Automatic Phrase Continuation from Guitar and Bass-guitar Melodies

Srikanth Cherla

Master's Thesis MTG - UPF / 2011
Master in Sound and Music Computing

Master's Thesis Supervisor:
Dr. Hendrik Purwins
Dept. of Information and
Communication Technologies
Universitat Pompeu Fabra

UNIVERSITAT POMPEU FABRA

Automatic Phrase Continuation from Guitar and Bass-Guitar Melodies

Srikanth Cherla

Music Technology Group
Universitat Pompeu Fabra
Tanger, 122-140, 3rd Floor
08018 Barcelona, SPAIN.

Master's thesis

Abstract

A framework is proposed for generating musically similar and interesting variations of a given monophonic melody. In this work, the focus is on rock/pop guitar and bass-guitar melodies with the aim of extensions to other instruments and musical styles. The original audio melody in audio format is first segmented into its component notes using onset detection and pitch estimation. Clustering is performed on the pitches to obtain a symbolic representation of pitch sequences in it. The note onsets are aligned with the estimated meter of the melody for a time-homogeneous symbolization of the rhythm in terms of onsets/rests and the metrical locations of their occurrence. A joint representation based on the cross-product of these two individual representations is used to train the prediction framework - the variable-length Markov chain. It is hypothesized that such a model will rearrange, while maintaining certain coherence, the segments of the original melody. The musical quality of the generated melodies was evaluated through a questionnaire by a group of experts and received an overall positive response.

Computing Reviews (1998) Categories and Subject Descriptors:

H Information Systems
H.5 Information Interfaces and Presentation
H.5.5 Sound and Music Computing

Key words:

Melody prediction, variable-length Markov Chains, stochastic music generation.

Contents

Chapter 1

Introduction

"Models of process are natural to musical thinking. As we listen, part of us drinks in the sensual experience of sound, while another part is constantly setting up expectations, and in so doing, constructing hypotheses of musical process."

-Curtis Roads

The idea of making music using stochastic and algorithmic methods is not something unheard of. Although it is only in the more recent times that labels like *algorithmic composition* or *stochastic music* have come to be associated with certain kinds music, there have always been *algorithmic* and *stochastic* elements in music composition and improvisation. Mathematics has been an intrinsic part of music right from its creation, to its expression and perception. Musical output has always involved the use of formulas and often unwritten, yet perceivable processes. The concept has always existed but its expression and interpretation have evolved over centuries.

The current thesis focuses on generating stylistically similar variations of basslines and guitar melodies based on the statistical analysis of patterns occurring in the original melody. While the scope of the work is limited to specific instruments and styles of music, it is proposed that the described method may also be extended to a broader context. This chapter provides the necessary background and historical references to put things into context for the reader before explaining the technical details later on.

1.1 Background

Prior to understanding machine-generated music, one must familiarize oneself with the idea of the musical process which is the cornerstone of all approaches that rely on machines to imitate human-like music. This section first introduces the reader to the general idea of a musical process and gradually narrows down the discussion to the Markov chain - a fairly popular representation the musical process, which is also employed here.

1.1.1 Music as a Process

The view of music as a process, as noted by Roads [Roa96] has been the key idea around which much of work related to *algorithmic/stochastic music* has revolved. A process, in a very general sense, can be defined as a series of actions, changes, or functions bringing about a result. Much like this general definition, a musical process also involves certain types of changes, actions, or functions that bring about a musical result - the composition. Anything that can be regarded as a process is governed by an underlying mechanism. The presence of such a mechanism in music can be seen in the difference between how it is perceived by the experienced and the inexperienced ear. The former would be more aware of the underlying mechanism of the musical process than the latter and would also be able to describe it in a simplified manner, grouping similar aspects, repetitions, highlighting variations and changes, etc. We often refer to aspects of the mechanism of a musical process at various levels when we speak in terms of *verse*, *chorus*, *key*, *rhythm*, *intervals*, etc. when referring to a musical piece.

While the awareness of the musical process may not be of much consequence to the casual listener, or someone who is only interested in experiencing the aesthetic effects of music, it is of great benefit to the musicologist, the musician or the composer. In order to create music, one must rely on a suitable representation for describing and expressing it. One of the earliest formalizations in Western music dates back to around 1026 when Guido d'Arezzo first introduced the *ut-re-mi* (now known as *do-re-mi*) system of associating a syllable with a musical note. He is also credited for having formulated the modern staff notation used in Western music. Any such notation or representation that one devises for a style of music, is simply a set of tools for the symbolic representation of the musical process.

1.1.2 Deterministic & Stochastic Models of Process

The interpretation of music as a process implies the existence of an underlying mechanism that is responsible for making the music sound the way it does. A composition, under this view, is essentially a complex set of rearrangements and variations of musical elements (interval, pitch, texture, timbre, etc.) based on some principles that determine its nature. One could as well say that two pieces that sound alike are due to the occurrence of similar musical patterns and variations in them. Creating systematically varying patterns out of musical elements involves a decision (or an event that may be interpreted as a decision) at each stage of the composition to create a change or an evolution of a certain nature. One may picture the evolution of a musical piece as a sequence of transitions between a set of musical *states* that are governed by decisions according to a pre-determined mechanism.

A set of instructions are required to realize the process of making these decisions, which is natural to humans, on a machine such as a computer. Several methods like finite-state automata, Markov chains and probabilistic grammars, have been applied over the years that attempt to simulate this decision process. These can broadly be classified into two approaches - deterministic and stochastic. The basic difference between models of these two types is that the former does not rely on randomness and the latter does. Processes are deterministic if there is a unique consequent to every state, or stochastic if there is a probability distribution of possible consequents.

In interest of relevance to the topic of the thesis, further discussions on models for music generation are restricted to the stochastic type. The reader is referred to [Roa96] for a review on both deterministic and stochastic models.

1.1.3 Stochasticity in Music

While, historically, attempts to introduce stochasticity in music have been many, this section covers some of the significant ones that would help establish some understanding of the idea. For a comprehensive review of approaches, the reader is referred to [Roa96], [Ame87] and [Hil70].

Aleatoric musical instruments existed in ancient Asia in the form of wind chimes. A wind chime consists of a system of similar resonating objects (such as metal tubes, sea shells, etc.) suspended around a *clapper* which strikes these objects to produce sounds when wind blows through the system. These were also used by many composers such as Olivier Messaien, Giles Swayne and David Sitek in recent times. In wind chimes, music is intended to be produced by natural movements or changes (wind blowing, motion of the support) and can be regarded as an early form of stochastic music.

Stochasticity in musical composition has existed for many centuries prior to the invention of computers. A popular game in post-medieval Western Europe that could be considered as an example of stochastic composition of the time, was the system known as *Musikalisches Würfelspiel*, meaning “Musical Dice Game”. This game essentially involved putting together pre-written measures of music together to create a Minuet. The element of chance was introduced into the Minuet composition through the use of dice. Each number possible on the dice was associated with a set of measures, one of which was picked up when that number turned up on the dice. Joseph Haydn’s *Philharmonic Joke* was composed in this manner. Mozart was also well-known for his Musikalisches Würfelspiel.

There have also been different mechanical musical instruments that were made for stochastic music composition [B7̈8]. One such instrument constructed by Dietrich Winkel in 1821 was the *Componium*. It was essentially a mechanical organ which consisted of two barrels that took turns to perform measures of randomly chosen music alternately. The Componium produced variations on themes that were programmed into it.

Iannis Xenakis, who is widely accepted as one of the most influential post-world war avant-garde composers, has composed several pieces using mathematics and stochastic processes. He authored the *Stochastic Music Program (SMP)*, which was based on formulas originally used to describe the behaviour of particles in gases [Xen01]. The composer specifies global attributes such as average duration of sections, timbre classes, density distributions of timbre classes, etc. to the program and executes it to generate a stochastic musical piece. This program helped create many of Xenakis’ works, including the well known *Eonta*, for piano and brass.

1.1.4 Markov Chains for Musical Composition

The Markov chain has been one of the most popular choices for algorithmic composers in the past. It is essentially a probabilistic framework that models the behaviour of sequences of events. The likelihood of a future event is determined by the nature of one or more events occurring in the immediate past. Such a framework models the evolution of a process much in the same way as musical composition where musical events of the immediate past play an important role in determining what comes next. And thus, they have been adopted in various computer-based compositional approaches.

The first ever application of Markov chains, by their creator Andrei Andreevich Markov, was to distil tendencies of spelling from the first $20,000$ characters in the text of Aleksandr Pushkin's *Eugene Onegin.* His intent was to model the alteration of vowels and consonants in Russian literary works. So, right from their inception, Markov chains have mainly been used for determining likelihoods of sequences in data which are based on a process, but whose nature cannot be precisely determined. A more detailed discussion of Markov chains, and their application in the current work is presented in later chapters. The reader is also referred to [GS97] for an introduction to Markov chains and to [Ame87] for a survey of their applications to musical composition prior to the 90s.

One of the first pieces of music ever to be written with the aid of a computer was the *Illiac Suite for String Quartet* which was the result of a series of four experiments (each one being a movement in the composition) by Hiller and Isaacson with the *Illiac I* digital computer [HI59]. These were pioneering experiments to see how successfully a computer would be able to compose a musical piece. The fourth movement of the Illiac Suite was composed using Markov chains.

Another notable application of Markov chains in music can be seen in Iannis Xenakis's electroacoustic composition *Analogique A + B.* This was based on his proposal to compose on the basis of "screens", a representation of regions of a *musical space* for each "slice of time". And the progression from one screen to the next is governed by Markov chains, thus incorporating a sort of "memory" into the temporal organization of the music [Har02]. This approach differs from that of Hiller et al. in the sense that it incorporates many simultaneous Markov chains.

The order of a Markov chain determines how much of the most recent past is considered when computing the likelihood of a certain outcome in the next time instant. While the increase in order of the Markov chain facilitates dealing with data more comprehensively, it requires a large amount of data to train and also becomes very rigid in its predictions. On the other hand, a first order Markov chain is often susceptible to produce seemingly random predictions. As a result, many of the recent approaches have adopted the variable-length Markov Chains (VLMCs), which provide a trade-off between the two extremes, for representing musical information [Mar10], [CW95], [Pac03]. At the same time, they can be represented with a parsimonious, efficient tree structure. A detailed introduction to VLMCs can be found in [BW99].

1.2 Motivation

Recent approaches of algorithmic/stochastic music have aimed at creating meaningful and interesting music using mathematical and statistical models with the aid of computers to realize these models. These approaches essentially play with the idea of programming computers with rules that humans generally follow while composing or improvising music to have them do the same, and sometimes maybe even with the hope of taking music creation to a new level. Computers are thus trained on rules of musical style and made to compose and sometimes even perform. This type of music gained prominence in the 1950s with composers Lejaren Hiller, Iannis Xenakis, Gottfried Michael Konig and others. One simple, yet very interesting example is the use of John Conway's "Life" algorithm [Gar70] to compose music. One branch of this very broad area of algorithmic composition focuses on style-oriented music generation. The goal is to devise algorithms or train models that can make computers produce music of a defined style.

Music generation from statistical models can be viewed as a series of sampling operations on a distribution [Con03]. So the distribution essentially models features of the music that is generated from it. The same distribution, when viewed as a representation of musical style, can be used in applications related to music information retrieval, score completion, audio mosaicking, etc. Research in this direction may also contribute to more reliable measures for musical similarity and would help get valuable insights into musical improvisation and composition.

Such work finds applications in musical ambiance generation in public places such as malls, restaurants and museums. It can be applied in education for musical training, compositional assistance and creation of new ideas for music composition. For example, Allan et al. [AW04] developed a system that automatically generates chorale harmonizations based on those composed by Johann Sebastian Bach. A system that is able to generate such novel musical ideas in the form of a score can be of great assistance in music training and compositional assistance for amateur musicians and students of music. It can be used, for instance, to extract a list of commonly occurring motifs or musical patterns in a certain style of music or works of a certain composer. These examples can then be used as a starting point to train students in composition of a certain style. The work of Cope [Cop96] serves as an excellent example of something that can be used for such an application.

Music therapy involves activities such as singing along, music improvisation, etc. where this type of technology can be useful. For example, Pachet's *Continuator* [Pac03] was applied to create an uplifting and cheerful mood among children who were very amused at the idea of a computer "responding" musically to something that they played on an instrument. This idea of carrying out a "musical conversation" with a system that responds favourably to a given musical stimulus could have positive effects on subjects, in a similar way, who attend a music therapy session. This is, however, in no way claimed to be a replacement to a conversation with a therapist, but only an interesting alternative.

Algorithmic composition tools in the form of programs and software for the computer also facilitate a shift of focus of the composer from fine details to broader picture. Rapid compositional prototyping tools such as *RapidComposer* [Mus11] already allow for such possibilities from computer software with features like automatic phrase, rhythm and chord generation. It is sufficient if there exists an idea in the mind of a composer about what he wishes his composition to convey and how he wishes it to sound as long as there is a suitable means (an interface) for him to communicate this to a machine. Another popular application of automatic music generation is in generating soundtracks for computer games. Context-specific music generation in games had been widely used by LucasArts in their games with the help of a system known as *iMUSE* [LM94]. There have also been some computer-based composition software in the past like Band-in-a-box [Ban11] and Impro-Visor [GTK10] that have been quite popular.

Several musical styles rely on reusing and transforming existing audio material to create new music. One could say that this trend of juxtaposing different audio segments to make music was born around the year 1942 in Pierre Schaeffer's *Musique Concrète*. This was the beginning of electroacoustic music, which is essentially music resulting from the manipulation of recorded or generated sound, emanating from loudspeakers, without an obvious human performer. The same concept has henceforth been adopted in numerous musical genres. A very well-known example of this is the hip-hop genre where, what are known as *samples* are extensively used as the basis for musical composition. These *samples* are essentially audio segments extracted from earlier songs which can be repeated, morphed or juxtaposed with other samples to create backdrops, themes, accompaniments, lead voices, etc. in making new songs. One extremely popular sample, known as "Amen Break", is a brief drum solo that occurs in the song *Amen, Brother* by the 1960s funk and soul outfit *The Winstons*. It has, for years, been used in numerous Hip-hop, Beakbeat and Pop songs.

The work presented in this thesis may be understood as stochastic composition in a Musique Concrète-like synthesis setting. This can be clarified further by dividing the entire process into two stages - analysis and generation. The initial analysis stage of the work involves, firstly, segmenting the audio signal and extracting pitch(or interval) sequences that make up the melody, and then quantizing them into appropriate clusters. Once such a quantized sequence is obtained, the variable-length Markov chain model is trained with it to build a probability distribution of subsequences that occur within it. This distribution is, simply speaking, a representation of the "style" of the melody.

Now during the segmentation stage, a collection of audio segments, each of which is an individual note in the melody, is also obtained. The generation stage essentially involves sampling a note at each time instant from the probability distribution for a subsequence, that has occurred in the immediate past, according to the learned Markov chain model. Once a sequence of notes has thus been sampled, appropriate audio segments are selected and juxtaposed to give the generated melody. The scope here, however, is a special case among methods that reuse or transform audio material to create new music (such as Musique Concrète) that is limited to audio consisting of monophonic melodies of the guitar and the bass-guitar.

Chapter 2

State-of-the-Art

"With the aid of electronic computers, the composer becomes a sort of pilot: pressing buttons, introducing coordinates, and supervising the controls of a cosmic vessel sailing in the space of sound, across sonic constellations and galaxies that could formerly be glimpsed only in a distant dream."

-Iannis Xenakis

Every model used for algorithmic composition is restricted by its nature. A model provides the composer with a specific methodology, a set of tools and intuition for composition. There can be variations in different parts of a model that may generate music of different types. In order to better highlight the specificities and novelties of attempts thus far in style-specific algorithmic composition, the entire process may be viewed in three stages: (1) Segmentation (2) Representation & Generation (3) Evaluation. Although every individual approach reviewed has had a different scope and application, it has contributed to one or more of these stages, and such a division of the process would help in organizing the approaches more clearly while keeping the general idea of the entire process intact.

2.1 Segmentation

Audio information from a source (a voice, a musical instrument, an orchestra, etc.) must be translated into a suitable format that can be processed by a computer. The options available at this stage are either to directly use the audio signal, or a symbolic representation based on MIDI or text. The latter does not require any segmentation of data available in the signal. Owing to the lack of a means to interface audio sources directly with a computer in the past, early approaches relied mainly on feeding it textual information. For example, the *Illiac Suite* of Hiller & Isaacson [HI59] was composed with a system of LISP programs that took an input stream of alphanumeric characters which represented a bass-line.

Majority of the recent approaches have used the MIDI representation for communicating musical information [Pac03], [Pai08], [Bil94], [Cop96]. This medium is preferred because one may avoid issues related to possible inaccuracy in segmenting audio data. and focus solely on the stylistic aspect of music. Moreover, the availability of various MIDI instruments for melody and percussion also makes these approaches feasible. However, it must be noted that, at the same time, the flexibility of such systems is also very limited to only a few instruments that are MIDI-compatible. Moreover, interfacing such systems with audio signal information would impair their performance when presented with a slightly imperfect input as a result of segmentation. However, such assumptions could still be considered valid when considering polyphonic data due to higher complexity of the audio signal.

The approach in this thesis proposes a more general framework that takes as input, directly, an audio signal (monophonic) and segments it into atomic components that serve as a basis for feature extraction. Marchini [Mar10] demonstrated a similar system in the context of generating rhythmic variations of a percussion track audio. This system was flexible and robust to audio recorded from several percussive sources such as drums, beat-box, etc. In the same spirit, Jehan [Jeh05] uses an intermediate minimal data representation directly obtained from the audio signal based on perceptual listening, for analysis and synthesis.

2.2 Representation & Generation

This is possibly the most important stage from the point-of-view of generating stylistically meaningful music. It is important to first have a set of instructions or rules that can describe logically or statistically the musical goal in focus. In the context of computer-based music generation, the task lies somewhere in between two extremes. Hiller and Isaacson note that most musical compositions reflect a balance between the extremes of order and disorder, and that stylistic differences depend to a considerable extent upon fluctuations relative to these two poles [HI59]. The choice of representation used for music generation is the key to achieving this balance depending on the context. As noted earlier, there are broadly two ways of approaching the problem of computer-based music generation (1) Deterministic (2)

Stochastic. Each of the approaches described below fall either strictly under one of these categories or combine aspects of both.

Probably the most popular example where computers are made to imitate musical style is David Cope's system called "Experiments in Musical Intelligence" (EMI). EMI analyses the score structure of a MIDI sequence in terms of recurring patterns (a signature), creates a database of the meaningful segments, and learns the style of a composer, given a certain number of pieces [Cop96]. His system can generate compositions with surprising stylistic similarities to the originals.

What has come to be known as *Evolutionary Music* involves solving problems by evolving a population of potential solutions to a problem, using standard genetic operations like crossover and mutation, until an acceptable solution emerges. This is based on the fundamental idea of how a genetic algorithm works. The generation process starts with a set of audio data (a piece, melody, or loop), which is initialized either randomly or based on human input. Then through the repeated application of computational steps analogous to biological selection, recombination and mutation, the aim is to produce more musical audio. An example of such a system developed for composing Jazz solos is *GenJam* [Bil94].

The Markov chain, owing to the fact that it incorporates sequential information into musical prediction, has been a frequent choice in research. Allan [All02] applies a HMM-based framework for harmonizing Bach chorales where the visible states are melody notes and the hidden states are a sequence of chords that would suggest possible harmonizations. A visible melody line is emitted from a hidden sequence of harmonies and the model also predicts three further notes at each time-step, one for each additional musical line in the harmonization. This is an example of the use of first-order Markov chains.

The approach by Paiement divides the task of melody generation into two parts: (1) rhythm generation and (2) melody generation on the generated rhythm. A rhythm is represented in terms of note onsets, note continuations and silences. A novel distance model, that uses Hamming distance to measure how similar two given rhythms are, is used to represent the rhythmic evolution of a melody over different constant-length time segments [Pai08]. Given the rhythmic sequence of a melody, the system uses an HMM trained on these distance sequences to predict a new sequence. A 5-tuple simplified Narmour feature, as proposed by Schellenberg [Sch96], is extracted from every set of three consecutive MIDI values of the melody. Another HMM is trained on this Narmour feature sequence, given the rhythm sequence. Finally, given this Narmour feature sequence, and a chord progression, a melody is generated using an IOHMM.

One major restriction on such HMM-based methods is imposed by the amount of training data required for learning aspects of musical style with HMMs. It is known that, in general, HMMs require a large amount of training data to be able to effectively generalize information, especially with increasing order, which also determines the performance of the model. The approach presented in this thesis employs an efficient and parsimonious tree-based representation of the variable-order Markov chain that can be utilized even in the presence of very limited training data. Talk of this representation now brings us to another recent and interesting experiment of Pachet which is known as *The Continuator* [Pac03].

The Continuator is intended to be a "Collaborative Musical Instrument". This system operates on short melodic phrases owing to the author's observation that Markov chains are (1) good at generalizing and learning musical style and (2) poor at representing long-term information. Various available MIDI parameters are used to obtain its symbolic representation with the help of a *reduction function* that parses a given input phrase based on various musical aspects such as pitch, velocity, duration, etc. or a combination of these. Sequences of symbols thus generated are parsed using an incremental parsing algorithm to train a variable-order Markov chain that maintains various possible sequences of symbols and their probabilities of occurrence. The system progressively learns more and more phrases from the musician to eventually develop a more accurate representation of her/his style.

Following a similar method, Marchini et al. [MP10] developed a system for the analysis of the structure and the style of a percussive audio sequence with the aim of generating an "arbitrarily long musically meaningful and interesting sound sequence with the same stylistic characteristics as the original". The framework was developed around a variable-length Markov chain that is used to predict and re-shuffle musical events. Applying various clustering thresholds simultaneously on segments obtained by onset detection, a multi-level discrete representation of the sound sequence is obtained. Then a regularity estimation of the levels is performed to determine the levels of refinement where regular time patterns appear. The periodic events are then used to estimate events, tempo, and meter. Based on this information, the Markov chains are used to generate continuations.

The classic paper by Assayag et al. [ADD99] discusses a dictionary-based prediction for automatic composition. Two dictionaries, namely, the "Motif Dictionary" and the "Continuation Dictionary" are used to represent and continue a given melody. The motif dictionary is composed of "motifs" which are elementary units that make up a melody. The continuation dictionary makes use of the motif dictionary information and estimates the probability of a single symbol (from the motif dictionary) continuing each motif minus the last symbol in it. A generation algorithm is used for continuation of a (so far) predicted sequence. A "context" variable is maintained which determines the maximum previous sequence to consider while making the prediction. The prediction is based on whether the context matches any of the motifs in the continuation dictionary. The probabilities in the continuation dictionary are used to choose the next symbol. Upon lack of any choice, the algorithm either stops or picks a symbol at random.

A multiple-viewpoint approach towards music prediction through the use of several Markov chains (each corresponding to a viewpoint) was proposed by Conklin et al. [CW95]. In this system, a collection of independent views of the musical surface each of which models a specific type of musical phenomena are used to train different Markov chains from data, which are then used for prediction. This method is applied to Bach chorales to generate new pieces. One obvious question that may be raised in regards to such a method is its musical meaning. Musical composition/improvisation involves intricate interplay between different musical phenomena ("viewpoints", in this context) rather than each one occurring separately by itself. For instance, the occurrence of notes in a melody of a certain style depends very much on the meter of the song which, perceptually speaking, determines at what metrical positions the listener perceives which notes. This combined information helps determine, for example, which mode a melody is in. Assuming independence between such facets of music, while possibly allowing for greater variation, also results in the loss of style-specific information.

The current approach uses a cross-product representation between pitch and metrical position (covered in 4) that, while limiting the number of possible continuation choices at a certain time instant, is also faithful to the element of style.

2.3 Evaluation

Once a symbol sequence has been generated according to a model on the computer, it has to be converted into an appropriate sound sequence (melodic, percussive, etc.) as musical output. Moreover, once the music has been generated, the question arises as to how one evaluates it. The frequently differing methods of evaluation employed on music generation systems have led to some stating that "this is a largely ignored aspect of research into algorithmic composition" [PW01]. There exist both objective and subjective measures to evaluate music generated by computers. Objective evaluation relies purely on certain formulae to determine the musical quality of output. A definite criterion, that helps objectify performance, is defined and how much the musical output adheres to this criterion is measured. That being said, the artistic nature of the output in the context of music generation creates room for subjective opinion as well. These are typically carried out via questionnaires provided to experts or naive listeners, as the situation demands. In this case, neither can be said to be better than the other and one must select an evaluation method that best suits the situation. A brief discussion on the question of evaluation of music composed using statistical models can be found in [Con03].

An example of an objective evaluation criterion can be seen in the work of Paiement [Pai08]. Here, a 100% accuracy is associated with a replication of the original melody. That is, depending on the note-to-note similarity of the generated melody to the originally presented melody, it has a higher or lower score. The problem with this type of evaluation is that it ignores the aesthetic value of the generated music. Using such a scheme, even those continuations which sound pleasing, but are symbolically dissimilar to the original would have a low score. Moreover, this is not the ideal evaluation scheme where one wishes the generation to vary in comparison to the original. It would seem more appropriate to consider a range of the defined "accuracy" to judge its goodness as a trade-off between originality and replication rather than going by the absolute performance measure which would favour the replication of the original melody.

The Continuator, being a *collaborative musical instrument*, was evaluated not only on the quality of music generated by it, but also on the possibly new modes of collaboration that it can help achieve. The author outlines various modes of collaboration such as *single & multiple autarcy, master/slave*, etc. that demonstrate the collaborative capabilities of the system. These capabilities themselves serve as a measure of the efficacy of one important aspect of the system. In addition to this, the musical quality of the system is evaluated with the Turing test where listeners are presented with a melody and asked to identify which has been played by a human and which one by the Continuator. It is also noted that the system passes the test only in the case of certain specific kinds of melodies. While the Turing test does indeed seem to be a very suitable way of evaluating the output of this thesis, it was not considered due to the fact that the generation, which involved transforming the original audio, often contained several synthesis artefacts in its timbre, tempo, etc. that would make the result obvious. With the correction of these artefacts not being the main focus of this work, the Turing test was avoided in this context.

From a probabilistic perspective, generating music similar to a given style can fundamentally be viewed as sampling from a distribution that is representative of the style [Con03]. Thinking along this line, one can possibly conclude that the nature of the chosen representation would greatly influence the musical output of the system. Based on this idea, a suitable judgement criterion for assessing the goodness of the model used for generating music is to measure how likely the composers original music is with respect to the model. In the Bach-chorale harmonization system in [All02], they look at how high a probability the model assigns to Bach's own harmonizations given the respective melody lines. So a high probability of

Bach's own chorales in the model would be a good sign that the model was able to generalize to Bach's style quite well. The author believes that these figures allow the model's performance to be directly compared with that of any future probabilistic harmonization system. In a similar way, a generation system that models musical style can also be trained with a multitude of styles and used as a classifier. This is what is has been done in the case of the *Audio Oracle* [DAC07]. For each style there is a different trained model, and new pieces of music are classified according to how likely it is for either model to produce this particular musical output. In the present situation, although this mode of evaluation is suitable, it is not used due to the absence of a large amount of data that would be required.

As one can see, there are several possibilities available for evaluating this type of work. The choice of an appropriate evaluation scheme depends on the important aspects of the work in question. As demonstrated by Marchini et al. [MP10], one may evaluate the system using feedback from experts. This would require the opinions of either professional musicians or even experienced listeners who have some idea of music theory, similarity, etc. Their system was evaluated by two professional percussionists. Each expert was presented with a questionnaire that asked about the performance of specific features of the system. Whether the musical goals of the system were achieved or not was assessed based on the response to this questionnaire. A similar scheme of evaluation was employed in this work as well. It shall be covered in more detail in Chapter 6.

Chapter 3

Segmentation

The segmentation stage involves extracting appropriate features from the given input for presenting to the generation algorithm. It converts the raw input data into low-level features that will serve as the basis for further analysis. In this work, the starting point is the signal corresponding to the melody read from an audio file in WAV format. This chapter describes the segmentation process that it undergoes prior to generating a mid-level representation using machine learning.

This section describes the stages involved in segmenting a given input melody to obtain low-level features that describe individual notes that make up the melody. The note segmentation process proceeds in two distinct steps (1) Onset Detection (2) Pitch Estimation.

3.1 Onset Detection

After the choice of input modality (as electric/bass guitar) has been made, a given melody played by this instrument is to be segmented into a sequence of notes or comparably short segments which form the atomic units that would be used for generation. On experimenting with different methods available for onset detection such as the Kullback-Leibler method, and those based on spectral difference, phase, energy, complex domain methods, etc., it was found that the onset detection based on *complex domain* [DBDS03] worked better than the rest for the chosen input modality. A detailed review and comparative study on different methods for onset detection, which served as a useful guide in making this choice, is available in [BDA+05]. The *Aubio* toolbox by Brossier [Bro06] contains implementations of a variety of onset detection algorithms, with the facility to adjust various algorithm parameters. The features of this toolbox can be integrated with Matlab code through the *Sonic-Annotator* command-line interface [Can11].

3.2 Onset Cleaning

Following onset detection, it was found that, while the chosen algorithm was able to detect note onsets fairly accurately, it also presented a problem of multiple detections in quick succession. Such misdetections often arose due to the guitar-pick or the guitarist's finger scratching against the string. It was observed that such multiple detections typically lied within a time-gap of around 100ms. As a post-processing step for the onset detection, a time-based onset cleaning step was introduced, where, if more than a single onset occurred within a time-span of 100ms, only the first one would be considered as a valid onset and the rest would be removed. Another onset

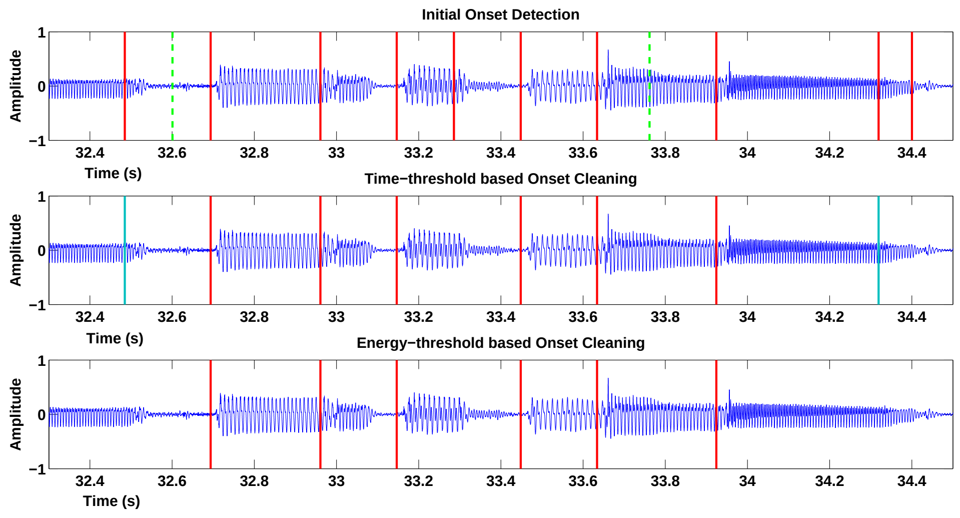


Figure 3.1: Different stages involved in onset detection and cleaning for an example segment of an audio signal. The first row shows the original onset locations. The onsets depicted in green-dashed lines are removed using the time-threshold. The second row shows the onset locations after this time-threshold based cleaning. The onsets depicted in blue lines are removed using the energy-threshold. The third row shows the final remaining onsets.

cleaning step was based on signal energy. In certain cases, (almost) silent segments were detected as separate notes. As it makes no sense in the pitch estimation of such silent segments, those segments that fell below an energy threshold of 40% of the average signal energy throughout the melody, were merged with the previous non-silent segments. Figure 3.1 illustrates the steps involved in onset detection. The onset detection also tends to overlook smooth changes in notes such as those due to a slide, hammer-on or pull-off as the onset is not strong enough. These are not considered as errors and are handled in the same way as those segments with a single note detected within it.

3.3 Pitch Estimation

The onset detection gives us possible candidates for notes. As only monophonic melodies are dealt with here, the features are based on pitch. The *YIN* algorithm [dCK02] is used for this purpose. Once again, the implementation of this algorithm available in the *Aubio* toolbox was used. Pitch detection is applied to each segment between two consecutive onsets and this yields a value for the pitch of the segment. For simplicity, the assumption here is that each segment between two consecutive onsets contains only one note. For example, if there exist pitch-bends, legatos where some of the note changes could not be detected, etc. in a segment, the pitch of the entire segment would be that of the initial time following the onset. This was determined by explicitly testing the pitch-estimation feature of the toolbox on these specific cases. An example output of the pitch estimation process is shown in Figure 3.2. As illustrated here, the onset detection is not ideal and there do exist missed onsets which are indicated here. Among those onsets that were missed, only those that signify the beginning of a riff are indicated due to their relatively greater importance. Pitch estimation is also susceptible to octave errors. As the reader may already be familiar, such errors involve the estimated pitch being an octave above ("too high" errors) or an octave below ("too low" errors) the actual pitch. An intrinsic part of segmenting the audio signal is to determine the correct set of parameter values for the onset detection and pitch estimation algorithms. Sonic Annotator provides a means to specify various parameter values for both. Several combinations of values were tried out, with the goal being a global set of parameters that work reasonably well for all the chosen data. Table 3.1 details the final values that were arrived at, and used for note segmentation.

In the end, the segmentation stage yields a set of onset times $\mathbf{t} = (t_1, t_2, \ldots, t_N)$ with associated pitches $\mathbf{p} = (p_1, p_2, \ldots, p_N)$.

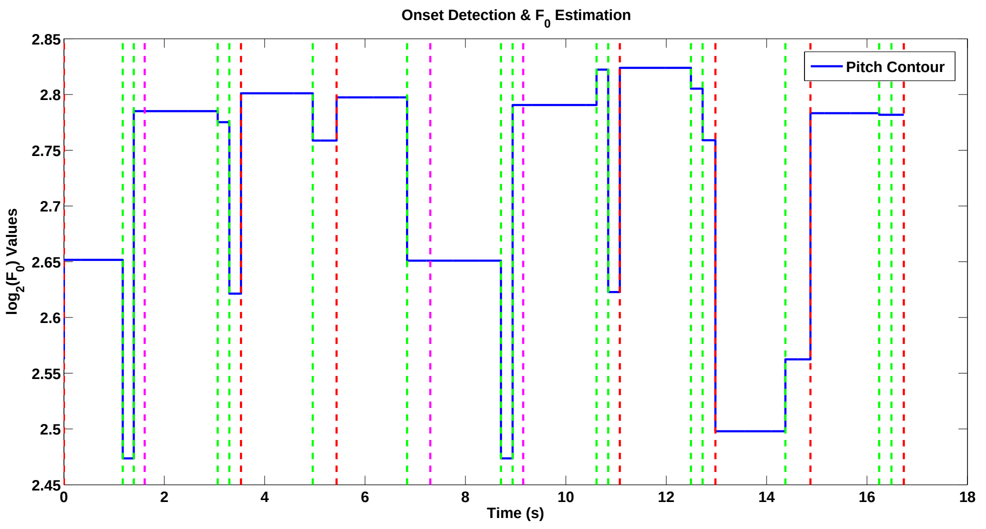


Figure 3.2: Pitch detection using YIN for a section of the intro bass-riff of the song *Calling Dr. Love* by the hard-rock group *Kiss*. The vertical dotted-lines refer to onset locations. Red signifies the beginning of each riff. Magenta indicates riff beginnings that were not detected.

Parameter	**Value**
Pitch Type	1 (*"yinfft"*)
Step Size	512
Block Size	2048
Max Pitch	127
Min Pitch	0
Onset Type	1 (*"complexdomain"*)
Peak Pick Threshold	0.5
Silence Threshold	-65
Wrap Range	0
Avoid Leaps	1

Table 3.1: The final set of parameters used for note segmentation with *aubionotes*.

Chapter 4

Representation

"...computer music systems are not neutral. Every system constrains musicians to a restricted set of operations; every view on a piece is a filter that biases the viewer's attention to a particular perspective. Indeed, the holy grail of a "universal" representation is antipodal to creative music. Music is constantly evolving, so perhaps one should not pray for a definitive solution to the questions of music representation."

-Herbert Brün

The segmentation stage, as described in Chapter 3, segments the input audio signal (ideally) into individual note segments and estimates a pitch value corresponding to each of these segments. Following the generation of this low-level representation, a higher level representation is derived from it that converts the sequence of pitch values into symbols for training the variable-length Markov chain. This approach explores two such high-level representations using (1) Pitch Clustering (2) Interval Quantization.

4.1 Pitch Clustering

At the present stage, we are given a sequence of pitch values (in Hz) as detected by the YIN pitch detection algorithm. This is depicted in Figure 3.2. We are faced with the problem of determining which notes these frequency values, estimated for each inter-onset segment, correspond to. This is owing to the fact that we do not assume any prior information about temperament, instrument tuning, number and pitch of scale notes or octave information of the given frequency values (in the case of equal temperament for example, a B_3, E_1, $F\#_2$, etc.). Moreover, while two frequency values may correspond to the same note, the detected frequency values are not identical to each other, but are very closely spaced.

Firstly, the pitch values (in Hz) are transformed into a logarithmic scale (of base 2) in order to apply a linear distance measure to cluster them. Grouping of similar pitch-values is realized using agglomerative single-linkage clustering. This yields a dendrogram representing their nested grouping and levels at which groupings change. That is to say, at the bottom of the dendrogram, each leaf-node corresponds to an individual log-pitch value. This corresponds to the finest similarity threshold value. At the root node, this value is maximum and all pitches are grouped into a single cluster. This gives us a coarse-to-fine (top-to-bottom in the dendrogram) cluster representation of the pitch data. Figure 4.1 shows example dendrograms for one of the songs from the input database. As the log-pitch values are scalar, the absolute value of their difference is used as the distance measure for linkage. For a review of the clustering algorithm employed here and a more detailed background on the topic, the reader is referred to [JMF99].

The above described method for clustering yields clusters at multiple distance-threshold levels. Often these are too many in number, all of which cannot be used. This necessitates selecting the level that corresponds to the "ideal" number of clusters. The task of determining the best number of clusters for a given data distribution is one that has received much attention over the years in a variety of research areas. Milligan & Cooper review the performance of 30 different criteria for this purpose in [MC85]. The variance ratio criterion (VRC) [CH74] that was verified as one of the more effective one in their analysis was chosen to be applied in the present context. This method estimates "the best sum-of-squares split" of the dendrogram using the Within-Group Scatter Sum (WGSS) and Between-Group Scatter Sum (BGSS). The idea is to have clusters that are well-separated (high BGSS) and, at the same time, compact (low WGSS).

4.1.1 The Variance Ratio Criterion

An explanation of the VRC for scalar-valued data is as follows. Suppose that we have a set of n individual log-pitch values $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$ in a one-dimensional Euclidean space. These can be represented by the $n \times 1$ data vector $\mathbf{p} = (p_1, p_2, \ldots, p_n)$ with the i^{th} row given by the log-pitch value p_i. Thus, the clustering of these n pitch values will be given by the partition of the rows of $\mathbf{p}$. Without loss of generality, it may be assumed that the center of gravity of the total n points is zero. Thus, the total scatter matrix of the n points is given by

$$T = \mathbf{p}^T \mathbf{p} = \sum_{i=1}^{n} p_i^T p_i$$

Now, suppose that we have a partition of the n log-pitch values into g groups with $n_1, n_2, \ldots, n_g$ values in each group, such that $n = \sum_{i=1}^{g} n_i$. Then, for the k^{th} group the row-elements of $\mathbf{p}$, p_{lk} (for $l = 1, \ldots, n_k$) represent the log-pitch values in group $\mathcal{G}_k$. One can now define the scatter for each group $\mathcal{G}_k$ with center of gravity c_k by

$$w_k = \sum_{l=1}^{n_k} (p_{lk} - c_k)^T (p_{lk} - c_k).$$

The pooled within-group scatter is defined by

$$w = \sum_{k=1}^{g} w_k.$$

The between group scatter is defined by

$$b = \sum_{k=1}^{g} n_k c_k^T c_k.$$

Hence, for each partition (at each clustering level) of the n log-pitch values into g partitions, we have the following identity

$$t = w + b.$$

What has been described here in the case of one-dimensional log-pitch values may be extended to a general case of n-dimensional data. This is clearly detailed in [FR67]. However, in this case, since the total scatter matrix t is fixed, a natural criterion for grouping is to minimize w. This is equivalent to maximizing b. The Variance Ratio Criterion (VRC) uses the values of b and w at each clustering level to find "the best sum of squares split" of the dendrogram by evaluating

$$VRC = \frac{b/(g-1)}{w/(n-g)}.$$

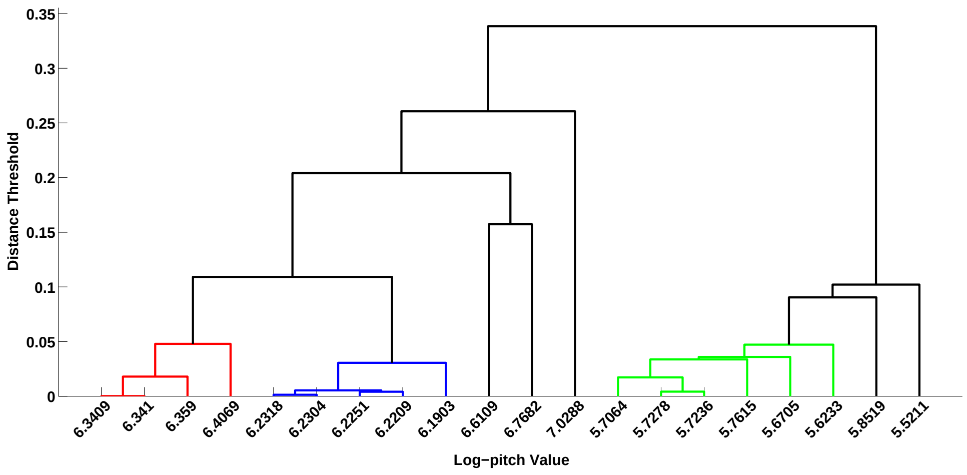
The VRC is an increasing function of the number of clusters. It is suggested in [CH74] that those numbers of clusters g (at certain clustering levels) be chosen for which the VRC has an absolute or local maximum, or at least a comparatively rapid increase (steep slope). Depending on the case, either situation might occur. And also that if there exist several local maxima, the most economical choice would correspond to the smallest value of g. Following this suggestion, in the present case, these conditions are evaluated in an order such that the local maxima are given the first preference starting with the level containing the least number of clusters, followed by the slope (only if no local maxima occur).

The log-pitch value cluster-levels that are the VRC maxima are sorted in increasing order of the number of clusters (or slope, if that be the case) and the top C levels are chosen. In the experiments, a value of $C = 4$ was used. This is done, firstly, as there is no a priori knowledge of the correct number of pitch-clusters that actually occur in the melody and as the VRC only provides an estimate of the best clustering levels. Secondly, selecting multiple cluster-levels also provides us with more patterns at different (coarse-to-fine) levels to learn from the data. This will be explained in more detail later in Chapter 5. It is not necessarily the case that each obtained cluster contains frequencies corresponding to a single note. The similarity threshold corresponding to each selected level determines the precision of clustering. For instance, among the selected levels, the one that contains the least number of clusters is more likely to have grouped two or more consecutive frequencies (that may even correspond to different notes) into the same cluster. The clustering only provides an abstract representation of notes at each level (henceforth referred to as *note-unit*). The number of note-units, depending on the melody, typically varied between 3 at the coarsest level (level 1) up to 30 in the finest level (level 4).

4.2 Interval Quantization

It was shown by Kim et al. [KCGV00] that rhythmic, and more importantly in the context of this work, melodic contour based on an interval representation was effective in order to identify melodic similarity. Their approach involves computing the contour of a given melody using only a few quantization steps to indicate the change of notes. The approach described here aims at deriving a similar representation. However, the difference being that, while in the case of [KCGV00], depending on the number of quantization steps, interval boundaries are pre-determined, the current representation approaches this problem in a data-driven manner for sequence generation.

For example, if Q is a vector used to represent different contour resolutions and



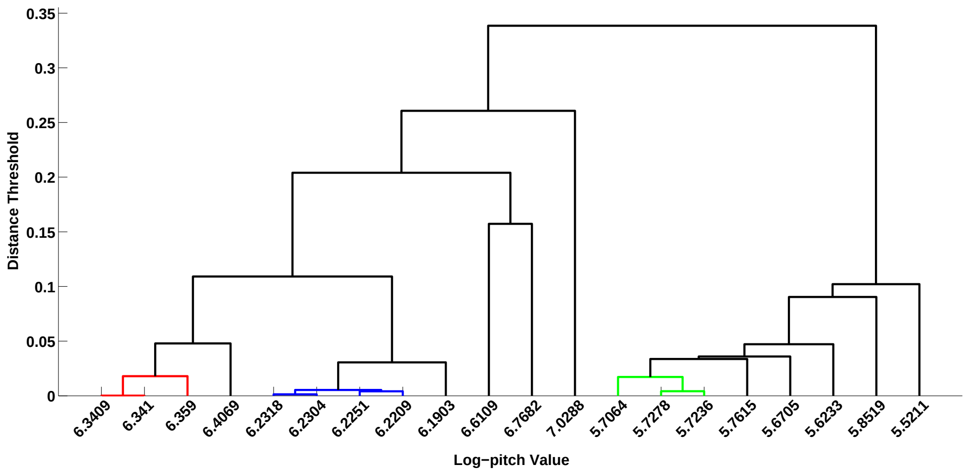


Figure 4.1: An illustration of the dendrogram yielded by agglomerative single-linkage clustering of the first 20 log-pitch values of the bass-line of the song *Hysteria* by the rock group *Muse*. In the top figure, clusters obtained with a distance threshold of 0.06 (8 clusters) are highlighted in different colours (with black being the case where the leaf node is its own cluster). While in the bottom figure, the same is shown when the threshold is 0.02 (13 clusters).

quantization boundaries, the length of Q indirectly reveals the number of levels of contour being used, and the individual values of Q indicate the absolute value of the quantization boundaries (in number of half-steps). $Q = [0\ 1]$ represents that interval changes are quantized into three levels, 0 for no change, + for an ascending interval (a boundary at one half-step or more), and for a descending interval. This representation is equivalent to the popular $+/-/0$ or U/D/R (up/down/repeat) representation. Similarly, $Q = [013]$ represents a quantization of intervals into five levels, 0 for no change, $+$ for an ascending half-step or whole-step (1 or 2 half-steps), $++$ for ascending at least a minor third (3 or more half-steps), $-$ for a descending half-step or whole-step, and $--$ for a descent of at least a minor third. It was also observed in their experiments that a 5-level representation (such as the one mentioned above) was effective in achieving some of the best results. However, the criterion for setting the interval boundaries in their approach, it must be noted, is manual.

This work adopts the quantized-interval representation of [KCGV00] for generating the continuation of a melody based on variation of its interval contour. A clustering approach will be explored for automatically determining boundaries for quantizing intervals based on the characteristics of a given melody. Such an approach can also, possibly, be extended to large databases containing similar melodies. Such a representation is also considered advantageous as, often, pitch-detection information extracted from audio files carries with it a small amount of noise about the actual value of the pitch of a note. An approach such as the one described below would help smooth-out such noisy information.

Following the segmentation process described in Chapter 3, we have a sequence of pitch-values in Hz. These are transformed into a logarithmic scale (of base 2) in order that equal differences in perceived pitch values are equal at different frequencies. Then, a sequence of intervals is obtained by taking the difference between consecutive log-pitch values. With this, negative values would indicate descending pitch-contour, while positive values would indicate ascent. We use the K-means clustering with a symmetry constraint (by manually selecting the value of K) in order to cluster the intervals. As the features being used here are one-dimensional interval values, it is expected that such a clustering method would provide a solution to the interval quantization boundary estimation problem at hand by obtaining the representation described in [KCGV00].

Although this representation was implemented, due to the limited time available for the completion of this work, it could not be evaluated as in the case of the pitch representation. Hence, only a theoretical explanation of the method is presented.

4.2.1 K-Means Clustering with Symmetry Constraint

As the name suggests, this method involves applying a K-means clustering over the interval values. The intervals are grouped into 3, 5 and 7 clusters. This would result in grouping of interval sizes progressively from large negative (descending) to near-zero (constant pitch or a very small change in either direction) to large positive (ascending). In addition to this, there is also a symmetry constraint imposed on the clustering. That is, the boundaries of clustering are assumed to be symmetric on either side of the cluster corresponding to a zero-interval (in the ideal case). To do this, the sign associated with the interval (ascending/descending) is first discarded and only its size is considered. On the resulting interval values, for example, if a 5-cluster representation is desired, a K-means clustering is performed with $K = 3$. This gives the boundaries in the positive (ascending) interval direction, which are then simply mapped in the negative (descending) direction about the zero-interval cluster to give 5 quantization levels. In experiments, the values of K are manually set in such a way that 3, 5, 7 and 9 quantization levels are obtained. Boundaries thus obtained are then used to quantize different interval values at multiple (in this case, 4) levels.

4.2.2 Interval-Pitch Matrix

A quantized-interval only gives us the approximate relation between two notes, but in no way the notes themselves. Assuming that we are given the note that precedes the interval, we need some kind of a mapping function that would help choose an appropriate set of notes that would follow the first one. We define an *Interval-Pitch Matrix* that, given an initial pitch and a quantized-interval that follows it, indicates the possible choices for the next pitch. Following this, various constraints are applied over these available choices to narrow down on a suitable pitch depending on the context.

The interval representation was implemented and also used for generating melodies, but owing to limited availability of time, could not be fully evaluated.

4.3 Metrical Analysis

Every melody has an underlying rhythm that gives it a certain structure according to a beat or metrical grid. Some of the prior approaches model the rhythmic structure of a melody explicitly to use the information while generating continuations [Pai08, Pac03]. This section describes the process of deriving the symbolic representation for the underlying rhythm of the melody. In the present approach, the onset times of the different notes in the melody are used to determine this rhythmic structure at an appropriate metrical level. What is meant by "appropriate" here will be explained below. While the rhythm is dealt with separately during this process, it shall be explained in Section 4.4 how the representation thus derived for the rhythm will be combined with that of the notes (or intervals) to obtain a joint representation that will be used to train the variable-length Markov chain.

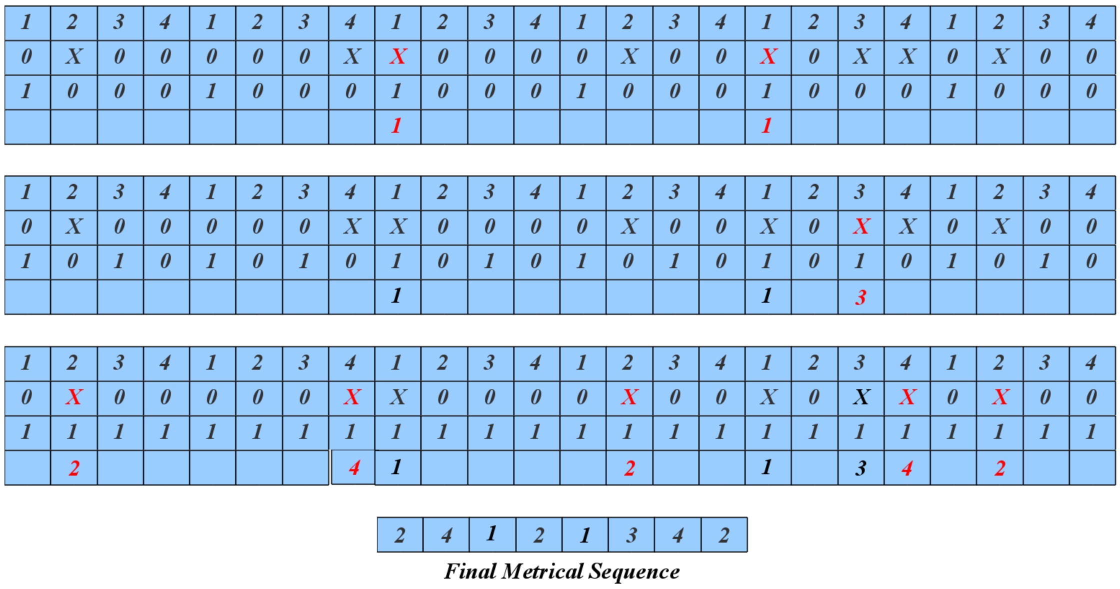
The idea here is to obtain a sequence of symbols that represent metrical locations (for example, from the set $\mathcal{M} = \{1, 2, 3, 4\}$, for a meter of length 4) that correspond to the locations of note onsets in the melody. The method can be best explained

1	*2*	*3*	*4*	*1*	*2*	*3*	*4*	*1*	*2*	*3*	*4*	*1*	*2*	*3*	*4*	*1*	*2*	*3*	*4*	*1*	*2*	*3*	*4*
0	*X*	*0*	*0*	*0*	*0*	*0*	*X*	*X*	*0*	*0*	*0*	*0*	*X*	*0*	*0*	*X*	*0*	*X*	*X*	*0*	*X*	*0*	*0*

Figure 4.2: The underlying rhythmic structure of an arbitrary melody where the 'X's indicate onset locations.

with the aid of an example. Consider the rhythm represented by the sequence of symbols in the lower row of Figure 4.2. Here, each X denotes the presence of an onset and 0 indicates the absence of one (a rest). The assumption is that the time-gap between successive symbols is equal. As can be seen, each pair of notes is separated by an arbitrary time-gap (a multiple of the time-gap between successive symbols). The upper row shows an imaginary sequence of metrical weights aligned with the rhythm as reference, for the sake of clarity. The "appropriate" metrical level mentioned earlier refers to the metrical level where the IBI (in this case, the temporal gap between 1 and 2, 2 and 3, etc.) is as small as the temporal gap between a pair of onsets that are closest to each other. A meter of length 4 is assumed here. The task at hand is to label the onsets of the lower row (symbolize them) according to the metrical positions in the upper row of Figure 4.2. It essentially generates a sequence of symbols (from the set $\mathcal{M} = \{1, 2, 3, 4\}$, in the current example) that indicates *where an onset occurs* in a beat sequence at the finest temporal resolution required to coincide the beats with every onset in the melody.

As the first step for metrical analysis, the beat-detection algorithm as proposed by Dixon et al. [Dix01] is applied to the given melody. The Java implementation of this algorithm, known as *BeatRoot* [Dix07] is used for generating a beat sequence underlying the given melody at an arbitrary metrical level. The metrical level at which beats are detected is assumed to be at the coarsest temporal resolution, which is the starting point for the method. For the example rhythm under consideration, such a beat sequence is depicted in the third row of the first grid in Figure 4.3. Each “1” indicates the beat location. It is to be noted that the beat detection algorithm may yield a beat sequence at a different metrical level as well. It can



1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
0	X	0	0	0	0	0	X	X	0	0	0	0	X	0	0	X	0	X	X	0	X	0	0
1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0
								1								1							

1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
0	X	0	0	0	0	0	X	X	0	0	0	0	X	0	0	X	0	X	X	0	X	0	0
1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
								1								1		3					

1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
0	X	0	0	0	0	0	X	X	0	0	0	0	X	0	0	X	0	X	X	0	X	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	2						4	1					2			1		3	4		2		

2	4	1	2	1	3	4	2

Final Metrical Sequence

Figure 4.3: Iterative representation of the process that determines the metrical locations of each of the onsets in the melody by progressively increasing the resolution of the underlying metrical structure.

be seen in the first grid of Figure 4.3 that some of the onsets (in red) coincide with the generated beat-sequence, which corresponds to the metrical location “1”. The procedure from here on is to progressively halve the inter-beat-interval (IBI), that is, to progressively increase the resolution of beats by a factor of 2 until all of the onsets coincide with a beat. This process is illustrated in Figure 4.3. In each iteration, the metrical location (indicated in the first row of every grid) of the onset that coincides with a beat is noted. In this way, we get a sequence of metrical locations that correspond to onsets of the melody.

However, in an real-scenario, detected onsets are not exactly coincident with the generated beats. To handle this, for each onset, it is checked whether it lies within a certain threshold in time from a beat closest to it. Following the approach of [Dix01], which relies on perceptual studies for discrete perception of closely spaced notes, this threshold was set to 70ms. So, in each iteration of the algorithm, the resolution of the meter is doubled and onsets that lie within a range ±70ms of a beat at that resolution are considered to be a match with it. This process is repeated until at least 90% of the onsets are matched with beats as it was found that, at times, the IBI tends to become as small as 140ms (a trivial case when the match threshold is ±70ms) for all the onsets to be matched. This would be too small an IBI.

Another interesting issue that arises here is the possibility of confusing a metrical location with one that occurs integer times a measure away from that location. Consider the first 8 symbols of the example of Figure 4.2 (measures 1 and 2), shown in Figure 4.4. The above described method for getting a sequence of metrical

1	*2*	*3*	*4*	*1*	*2*	*3*	*4*
0	*X*	*0*	*0*	*0*	*0*	*0*	*X*

Figure 4.4: A section from the example that illustrates the limitation of the metrical analysis method.

locations would give us the sequence $(2, 4)$ for this section, but the information about the measure to which these locations belong is lost. In this representation, one could assume that the two onsets are in the second and fourth metrical positions of the same measure. That is, if we wish to re-generate an onset sequence from the metrical locations determined by the algorithm, we would get what is shown in Figure 4.5. The result of this error would be an incorrect variation of the actual tempo of the melody in the representation. To handle such a case, initially, a simple

1	*2*	*3*	*4*
0	*X*	*0*	*X*

Figure 4.5: A section from the example that illustrates the limitation of the metrical analysis method.

workaround was proposed. The assumption was that it would suffice as long as we are able to reproduce these measure considerations not in the exact, but a rough sense while generating a melody from the example. That is, the overall trend of how often a certain metrical location symbol that follows another is in the same or a different measure as the first is to be maintained. First, a look-ahead variable is defined that determines the upper-limit on the number of measures after which the next metrical location can occur. This value is derived from the specific melody and is equal to the number of measures corresponding to the longest note duration in it. Presently, the lookahead variable's value is assumed to be 2 for explanation. Next, conditional probability distributions of the form $p(m|l)$ are defined, where,

$m = \{0, 1, 2\}$ refers to the measure for the case where the look-ahead variable is 2. Each distribution essentially defines the probability of the next onset occurring m measures away from the current onset, given that the current onset is on a metrical position l. In the example of Figure 4.4, the metrical location “4” occurs $m = 1$ measures after the metrical location “2” (a “0” would mean the same measure). Here, $l = 2$ as the metrical location of the current onset is “2”. It is now possible

$$\begin{pmatrix} 1/2 & 1/2 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

Figure 4.6: The *Measure Matrix* for the example in Figure 4.2. Each row corresponds to a metrical position $1, \ldots, 4$, and each column corresponds to a measure $0, \ldots, 2$.

to define a $l \times m$ matrix A, where $a_{i,j} = p(j|i)$ according to the above definition. For the example in Figure 4.2, the matrix A is shown in Figure 4.6. So given that we have a generated sequence of metrical locations, we determine the duration of a note corresponding to the current metrical location depending on the metrical location of the next note *and* the number of measures after which it occurs (that is determined by the *Measure Matrix*).

In practice, however, this representation performs rather poorly by producing notes that are often either too long or too short. Moreover, this representation also counters the reason for using Markov chains for representing the rhythm structure of the melody. The VLMC in this context aims at achieving as sort of “variable memory” representation where symbol sequences of different lengths would represent different memory lengths. For such a representation to be theoretically correct, it is also necessary that each pair of consecutive symbols are equally spaced in time. This is to ensure that a sequence of, say, n symbols always corresponds to the same duration of time t_n seconds. If this condition isn’t satisfied, a sequence of, say n symbols, could refer to any arbitrary duration of time. The previously described representation suffers from this drawback.

Hence, as an improvement over the previous representation, a slight modification of it is used. As explained previously, correspondence between onsets and beats is established in much the same way. The main difference here is that, while in the previous representation, only onset locations are assigned a symbol value, in the improved representation, rests (metrical position where there is no onset) are also assigned a symbol value. For example, like in the previous example, say we consider

$\mathcal{V}/\mathcal{M}$	1	2	3	4
0	**1**	**2**	**3**	**4**
1	**5**	**6**	**7**	**8**

Table 4.1: A depiction of the symbolic representation for rhythm when the assumed meter length is 4. Here, $\mathcal{M} = \{1, 2, 3, 4\}$, $\mathcal{V} = \{0, 1\}$ and $\mathcal{R} = \{1, 2, 3, 4, 5, 6, 7, 8\}$.

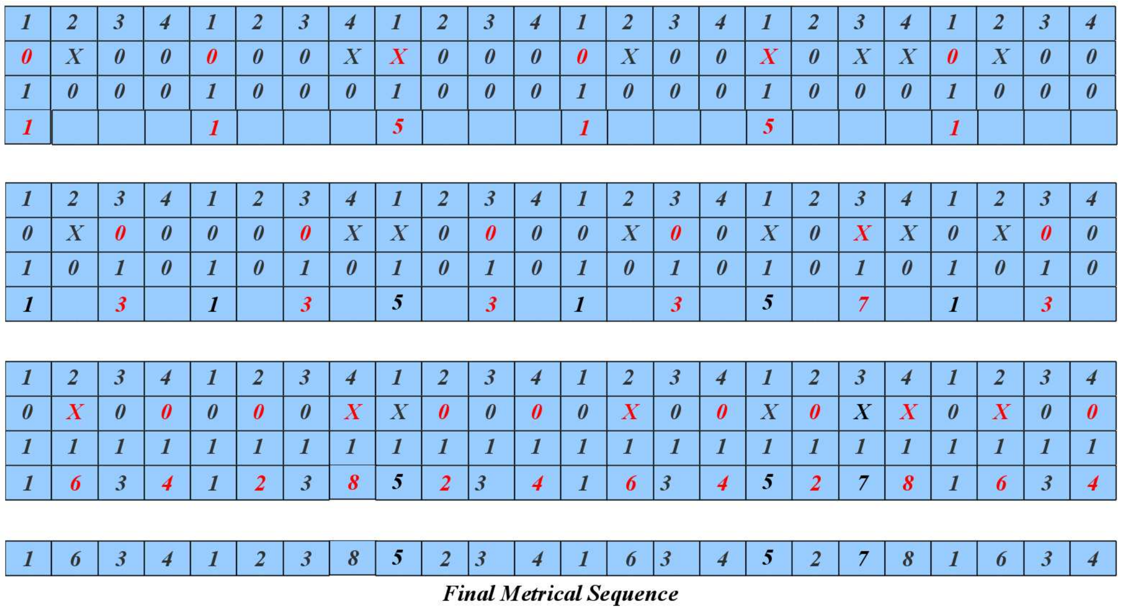
a meter length of 4 where the symbols for each metrical location are denoted by $\mathcal{M} = \{1, 2, 3, 4\}$. Now, we also consider two additional cases, denoted by $\mathcal{V} = \{0, 1\}$, where 0 indicates the presence of an onset and 1, the absence of it. We obtain a symbolic representation $\mathcal{R}$ as a cross-product $\mathcal{M} \times \mathcal{V}$ of these two sets of values as shown in Table 4.1. In this representation, a symbol is generated corresponding to every beat (at the final metrical level where at least 90% of the onsets match with beats) irrespective of whether there exists an onset there or not. For example, at metrical location 1, the symbol assigned when there is a rest, is 1. While, when there is an onset, the assigned symbol is 5. More generally at a homogeneous time instant i, given a metrical location $m_i \in \mathcal{M}$, and a onset-type symbol value $v_i \in \mathcal{V}$, the cross-product rhythm symbol r_i of these two can be written as the ordered pair

$$r_i = (m_i, v_i)$$

This symbolization scheme is implemented at multiple meter-length levels, namely 1, 2 and 4. At each level, the number of symbols for representing the rhythm is twice as many as the meter length itself. This representation also has the advantage that it is homogeneous in time and, hence, a uniform representation of the variable-length memory concept that is desired to be achieved using the VLMC. This essentially means that a sequence of metrical location symbols (learned or generated) in this representation are equally spaced in time. For the example onset sequence in Figure 4.2, the derivation of a symbol sequence according to this new method is shown in Figure 4.7. Once again, the starting point is the metrical level corresponding to the detected beats. Here, only a few onsets are matched with the beats. Then, iteratively the IBI is halved to get a new set of beats that will match with more onsets. The criteria for termination here, as before, is a match between the beats and at least 90% of the onsets.

4.4 Final Melody Representation

Sections 4.1, 4.2 and 4.3 described how, individually, note and rhythm patterns of the melody are captured. In order to finally represent the entire melody, two


Final Metrical Sequence

1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
0	X	0	0	0	0	0	X	X	0	0	0	0	X	0	0	X	0	X	X	0	X	0	0
1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0
1				1				5				1				5				1			

1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
0	X	0	0	0	0	0	X	X	0	0	0	0	X	0	0	X	0	X	X	0	X	0	0
1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
1		3		1		3		5		3		1		3		5		7		1		3	

1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
0	X	0	0	0	0	0	X	X	0	0	0	0	X	0	0	X	0	X	X	0	X	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	6	3	4	1	2	3	8	5	2	3	4	1	6	3	4	5	2	7	8	1	6	3	4

1	6	3	4	1	2	3	8	5	2	3	4	1	6	3	4	5	2	7	8	1	6	3	4

Figure 4.7: Illustration of the iterative process that determines the time-homogeneous metrical symbols of each of the onsets (and rests) in the melody by progressively increasing the resolution of the underlying metrical structure.

possible options are explored (1) Disjoint VLMCs for rhythm and notes (2) A single VLMC using a combined representation.

4.4.1 Disjoint Representation

This served as an intermediate step prior to obtaining the combined representation (described in Section 4.4.2 that was musically more sensible and indeed produced better results). In the disjoint representation, the idea was to use two independent Markov chains, one for generating a rhythm sequence and the other for generating an interval/pitch sequence. This approach is similar in spirit to that of [Pai08, CW95].

In this method, a symbolic representation of the underlying rhythm of the melody was generated much in the same way as described in Section 4.3. Alongside this, the symbol sequences of pitches/intervals in the melody were also generated. Each of these sequences was then used to train a different VLMC, one for rhythm, and another for either pitches or intervals. The idea was to first generate a sequence of rhythmic symbols of a certain length. Then those symbols corresponding to onset locations in the generated rhythm symbol sequence were enumerated. Then, a sequence of intervals or notes (depending on the representation in use) of length equal to the number of onset locations in the generated rhythm sequence, is generated. As noted earlier, this method, while producing more variations, is not musically meaningful as the combined representation that is explained next.

4.4.2 Combined Representation

The representation indicated in Table 4.1 is obtained as a cross-product between two sets of values, one corresponding to metrical locations and the other to the presence or absence of onsets at a metrical location. In a similar way, the final representation also relies on such a cross-product representation between the symbols generated for intervals/pitches and those generated for the homogeneous rhythm representation described in Section 4.3. This section describes the representation.

Consider a set of symbols $\mathcal{R} = \{r_1, \ldots, r_R\}$ that represent R homogeneous metrical symbols generated for a set of metrical locations $\mathcal{M} = \{m_1, \ldots, m_M\}$, where $R = 2M$. And let $\mathcal{N} = \{n_1, \ldots, n_N\}$, a set of N elements that represents the pitch clusters, or similarly, interval quantizations. The combined representation essentially involves generating the set $\mathcal{S} = \{s_1, \ldots, s_S\}$ such that $\mathcal{S} = \mathcal{N} \times \mathcal{R}$. This would result in a total of $S = N \cdot R$ symbols. For example, if the metrical length $M = 4$, then $R = 8$ and $\mathcal{R} = \{1, 2, 3, 4, 5, 6, 7, 8\}$. And say the interval quantizations are $\mathcal{Q} = \{--, -, 0, +, ++\}$, then $N = 5$ and $\mathcal{N} = \{1, 2, 3, 4, 5\}$. Therefore, the number of symbols in the combined representation would be $S = 40$.

While this is the general idea of the combined representation, it assumes two different interpretations with the interval and pitch representations respectively (with the rhythm representation being common to both).

Combined Pitch-Metrical Representation

In the case of the combined pitch-metrical representation, the assumption is that a new pitch occurs at a homogeneous time instant where there is an onset and extends until the next onset. Hence, at a certain homogeneous time instant i, given the pitch cluster value p_i and rhythm symbol value r_i, the cross-product symbol value s_i between the two can be given by

$$s_i = (r_i, p_i)$$

	1	2	3	4	5	6	7	8
1	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
2	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
3	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**
4	**25**	**26**	**27**	**28**	**29**	**30**	**31**	**32**

Table 4.2: A depiction of the combined representation for pitch and rhythm when the assumed meter length $M = 4$ (columns) and number of pitch clusters is $N = 4$ (rows).

For example, say at a certain clustering level, there exist 4 pitch clusters ($N = 4$). Given a meter length $M = 4$ (which means $R = 8$), Table 4.2 gives the matrix of symbols.

Unlike the case of intervals in Section 4.4.2, all the symbols in the matrix of Table 4.2 are possible, and hence, the number of symbols S is given by

$$S = N \cdot R$$

where the number of pitch quantizations is N and meter length is M ($R = 2M$).

Combined Interval-Metrical Representation

In the case of the combined interval-metrical representation, it is assumed that an interval can occur only at a homogeneous time instant (explained in Section 4.3) that immediately precedes that of an onset. In every other case (where an onset does not occur), a zero-interval is assumed. That is, at a homogeneous time instant i, the value n_i of the interval is given by

$$n_i = \begin{cases} k \in \{1, \ldots, N\} & \text{if onset at time } (i+1) \\ 0 & \text{if rest at time } (i+1) \end{cases}$$

Hence, if r_i and n_i are the homogeneous rhythm symbol and quantized interval values at homogeneous time i respectively, the cross-product symbol s_i of these two is given by the ordered pair

$$s_i = (r_i, n_{i-1})$$

As an example, consider the case where the meter length $M = 4$ (hence $R = 8$), and the quantized interval representation $\mathcal{Q} = \{-, 0, +\}$ (which means $N = 3$). In

$\mathcal{N}/\mathcal{R}$	1	2	3	4	5	6	7	8
1 (−)	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
2 (0)	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
3 (+)	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**

Table 4.3: An illustration of the combined-representation for intervals and meter when the assumed meter length (columns) is $M = 4$ (Table 4.1) and interval quantization (rows) is $\{-, 0, +\}$.

this case, firstly, the homogeneous cross-product representation for the rhythm is given by Table 4.1. Next, a cross-product between these and the interval quantization symbols is taken according to Table 4.3

More generally, at homogeneous time i, the cross-product symbol value c_i (corresponding to the ordered pair s_i above) for the metrical symbol r_i, the pitch cluster (or interval quantization) symbol n_i, given a meter length M, is given by

$$c_i = (i_{i-1} - 1) \cdot R + r_i$$

It is to be noted here that not all symbols in the final cross-product occur in any given sequence. This is because, we do not assume the presence of a non-zero interval (+, −−, etc.) unless there is an onset. Hence, those symbols that correspond to a non-zero interval and a rest (absence of an onset) will never occur in the sequence. For example, in Table 4.3, the symbol values 1, 2, 3, 4, 17, 18, 19 and 20 will never occur in a sequence. Hence, given N interval quantizations and a meter length of M, the maximum number of symbols S' that can occur in a given sequence is not $N \cdot R$, but can be given by

$$S' = (N + 1) \cdot M$$

This completes the description of the symbolic representation. In summary, firstly, a metrical symbol is obtained as a cross-product between the metrical location (of a time-homogeneous representation with a given meter length) and the type of onset (onset or rest). Following this, the final melody symbol is obtained as a cross-product of the metrical symbol and interval quantization/pitch cluster symbol. This process is repeated at multiple levels of meter length, interval quantizations and pitch clustering. Symbol sequences thus generated from the original pitch-value sequence are used for training the VLMC that is then used for generation.

Chapter 5

Generation

"Music is composed, to an important degree, of patterns that are repeated and transformed. Patterns occur in all of music's constituent elements, including melody, rhythm, harmony, and texture. "

-R. Rowe

One may define musical style in several ways. There are often external, non-musical factors that come into play (theme, musical era, cultural factors, etc.) while regarding a piece of music as belonging to a certain style. The analysis employed here relies only on the application of statistics and information theory to the segmented monophonic melody in order to identify the style. Here, style can be seen as the prevalence or the occurrence of certain characteristic melodic patterns along with rhythmic ones that make the melody sound the way it does. The Markov chain, from its definition and prior usage in other domains such as text and speech, serves as a suitable statistical model for modelling temporally changing (musical) patterns.

Here, one must return to the interpretation of music as a stochastic process that was introduced earlier in Chapter 1. Modelling the style of music can be translated into the problem of identifying the mechanism of the stochastic process, or more generally, the information source generating the music. As a continuation of what has been introduced in Chapter 1, this chapter provides a theoretical background of stochastic processes, Markov chains and their application to this work. The following description closely follows that of [Mar10].

5.1 Stochastic Process

Let $(\Omega, \mathcal{F}, P)$ be a probability space and $\mathcal{X}$ a finite alphabet (a set whose elements are called symbols). A *stochastic process* is a series $(X_n)_{n\geq 0}$ of random variables $X_n : \Omega \to \mathcal{X}\ \forall n \in \mathbb{N}$. Moreover, the process is *stationary* if $P(X_1 = a_1, \ldots, X_n = a_n) = P(X_{1+k} = a_1, \ldots, X_{n+k} = a_n)\ \forall a_1, \ldots, a_n \in \mathcal{X}, \forall k, n \in \mathbb{N}$.

A particular case of a stochastic process is a *Markov chain*. In the context of this work, the stochastic processes is the Markov chain.

5.2 Markov Chains

A Markov chain is a "chance process" [GS97] in which the predictions for future experiments are influenced by the outcomes of previous ones. The relation between two consecutive experiments is referred to as a *transition*, and the nature of these transitions is determined by what is known as the *transition nucleus* of the particular Markov chain. Hence, to explain a Markov chain the concept of the transition nucleus is to be introduced first.

5.2.1 Transition Nucleus

Let $(\Omega, \mathcal{F})$ be a measurable space and $\mathcal{X}$ a finite set. We define a *discrete transition nucleus* in $\mathcal{X}$ any real function

$$N : \mathcal{X} \times \mathcal{X} \to [0, \infty]$$

such that:

$$\sum_{y\in\mathcal{X}} P(x, y) = 1 \quad \forall x \in \mathcal{X} \tag{5.1}$$

If $\mathcal{X}$ is a finite set with n elements ($|\mathcal{X}| = n$) $\mathcal{X} = \{x_1, \ldots, x_n\}$, a discrete transition nucleus N on the set $\mathcal{X}$ is completely determined by the real $n \times n$ matrix $(a_{i,j})_{i,j=1,\ldots,n}$ where the generic element $a_{i,j}$ is equal to $N(x_i, x_j)$. Such a matrix is called *transition matrix* for the nucleus N and, vice versa, an $n\times n$ matrix such that the sum of the elements on each row is 1 uniquely determines a transition nucleus.

In theory, it can be said that every single past outcome until the current experiment plays a role in determining that of the current one. In practice, however, it is assumed that the influence is limited to only O previous outcomes. The value O is known as the *order*, or *length* of the Markov chain. The *homogeneous Markov chain* is a specific case of the Markov chains that will first be introduced before going on to the variable-order Markov chains used in the present work.

5.3 Homogeneous Markov Chains

Let $(\Omega, \mathcal{F}, P)$ be a probability space equipped with a *filtration*, i.e. a non decreasing sequence $\mathcal{F}_k$ of σ-algebras[1] in $\mathcal{F}$.

Definition: Let $(X_k)_{k\geq 0}$ a series of random variable with values in $\mathcal{X}$ (with the measure of the discrete space). The series is said to be a *homogeneous Markov chain* of nucleus N if it is adapted to the filtration (i.e. for all $k \geq 0$ the random variable X_k is $\mathcal{F}_k$-measurable) and moreover the following equation holds:

$$P(A, X_k = a, X_{k+1} = b) = P(A, X_k = a)N(a, b) \tag{5.2}$$

for all $k \geq 0$ and for all the possible choices for the event A in $\mathcal{F}_k$ and for all $a, b \in \mathcal{X}$.

If B is a subset of $\mathcal{X}$, then by posing

$$N(a, B) := \sum_{b\in B} N(a, b) \quad (\leq 1 \text{ for } 5.1) \tag{5.3}$$

we have

$$P(A, X_k = a, X_{k+1} \in B) = P(A, X_k = a)P(a, B). \tag{5.4}$$

Where the latter is the more general formulation of equation 5.2. In the case of the event $\{A, X_k = a\}$ having a non-null probability, Equation 5.4 can be re-written in terms of conditional probability as

$$P(X_{k+1} \in B|A, X_k = a) = P(a, B). \tag{5.5}$$

The relation 5.5 states that the position of the system at time $(k+1)$ conditioned on an event of type $\{A, X_k = a\}$, where A is in the past of k, it is given by $P(a, \cdot)$ and does not depend on the choice of A. To summarize, once the actual position is

[1]Intuitively, the elements of $\mathcal{F}_k$ represent the events of the past history up until the time k included. In other words, by knowing the history of the system up to time n included, one can claim with certainty if the generic event $A \in \mathcal{F}_k$ occurred or not.

known, any other information on the past history does not change the distribution for the next state.

More generally, it is possible to define *non-homogeneous* Markov chains where the nucleus N in 5.2 depends on k, but for sake of simplicity, these will also be referred to in terms of homogeneous Markov chains. Moreover, it is possible to define Markov chains of order m where the distribution over the next state only depends on the last m positions. It is, however, possible to see a m-order Markov chain as a first order Markov chain built on an alphabet made of vectors of length m with components in $\mathcal{X}$. For example, a second order Markov chain can be translated into a first order one by substituting the series:

$$\ldots a_0, a_1, a_2, \ldots, a_k, \ldots$$

with the following

$$\ldots a_{-1}a_0, a_0a_1, a_1a_2, \ldots, a_{k-1}a_k, \ldots$$

and the new nucleus is defined by:

$$N(x_1x_2, y_1y_2) = \begin{cases} 0 & \text{if } x_2 \neq y_1 \\ P(X_2 = y_2 | X_0 = x_1, X_1 = x_2) & \text{otherwise} \end{cases}$$

5.4 Variable Length Markov Chains

Let $\mathcal{X}$ be the usual finite alphabet, a *set of strings* over the alphabet $\mathcal{X}$ is defined as the set

$$\mathcal{X}^* := \bigcup_{k \geq 0} \mathcal{X}^k$$

whose elements are called *strings*. Note that by convention $\mathcal{X}^0 = \{\mathbf{e}\}$ where $\mathbf{e}$ is the void string.

Let $(X_t)_{t\in\mathbb{Z}}$ be a time-homogeneous Markov chain of order m with values in $\mathcal{X}$. The symbol x_i^j represents the following string (written in reversed order)

$$(x_j, x_{j-1}, \ldots, x_i)$$

where $i < j, i, j \in \mathbb{Z} \cup \{-\infty, +\infty\}$ and where $wu = (w_{|w|}, \ldots, w_2, w_1, u_{|u|}, \ldots, u_2, u_1)$ is the concatenation of the strings w and u.

From these introduced conventions,

$$\mathbb{P}[X_1 = x_1 | X_{-\infty}^0 = x_{-\infty}^0] = \mathbb{P}[X_1 = x_1 | X_{-m+1}^0 = x_{-m+1}^0], \text{ for all } x_{-\infty}^1. \qquad (5.6)$$

The concept of the *variable-length Markov chain* (VLMC) can be seen as re-grouping of several states x_{-m+1}^0 in 5.6.

5.4.1 Context Function

Let $(X_t)_{t\in\mathbb{Z}}$ be a stationary process with $X_t \in \mathcal{X}$, $|\mathcal{X}| < \infty$. Moreover, let $c : \mathcal{X}^\infty \to \mathcal{X}^\infty$ be a function such that:

$$\begin{aligned} &c : x_{-\infty}^0 \mapsto x_{-\ell+1}^0, \text{where } \ell \text{ is defined by} \\ &\ell = \ell(x_{-\infty}^0) = \min\{m; \mathbb{P}[X_1 = x_1 | X_{-\infty}^0 = x_{-\infty}^0] = \mathbb{P}[X_1 = x_1 | X_{-m+1}^0 = x_{-m+1}^0] \\ &\qquad\qquad \text{for all } x_1 \in \mathcal{X}\}, \\ &\text{and the case } \ell = 0 \text{ holds when there the next state is independent from the context.} \end{aligned} \tag{5.7}$$

Then, $c(\cdot)$ is called *context function* and for all $t \in \mathbb{Z}$, $c(x_{-\infty}^{t-1})$ is called *context* for the variable x_t. The context is precisely the part of the past that influences the current value.

Let $(X_t)_{t\in\mathbb{Z}}$ a stationary process where $X_t \in \mathcal{X}$, $|\mathcal{X}| < \infty$ with corresponding context function $c(\cdot)$ as before. Let $0 \le m \le \infty$ be the smallest integer such that

$$|c(x_{-\infty}^0)| = \ell(x_{-\infty}^0) \le m \text{ for all } x_{-\infty}^0 \in \mathcal{X}.$$

Then $c(\cdot)$ is called context function of order m, and if $m < \infty$,the process $(X_t)_{t\in\mathbb{Z}}$ is called *variable-length Markov chain* (VLMC) of order m.

The VLMC $(X_t)_{t\in\mathbb{Z}}$ will henceforth be referred to with its law P_c (in the space $\mathcal{X}^{\mathbb{Z}}$). Moreover, if P is a probability measure over $\mathcal{X}^{\mathbb{Z}}$, the following notations will be used,

$$\begin{aligned} P(x) &:= \mathbb{P}_P[X_1^m = x] \quad (\forall x \in \mathcal{X}^m) \\ P(x|w) &:= \frac{P(xw)}{P(w)}. \end{aligned}$$

5.5 Tree representation of VLMCs

Because of the homogeneity a VLMC P_c is completely determined by the following transition probabilities:

$$\mathbb{P}_{P_c}[X_1 = x_1 | X_{-\infty}^0 = x_{-\infty}^0] = p(x_1 | c(x_{-\infty}^0)), x_{-\infty}^1 \in \mathcal{X}^\infty.$$

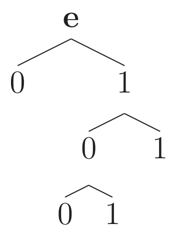


Figure 5.1: Context tree of the function $c(\cdot)$ in the example.

The states determining such transition probabilities are given by the image of the context function $c(\cdot)$. It is convenient to represent those states in the form of a tree.

The trees here contain a root node on the top with branches growing from top to bottom. From each node a maximum number of $|\mathcal{X}|$ branches are growing. The tree is built in such a way that each value of the function $c(\cdot) : \mathcal{X}^\infty \to \bigcup_{k=0}^{n} \mathcal{X}^k$ is represented by a terminal node. The context $w = c(x_{-\infty}^0)$ can, in fact, be found in the tree starting from the root node and following branch given by x_0, then the sub-branch given by x_{-1} and so on until the sub-branch determined by $x_{-\ell(x_{-\infty}^0)+1}$ is reached. The resulting tree is not necessarily complete i.e. the nodes of the tree might have any number of branches equal or less then $|\mathcal{X}|$.

The following context function:

$$c(x_{-\infty}^0) = \begin{cases} 0, & \text{if } x_0 = 0, \text{ and for any } x_{-\infty}^{-1} \\ 1,0,0, & \text{if } x_0 = 1, x_{-1} = 0, x_{-2} = 0, \text{ and for any } x_{-\infty}^{-3} \\ 1,0,1, & \text{if } x_0 = 1, x_{-1} = 0, x_{-2} = 1, \text{ and for any } x_{-\infty}^{-3} \\ 1,1, & \text{if } x_0 = 1, x_{-1} = 1, \text{ and for any } x_{-\infty}^{-2} \end{cases}$$

is represented by the tree of Fig. 5.5.

A formalization of the above is presented here.

Let $c(\cdot)$ be the context function of a VLMC stationary. The contextual tree τ is the set:

$$\tau = \tau_c = \{w; w = c(x_{-\infty}^0), x_{-\infty}^0 \in \mathcal{X}^\infty\} = c(\mathcal{X}^\infty),$$

and its elements are called *nodes*. The "topology" of such a tree (the interconnection between the nodes in τ) is given by the following rule in $\omega, \tilde{\omega} \in \tau$:

$$\tilde{\omega} \text{ is child of } \omega \iff \exists \sigma \in \mathcal{X} : \omega\sigma = \tilde{\omega}.$$

This rule is recursively applied to generate the nodes of the tree, following which all the possible missing prefixes of the strings in τ are added including the void prefix

e. Thus, a tree with root node **e** where the nodes of level i are strings in τ with i symbols is obtained. For the context function c of example 5.5, the case is as follows

$$\tau_c = c(\{0,1\}^\infty) = \{0, 100, 101, 11\}.$$

Addition of all the prefixes gives

$$\tilde{\tau} = \{\mathbf{e}, 0, 1, 10, 11, 100, 101\}$$

where, with rule 5.5 it implies that 0 and 1 are sons of **e**, 10 and 11 are sons of 1, 100 and 101 are sons of 10. Thus, the tree of Figure 5.5 is constructed.

5.6 Application to the system

Having introduced the theory behind the variable-length Markov chains (VLMCs) and their tree-based representation, their application in the statistical analysis of the symbol sequence derived in Chapter 4 will be presented in this section. This involves the induction of a VLMC from the obtained sequence of symbols.

In [BW99, RST96], a general method for inferencing long sequences is described. The simplified implementation as described in [Pac03] is used here for faster computation. As the intent is to experiment with multiple context-lengths, a tree is constructed with the maximum expected context-length (or 32 symbols). Each node of the tree represents a specific context that had occurred in the past. In addition, as explained in [Pac03], each node carries a list of continuation indices corresponding to block indices matching the *context*. This process is repeated for all the different levels obtained after the cross-product (Section 4.4.2)

For audio, a different approach has been applied in [DAC07]. This method does not require an event-wise symbolic representation since it employs the factor oracle algorithm. The VLMC has not been applied to audio prior to the work of [MP10] due the absence of an event-wise symbolic representation. Chapter 4 presents a method to obtain such a representation, similar in spirit to that in [MP10]. Given a symbolic representation, the statistical analysis of symbol sequences is explained here with the aid of a simple example.

Assume that the tree of Figure 5.2 has been constructed from the symbol sequence of one level. A continuation of the sequence (A, B) is to be generated. Starting at the root node, the tree is traversed considering the sequence of the given context in reverse order (from the most recent state to the first). The following continuation indices are available

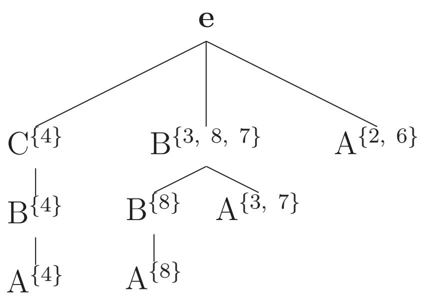


Figure 5.2: Context tree built from the analysis of the sequences (A, B, C, D) and (A, B, B, C).

$$\text{Continuation\_List((A, B))} = \{3, 7\}.$$

Those indices correspond to the symbols (C, B) respectively. To generate the continuation, one of these is chosen at random. Suppose the choice is B. Then the new sequence is (A, B, B) which has the possible continuations

$$\text{Continuation\_List((A, B, B))} = \{8\}.$$

At this point, there exists only one choice (index 8) corresponding to C. Thus, there is now the sequence (A, B, B, C). Going a step further, we have to continue using the longest context which is (B, C). We can only choose the index 4 as continuation. So we have obtained the sequence (A, B, B, C, D) that is a new sequence that has never occurred in the examples provided. However, this does not have a continuation either. When this happens, a simple and obvious solution is to use a blank context **e** which means choosing randomly between all of the indices. There is, thus, a "discontinuity" in the generation. A more elegant solution would be to exploit the multi-level representation that is available. A generation strategy is outlined which determines the choice to be made in case of such and exception.

5.7 Generation Strategy

If a particular level is fixed, the continuation indices are drawn according to a posterior probability distribution determined by the longest context found. But the question arises as to which level has to be chosen. Depending on the sequence, it could be better to predict based either on a coarse or a fine level. But there is not rule as to what should be preferred. A trade-off exists between the level and the number of choices that it makes available. Selecting a lower level at which a context of at least $\hat{l}$ exists (for a predetermined fixed $\hat{l}$, usually $\hat{l}$ equal 6 or 8). This works quite well for many examples. But in some cases a context of that length does not exist and the system often reaches the higher level where too many symbols are provided inducing too random generations. On the other hand, it occurs very often that a lower level is made of singleton clusters that have only one instance. In this case, a long context is found in the lower level but since a particular symbol sequence only occurs once in the whole original segment the system replicates the audio in the same order as the original. This behaviour often leads to the exact reproduction of the original until reaching its end and then a jump at random to another block in the original sequence.

In order to increase recombination of symbols and still provide good continuation, some heuristics are employed taking into account the multiple levels available for the prediction. A recombination value p, in the range $[0, 1]$ is also set. The following heuristics are used to generate the continuation in each step:

- Set a maximal context length $\hat{l}$ and compute the list of indices for each level using the appropriate suffix tree. Store the achieved length of the context for each level.
- Count the number of indices provided by each level. Select only the levels that provide less than 75% the total number of symbols.
- Among these level candidates, select only the ones that have the longest context.
- Merge all the continuation indices across the selected levels and remove the trivial continuation (the next onset).
- In case there is no level providing such a context and the current block is not the last, use the next block as a continuation.
- Otherwise, decide randomly with probability p whether to select the next block or rather to generate the actual continuation by selecting randomly between the merged indices.

5.8 Interpretation

The theory covered so far in this chapter explains the general idea of the VLMC with some examples on how it can be applied in the context of generating new symbol sequences from given examples. In the present work, there are two representations, namely, the pitch and the interval representations that are obtained from the audio data for melody generation. Although the basis for generation in both these representations is the same, which has been explained in this chapter so far, the present section provides a brief note on the difference in their interpretations.

In the case where the VLMC framework is used to predict indices corresponding to the combined pitch-metrical representation (Section 4.4.2), each predicted index corresponds to a unique audio segment (or pitch) in the input melody. Hence, in this case, the choice of the pitch is obvious given the index predicted by the VLMC.

However, in the case of the combined interval-metrical representation (Section 4.4.2), it is not so straightforward. In this representation, every predicted index corresponds to a quantized interval (such as $++$, $-$ or 0). This does not directly correspond to an audio segment. Hence, while using the VLMC predictions with this second representation, it would be necessary to have an additional stage of selecting the audio segment, given the pitch of the previous time-step and the current quantized interval. In such a situation, there will be several audio segments that satisfy this condition. One may apply additional constraints such as those based on harmony, tessitura, etc. in order to narrow down the list of possible choices before selecting one.

Chapter 6

Evaluation

"In the final analysis, randomness, like beauty, is in the eye of the beholder."

-R. W. Hamming

6.1 Database

The present work uses monophonic audio of guitar and bass-guitar melodies for the analysis of patterns in them and in turn exploits this information to generate variations of these melodies. The use of audio data is mainly motivated from the fact that a majority of related approaches in the past have skipped the stage of segmenting musical information from audio and have focused specifically on labelled symbolic representations such as MIDI. Recent approaches (among the few) that deal directly with audio information in the context of music analysis for generation are the *Audio Oracle* [DAC07], the works of Jehan [Jeh05] and Marchini [MP10]. Moreover, it was thought that an assessment of how state-of-the-art segmentation methods and those for analysis, which are often isolated from the process of segmentation, work in conjunction would be beneficial. It would also be interesting to see in what way these ideal-case analysis methods would have to be adapted/modified in order to handle segmented audio data. Although this thesis focuses on only guitar melodies, it is to be noted that the general method may be extended to other instruments such as violin, piano, flute, etc. with appropriate modifications to parameters of the segmentation algorithms used. Similar results may be expected in those cases as well.

Song	**Artist**	**Type**	**Source**	**Duration**
More than a Feeling (1)	Boston	Riff	Multi-track	0:23
More than a Feeling (2)	Boston	Riff	Multi-track	0:21
Truckin' (1)	The Grateful Dead	Solo	Multi-track	0:23
Truckin' (2)	The Grateful Dead	Riff	Multi-track	0:19
Calling Dr. Love (1)	Kiss	Riff	Multi-track	0:16
Calling Dr. Love (2)	Kiss	Riff	Multi-track	0:07
Amazing Journey	The Who	Solo	Multi-track	0:47
Sweet Child of Mine	Guns N' Roses	Riff	Recorded	0:15
Another Brick in the Wall	Pink Floyd	Solo	Recorded	0:45
Hysteria	Muse	Riff	Multi-track	0:20

Table 6.1: Database of melodies used for analysis.

The analysed database consists of a variety of pop/rock songs. These have either been recorded earlier or obtained as multi-track recordings such that various instruments in the recording are available as separate audio files. The main focus was on bass-lines with some importance given to guitar/bass solos. Bass-lines usually had a riff-like structure in which the same melodic segment repeats over a short period of time (typically 2s to 5s) with minor (if not any) variations in each repetition. Typically, each riff-type track contains around 4 repetitions of the riff. And solos are long melodic phrases that don't necessarily have an overall repetitive structure like riffs. They are typically between 20s and 50s in length. Those melodies that employ effects such as delay, distortion, reverb, etc. along with the instruments were avoided for ease of segmentation. Table 6.1 lists the melodies that were used for analysis and generation. The evaluation considered a subset of six of the ten melodies in Table 6.1. These are listed in Table 6.2.

Song - Artist	**Instrument (type)**	**Accompaniment**
More than a Feeling (1) - Boston	Bass (Riff)	Yes
Truckin' (1) - The Grateful Dead	Bass (Solo)	Yes
Truckin' (2) - The Grateful Dead	Bass (Riff)	Yes
Calling Dr. Love (1) - Kiss	Bass (Riff)	Yes
Sweet Child of Mine - Guns N' Roses	Guitar (Riff)	No
Another Brick in the Wall - Pink Floyd	Guitar (Solo)	No

Table 6.2: List of melodies used for evaluation.

6.2 Onset Detection

This section presents a summary of the evaluation of onset detection performed on the six selected examples. The *Aubio* implementation of the *complex domain* method for onset detection [DBDS03] was employed here. A detailed list of parameters used can be found in Table 3.1. The six melodies were of varying length and contained a total of 331 onsets.

A precision-recall based method was used to determine the performance of onset detection. This is a commonly employed performance measure in information retrieval tasks to measures how accurate, and at the same time, how exhaustive a certain retrieval operation was. In this context, the data to be retrieved is the set of onsets of the original melody. Precision is defined as

$$precision = \frac{|\mathcal{L} \cap \mathcal{T}|}{|\mathcal{T}|}$$

and recall is defined as

$$recall = \frac{|\mathcal{L} \cap \mathcal{T}|}{|\mathcal{L}|}$$

Where, $\mathcal{L}$ is the set of ground-truth onsets and $\mathcal{T}$ is the set of retrieved onsets. Simply speaking, a high recall would mean that nothing has been missed but there may be a lot of useless results to sift through (which would imply low precision). High precision means that everything returned was a relevant result, but all the relevant items may not have been found (which would imply low recall).

Using these two, a third measure, known as the f-Measure(f_M), is defined as the harmonic mean of the precision and recall.

$$f_M = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

Table 6.3 shows the evaluation of onset detection according to the described measures, with an overall precision of 94.58%, recall of 86.42% and f-measure of 89.11%.This, although not perfect, could be considered satisfactory in the present case. A discussion of the effects of missed and wrongly detected onsets on the analysis that followed is presenting in Chapter 7.

6.3 Expert Evaluation

A subjective evaluation, based on the opinion of experts, was chosen to assess the musical quality of the melodies generated by the current system. The qualitative

Track No.	**Precision**	**Recall**	**f-Measure**	**Onset Count**
1	1.0000	0.9375	0.9677	60
2	1.0000	0.9117	0.9538	31
3	0.9629	0.8387	0.8965	54
4	0.8923	0.9206	0.9062	65
5	0.8198	0.7280	0.7711	111
6	1.0000	0.7407	0.8510	20
Overall	0.9458	0.8642	0.8911	331

Table 6.3: Table illustrating the performance of the onset detection on the evaluated examples.

assessment of some specific aspects of it was deemed ideal for evaluation here. A quick summary of what is discussed in more detail in Chapter 2 in regards to the preference for the present evaluation method is presented here.

An evaluation based purely on “accuracy”, as in the case of [Pai08], was deemed unsuitable in the context of the current work as variations in the original melody are a necessary condition here. A criterion that favours similarity with the original melody would have an adverse effect on the performance in the presence of variations, and hence be against the point of having a system that aims at generating melodic variations. Another option is a classifier-based evaluation, as that employed by [All02, DAC07]. It involves using the generation system as a classifier and measuring the accuracy of its classification of unseen data of different styles. However, in the present case, due to the unavailability of sufficient amount and variety of audio data required for conducting such an evaluation, it is avoided. The third option is a Turing test that determines to what extent a listener can be deceived into believing that a melody actually generated by the system is that played by a human (or vice versa). Such an evaluation method was used as a part of the evaluation in [Pac03]. Due to unresolved issues related to synthesis artefacts occurring in the output of the present system, the choice for the listener would become obvious and hence this method is also avoided.

In the present work, it is required that certain specific aspects of the system, such as metrical coherence, equivalence of tempo, occurrence of repeating patterns, regularity of riff structure, etc. be evaluated. It was thought that such specific aspects can be focused on in the evaluation through a questionnaire. Experts - individuals with an educational background in music or extensive music performance experience - were considered to be ideal candidates for providing feedback on the quality of the generated melodies. Four experts were consulted for their feedback on the output of the system. Each of them is either a professional/session musician or holds a diploma in classical music or both.

A questionnaire (Figure 6.1) was prepared and presented to each of the experts as a part of an evaluation package. The evaluation package contained six folders, each containing a copy of the questionnaire, an audio file of the original melody and another of a generation corresponding to the original. This number was intentionally kept small in order not to lose the attention of the listener over the course of the evaluation. Two recorded melodies and four multi-track melodies were used in the evaluation. These are listed in Table 6.2. In the case of multi-track melodies, both the original and the generation, which in all cases were bass-lines, were overlayed with the guitar and drum tracks as in the original song. In each case, it was specified in the instructions that the bass-line was to be focused on. This was not possible with the recorded tracks and they were played in isolation. The experts were first asked to listen to the original track any number of times until they developed a fair idea of the melody. Following this, they were asked to listen to the generated melody and answer the questionnaire.

6.3.1 Questionnaire

The questionnaire was prepared keeping in mind the different musical facets of the melody that were handled in the method. It was considered to be of prime importance that, in the least, one can observe note sequences (patterns) from the original melody from time to time in the generation. This is a distinctive characteristic of Markov chains as compared to other statistical models. In contrast, a random sampling method (which was also examined at an earlier stage of the work), while maintaining the overall distribution of notes does not ensure that they occur in order as in the original melody.

Once it has been confirmed that note segments from the original melody do occur in the generation, the next question is at what locations in the melody do they occur. As explained in Chapter 4, the metrical locations of different notes are also included in the cross-product that generates the final symbolic representation of note sequences. If this formulation works, it would imply that notes chosen for the generation also occur in the correct metrical location. While the statement of question 2 in Figure 6.1 applied to the guitar/bass solo case, it was modified in the case of a riff as follows

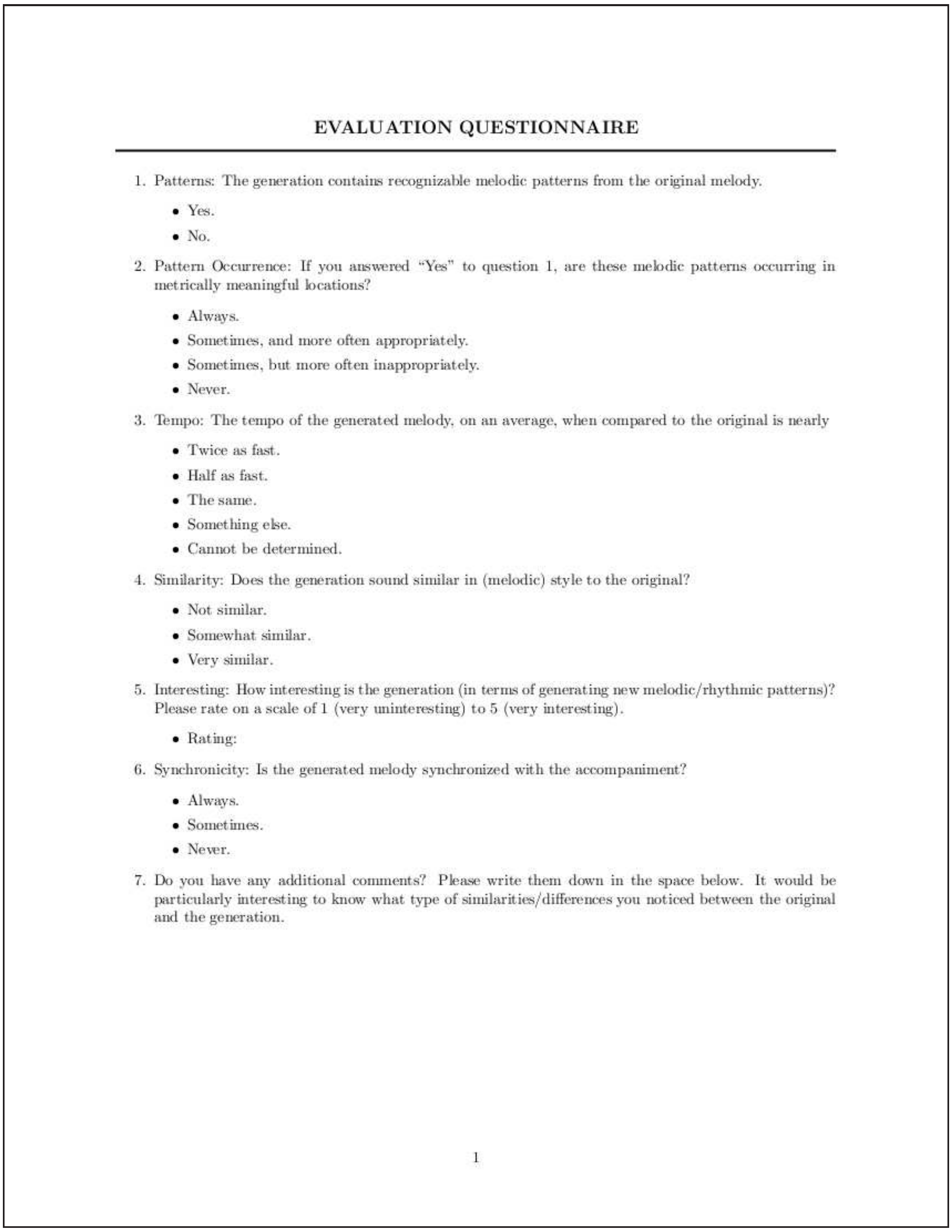
"If you answered "Yes" to question 1, are these melodic patterns occurring in such a way that a riff structure is evident?"

It is also important that the generation be of, more or less, the same tempo as the original melody. This question was considered necessary due to the method employed for metrical analysis. As explained in Section 4.3, the metrical level of generation of the new melody is the same level at which at least 90% of the onsets of the original melody are matched with beats. In this way, the tempo of generation is typically twice or thrice the actual tempo of the original melody. It was considered necessary to verify that, firstly, a stable tempo was observable in the generation, and secondly, this tempo was, more or less, the same as that of the original.

Questions 4, 5 and 8 of the questionnaire go hand-in-hand. While the question of how similar or interesting a generation seems is highly subjective and depends to the great extent on the listener in question, it was found that the last question that sought comments from her/him often had a response that helped understand the listener's understanding of similarity and the "interestingness" rating. In general, it is hoped that the expert provides a high rating for the interestingness of the generations and considers them to be "somewhat similar" or "very similar" to the original. At times, a similarity rating of "very similar" could mean that there aren't sufficient variations in the generation as compared to the original. Question 8, once again, helps clarify this opinion.

The question about synchrony mainly refers to the synthesis aspect of the generation. It is often the case that the chosen note and the duration assigned to it in the generation are not equal. In such a case, the time-stretch library *Rubber Band* [Rub11] was used to stretch/shrink the note to the appropriate duration. It was observed that, on overlapping the generated melody with the tracks of other instruments accompanying the original track, there were some occasions where the synchrony between them is lost.

It was hoped that feedback from the experts based on this questionnaire would help assess the musical quality of the generated melodies, and, at the same time also provide some valuable insight into errors and room for possible improvements. Table 6.4 summarizes the results of the expert evaluation which are analysed in detail in Chapter 7.



EVALUATION QUESTIONNAIRE

1. Patterns: The generation contains recognizable melodic patterns from the original melody.
   - Yes.
   - No.
2. Pattern Occurrence: If you answered "Yes" to question 1, are these melodic patterns occurring in metrically meaningful locations?
   - Always.
   - Sometimes, and more often appropriately.
   - Sometimes, but more often inappropriately.
   - Never.
3. Tempo: The tempo of the generated melody, on an average, when compared to the original is nearly
   - Twice as fast.
   - Half as fast.
   - The same.
   - Something else.
   - Cannot be determined.
4. Similarity: Does the generation sound similar in (melodic) style to the original?
   - Not similar.
   - Somewhat similar.
   - Very similar.
5. Interesting: How interesting is the generation (in terms of generating new melodic/rhythmic patterns)? Please rate on a scale of 1 (very uninteresting) to 5 (very interesting).
   - Rating:
6. Synchronicity: Is the generated melody synchronized with the accompaniment?
   - Always.
   - Sometimes.
   - Never.
7. Do you have any additional comments? Please write them down in the space below. It would be particularly interesting to know what type of similarities/differences you noticed between the original and the generation.

1

Figure 6.1: The questionnaire presented to the experts for their feedback. This questionnaire was to be answered for each of the original/generated melody pair.

	Patterns	**Patt. Occurrence**	**Tempo**	**Similarity**	**Interesting**	**Synchrony**
Melody 1 (*Sweet Child of Mine* guitar riff)						
Expert 1	Yes	Appropriately	Same	Somewhat similar	3	N/A
Expert 2	Yes	Inappropriately	Same	Very similar	3	N/A
Expert 3	Yes	Appropriately	Same	Very similar	4	N/A
Expert 4	Yes	Inappropriately	Same	Very similar	2	N/A
Melody 2 (*More Than a Feeling* bass riff)						
Expert 1	Yes	Always	Same	Very similar	2	Sometimes
Expert 2	Yes	Appropriately	Same	Very similar	3	Sometimes
Expert 3	Yes	Appropriately	Sth. else	Very similar	3	Sometimes
Expert 4	Yes	Always	Cannot be det.	Very similar	3	Sometimes
Melody 3 (*Truckin'* bass riff)						
Expert 1	Yes	Appropriately	Sth. else	Very similar	4	Sometimes
Expert 2	Yes	Appropriately	Same	Somewhat similar	5	Sometimes
Expert 3	Yes	Appropriately	Same	Very similar	5	Sometimes
Expert 4	Yes	Always	Same	Very similar	5	Always
Melody 4 (*Another Brick in the Wall* guitar solo)						
Expert 1	Yes	Inappropriately	Same	Somewhat similar	1	N/A
Expert 2	Yes	Appropriately	Same	Very similar	3	N/A
Expert 3	Yes	Inappropriately	Cannot be det.	Very similar	3	N/A
Expert 4	Yes	Appropriately	Same	Very similar	4	N/A
Melody 5 (*Truckin'* bass solo)						
Expert 1	Yes	Inappropriately	Sth. else	Somewhat similar	1 (or) 5	Never
Expert 2	Yes	Inappropriately	Cannot be det.	Somewhat similar	2	Sometimes
Expert 3	Yes	Inappropriately	Same	Somewhat similar	3	Sometimes
Expert 4	Yes	Inappropriately	Sth. else	Very similar	1	Sometimes
Melody 6 (*Calling Dr. Love* bass riff)						
Expert 1	Yes	Always	Same	Very similar	1	Sometimes
Expert 2	Yes	Inappropriately	Cannot be det.	Very similar	2	Sometimes
Expert 3	Yes	Always	Same	Very similar	4	Always
Expert 4	Yes	Inappropriately	Same	Somewhat similar	3	Sometimes

Table 6.4: Expert Evaluation Results

Chapter 7

Discussion

The approach described in this thesis deviates from similar ones in the past [Pai08, Pac03, DAC07] in that while these prior approaches start directly with MIDI input, the present one contains an additional segmentation stage that transforms the input signal into a relevant symbol sequence. The presence of this segmentation stage produced some discrepancies in the expected outcomes at different stages of the symbolization and analysis stages that followed, that also affected the final result. This chapter discusses these discrepancies, how they were handled and relates them to the musical output generated by the system. The impact of each of these on the system varies, but they are covered here nevertheless for academic interest. Some of them will also be used to explain the feedback received from the experts.

7.1 Onset Detection

One of the factors that contributes to the inaccuracy in the generation of a melody is onset detection. As discussed in 6.2, not every note onset in the melody is detected. This is due to the limitations of the onset detection method in use with respect to the instrument at hand. The imperfect onset detection results in, firstly, inaccurate onset locations and secondly, multiple notes being segmented as a single note. The former, along with its consequence (discussed in Section 7.3), is illustrated in Figure 7.2. The latter is illustrated in Figure 7.1. It affects the output by causing, what is expected to be a single note chosen to be used in the generation, to be a sequence of notes. It should be noted here that this creates an effect that, though not ideally wanted, doesn't necessarily affect the output in a negative way. Additionally, it also has an impact on interval representation (see Section 7.2).

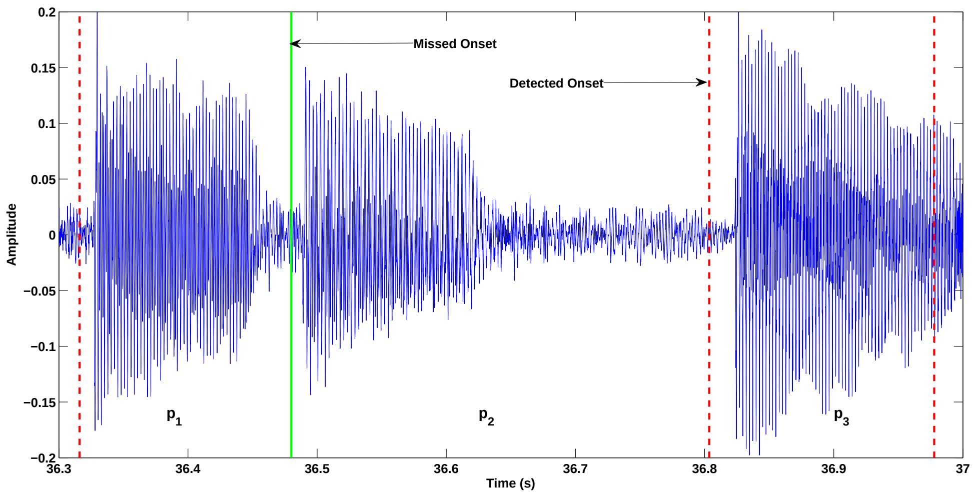


Figure 7.1: Illustration of a case where a missed onset leads to a mistake in the interval sequence. The actual interval sequence here is $i_1 = p_2 - p_1$, $i_2 = p_3 - p_2$. However, when one onset (green) is not detected, the interval sequence becomes $i_1 = p_3 - p_1$.

7.2 Pitch Detection

Another possible place for errors is the pitch detection. While these errors are not very prevalent on carefully setting the algorithm parameters, they do occur at times and are worth noting as they affect the performance of the system. Pitch detection errors are typically of two kinds. The first and most common are octave errors (Section 3.3). As a result of octave errors, certain pitches are wrongly clustered along with others that are nearly an octave apart. And during generation, if these wrongly clustered pitches are selected, they result in an abrupt jump of a very large interval from the previous pitch in the generated melody.

And secondly, there also seemed to be certain semi-tone errors. In such errors, the estimated pitch differs from the actual pitch by a small amount, typically less than or equal to a semi-tone. These errors, however, do not pose a very serious problem to the method as clustering is performed for the symbolization of pitch-values anyway which leads to the grouping of such segments with those nearest to them in pitch.

7.3 Beat Detection & Onset Matching

The beat detection stage was necessary to get the underlying rhythmic structure of the melody. In spite of a few missed onsets, the beat detection algorithm [Dix01] detected the beat structure of the input melody fairly accurately. However, what followed, namely, the process of matching onsets with beats relied heavily on the accuracy of onset detection. This is becuase, the metrical positions at which notes would be placed in the generation relies entirely on those positions to which onsets are matched in the first place. For instance, if an onset that actually occurs on metrical position 1 is detected with some amount of error. It is possible that this onset is matched with a different metrical position, say 2. Such a case is illustrated in Figure 7.2. If this happens, each time the note at this onset is selected in the generation, it would be placed at metrical position 2, instead of 1. In this stage, any onset that did not match a beat at the detected, or even a finer metrical level, was discarded.

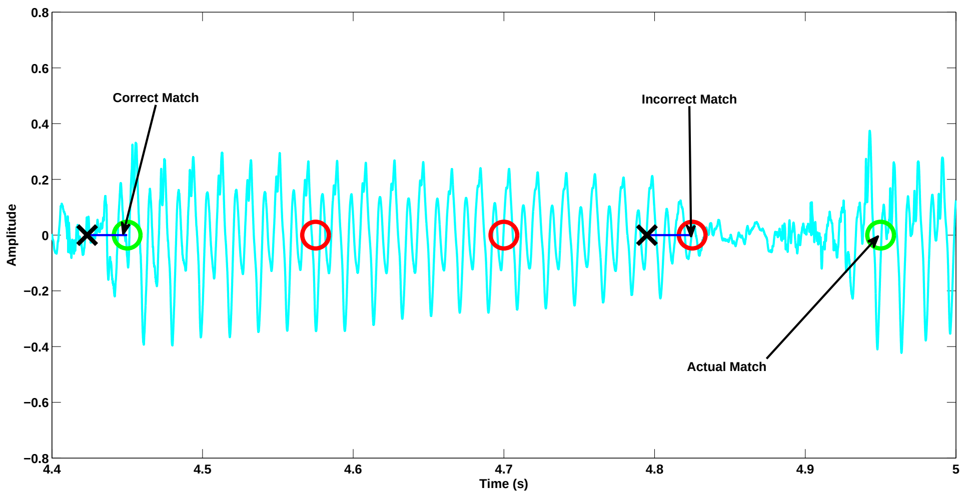


Figure 7.2: Illustration of a case where an onset is matched to the wrong metrical position. Each of the circles indicates the ideal metrical locations. The green ones indicate the metrical location 1. In the example, due to the early detection of the onset, it is matched to the metrical location 4 instead of 1.

Another drawback of this approach for matching onsets and beats is that it cannot detect triplet notes. However, as the detection resolution is successively doubled, at one stage, if a triplet note onset is sufficiently close to a beat, it is matched to it. Triplets that are thus detected appear, not as triplets, but as a sequence of half (quarter, eighth, etc. depending on the tempo) notes in the generation. This was observed in some of the examples.

7.4 Pitch & Interval Representation

The motivation for using the pitch and interval representations as described in Chapter 4 arises from the desire to use a data-driven approach to determine similar sounding melodic events. These events can be pitches or intervals, both of which are considered to be strong indicators of style [GKM03]. The multi-level representation provides a coarse-to-fine grouping of these events and helps obtain symbol sequences of the same. A more detailed review of some of the results of this clustering would be informative and is presented here.

The clustering method applied on the pitches and intervals, being a data driven one, is oblivious to any prior assumptions that can be made on the distribution of notes in the melody. It is hoped that whatever be the nature of the distribution, it would be captured to a fair amount of accuracy by the clustering process.The result of the clustering process mainly varied with the length of the melodies. In this work, the melodies used were typically ranging between 15 seconds and 1 minute.

Table 7.1 illustrates the clusters formed at four levels $\{L_1, \ldots, L_4\}$ chosen by the VRC for an example melody. As one goes from L_1 (coarse) to L_4 (fine), in each successive level, more clusters are formed by the division of those from the previous level. This is a relatively short melody of 24 seconds in length and a similar trend in clustering was observed with other melodies of similar lengths.

The variance ratio criterion aims at minimizing the within-group scatter while maximizing the between-group scatter of a given data distribution (pitch distribution) that is to be clustered. It identifies those levels that best satisfy this criterion. Pitches thus grouped together are those that satisfy the VRC. And as the clustering approach employed is a purely unsupervised one that depends on the data, it was observed that occasionally pitches that do not correspond to the same note are clustered together. Again, this results in a variation when the segments corresponding to these pitches are chosen (although a different pitch is expected). Although this is not ideally desired, in the present generation context, it often works well.

In comparison to the pitch representation, the interval representation (that could not be formally evaluated due to lack of time) seemed to have a more loose structure. This arises from the fact that the continuation suggested by the Markov chain in this case is an interval and not a note. The Interval-Pitch Matrix (see Section 4.2.2),

P_i	L_4	L_3	L_2	L_1
50.38	1	1	1	1
52.61	1	2	2	2
52.69	1	2	2	2
52.73	1	2	2	2
52.97	1	2	2	2
53.29	1	2	2	2
53.47	1	2	2	2
53.70	1	2	2	2
54.63	1	2	2	2
54.93	1	2	2	2
55.23	1	2	2	2
56.36	1	2	2	2
56.48	1	2	2	2
57.18	1	2	2	2
58.88	1	2	3	3
59.01	1	2	3	3
60.69	1	2	3	3
60.82	1	2	3	3
63.63	1	3	4	4
69.16	2	4	5	5
69.21	2	4	5	5
69.25	2	4	5	5
69.31	2	4	5	5
69.33	2	4	5	5
69.43	2	4	5	5
69.65	2	4	5	5
69.83	2	4	5	5
69.93	2	4	5	5
72.43	2	4	6	6
75.15	2	5	7	7
75.89	2	5	7	7
76.80	2	5	7	7
79.89	2	6	8	8
80.74	2	6	8	8
80.89	2	6	8	8
80.93	2	6	8	8
80.97	2	6	8	8
81.03	2	6	8	8
81.03	2	6	8	8
81.14	2	6	8	8
81.26	2	6	8	8
81.39	2	6	8	8
81.58	2	6	8	8
81.64	2	6	8	8
81.70	2	6	8	8
81.70	2	6	8	8
81.71	2	6	8	8
81.71	2	6	8	8
81.81	2	6	8	8
81.81	2	6	8	8
81.87	2	6	8	8
81.91	2	6	8	8
81.94	2	6	8	8
82.53	2	6	8	8
84.72	2	6	8	8
87.31	2	6	9	9
91.50	2	7	10	10
91.58	2	7	10	10
91.68	2	7	10	10
101.50	3	8	11	11
102.30	3	8	11	11
103.93	3	8	11	11
108.24	3	9	12	12
108.77	3	9	12	12
122.85	4	10	13	13

Table 7.1: Illustration of pitch clustering at multiple (coarse-to-fine) levels of various pitch values occurring in one bass-riff of the song *"Truckin'"* by the rock group *The Grateful Dead*. P_i refers to a pitch value and L_j, $(j = \{1, 2, 3, 4\})$ refers to a certain clustering level. One can see how pitch grouping changes with a change in level.

along with further constraints on metrical position, harmony, etc. are used to narrow down possible choices of notes, out of which one is chosen at random. This procedure is subject to more variability when compared to the pitch representation, where each continuation index corresponds to a single audio segment. A more detailed analysis and evaluation of the interval representation could not be performed due to lack of time.

7.5 Context length & Recombination factor

In the generation stage, two parameters that play a key role in determining the nature of the generated melody are the *context-length* and the *recombination factor*. Whether a generation sounds very similar to the original or otherwise, from the representation perspective, is solely determined by these two. Before further explaining the significance of these parameters during generation, it would benefit to go over some terms that will be used frequently in this section.

- **Symbol Index:** Following segmentation and time-homogenization of the audio into a symbol sequence (see Chapter 4), each symbol in the sequence that represents the original melody is associated with an index. The index corresponding to each symbol gives its position in the original melody. For instance, the first symbol in any melody would be assigned the index 1, the second the index 2 and so on.

- **Current Index:** The current index is the index of the symbol at a certain homogeneous time instant from where the continuation is being generated.

- **Continuation Index:** A continuation index refers to the symbol index that is chosen as the continuation to follow the current index.

- **Recombination Graph:** A recombination graph basically shows how symbol indices are chosen during melody generation. The Y-axis depicts time in homogeneous time-steps. The X-axis contains symbol index values corresponding to the original melody. In the case where the generation and the original are identical, this graph would show a straight diagonal line starting at the origin. In the same way, if there exists a sequence of consecutive indices, these would also appear as a diagonal line in the recombination graph. A choice of non-consecutive indices during generation is seen as a series of "jumps" along the X-axis with increase along the Y-axis. (increasing homogeneous time-steps) Figure 7.3 shows an example recombination graph.

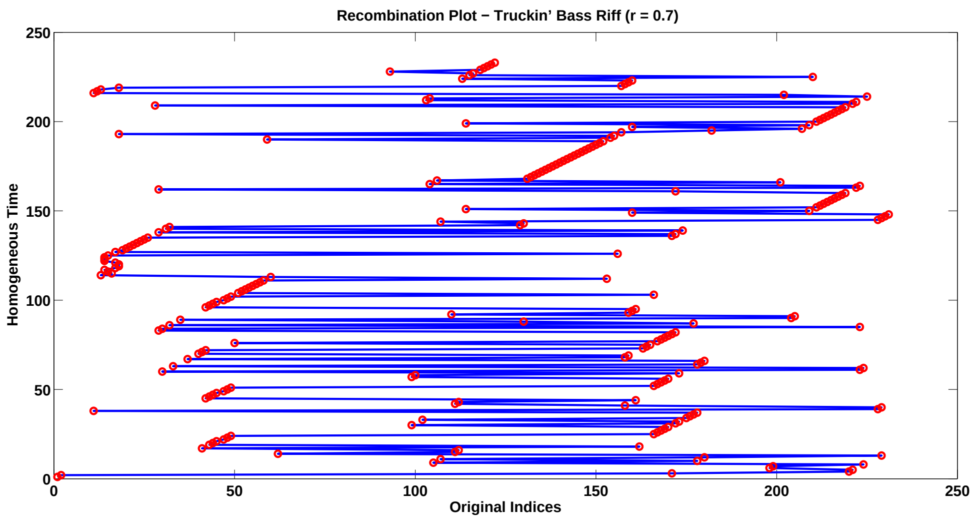


Figure 7.3: The recombination graph for the generation best rated by the experts (Grateful Dead - *Truckin'*).

The context-length specifies the maximum history (in terms of homogeneous time-steps) that the variable-length Markov Chain (VLMC) would consider in order to predict possible continuations. With reference to the theory of Chapter 5, this parameter represents the maximum *order* of the VLMC. With the generation algorithm currently being followed, the longer the context length, the more similar the generation will sound like the original. This happens because when the context-length for generation is increased, owing to the fact that the algorithm chooses only those levels that correspond to the maximum possible context-length, the number of choices that correspond to this longest context-length are also very few. Hence, the continuation index given by the VLMC would, in most cases, be the same as that in the original. The recombination graphs of Figure 7.4 show how an increasing context-length affects the choice of continuation indices during generation.

In the experiments, context-lengths of 2, 4, 6, 8, 16 and 32 were used. What is to be noted here is that not one context-length value was found to be ideal for all the cases. The choice depended very much on the tempo of the melody. In the case of riffs, for example, the context-length was found to be proportional to the duration of the riff. Choosing an appropriate context-length in this way ensured that the riff structure was preserved well. Too short a context-length resulted in broken riffs and abrupt changes in between them. The only advantage of having a sufficiently small context-length is that it would allow for variations, whose occurrence would be even more scarce when the context-length is long. On an average, for the melodies

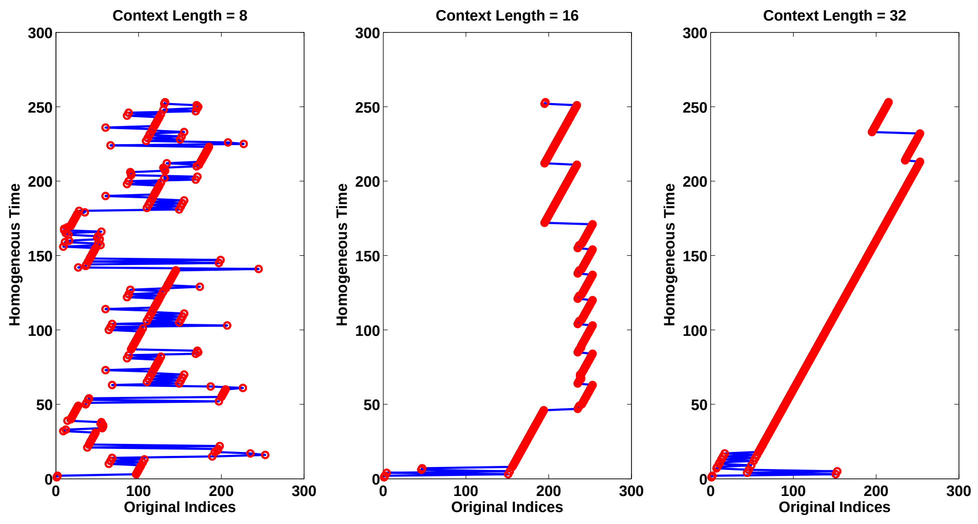


Figure 7.4: An example of how change in context-length affects continuation index selection in the generation. From left to right, it can be observed that there is an increasing tendency for consecutive indices (less jumps).

considered in the evaluation, a context-length of 8 was found to be reasonably good.

The recombination factor here is a value that has the range $[0, 1]$. Its value essentially determines the probability to continue with the continuation index of the segment following the current one in the original melody. In the extreme case of the recombination value being 0, the VLMC is completely ignored and the continuation index is what immediately follows the current index in the original melody. Given that one starts the generation with the first index of the original, this would result in an identity between the generation and the original. Hence, a smaller recombination value implies more similarity due to consecutive segments from the original melody occurring more often in the generation. In the case that the recombination factor is 1, only the continuation index suggested by the Markov chain is considered. This case is, in fact, an issue which is the main reason for having this parameter. Figure 7.5 shows how the number of discontinuous indices gradually decreases with an increase in the recombination value (keeping context-length constant).

In the experiments, recombination factor values of 0.1, 0.3, 0.5, 0.7 and 0.9 were chosen. The expected trend was also observed in the generated melodies. Those that used smaller values of the recombination factor had longer segments that were identical to those occurring in the original melody. This was more rare when higher values were used. However, as explained before, generation that use higher

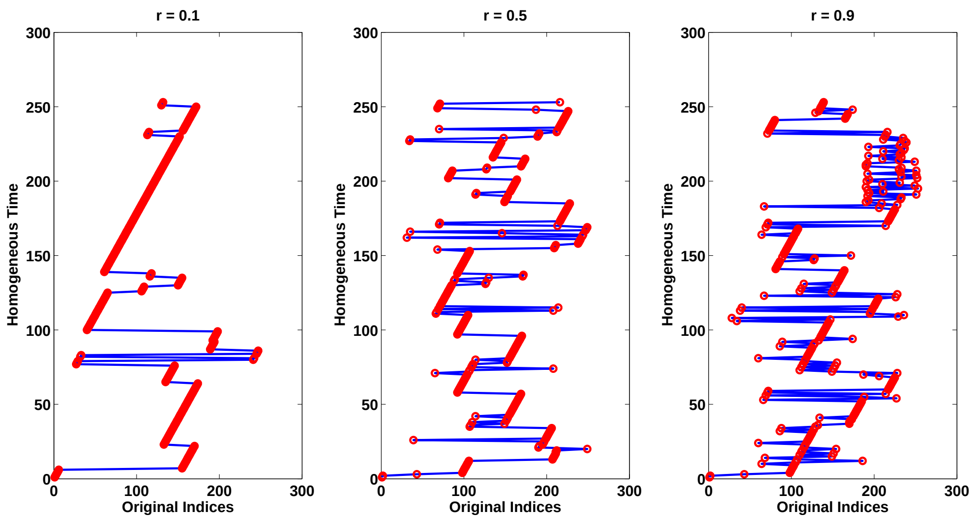


Figure 7.5: An example of how change in recombination value affects continuation index selection in the generation. From left to right, it can be observed that there is a decreasing tendency for consecutive indices (more jumps).

recombination factor values are susceptible to what sound like "loops". A "loop" occurs if, for a certain index there are no options except the direct continuation (the index that follows it in the original melody) and then at some point there is, again, only one option back to an index that occurred shortly before (the start of the loop). A typical loop is shown in the second plot of Figure 7.4. It was found that, among the chosen examples, values of 0.5 and 0.7 maintained a fairly good balance between identity and variability. That said, at times, even values as low as 0.3 resulted in nice sounding melodies, but with only very few variations.

7.6 Expert Evaluation

Evaluation of the quality of the generated melodies received from experts through the questionnaire, while confirming certain claims and expectations about the performance of the system, also revealed some issues and drawbacks in it. Several responses provided by them were in agreement with one another and helped make interesting inferences about the system. Some useful suggestions and opinions were also expressed and will be discussed in this section. In order to understand the results and their analysis better, Table 7.2 shows the context-length and recombination factor values used for each of the evaluated generations.

Song - Artist	**Context-length**	**Recombination**
Track 1	6	0.9
Track 2	32	0.5
Track 3	8	0.7
Track 4	8	0.9
Track 5	8	0.5
Track 6	8	0.9

Table 7.2: List of melodies used for evaluation.

One response that was consistently the same from all the experts for all the tracks was the observability of patterns in the generations that also occurred in the original. This is well in accordance with the general claim that Markov chains do reproduce melodic patterns in a musical context. Moreover, it is also indicative of the efficacy of the representation in use to capture recurring segments or *motifs* from the melody.

But when it came to the occurrence of these patterns in the generations, there were varying, and sometimes contradicting opinions. It was pointed out that, on certain occasions in the generated melody, note accents changed. That is, certain notes which originally occurred on a strong-beat shifted to a weak-beat and vice versa. This was, in fact, something expected due to the incorrect matches that occurred in the onset-beat matching stage for symbolizing rhythm (Sections 4.3, 7.3). The errors that originated during segmentation, thus propagated into the representation and manifested in this form during generation. However, there were mixed responses to the resulting effect which were evident from some of the experts' comments.

> *"I liked the fact that the "bass player" in the generated melody played a little longer one note, and changed the accent - the phase of the bass riff. Jazz music! I liked it!"*
>
> *-Expert 4 on Melody 3*

> *"The main difference from my point of view is the lack of the rhythmic support, and the shifting of the musical accents. Even more, the underlying harmony of the new melody is not clear, and that's why there's a certain sensation of chaos."*
>
> *-Expert 2 on Melody 1*

There was a fairly positive feedback in regards to the tempi of the generated melodies, with two-thirds of all the responses saying that they were the same as those of their respective originals. It is interesting to see that even in those cases where the tempo of a generation was considered not to be the same as that of the original, it was never found to be double or half of it. This observation could possibly be due to the accent-shifting effect (Section 7.3). This is supported by the fact that in most of the cases where the tempo of the generation and the original were not considered to be the same, the experts also pointed out that melodic patterns were occurring in inappropriate metrical locations (or broken riffs) in the generation. It may, therefore, have been the case that due to irregularity in reproduction of certain onsets in their appropriate locations, it was not very straightforward to keep a track of the tempi in the case of some of the melodies (especially ones without accompaniment).

The question on overall similarity between the generation and the original received a fairly good feedback with all responses being either "Very similar" or "Somewhat similar". Although, it must be noted that in certain cases (tracks 2 & 6), the generations were found to be too similar to the original which adversely affected their interestingness rating. As one of the experts pointed out,

"It is rather uninteresting as it introduces very few new elements."

-Expert 1 on Melody 6

On revisiting the database and generation parameters, it was found that the generation for track 2 used a very long context (32 symbols) as a result of which there were long segments replicated from the original. In the case of track 6, the riff itself was composed of few notes and with minimal variations. Hence, one may even consider it to be an encouraging sign that at least a few variations were produced on this track. Moreover, the riff structure and rhythmic evolution of the generation in this case are almost identical to those of the original, and when played with the accompaniment could have created such an impression. It is, however, surprising that two of the experts considered the riff structure to be occurring inappropriately. This is yet to be clarified.

When it came to interestingness, from the responses given, it seemed like the experts found interesting those cases where there were at least some noticeable variations from the original. There were instances where, although a particular generation received good feedback for other questions, it received a poor interestingness rating (The rating of Expert 1 for Melodies 2 and 6). It is not very easy to generalize the basis of these ratings due to their subjectivity, but one can get an idea of what was appreciated in the generations from some of the comments by the experts.

One aspect of the generation of Track 1 that everyone noticed and, in particular, did not like, was the repetition of the same notes in succession. It should be noted about this example that a very high recombination factor value was used (0.9). As a result of this the "loop-effect" that was explained earlier was observed in the form of repetition of the same note. However, this was quite effectively handled by the use of a slightly lower value of the recombination factor.

The generation framework used here seems to be more suitable in the case of short melodic segments rather than long solos. The main problem with longer melodies is that an overall higher-level structure, which is often ensured by the musician, is absent. As the temporal scope considered by the system is limited to around a single measure, it does not make any distinction between notes occurring at different time locations in the entire melody. Keeping this in mind, a more suitable application for such a system would be to learn motifs occurring in a longer melody and reproduce or generate variations of these motifs.

The generation for track 3 received significantly better feedback and ratings than the others consistently from all the experts. It would help to use this track as a reference in any future work to assess the relation between the model parameters and different facets of the melody in order to improve generations related to other melodies as well.

It was also appreciated that the system was able to reproduce even silent pauses from the original melody on several occasions. Except melodies 5 & 6, the generation of every other track had at least three experts give it a rating of 3 or higher.

Overall, the feedback given by the experts, although not extremely positive, was in fact encouraging. All of them found at least two generations out of the six genuinely interesting and expressed that with some minor improvements in metrical analysis, synthesis and overall structure of the generations, the others could also sound much better. One of them even pointed out that the system does at times generate interesting variations that were "inspiring".

> *"... an interesting evolution of the bass line that could be useful as an inspiration for musicians."*
>
> *-Expert 2 on Melody 2*

Chapter 8

Conclusions and Future Work

Feedback received from the experts who evaluated the system's musical output was indeed encouraging and indicative that the results were satisfactory. However, there still remain some issues that need to be addressed. Moreover, the present work laid the foundation for a system that generates stylistically similar melodies and there is still room for improvements to enhance its current performance.

8.1 Future Work & Improvements

This work, while mainly focussing on the pitch-based representation for symbolizing audio data, also proposes exploiting interval and simplified Narmour feature information towards the same end. Due to limited time, these representations could only be implemented, but not explored in depth or evaluated. Given that these differ significantly from the pitch representation, it is expected that the musical output using these representations could also be different and worth looking into.

It was also observed during evaluation that there didn't exist a single value of context-length that would suit all the examples. However, a possible cue, which could be explored to automatically estimate its value, is the average tempo of the melody. A resolution of this issue could also be a task in the future. Another related issue that came to light in the evaluation was the relative success of riff-type melody generation over the solo-type. This was mainly due to the lack of an explicit mechanism to handle higher-level structure of the melody. A possible extension to this work could be to incorporate methods that segment the melody into regions based on higher-level similarity that would help reproduce its global evolution [Moz94], and use what could be short *motifs* that are generated by the VLMC locally.

The selection of the number of levels for pitches and meter was handled manually in this work. A measure of the complexity of the different facets of the melody could serve as indicators to determine these numbers dynamically depending on the melody. Additionally, with these improvements, the present approach can also be extended to melodies of other musical styles played by different instruments.

One of the main reasons why the Turing test could not be applied in the evaluation of this work was the fact that there were various synthesis artefacts in the generated melody. These were also pointed out by some of the experts. Some effort in this direction to improve the synthesis quality would also be worthwhile.

8.2 Conclusions

This thesis presents a framework to generate guitar and bass melodies that contain stylistically similar variations of a given original, based on the statistical analysis of melodic patterns in it. The said variations are produced by the re-shuffling of audio segments (notes) occurring in the original melody itself, using the variable-length Markov chain model for learning meaningful arrangements of them. It forms a part of the broad and fascinating area of stochastic music generation with computers. This work addresses the much ignored problem of generating stylistically similar melodies directly from audio, instead of symbolic data (MIDI, MusicXML, etc.). This is, in general, a more difficult problem due to the occurrence of segmentation errors that tend to propagate into any symbolic representation that is in use for generating music. The work also highlights, and to some extent handles, various issues that arise when dealing with audio data directly. A multi-level representation, similar in spirit to that of [MP10] is employed here for the symbolic representation of notes in the melody and their metrical positions. A novel application of the clustering-level selection method as proposed by [CH74], in the context of pitch data is also applied towards this end. Although not to completion, the work also explores the possibility of using the simplified Narmour features and an interval-contour representation as a replacement for pitch representation. This has been left out for future experiments. The pitch-based representation produced satisfactory results which were appreciated by a group of experts who evaluated the musical output of the system.

Bibliography

[ADD99] Gerard Assayag, Shlomo Dubnov, and Olivier Delerue. Guessing the Composers Mind: Applying Universal Prediction to Musical Style. In *Proc. International Computer Music Conference*, pages 496–499, 1999.

[All02] Moray Allan. Harmonizing chorales in the style of johann sebastian bach. Master thesis, School of Informatics, University of Edinburgh, 2002.

[Ame87] Charles Ames. Automated Composition in Retrospect: 1956-1986. *Leonardo Music Journal*, 20(2):169–185, 1987.

[AW04] Moray Allan and Christopher K. I. Williams. Harmonizing Chorales by Probabilistic Inference. In *Advances in Neural Information Processing Systems*, 2004.

[B7̈8] Alexander Büchner. *Mechanical Musical Instruments.* London Batchworth Press, 1978.

[Ban11] Band-in-a-box, 20th August, 2011. http://www.pgmusic.com/.

[BDA$^+$05] Juan Pablo Bello, Laurent Daudet, Samer Abdallah, Chris Duxbury, Mike Davies, and Mark B. Sandler. A Tutorial on Onset Detection in Musical Signals. *Speech and Audio Processing, IEEE Transactions on*, 13(5):1035–1047, 2005.

[Bil94] J. Biles. Genjam: A genetic algorithm for generating jazz solos. In *International Computer Music Conference*, 1994.

[Bro06] Paul Brossier. *Automatic Annotation of Musical Audio for Interactive Applications.* PhD thesis, Queen Mary, University of London, 2006.

[BW99] Peter Bühlmann and Abraham J. Wyner. Variable length markov chains. *The Annals of Statistics*, 27(2):480–513, 1999.

[Can11] Chris Cannam. “Sonic-Annotator”. http://omras2.org/SonicAnnotator, April 2011.

[CH74] T. Calinski and J. Harabasz. A dendrite method for cluster analysis. *Communcations in Statistics - Theory and Methods*, 3:1, 1974.

[Con03] Darrell Conklin. Music Generation from Statistical Models. In *AISB 2003 Symposium on Artificial Intelligence and Creativity in the Arts and Sciences, In Proceedings of*, pages 30–35, 2003.

[Cop96] David Cope. *Experiments in Musical Intelligence.* A-R Editions, Madison, Wisconsin, 1996.

[CW95] Darrell Conklin and Ian H. Witten. Multiple viewpoint systems for music prediction. *Journal of New Music Research*, 24(1):51–73, 1995.

[DAC07] Shlomo Dubnov, Gerard Assayag, and Arshia Cont. Audio oracle: A new algorithm for fast learning of audio structures. In *International Computer Music Conference*, pages 224–228, 2007.

[DBDS03] Chris Duxbury, Juan Pablo Bello, Mike Davies, and Mark B. Sandler. Complex domain Onset Detection for Musical Signals. In *6th Conference on Digital Audio Effects (DAFx-03)*, 2003.

[dCK02] Alain de Cheveigne and Hideki Kawahara. Yin, a fundamental frequency estimator for speech and music. *The Journal of the Acoustic Society of America*, 11(4):1917–1930, 2002.

[Dix01] Simon Dixon. Automatic Extraction of Tempo and Beat from Expressive Performances. *Journal of New Music Research*, 30:39–58, 2001.

[Dix07] Simon Dixon. Evaluation of the audio beat tracking system beatroot. *Journal of New Music Research*, 36(1):39–50, 2007.

[FR67] H. P. Friedman and J. Rubin. On some invariant criteria for grouping data. *Journal of the American Statistical Association*, 62(320):1159–1178, Dec 1967.

[Gar70] M. Gardner. Mathematical games: The fantastic combinations of john conway’s new solitaire game “life”. In *Scientific American*, volume 223, pages 120–123. 1970.

[GKM03] Emilia Gómez, Anssi Klapuri, and Benoit Meudic. Melody description and extraction in the context of music content processing. *Journal of New Music Research*, 32, 2003.

[GS97] Charles M. Grinstead and J. Laurie Snell. *Introduction to Probability*, chapter 11, pages 405–452. American Mathematical Society, 2 revised edition edition, 1997.

[GTK10] Jon Gillick, Kevin Tang, and Robert M. Keller. Machine learning of jazz grammars. *Computer Music Journal*, 34(3):56–66, 2010.

[Har02] James Harley. The Electroacoustic Music of Iannis Xenakis. *Computer Music Journal*, 26(1):33–57, 2002.

[HI59] Lejaren A. Hiller and Leonard M. Isaacson. *Experimental Music: Composition with an Electronic Computer*. McGraw-Hill, 1959.

[Hil70] Lejaren Hiller. *Music Composed with Computers: A Historical Survey*. Cornell University Press, 1970.

[Jeh05] Tristan Jehan. *Creating Music by Listening*. PhD thesis, Massachusetts Institute of Technology, 2005.

[JMF99] Anil K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A survey. *ACM Computing Surveys*, 31(3):264–323, 1999.

[KCGV00] YoungMoo E. Kim, Wei Chai, Ricardo Garcia, and Barry Vercoe. Analysis of a Contour-Based Representation for Melody. In *Proceedings of International Symposium on Music Information Retrieval*, 2000.

[LM94] Michael Z. Land and Peter N. McConnell. Method and apparatus for dynamically composing music and sound effects using a computer entertainment system, 1994.

[Mar10] M. Marchini. Unsupervised generation of percussion sequences from a sound example. Master's thesis, Universitat Pompeu Fabra, 2010.

[MC85] Glenn W. Milligan and Martha C. Cooper. An Examination of Procedures for Determining the Number of Clusters in a Data Set. *Psychometrika*, 50(2):159–179, June 1985.

[Moz94] Michael C. Mozer. Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multi-scale processing. *Connection Science*, 6(2–3):247–280, 1994.

[MP10] Marco Marchini and Hendrik Purwins. Unsupervised generation of percussion sound sequences from a sound example. In *Sound and Music Computing Conference*, 2010.

[Mus11] Rapidcomposer, 20th August, 2011. http://www.musicdevelopments.com/rapidcomposer.html.

[Pac03] Francois Pachet. The Continuator: Musical Interaction With Style. *Journal of New Music Research*, 32(3):333–341, 2003.

[Pai08] Jean-Francois Paiement. *Probabilistic Models for Music*. PhD thesis, Ecole Polytechnique Federale de Lausanne, 2008.

[PW01] Marcus Pearce and Geraint Wiggins. Towards a framework for the evaluation of machine compositions. In *In Proceedings of the AISB01 Symposium on AI and Creativity in Arts and Science. AISB*, pages 22–32, 2001.

[Roa96] Curtis Roads. *The Computer Music Tutorial*, chapter 18, 19, pages 819–909. The MIT Press, 1996.

[RST96] Dana Ron, Yoram Singer, and Naftali Tishby. The power of amnesia: Learning probabilistic automata with variable memory length. *Machine Learning*, 25(2–3):117–149, 1996.

[Rub11] Rubber band library, 20th August, 2011. http://breakfastquay.com/rubberband/.

[Sch96] E. Glenn Schellenberg. Expectancy in melody: tests of the implication-realization model. *Cognition*, 58:75–125, 1996.

[Xen01] Iannis Xenakis. *Formalized Music: Thought and Mathematics in Composition (Harmonologia Series, No 6).* Pendragon Pr, 2nd edition, March 2001.