samples," 27.2.2011. [Online] Available: `http://www.drumsamples.org/`. [Accessed 27.2.2011].

[44] T. D. Rossing, F. R. Moore and P. A. Wheeler, *The Science of Sound.* Reading, Massachusetts: Addison-Wesley, 2001.

[45] "EZDrummer Software," 27.8.2010. [Online] Available: `http://www.toontrack.com/products.asp?item=7`. [Accessed 27.8.2010].

[46] S. Bilbao, "Percussion Synthesis Based on Models of Nonlinear Shell Vibration," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 4, May 2010, pp. 872–880 .

[47] J. Seppänen, "Computational Models of Musical Meter Recognition," MSc. thesis, Tampere University of Technology, 2001.

[48] A. Klapuri, "Signal Processing Methods for the Automatic Transcription of Music," PhD thesis, Tampere University of Technology, Finland, 2004.

[49] J. Franklin, "Multi-Phase Learning for Jazz Improvisation and Interaction," in *Proceedings of the Eighth Biennial Symposium on Arts and Technology,* New London, USA, 2001, pp. 1–10.

[50] G. Weinberg and B. Blosser, "A Leader-Follower Turn-Taking Model Incorporating Beat Detection in Musical Human-Robot Interaction," in *Proceedings of the 4th ACM/IEEE International Conference on Human Robot Interaction,* La Jolla, USA, 2009, pp. 227–228.

[51] G. Weinberg and S. Driscoll, "Toward Robotic Musicianship," *Computer Music Journal*, vol. 30, Issue 40, Winter 2006, pp.28–45.

[52] B. Nettl, "Improvisation," *Oxford Music Online,* 26.8.2010. [Online] Available: `www.oxfordmusiconline.com`. [Accessed 26.8.2010].

[53] M. Levine, *The Jazz Theory Book.* Petaluma, CA: Sher Music Co, 1995.

[54] R. Rawlins and N. Bahha, *Jazzology: The Encyclopedia of Jazz Theory for All Musicians.* Cheltenham, Victoria: Hal Leonard Australia, 2005.

[55] K. Backlund, *Improvisointi pop/jazzmusiikissa.* Helsinki: Musiikki-Fazer, 1983.

[56] M. Säily, " "Philly-Joe" Jonesin jazz-rumpukomppaus: improvisoitu säestys ja vuorovaikutus kappaleessa 'Blues for Philly Joe'," MA thesis, University of Helsinki, Department of Art Studies, Musicology, 2007. [Online] Available: `https://oa.doria.fi/handle/10024/5816`. [Accessed 8.9.2010].