Perception based approach on pattern discovery and organisation of point-set data

Mikko Pelkonen

Pro graduate thesis
UNIVERSITY OF HELSINKI
Department of Computer Science

Helsinki, March 11, 2020

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty Faculty of Science	Laitos — Institution — Department Department of Computer Science

Tekijä — Författare — Author Mikko Pelkonen
Työn nimi — Arbetets titel — Title Perception based approach on pattern discovery and organisation of point-set data
Oppiaine — Läroämne — Subject Computer Science

Työn laji — Arbetets art — Level Pro graduate thesis	Aika — Datum — Month and year March 11, 2020	Sivumäärä — Sidoantal — Number of pages 56

Tiivistelmä — Referat — Abstract

The general topic of the thesis is computer aided music analysis on point-set data utilising theories outlined in Timo Laiho's Analytic-Generative Methodology (AGM) [19]. The topic is in the field of music information retrieval, and is related to previous work on both pattern discovery and computational models of music. The thesis aims to provide analysis results that can be compared to existing studies.

AGM introduces two concepts based on perception, sensation and cognitive processing: interval–time complex ($IntiC$) and musical vectors (muV). These provide a mathematical framework for the analysis of music. $IntiC$ is a value associated with the *velocity*, or rate of change, between musical notes. Musical vectors are the vector representations of these rates of change. Laiho explains these attributes as meaningful for both music analysis and as tools for music generation. Both of these attributes can be computed from a point-set representation of music data.

The concepts in AGM can be viewed as being related to geometric methods for pattern discovery algorithms of Meredith, Lemström et al. [24] who introduce a family of 'Structure Induction Algorithms'. These algorithms are used to find repeating patterns in multidimensional point-set data.

Algorithmic implementations of $intiC$ and muV were made for this thesis and examined in the use of rating and selecting patterns output by the pattern discovery algorithms. In addition software tools for using these concepts of AGM were created. The concepts of AGM and pattern discovery were further related to existing work in computer aided musicology.

Avainsanat — Nyckelord — Keywords
music information retrieval, pattern discovery

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Contents

1 Introduction

Perception of music and the inference of structure are often studied in the field of computer aided musicology in an effort to further the understanding of our cognitive capacities and how we hear music. This has produced a wealth of interesting theories, models and algorithms, as well as problems.

This thesis focuses on two different directions of research. The perception based approach to music analysis in Timo Laiho's Perception, Time and Music Analysis — Analytic Generative Methodology [19] is investigated from the point of view of computer aided music analysis, with the aim of the implementation of it's core concepts and the evaluation of their use in music analytic tasks. Specifically the concepts are viewed in conjunction with the family of geometric pattern discovery algorithms introduced by Meredith et al. [24, 21].

In Analytic Generative Methodology (AGM) Laiho presents a new methodology for music analysis. He calls the methodology *perception based* and views music analysis through the perspective of musicology, physics, cognitive science and philosophy. AGM includes two mathematical concepts which are called interval–time complex (*intiC*) and musical vectors (muV). Both of these describe the context based perception of movement, or the *velocity*, of pitches. These concepts were studied and implemented and works of computer aided musicology were reviewed to provide a context for their application.

The pattern discovery algorithms by Meredith et al. [24, 21] are used for the discovery of recurring patterns in multidimensional point-set data, especially data representing polyphonic music. These patterns can be used to for example infer the structure of a musical work. The algorithms suffer from the problem of finding perceptually significant patterns. They may output hundreds of thousands of different patterns for a single musical composition and only a fraction of these could be thought of as perceptually significant or interesting. As the aim of AGM is to explain perception based phenomena it was evaluated in the use of solving this problem and inspecting whether the combination of AGM and pattern discovery had combined value as tools for computer aided music analysis. The results of this thesis include the association of musical vectors and patterns which were be visualised as vector graphs that demonstrated aspects of structure in musical works.

As the pattern discovery algorithms operate by calculating vectors between coordinate values of datapoints and the concept of musical vectors from AGM can also be described as vector based relations between the points of point-set data, the techniques from these pattern discovery algorithms were used in the implementation of the computation of muVs.

Music theoretical methodologies, computational models of music and other works of computer aided musicology were reviewed and potential connections to AGM were made. Context based time and pitch-sensitive concepts are found in works studying the grouping structure of music [3] and the concepts of high- and low-level cognitive processing regarding the perception of music, which was a central concept in AGM, can be found in computational models of music [2, 26, 9].

The rest of this thesis is organised as follows. Related work is introduced first in separate chapters. For computational models of music, theoretical approaches to musical structure and models of music that employ aspects of perception are presented. Then pattern discovery algorithms are inspected in detail, including the description of the geometric representation of multidimensional point-set music data used in the implementations of the algorithms in this thesis. Next Analytic Generative Methodology is introduced and within it begin the contributions of this thesis: the implementations of the core concepts of AGM and the algorithms for the computation of musical vectors. In it's own chapter the usage of AGM software for music analysis is briefly discussed. The chapters regarding the combination of the perception based approach of AGM with pattern discovery include utilisation of the algorithms developed for this thesis with pattern discovery algorithms, as well as empirical experiments. Finally the results of using the presented algorithms are reviewed and their usefulness and potential applications are discussed.

2 Computational models of music

Previous research exists on several computational approaches to implementing different musicological models and theories. A central component of this thesis is a music analytic methodology, AGM, so it has been necessary to investigate existing computational music analytical models. Different approaches to computer aided musicology were reviewed. They could be broadly divided into statistical modelling and algorithmic, or rule based, systems. This division is not concrete as there is overlap between the groups. The reviewed systems using statistical modelling acquire knowledge from a corpus of music and then for example may use some probabilistic approach to generate outputs using this knowledge. Rule based systems on the other hand may utilise algorithmic solutions implemented based on expert knowledge on the subject matter. Popular tools in this case include pattern discovery and formal grammars used in natural language processing. Each approach has a unique background on which their motivation, chosen methods and goals are based. Theories and analytic models are focused on creating structural descriptions of musical works and statistical models can be used to create outputs such as predictions of notes not yet heard in a melody and new works of music.

In the influential book Generative Theory of Tonal Music (GTTM) Lerdahl and Jackendoff [20] outline a comprehensive theory for western tonal music. The word generative in this case refers to generative linguistic theory, the approach which they have adopted for music analysis. In AGM the word generative on the other hand only refers to the generation of new output, not an other existing paradigm. In GTTM a piece of music is described as a mentally constructed entity and that the central task of music theory should be to explicate that “mentally produced organisation.” The theory should include both artistically interesting aspects of musical structure and principles that account for simpler musical phenomena. Many of the concepts used in computer aided musicology are related to this or similar theories.

GTTM uses a concept of musical intuition of the experienced listener. The theory aims to describe how this idealised listener hears, understands and structures music. It tries not to just describe the conscious understanding of musical structure but rather the unconscious knowledge which the listener uses to organize and process patterns of pitch and other properties of notes such as duration, dynamics, timbre. The experienced listener differentiates between typical or anomalous elements such as errors made by a performer which may possibly be found ‘ungrammatical’.

The authors’ view is that a theory of a musical idiom should characterise the mental organisation in terms of a formal grammar that models the connections the listener makes between the presented musical surface of a piece, i.e. the music the listener hears, and the structure he attributes to the piece, and that such a grammar consists of a system of rules that can be used to assign analyses to pieces. The theory is restricted to parts of musical intuition that are characterised as being hierarchical in nature. These are divided into four different structural descriptions. *Grouping structure* is a hierarchical segmentation of the piece of music into motives, phrases and sections. *Metrical structure* expresses that events of the piece are related to rhythmic changes of strong and weak beats at different hierarchical levels. *Time-span reduction* assigns a hierarchy of ‘structural importance’ to the pitches of the piece. This is determined by their position in the grouping and metrical structures. Finally *prolongational reduction* designates a hierarchy of both harmonic and melodic tension and relaxation, and the progression and continuity of the pitches.

The approach of GTTM is adopted from the generative-transformational grammar school of the study of language. The generative linguistic theory is described in GTTM as an effort to characterise what humans know when they know how to speak a language. It is said to aim to explain what enables the understanding and creation of an indefinite number of sentences most of which a person speaking has never heard before. Linguistic theory models unconscious knowledge, which has not been acquired by direct instruction or is not available to conscious introspection. This is done with a formal system called a grammar which describes, or is used to ‘generate’, all the possible sentences of a language.

The approach in GTTM is compared to preceding theories that had used linguistic approaches. It is said to differ by combining psychological concerns and the formal nature of the theory to serve musically or psychologically interesting generalisations. In addition the authors claim that previous approaches had attempted literal translations of aspects of linguistic theory into musical terms, for instance by looking for musical 'parts of speech', transformations, deep structures or semantics. These have been replaced in GTTM with concepts of musical structure such as organisation of rhythm and pitch, the differentiation of dynamics and timbre and motivic-thematic processes.

The intention of GTTM is the structural description of any tonal piece of music from the viewpoint of what the experienced listener infers in hearing of the piece, with the understanding of musical cognition as the goal. In addition to the assignment of structural descriptions to a piece, the theory tries to differentiate the descriptions along a scale of coherence, weighting them as more or less 'preferred' interpretations. The experienced listener is thought to more likely attribute some structures to the music than others. For these purposes the theory employs rules which are divided into two types. The *Well-formedness rules* (WFRs) specify the possible structural descriptions of music and the *preference rules* (PRs) designate which structural descriptions out of all the possible ones correspond to the experienced listeners hearing of a particular musical work. The four different structural descriptions of music in GTTM are connected to four processes: grouping structure analysis, metrical structure analysis, timespan reduction analysis and prolongation reduction analysis which are evaluated using the two types of rules.The concepts of GTTM or their analogies, recur in many computational approaches to modelling aspects of music.

Hamanaka et al. [16] have made a computational implementation of the GTTM which includes algorithms for an Automatic Time-Span Tree Analyser and σGTTM Analyser which detects local grouping boundaries. The latter is done using a statistical learning method with a decision tree. With the implementations they encountered issues with ambiguous rule definition and conflicts among preference rules. Additionally they bring up three problems related to computation and music analysis. There exists *ambiguity in music analysis* — a piece of music will typically have more than one interpretation. They consider this to be a major obstacle. The results are susceptible to *context dependence.* They depend on many different factors such as rhythm, chord progression, melody of other parts, historical context and other yet unknown factors. Finally there exists a *trade-off relationship in music analysis* between the level of automation of the analytic process and the amount of variation in the input data. Where automating the analysis of a more varied dataset becomes increasingly problematic. These are similar issues to what Laiho [19] criticises of axiomatic theories and methodologies of music analysis.

In his doctorate thesis [2] Emilios Cambouropoulos has formulated a theory for obtaining structural descriptions of musical surfaces. The goal is otherwise similar to GTTM but is based solely on creating computational methods for such a theory. The aim of the theory is that computed descriptions of a musical work could be acceptable by a human music analyst. The theory and its implementation include both pattern discovery and, in a subsystem for local grouping of a melodic music surface (Local Boundary Detection Model), rules that relate to principles of proximity and similarity which are similar to the core concepts of AGM.

Cambouropoulos describes that the primary aim of computational models is to assist in the formulation of theories that describe *musical activities and tasks* consistently rather than searching solutions to musical problems. He states that theories allow the formulation of hypotheses and models. These can be implemented as computer programs and evaluated. The results can then be used in the re-examination and adjustment of the initial theories. One of the goals of this thesis is to implement the central components of AGM in part to help examine the theories it presents.

Cambouropoulos describes the overall form of his theory as follows. Musical notes are used as discrete events that make up a 'musical surface (0)'. These events are then converted to another 'musical surface (1)' that is comprised of a number of musical interval profiles. For pitch this includes exact pitch intervals, scale/step intervals, step/leap intervals and a contour.

Next a process for discovering potential local boundaries is applied to the musical surface (1). This task is called the Local Boundary Detection model (LBDM) [3]. Local discontinuities and changes are used to provide cues of possible points where local boundaries may be detected. With the assumption that notes that are immediate neighbours of strong boundaries will tend to be perceived as being more prominent than others, accents of individual note events may be calculated. Cambouropoulos hypothesises that these accents are the key to determining a low-level metrical structure where one exists.

This 'proto-segmentation' is then accompanied by higher-level components to provide a more valid *segmentation*. These components are based on, what Cambouropoulos calls, parallelism and similarity. The term parallelism refers to the concept of repeating patterns in music which is explored in more detail later in this thesis. Resulting recurrent musical patterns are perceived by a listener or analyst and suggest partitioning of a musical surface. The partitioning might contain ambiguous boundaries and overlapping segments. The results may contain boundaries that are incompatible with each other or contradicting with locally detected boundaries. The local boundaries and higher-level segmentation boundaries can then be combined together for a more complete segmentation.

Using the low-level properties of the musical structure from previous steps of the system the parallelism component may be applied on reduced versions of the surface as well. For example by using only notes on metrically strong positions or accented notes.

The musical segments produced by the above operations are organised and labelled into categories based on their similarity. A 'goodness' measure of these categorisations is used to determine which of alternative segmentations should be preferred. In addition Cambouropoulos suggests that the discovered categories can be organised by their ordered in-time relationships and finally the GCTMS-algorithms can be applied to the new sequences of labelled musical segments, for example motives, to derive higher-level structural descriptions. This final part has not been examined in publications.

As a third example of musical structure we may look into pattern discovery algorithms which will be discussed more thoroughly later. An algorithm that finds a set of repeating patterns in a musical work that can be used to present the whole work can be thought of as an implementation of the concept of grouping structure. An example of this, the COSIATEC algorithm [24, 21], generates a list of repetitive patterns that can be used to represent a musical work without overlap. This kind of an approach can be thought of as an algorithmic structural analysis of a musical work. Meredith [23] has conducted a review of a large selection of such algorithms to evaluate which of them perform well in such music analytical tasks.

2.1 Stylistic composition

As AGM is based in a cognitive model developed to analyse music, the work of Pearce [28] and Pearce et al. [25, 26] with statistical modelling and hypotheses of cognitive processing in music was especially relevant. In addition similar approach using Random Generation Markov Chains (RGMCs) by Collins et al. [9] is looked at as it has been made to include pattern discovery. Both of these lines of research center around acquiring knowledge from a corpus to build a statistical model of music. One major goal in these works is stylistic composition, i.e. creating works in a style of a composer or a period of music, differing from free composition — any work that is not a pastiche.

Pearce [28] proposes that "statistical models which acquire knowledge through induction of regularities in corpora of existing music can, with appropriate methodologies, provide significant insight into the cognitive processing involved in music perception and composition." His research examines music, specifically, from the point of view of Artificial Intelligence. Pearce conducts an empirical examination of modelling techniques in order to develop statistical models of musical structure which have the potential to account for aspects of the cognitive processing of music. The best performing models are applied to examine specific hypotheses regarding processing in music perception and cognition. To reduce the complexity of the task the research is limited to modelling monophonic music, focusing only on pitch structure. The influences of tonality, rhythm, meter and phrase on pitch structure are accounted for. A symbolic presentation of musical surface is used, and features such as tonal centres and phrase boundaries are acquired directly from score sheets. As the modelling technique Pearce focuses on using finite context models, also known as n-gram models, to examine the minimal requirements placed on the cognitive processing of melodies. Pearce [28] refers to Wiggins and Smaill [29] who note that "motivations for applying AI techniques to the musical domain can be drawn out on a continuum between those concerned with understanding human musical abilities at one extreme (cognitive science) and those concerned with designing useful tools for musicians, composers and analysts at the other (applied AI)." AGM seems to be applicable to both ends of the spectrum Wiggins & Smaill propose. Pearce uses similar definition regarding the existence of the different motivations for applying AI techniques to the musical domain. The motivations originate from natural science, engineering, engineering science, the arts and the humanities. Further, "motivations drawn from different disciplines imply different goals and methodologies for achieving those goals." [28] Pearce develops a system that is based on and evaluated using methodologies drawn from basic AI research and applies his system to the cognitive modelling of music perception and composition. The current understanding of music cognition is said not to be as advanced as that of other areas of human psychology, for example visual perception and memory. Music cognition draws on knowledge in several different domains and at different levels of description, and it is hinted that purely algorithmic model seems unrealistic.

Pearce and Wiggins [26, 27] have further studied auditory expectation using probabilistic modelling. The focus of their work is around a model of prediction in musical melodies, but they have used similar techniques for predictions of musical harmony and even expectations in speech [27]. They state that expectations play a role in a multitude of cognitive processes, listing sensory perception, learning and memory, motor responses and emotion generation. Pearce and Wiggins write that "Accurate expectations allow organisms to respond to environmental events faster and more appropriately and to identify incomplete or ambiguous perceptual input. To deal appropriately with changes in the environment, expectations must be grounded in processes of learning and memory." [27] Their their work is presented around the idea of information transmission by musical structure during the listening experience, with the context of the producer of the music and the listener sharing knowledge. The cognitive processes used in such a transmission and their relationship with other more general processes in human cognition are explored in [27].

The approach is developed and evaluated in the model called the "Information Dynamics of Music (IDyOM) model of musical melody processing". [27] It is divided in separate layers with two way information flows, processes and phenomena presented between each layer. The model itself does not incorporate the connection in both ways as it is organised as a strict bottom-up hypothesis, but the authors acknowledge the existence of a two way interaction between the layers. The layers are starting from bottom: Auditory stimulus $\rightarrow$ pitch/time percepts in sequence $\rightarrow$ learning system $\rightarrow$ expectations $\rightarrow$ segmentation ($\rightarrow$ conscious experience). The core is a model of human melodic pitch perception that uses Markov models, or more specifically n-grams [28, 27]. IDyOM encounters a musical corpus from which it learns and creates a compact representation of the data by matching new note sequences against previously encountered ones. The model allows predictions of n-gram models of all possible orders to contribute probability mass to each predicted distribution.

The model [27] has five different configurations: long-term model (LTM), short-term model (STM), their combinations and variations. LTM is exposed to an entire corpus, which models the listener's learned experience and context for information theoretic analysis. This comes with a variation that learns as stimulus proceeds (LTM+). STM is exposed only to the current melody. In addition there were models which combined the short- and long-term models. These were determined to be serious candidates as models of human cognition and the single models were said to be informative regarding musical structure.

Data is represented in the model with multiple features for each note using the multiple viewpoints approach [11]. The viewpoints are derived from pitch, time or a combination of them both. In addition the model uses an explicit representation of a sequence in time. Basic viewpoints are selections of note features: pitch, the start time of the note, duration and mode. A derived viewpoint is for example a pitch interval. The viewpoints may be linked as compound viewpoints AxB. Threaded viewpoints are selected elements of a sequence: the scale degree of the first note in each bar of a melody when metrical information is available. The viewpoints are thought of as sequences of the values and each viewpoint models a percept which is expressed and used in music theory.

Despite of this theoretical foundation the authors hold it important that they are not predisposing the system in a hard-coded or rule-based way and that the features are only the properties of the data level of abstraction below the level of interest of their study. This would not contradict their claims of "domain-generality and methodological neutrality at the level of interest of sequence processing." [27] Additionally the authors speculate about a system that would construct its own viewpoints with machine learning methods. The model is limited to monophonic music, only one aspect of the significantly multidimensional range of music. The authors regard their memory model as inadequate as it employs total recall which never fails and outperforms humans in implicit learning tasks. [27]

As another example of a model usable in stylistic composition Collins et al. have developed two computational models of stylistic composition using a constrained Markov model. [9] With one of the systems they have included SIACT pattern discovery algorithm with a "perceptually validated formula" for rating pattern importance in guiding of target generation [6]. The algorithm is described as a "within-domain analogy-based design system" [9] which consists of a target design and a source design. The target is a new passage of music to be generated and the source is an excerpt of human-composed music in an intended style. Large-scale repetitive structures have been taken into account in the system, and the authors state this as the most important contribution of the work, noting that other current approaches to music generation tend to be focused on small-scale relationships. The small-scale relationships in their system are acquired using a constrained Markov model calculated over a database of pieces in the intended style.

Experiments in Musical Intelligence (EMI) created by David Cope [12] is an algorithmic, or rule based, system for stylistic composition of music. Collins et al. [9] use EMI and its output as a reference point in their research on probabilistic models of stylistic composition. EMI is built upon using musical *signatures*, or often used patterns that signal a composer's style. Cope argues that composers create music by mixing such signatures and using *recombinancy* — the recombination of the elements found in their other works and in the music of other composers. In his books Cope examines this concept through various examples of western classical music. EMI has not been been published in peer-reviewed journals but has been referenced and reviewed by others.

Both Cope and Collins refer to *Musikalisches Würfelspiel*, or the *musical dice game*, as one of the earliest examples of formal combinatorial probabilistic music. It is a system that was popular throughout the 18th century in Western Europe where the player of the game rolls dice to generate bars of a music using a stylistic template musical score, in other words "Picking segments from a database score sheet". Cope has used EMI to produce new works in the styles of composers such as Stravinsky, Palestrina and Scott Joplin.

EMI attempts to create a Musikalisches Würfelspiel out of music that is not specifically designed to be such. It is done with the help of pattern matching to avoid deconstructing the musical signature elements, which are integral to the style of the music. Music is first entered to the EMI in the form of events. The events contain the information of note attributes such as pitch, timing, duration etc. Then the music is analysed according to a system of identifiers and pattern matching is used to protect signatures from recombinancy. A deconstruction phase places musical segments in a lexicon according to a meaning that is attributed to the segments. Finally these musical segments are reconstructed according to an augmented transition network (ATN)

Collins et al. have developed and conducted a review of algorithms for stylistic composition [9]. Focusing on several issues: avoidance of replication, database construction, level of disclosure, i.e. to what extent is the model reproducible, and the rigour and extent of evaluation. Different stylistic composition briefs are used in the paper to evaluate how a model succeeds in composition tasks. Ground bass and fugal exposition involve differing compositional strategies. Chorale harmonisation is more concerned with harmony than counterpoint. Other briefs are classical string quartet, Chopin mazurka opening section and advanced tonal composition. The paper focuses on the implementation of a Chopin mazurka -brief to test whether random generation Markov chains (RGMCs) can be applied to music from different composers or periods. The authors chose Chopin's mazurkas as a corpus with around 50 compositions to build an example database with enough characteristic features. In addition this enabled them to compare the results with EMI's mazurkas.

A general description of Markov models for music is included in [9], where a model consists of a state space, in this case the set of pitch classes, and a transition matrix describing the probabilities that a state is followed by another. As another demonstrative example the *Musikalisches Würfespiel* is described as a Markov model where states are bar-length musical sequences with uniform transition matrix and initial distribution. Additionally the authors cite several other examples of HMMs and n-gram models in music computing.

Collins et al. [9] define an example Markov model for melody with pitch classes that form the state space of the Markov chain, for example the state space for a piece of music with natural pitch classes plus B♭ would be

$$I = \{F, G, A, B\flat, C, D, E\}.$$

For the transition matrix for each $i, j \in I$ the number of transitions from i to j is counted and these counts are divided by the total number of transitions from state i. An initial state for the transition matrix is defined as for example

$$\boldsymbol{a} = (\frac{1}{2}, 0, \frac{1}{2}, 0, 0, 0, 0, 0, 0)$$

which gives a probability of $\frac{1}{2}$ for initial pitch classes F and A. This kind of a definition can then be used in a Markov chain to provide a sequence of possible events. The above definition is limited to the considerations of monophonic melodies. It is often a starting point of a compositional strategy where it is followed by harmonic or contrapuntal development. The final models proposed in [9] differ from this as they begin with a predominantly harmonic, or vertical, *full texture*. A state in the state space of this kind of a model consists of two elements: a beat of the bar on which a particular *minimal segment* begins, and the spacing in semitone intervals of the set of pitches that is sounding concurrently, without change. This state is referred to as a *beat/spacing state.*

The models developed use an RGMC where a random number is used to select an element of an initial distribution list. N-1 random numbers are then used to select elements of transition list L, dependent on the previous selections. The result of this is a list of state-context pairs $((i_0, c_0)), (i_1, c_1), \ldots, (i_{N-1}, c_{N-1})$, referred to as the generated output. An instance of the model used in the empirical evaluation has a state space I for a first-order Markov model containing all *beat/spacing* states found over thirty-nine Chopin mazurkas. In addition the model uses a template that consists of instructions for tempo, key signature, time signature, partition points etc.

Some shortcomings of using RGMC for stylistic composition are worth pointing out. Collins et al. discuss addressing these mostly regarding harmony in tonal music but applying principles of AGM could be another way to approach some of the issues. RGMC works well on beat-to-beat level, but does not guarantee same results at higher levels, with phrases strung together without direction or large scale structure. Another issue mentioned was a lack of awareness of the distribution of notes within a chord. There should be sensitivity to the positions of lowest- and highest-sounding notes in a chord. Finally a generated output might not convey a sense of arrival or departure. Traditionally this can be addressed by composing the end of a phrase first and merging the forwards and backwards processes which is feasible with RGMCs.

The SIACT pattern discovery algorithm [10, 6] is used to try to ensure that patterns from a template piece are inherited by the generated passage. The patterns found by SIACT are filtered and rated by a *perceptually validated formula* [8], which predicts musical importance of the patterns using observed ratings related to their quantifiable properties. The formula uses a linear combination of three factors: compactness, compression ratio and pattern's expected number of occurrences.

A template with patterns is defined in [9] to include additional information when patterns are discovered in an excerpt. For each discovered pattern $P_{i,1}$ the onset time of first and last datapoints and their translators which bring $P_{i,1}$ to the other occurrences $P_{1,1}, P_{2,1}, \ldots, P_{M,1}$ are stored. This information ensures the retaining of large scale structures in the generated output. This, main contribution of the paper, is described as the extraction and transferring of repetitive structures from an existing composition to a new one. The authors consider repetitive structures as being often hierarchical and try to transfer this to the target design as well. The system begins with the most nested repetitive element in the source design and generates material consistent with that information for the target design. This is then repeated for the next most nested element from the source design and more material is added for the target design while keeping any material that is already added in place. Generation of this local material is done using a Markov chain and continued to the completion of a large-scale structure of the target design.

Even though generally the algorithms did not achieve the level of results when compared to the output of EMI or human compositions the results showed some promise — in a few cases the output was scored better than amateur human compositions. The inclusion of repeated patterns did not improve the scores of the generated output. The authors emphasised that the experiment was informative regarding the development of music systems that take into account large scale structures of compositions and the experiments indicated that there were other factors that would need to be addressed. Another factor that might have affected the results negatively was the setup used for evaluation. The main issue being the fact that the judges had a limited time to engage with the stimulus, 1 hour for 32 excerpts of music. This had likely diminished the perception of repeated patterns.

3 Geometric pattern discovery in music

Meredith et al. introduce a family of Structure Induction Algorithms (SIA) for discovering repeated patterns in point-sets [24]. The algorithms are applied to multidimensional point-set data for the discovery of perceptually significant repeated patterns in music.

The original motivation behind the algorithms is founded on the authors' desire to develop a computational model of expert music cognition. The paper refers to numerous music psychologists and music analysts with the statement that "identifying significant repetitions in a piece of music is an essential part of achieving a rich and satisfying interpretation of it." [24]

Collins et al. [10] try to make distinctions between terms pattern 'discovery', 'extraction', 'identification' and 'mining'. They state that this distinction is not often clear in MIR. In addition they use the terms 'intra-opus' discovery, which concentrates on patterns that occur within pieces, and 'inter-opus' discovery, where patterns are discovered across multiple pieces of music. This thesis is mainly concerned with the intra-opus discovery of patterns that occur within pieces of music, and the related algorithms are referred here as 'pattern discovery algorithms'.

Several modifications and additions to the algorithms have been developed subsequently. Notably Collins et al. have developed SIACT [10] to try to address some issues related to previous algorithms and SIARCT-CFP, [7] an algorithm for detecting patterns with inexact pattern matching, with a review of the state of pattern discovery in music. Janssen has conducted a review of research on pattern discovery in music [18]. They have included multiple geometric pattern discovery methods alongside string-based methods. Meredith [23] has reviewed compression based pattern discovery algorithms for music analysis tasks.

Meredith et al. [24] list scientific and engineering applications of an algorithm for discovering repetitions in music: a component in a computational model of expert music cognition, in software tools for music analysts and composers. Meredith [21] mentions the indexing of collection of music documents for rapid searching and tools for music analysis and composition as additional use cases for this type of pattern discovery. Collins utilises SIA based geometric pattern discovery in stylistic composition algorithms and pattern discovery from audio signals [9]. Even though the research focuses on musical applications, it is stated that the algorithms could be used to the processing of *any* data that can be represented as a multidimensional dataset. This could include audio recordings, images, video and 3D-molecular models [24].

3.1 Repeated patterns in music

A pattern discovery algorithm could be used for discovering characteristic structural features in the works of a composer, for analysing the structure of a work into subcomponents, or an unfinished work could be processed to discover repeated structures and gain a new perspective to further progress the work. The identification of perceptually significant repetitions is thought to be an essential step in the process by which musical work is interpreted by an expert listener. [24] Both Cope's EMI [12] and Cambouropoulos' GCTMS [2] use pattern detection as a step in their computational models of music. In addition Collins et al. attempt to incorporate pattern discovery in their model of stylistic composition [9].

Pattern discovery with SIA discovers all exact repeating patterns in a dataset and most of these exact repetitions in music are not perceptually significant. There is a need to formally characterise what distinguishes these interesting repetitions from repetitions that a listener does not notice or that are not considered important by an analyst. There is also a significant amount of diversity in perceptually significant repetition [24]. Patterns involved in repetitions vary widely in their structural characteristics and a pattern can be modified in many ways to give other patterns that are perceived to be versions of it. A pattern may be a small few note long motif, or a whole section of a work. A pattern in polyphonic music with unambiguously identifiable voices may have notes from one or any number of the voices. Occurrences of a pattern may overlap in time, patterns may occur consecutively or they may be widely separated in time. A variety of transformations that can be used to create a new pattern out of another pattern is presented – a pattern may be *truncated*, *augmented*, *diminished*, *inverted*, *reversed* and *embellished.*

3.2 Representation of music data

Music needs to be represented in some discrete format for the use of algorithms. Most importantly the representation of music as events describing notes rather than audio recordings of music is the focus of most, if not all previous work regarding this thesis. Typically in musicology western music notation is used. Music notation can be represented digitally by various music software and formats such as MusicXML and then be parsed to be input to algorithms. An alternative format for the input of music data could be other digital formats, such as MIDI-files containing note information. For this thesis *music21*, a toolkit for computer aided musicology and symbolic music data, was used to parse music corpuses in various formats [13]. The parsed data of a musical work was stored in a list containing numeric representations of pitches and onset times of notes. In addition note lengths and information regarding voice and dynamics of the notes could be used, as the algorithms are applicable to multidimensional point-set data, but they were not yet considered relevant for the inspected algorithms.

In addition to this *point-set* representation of music other approaches have previously been used. String matching algorithms, that operate on symbolic sequences of musical data, have for example been used in repetitive pattern discovery algorithms but they have issues especially with polyphonic music. Meredith et al. [24] state that such algorithms cannot deal with unvoiced polyphonic music, such as keyboard music, and that string-matching also causes problems in finding patterns which are distributed between several voices, or transposed occurrences of patterns and patterns with gaps.

Multiple Viewpoints by Conklin and Witten [11] are another popular form of music representation in computer aided musicology. Multiple independent views of musical surface, or parameters for each note, are stored as viewpoints and music is represented with sequences of these viewpoints. They may contain different interval profiles, dynamic, timbral and other contextual information. This approach is mainly applied to monophonic music, or monophonic parts extracted from larger musical works. Models of musical style by both Pearce et al. [26] and Collins et al. [9] employ this approach. In addition Collins et al. have combined this with information of patterns extracted from point-set data.

3.3 Geometric representation

Meredith et al. [24] define concepts and formalisms for geometric representation of *multidimensional datasets* and *patterns* which will be used throughout the rest of this thesis. A *vector* is a k-tuple of real valued numbers. It is a member of a k-dimensional Euclidean space, represented as an ordered set of k real numbers. A *k-dimensional vector* is a k-tuple of real numbers. A *k-dimensional vector set* is a set of vectors in which every vector has the cardinality k. A *datapoint* is a vector in a pattern or a dataset. The term *dataset* is usually reserved for a k-dimensional vector set that represents a complete set of data that is being processed. The term *pattern* is reserved for a k-dimensional vector set that is a subset of a dataset or a transformation of a subset of a dataset. When searching for occurrences of vector set P in vector set D, the set P is referred as a pattern and the set D as a dataset. Patterns P_1 and P_2 are defined to be *translationally equivalent* if an only if there exists a vector $\boldsymbol{v}$ such that translating P_1 by $\boldsymbol{v}$ gives P_2. A multidimensional dataset can be orthogonally projected to give another multidimensional dataset.

Music can be represented in many appropriate ways as a multidimensional dataset. A tiny 5-dimensional example set of three notes could be defined as follows:

$$\begin{gathered}\{< 0, 27, 16, 2, 2 >,\\ < 2, 44, 26, 1, 1 >,\\ < 4, 47, 28, 1, 1 >\}\end{gathered}$$

First element is the onset time of a note as the number of semiquavers that have elapsed by the time the note occurs. Second element represents the *chromatic pitch* of the note. It can be described as a numerical representation of the key on a piano keyboard that when pressed plays the note. The lowest note would normally be $A\natural_0$ which is defined to be 0. The note semitone above $A\natural_0$ is $B\flat_0$ and is therefore 1. The chromatic pitch of middle C ($C\natural_4$) is 39. Another way to represent the pitch of the note would be *morphetic pitch* [22] which indicates the position of the notehead of the note on the staff. The morphetic pitch of $A\natural_0$ is defined to be 0. The morphetic pitch of middle C ($C\natural_4$) is 23, D above middle is 24 and so on. This is included as the third element. A fourth element represents the duration of the note measured in semiquavers and the fifth element represents the voice in which the note occurs. Meredith suggests [23] using morphetic pitch for modal and major-minor tonal music and chromatic pitch for pieces not using a modal or tonal system. Instead of morphetic pitch, a similar pitch representation, diatonic note number, provided in the *music21* toolkit [13] was used in this thesis. It identifies the diatonic version of a note ignoring accidentals. In there $C\natural_0$ is defined to be 0, $G\natural_0 = 5$, $C\natural_1 = 8$ and so forth. Notes lower than $C\natural_0$ are represented with negative numbers. The different pitch representations of a dataset are considered to be different projections of the dataset. The research on geometric pattern discovery often focuses on discovering repeated patterns in 'piano-roll' type representations such as MIDI data, in which pitches are represented with chromatic pitch numbers.

As AGM was not thought to be dependant on existing music theoretical frameworks, chromatic pitch representations were used with it's implementation and in the inspection of specific patterns and their vectors. For the selection of perceptually significant patterns diatonic pitch representations were additionally used as most of the datasets for pattern discovery consisted of tonal or modal music.

3.4 Structure Induction Algorithms

SIA discovers maximal repeated patterns by taking a multidimensional dataset as input and finding the patterns in the dataset that when translated by all possible vectors give other patterns in the dataset. A pattern is *maximal translatable*, or an MTP, for a vector if it is the largest pattern that can be translatable by it to give another pattern in the dataset. SIA discovers all non-empty MTPs in the dataset. [24, 21]

The algorithm operates in the following manner. First it sorts the dataset D and constructs a *vector table* W from the sorted dataset. The heads of both the rows and the columns contain the datapoints of D. A cell in the vector table contains the vector from the datapoint at the head of the column of that cell to the datapoint at the head of the row of the cell. For V all the values in the vector table below the leading diagonal are computed, which are all the vectors from a datapoint to every other datapoint greater than it. In addition the items in V contain a pointer to the vector's origin data point in D. The vectors in vector table are sorted, the original implementation [24] uses a slightly modified merge sort that benefits from having the columns in the input table already sorted. The complete set of non-empty maximally translatable patterns can be obtained by scanning the list containing V once, reading off datapoints and starting a new pattern when the vector changes. The most expensive step is sorting vectors $O(kn^2 log_2 n)$ for a k-dimensional dataset of size n, with the space complexity of $O(kn^2)$. An implementation of SIA was developed for this thesis using array operations from the *numpy*-library for Python. The vector table V in this implementation is computed from the whole vector table W. Code listings of the implementations of both SIA and SIATEC are included in appendix A.

SIATEC finds for all translationally equivalent patterns their instances and the associated translation vectors. TEC stands for 'translationally equivalent class'. First a modified version of SIA generates all the MTPs and then the SIATEC algorithm finds all the occurrences of each MTP. The basic implementation of SIA needs to only compute the vectors below the leading diagonal — the MTP of $-\boldsymbol{v}$ is the same as translating the MTP by $\boldsymbol{v}$. Finding all the occurrences of any pattern within a dataset is more efficient when the whole vector table W is computed.

The dataset is sorted so that vectors increase when descending a column and decrease when moving left to right along a row. A column in the vector table contains all the vectors so that the datapoint at top can be translated by them to give another point in the dataset. Finding all the occurrences of a pattern means finding all the vectors that the pattern is translatable by. The worst-case time complexity of SIATEC is $O(kn^3)$ for a k-dimensional dataset of size n.

In addition to the intra-opus pattern discovery algorithms these techniques can be used for pattern matching. That is to say finding a multidimensional query pattern from a multidimensional dataset. The SIAMESE pattern matching algorithm [24] works essentially in the same way as SIA. The points in the query pattern and the points in the dataset are sorted and a vector table is constructed. Each entry in the vector table gives the vector from the query datapoint at the head of the column to the dataset point at the head of the row, with a pointer back to the pattern datapoint. All the vectors are sorted to a list which gives all the vectors that the query pattern can be translated by to give a non-empty match in the dataset.

3.5 Perceptually significant repetitions

Experiments [24] suggest that many of the perceptually significant repeating patterns are MTPs discovered by SIA and SIATEC. However, SIATEC may typically discover tens of thousands of TECs even in relatively short pieces of music. Only a very small proportion of these TECs are perceptually significant or analytically interesting. Rachmaninoff Prelude, Op.3 No.2 contains 70000 MTPs and probably less than 100 of them would be considered interesting by an analyst [21]. The power set of a dataset D contains $2^{|D|}$ different patterns and the number of MTPs generated by SIA for a dataset D is less than $\frac{|D|^2}{2}$. [24].

There are several ways in which repeated musical structures might be perceptually significant or analytically interesting. There exists different "structural features that a repeated musical pattern might be able to tell us something about." [24] One example is to find theme-like and motif-like patterns in a passage. Meredith et al. suggest using different kinds of heuristics and rules to isolate perceptually significant repetitions from the rest, stating that there might not be a single set of rules capable of isolating all and only interesting repetitions. Various different algorithms have been developed for this task.

3.5.1 Heuristics

Different heuristics that could be used to measure the significance of patterns have been suggested [24, 21]. Meredith [21] explores three heuristics for isolating the themes and motives in a piece of music: *coverage*, *compactness* and *compression ratio*.

The coverage of a pattern is the number of datapoints in the dataset that are members of occurrences of pattern. Coverage is generally greater for patterns whose occurrences overlap less, larger patterns and those that occur more often. "In general it seems theme-like and motif-like patterns have relatively high coverage". [21] Compactness is the ratio of the number of points in the pattern to the total number of points in the dataset within a region spanned by the pattern. The region can be defined in different ways: smallest time segment that contains the pattern, bounding box or convex hull of the pitch-onset-time graph. Typically at least one occurrence of a theme-like pattern will have high compactness value. For isolation of theme-like patterns the third interesting heuristic is the compression ratio. It is defined as the ratio of the set of all the points covered by all the occurrences of a pattern by specifying one occurrence of the pattern with all the non-zero vectors by which the pattern is translatable within the dataset.

Using previously described heuristics in conjunction with SIATEC a compressed or efficient representation of a dataset can be generated [21]. The COSIATEC algorithm takes a dataset as input and generates a list of TECs that cover the input dataset without any overlap. First SIATEC is run on the dataset and it generates a list of <pattern, translator set> pairs. The heuristics compression ratio, coverage and compactness are used to choose the 'best' pattern P. P is output together with its translator set. All the points covered by the P-set are removed and if the dataset is empty, COSIATEC terminates. If points remain, SIATEC is run again on the remaining dataset. The results are the best pattern and its translators for each iteration. The degree of compression directly depends on the amount of repetition in the dataset. The results in some cases resemble thematic and motivic analyses carried out by music analysts [21].

In Modelling Pattern Importance in Chopin's Mazurkas, Collins et al. [8] introduce characteristics of a musical pattern with subjective assessment of a pattern's salience. Their objective was to identify how to order the output of a pattern discovery system. This would allow the discarding of uninteresting patterns. They set out to empirically validate known and novel concepts and formulae that have been used to address this problem. In an empirical study music undergraduates examined excerpts taken from Chopin's Mazurkas and rated discovered patterns giving high ratings to patterns that they thought were noticeable or important. Half of the patterns that were used in the review were hand picked by the authors and half were a random selection of the output of the SIATEC algorithm. The authors used linear regression models for rating discovered patterns in music with variables that were included by forward selection and backward elimination. The resulting model included parameters known from the COSIATEC-algorithm, with the formula for the final pattern rating being:

$$\begin{aligned} rating = 4.28 &+ 3.42 \cdot compactness \\ &- 0.04 \cdot expected\_occurrences \\ &+ 0.65 \cdot compression\_ratio \end{aligned}$$

3.6 Further developments on the geometric pattern discovery algorithms

Collins et al. have developed modified versions of the translational pattern discovery algorithms [10, 6, 7]. They formulate the problem as discovering translational patterns from a given piece of music in a semi-symbolic representation, noting that these are not the only type of pattern that could matter in music analysis, but that music analysts do acknowledge that such discovery task forms part of the preparation when writing an analytic essay. Their motivation behind studying translational patterns is the prospect of automating the aforementioned pattern discovery task, and they consider the problem to be an open problem in MIR. Collins' et al. initial work comprises of an evaluation of different pattern discovery algorithms applied to Baroque keyboard works. [10] In addition they contribute modified versions of the SIA-based geometric pattern discovery algorithms.

Forth and Wiggins [15] have developed a non-parametric version of COSIATEC which requires only one run of SIATEC. This reduces the computational complexity, but means that the output is always a subset of the set of maximal patterns F. The original version of running SIATEC successively makes it possible to discover patterns further in $G \setminus F$, where

$$G = TEC(P_0, D_0), \dots, TEC(P_N, D_N).$$

which is the output of COSIATEC.

3.7 Issues with Structure Induction Algorithms

Collins et al. [10] describe the 'problem of isolated membership' where "a musically important pattern is contained within an MTP along with other datapoints that might not be musically important." They propose the following solution to this problem: First SIA is run, then the inside of each MTP of it's output is 'trawled' and the subsets that contain at least b points and have a *compactness* greater than some threshold a are returned. Compactness is the ratio of number of points in a pattern to the number of points in the region of the dataset the pattern spans.

The region can be defined in different ways which affects the compactness. Collins et al. employ a lexicographically ordered scheme, where points are ordered based on pitch first and onset time second. This has a low computational complexity of $O(kn)$.

They define the *compactness* of a pattern P in a dataset D, where $P = p_1, ..., P_l$ as

$$c(P, D) = l / |d_i \in D : p_1 \preceq d_i \preceq p_l$$

Alternatives to the *trawling* scheme are presented [10]. First is Meredith et al.'s [24] suggestions for improving or extending the SIA family with developing an algorithm that searches the MTP TECs generated by SIATEC and then selects all and only TECs that contain 'convex-hull compact patterns'. Other options include segmenting the dataset first. Usually the pattern discovery guides segmentation and it is not clear how to do it in the opposite order. Usage of a 'sliding window' with SIA is suggested, but this may prevent the discovery of important patterns if their generating vectors are outside the window. Finally they "consider the set of all patterns that can be expressed as an *intersection* of MTPs." This would not be susceptible to isolated membership problem, but is more computationally complex.

The *compactness trawler* by Collins' et al. [10] is defined as follows:

1. $P = \{p_1, \dots, p_l\}$ is a pattern in a dataset D and $i = 1$.
2. Let j be the smallest integer such that $i \leq j \leq l$ and $c(P_{j+1}, D)$ where $P_{j+1} = p_i, ..., p_{j+1}$.
3. If such integer does not exist then set $P' = P$, else set $P' = p_i, ..., p_j$.
4. Return P' if it contains at least b points, else discard it.
5. If j exists in step 2, set P in step 1 to equal $p_{j+1}, ..., p_l$, set $i = j + 1$, and repeat steps 2 and 3. Otherwise set P as empty.
6. After a number of iterations P will be empty and the output can be labelled $P'_1, ..., P'_N$, which are N subsets of the original P, where $0 \leq N \leq l$.

The complete algorithm is called 'structural inference algorithm and compactness trawler' (SIACT). In it MTPs are first calculated with SIA and the compactness trawler is applied to each MTP. The compactness trawling of the algorithm requires $O(kmn)$ calculations where m is the number of MTPs returned by SIA.

A comparative evaluation of SIA, COSIATEC and SIACT is presented [10] with a dataset consisting of baroque keyboard music analysed by a music analyst, the second author of the paper, with a similar brief to the intra-opus discovery task: discovering translational patterns that occur within a piece.

Meredith criticises SIACT [23] stating that when scanning a pattern P from different directions, left or right, different points may or may not be considered 'isolated'. He demonstrates that in some cases using the original SIA more complete patterns are found than with SIACT.

Another pattern discovery algorithm by Collins et al. [7] aims to solve an *inexactness problem* and a *precision problem* of point-set pattern matching with a novel algorithm called SIARCT-CFP. [7] As the first problem of the point-set approach they describe that the algorithms extend only to a limited number of inexact repetition types. The exact pattern matching algorithms for example can not account patterns with rhythmic variation. Poor precision means that the algorithms return many false positives, numerous patterns that are not significant in any way. The inexactness is addressed with a *fingerprinting* algorithm. If the output of a pattern discovery algorithm finds at least one exact repetition, the fingerprinting algorithm is run to find inexact occurrences of this pattern. The precision is addressed by a categorisation process where similar patterns are grouped together and each group is represented with one exemplar pattern.

In Analysing Music with Point-Set Compression Algorithms [23] Meredith reviews several point-set pattern-discovery and compression algorithms.

Regarding the focus on compression algorithms Meredith explains that in this case music analysis is inspected with a “hypothesis that the best analyses of a piece correspond to the shortest possible descriptions of it.” The algorithms employ different strategies aimed at selecting a set of MTP TECs that collectively cover, or almost cover, the input dataset in a way that maximizes compression. Many known SIA-derived algorithms are included. They are evaluated on two musicological tasks: classifying folk song melodies into tune families and discovering repeated themes and sections in pieces of classical music. The COSIATEC algorithm, which achieved the best compression in general, was also found to be the best-performing algorithm on the first task and achieved the second-best score overall on the latter task.

In conclusion there have been multiple different variations in the SIA-family of pattern discovery algorithms. Different approaches to selecting or rating interesting patterns have been made. They have mostly focused on creating structural descriptions of musical works in a manner that would be similar to how an experienced listener or music analyst would hear or organise music. Many empirical evaluations and applications to finding interesting patterns in musical works have been done. As a different practical example Collins et al. [9] have incorporated pattern discovery to the task of modelling musical style. One of the main problems, the determination of which patterns are perceptually significant, or important, remains an open one as there exists more than one way to think about how people hear and understand music.

4 Analytic Generative Methodology (AGM)

In Analytic Generative Methodology (AGM) Timo Laiho [19] presents a new methodology for music analysis. AGM is presented as a *perception based* methodology and is set to address the lack of the role of perception in traditional music analysis. As one of the main components of AGM Laiho introduces an analytic concept of Musical vector (*muV*). They can be described as a two dimensional graphical representation of the relationship of pitch occurrences (y-axis) in time (x-axis). This relationship is further explained as a unit of difference, or the *differential* between successive notes, which Laiho calls Interval-time Complex, or *intiC*. Laiho describes the concepts of *intiC* and *muV* as a new kind of analytic dimension which do not refer to isolated static elements but describe the *temporal relations* of pitch occurrences. These concepts are founded on a music analytic theory and a cognitive model of human perception and sensation which Laiho develops in his thesis. Laiho’s theory is founded upon existing work and models in musicology, cognitive science, philosophy and David Bohm’s work on *implicate order* and views on physics. Laiho contrasts his theory to the currently dominant structural music analytic methods, namely pitch-class set theory and Schenkerian analysis, but relates it to Schenker's concept of *prolongation* and Adorno's *directionality* and *becoming*, and makes a connection to the *temporal* development of musical structures in Schenkerian analysis.

A principal component of AGM, the Musical vector (muV) can be defined as a vector between two pitches, or points of point-set data representing pitches. Therefore it is directly related to translation vectors used in Structure Induction Algorithms and can be computed using similar methods. In addition the analytic properties of muVs can be used to examine perceptually significant repetitions found using the algorithms in the SIA family.

In relation to this thesis and the computation of perception based aspects of point-set data, certain parts of Laiho's thesis warrant being brought out. Laiho coins a concept of *perception/sensation* that combines both the sensing of information about the world and turning it into neural signals, with perception which deals with the interpretation of those signals. Laiho evaluates music analytical concepts through cognitive science, with the view of human perception and cognition as bottom-up and top-down processes and their interaction. Top-down processes are described as axioms and syntactic rules of formal languages, which represent high-level organisation and can be used to generate sentences, or musical phrases, containing items of low-level structures. These for example include the Schenkerian concept of prolongation. Laiho assumes that with this approach it remains unclear if the generated sentences have any actual relevance to perception/sensation. The higher-level concepts are in addition described to be capable of in a reductive manner *decoding* the "state of affairs" of low-level properties, but this is stated to leave out essential time-dependent contextual factors evident in perception/sensation. In the opposite direction the bottom-up processes start from low-level sensory information which can be reducted to compound higher-level structures. This is associated with connectionism in cognitive science and the computational models of artificial neural networks.

The division of higher level structures and low-level sensory information is found in concepts of high and low level processing in musicology for example in differentiation between psychoacoustics and music theory and "primitive" and "Schema based" streaming in Bregman's Auditory Scene Analysis [1]. Bregman considers the concept of "auditory stream", instead of a "sound-object" to be a perceptual *unit* representing a single (precursory) auditory event. Laiho connects this time-dependent concept to the analytic concepts of AGM.

The main criticism of music analysis methodologies such as Schenkerian pitch-class set theory are directed towards non-temporal assumptions of "higher-order" reduction and it's relationship to the reductive principles of a strict, hierarchical axiomatic system. Although AGM is said not to intend to abandon structural hierarchy and principles of reduction in the context of cognitive processing. On the problem of grouping, or segmentation, Laiho quotes Lerdahl [20]: “the hierarchical structure is known to be central to learning and memory, both in the general case and for music in particular” and that “the psychological evidence points to the central role of hierarchies in cognition” adding that this does not imply the direct application of a strictly ruled, formal axiomatic structure in analysis.

Some critique towards artificial intelligence and computational approaches to human cognition is presented, but it is worth to note that this is mostly aimed towards approaches and models of which most date latest to 1990s. The issues are regarding the earlier focus on high-level cognitive processes with methods such as predicate logic, semantic networks, the lack of methods on the lower level of sensing and perception and the gap between low and high level processing. In recent years research in machine learning, statistical modelling, and in deep neural networks, with the application of methods similar to and simulating human cognition and perception, has turned the focus on computational research to more low level processing of information and brought forward tools that may bridge gaps in the understanding of sensory and perceptual cognition.

The above criticisms boil down to two points regarding an answer to the question of what kinds of mechanisms constitute the possibility of musical hearing: time-dependent perceptual/sensational factors are necessary for the ability of structural musical hearing and these factors cannot depend on computation of *discrete* entities.

Laiho illustrates another problem regarding high and low levels of cognitive processing and a “meaning barrier” between them with The Chinese Room argument of Searle. In The Chinese room -argument the “closure” of an inner structure in cognitive processes is something that prevents “understanding” of the Chinese stories. Laiho claims that any kind of *boundary condition* inherent in formal languages inevitably obstructs the two-way *interaction* of the inside-outside processes of cognition and prevents *meaning* production and understanding. He argues that in music analysis the analytic descriptions must take into account time dependant perception/sensation properties. AGM aims to overcome this “meaning barrier” by considering the *interactivity* of *inside* and *outside structures* of a system showing that the structurally “open” organisation (of language) functions as a basic cognitive principle which is essentially related to perception/sensation capacities. The structural basis of AGM is formed with this interactive model of cognitive processing.

Finally with a third concept, *milieu-territorial assemblages*, Laiho tackles the problems of higher order structural organisation of music while retaining the connection to the context and temporally dependant concepts of perception/sensation. Laiho organizes *muV*s into what he calls milieu and territorial assemblages which correspond to higher level structures and organisational concepts of music analysis. These assemblages are however not strict or concrete structures with well-defined borders but relations which build upon each other in a recursive manner. This could be thought of as all the different perceptible combinations of sets of musical vectors.

In the computational models of music and related algorithms reviewed in this thesis the separation of high and low-level structures and processes varies. In the most strict sense an axiomatic structural analysis would be represented by the structures based on pattern discovery with algorithms such as COSIATEC. Meredith [23] has evaluated the use of these kinds of algorithms in music analytic tasks. On the other end of the spectrum Pearce [26] approaches the problem of building a model of musical style that is based on hypotheses of cognitive processing. This statistical model is trained with a corpus of melodies from which, in a bottom-up manner, it models cognitive learning processes. The work of Cambouropoulos [2, 3] combines the results of pattern discovery with the discovery of local pitch based phenomena to infer grouping structures. This hints to the usage of interactivity of bottom-up and top-down processes in an algorithmic system. Similarly the model of Collins et al. [9] combines low-level learning with information from a pattern discovery algorithm which could be thought of as higher-level information.

Laiho has tried to answer the question of what is the universal perceptual factor of hearing, listening and understanding music that can be analytically defined. According to Laiho this is the time and context based perception of the movement i.e. velocity of pitches and other musical variables such as dynamics or timbre.

4.1 Interval-time complex (*intiC*)

The *interval-time complex* (*intiC*) is introduced as a contextually and temporally dependant relationship between two pitch occurrences. It can be for example used to describe dramaturgic intentions of music based on *qualitative differentiation*.

Laiho describes using *intiC* in contrasting qualitative differentiation, stating that music and art analysis that focuses in the structure of the differential movement using the analytic concept of *intiC* brings forth conceptual and temporal factors which underlie the perceptual/sensational organisation of a work of music. Such an analysis provides a dynamic profile and hints at features of differentiation from the point of view of musical interpretation and performance. Algorithms that can be used to conduct *intiC* and *muV* analyses on music score sheets were developed for this thesis.

Calculating *intiC*-values for a set of <time, pitch> -pairs in a dataset D can be done as follows:

$$\begin{aligned} n &= |D| \\ delta &= (D[1] - D[0]) \ldots (D[n] - D[n-1]) \\ intics &= delta[p_i]/delta[t_i] \mid i \in (0 \ldots n), \end{aligned}$$

where $D = (t_0, p_0) \ldots (t_n, p_n)$ and t_i are the onset times of pitches p_i.

The values can then be plotted on a graph where x-axis corresponds to time, and y-axis to *intiC*-values as shown in figure 1. On the x-axis the values are plotted based on the datapoints of D starting from the second point, which is the time step where we have our first *intiC*-value.

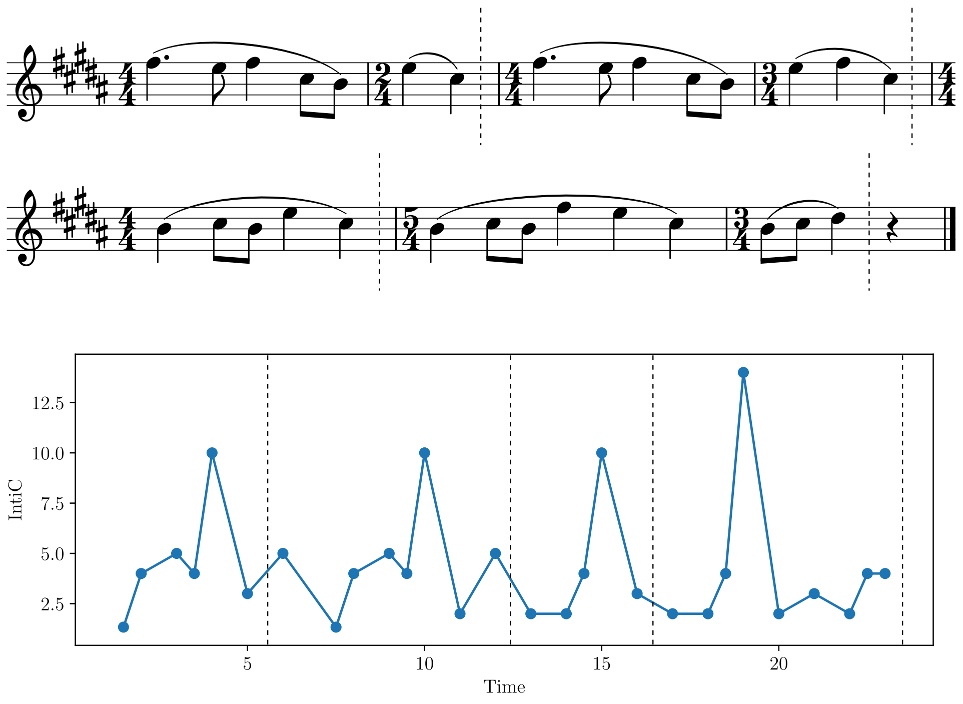


Figure 1: Example of Interval-Time Complexes of solo violin from Stravinsky's Rite of Spring m. 91-92

Calculating the *intiC* values this way gives the value 0 for repeating pitches. For simultaneous pitches, such as the pitches of a chord, the values approach infinity. This is one reason why for visualisation and music analytic purposes the *intiC* values computed this way are mainly suited to be used with monophonic music data.

IntiC-based analyses using software developed for this thesis have been used by students at the department of musicology at University of Helsinki to examine the differing dynamics of for example sung popular music in comparison to what is printed on sheet music of Olavi Virta, and for studying the rhythmic dynamics of jazz-performances of Gil Evans.

Smoothing is used to bring out more dominant changes in *intiC* values by clearing out more arbitrary variation. Laiho uses weighed averages of neighbouring values (0.3 preceding value, 0.5 current and 0.3 following) as the smoothing formula. For the software implementation of this thesis the smoothing is done by using a convolution function over a window w:

$$w = [0.3, 0.5, 0.2]$$
$$smooth = convolve(w/sum(w), intics)$$

which gives us weighted averages over the previous, current and next *intiC* values. This can be done repeatedly to give a series of more averaged out *intiC*-curves which can be used to visually bring out the largest changes of the values.

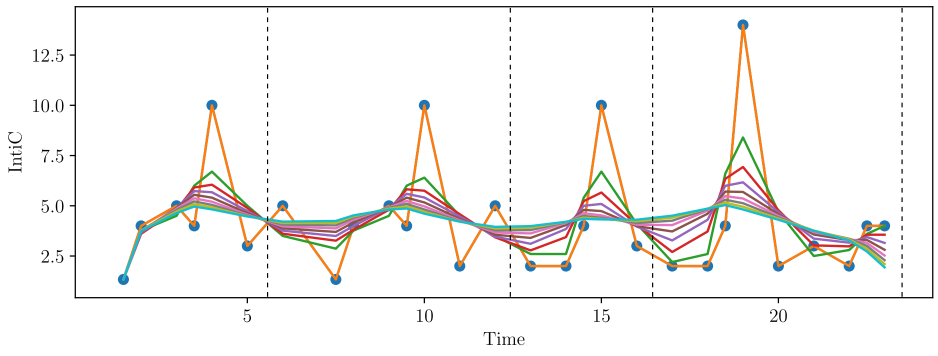


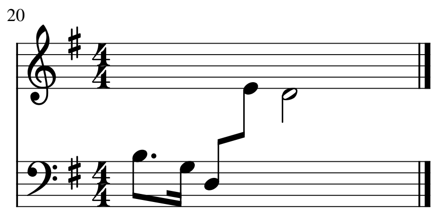
Figure 2: Nine iterations of smoothed Interval Time Complexes from figure 1

Different iterations of these *intiC*-values could be useful for example for segmentation or boundary detection for monophonic music. A segment boundary could be seen to be found at a local minimum or maximum of the *intiC*-curve. This is somewhat similar to the approach of Cambouropoulos [2, 4] where a Local Boundary Detection Model (LBDM) includes rules regarding where a local grouping boundary exists. These rules that affect *boundary strength* include the degree of change between two consecutive intervals using proximity in both temporal and interval domain.

In analysing musical grouping processes, or segmentation, Cambouropoulos uses structural repetition and similarity in combination with these local *Gestalt*-based factors that are used to identify points of local maximal change in various musical parameters, including pitch intervals, dynamic changes and so on [4]. The *intiC* values could be calculated for dynamic or timbral changes as well as any temporally variable information. Cambouropoulos also refers to the usage of memory-based model [14] for melodic segmentation where Markov models are used for acquiring melodic regularities with the assumption that segmentation boundaries occur more likely close to accentuated changes in entropy. This could be thought of as an option to combine these low-level properties with higher-level concepts.

4.2 Musical vectors (*muV*s)

Musical vectors, or *muV*s, are another way to conceptualise the time dependant relationship of pitches. The *intiC* can be thought of as the differential of the *muV* — a vector between two pitch points. As mentioned previously a subset of the *muVs* can be graphically presented as vectors between adjacent points that represent pitch onsets. When only these vectors are taken into consideration we get a pitch contour that takes into account the temporal relationship between the pitch occurrences. In figure 3 which is an adaptation of an example in AGM [19] of counter-subject of Mahler's Fourth Symphony (m. 20), the pitch contour is shown in blue. We can get the set of vectors for the pitch contour by calculating $V_{contour} = \{p_{i+1} - p_i | i \in |D| - 1\}$, where D is a temporally sorted point-set. *MuV*s include a much larger set of vectors than just those between adjacent points. The potential *muV*s that can signify possible connections towards points further in the dataset, or that suggest the positions of likely new points that could be inserted to the set are shown in green. The vector shown in red is the vector sum that describes the whole pitch contour of the counter-subject.



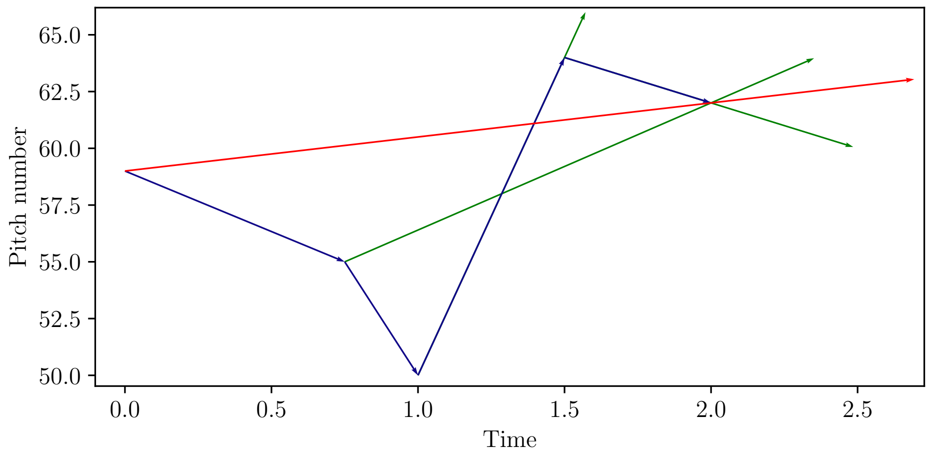


Figure 3: Example of potential musical vectors for a counter-subject of Mahler's Fourth Symphony (m. 20) [19] p. 37

A basic definition of the *muV* set is as follows. Given a set of points D, for each point $p_i \in D$ we have points $p_j \in D$, where $j \in i+1, ..., n$ and $n = |D|$, that occur after p_i, the rest of the dataset. For each of these points we can calculate a vector $\boldsymbol{v}$. These are all the *potential muV*s that can be found.

As mentioned earlier, *muV*s are a subset of the set of all the vectors of a vector table used in Structure Induction Algorithms [24]. Specifically, as we are only interested in vectors with positive values that correspond to the x-axis i.e. time, the set of all the potential *muV*s is equivalent to the vector table V which contains the lower diagonal of the vector table W. The size of this set of all potential musical vectors illustrates the problems that arise when comparing every point of the dataset with each other. There are issues with both the time complexity with performing computations on the vectors, and with the size of the resulting outputs of the algorithms. As one solution *perceptually interesting muV*s could be computed related to concepts of heuristics described previously.

Some heuristics can be used based on the definitions Laiho has outlined. A core concept is the detection of the directionality of movement i.e. the *velocity.* The *muV*s are a description based on temporally oriented diagonal relationships, which can also be thought of as velocities. The differences of the *muV*s can be calculated, and the *muV*s can be organised based on the changes of the velocities. A related attribute we can use is *invariance*: consider a point $x \in D$, all the vectors from x to points that are in the same or similar enough direction describe the *invariance of movement.* The vectors can be thought of as being *extended* according to this heuristic. Essentially we might only take into consideration the vector sum of all adjacent vectors with the same direction as was illustrated in figure 3.

A simple definition can be made based on above properties. A vector $\boldsymbol{a}$ is a *musical vector* if and only if there exists another vector $\boldsymbol{b}$ with the same starting point A and an angle $\theta \leq t$ between them, where t is some threshold value. For all musical vectors with the same same starting point A within an angle $\theta \leq t$, the *descriptive musical vector* is the vector with the largest magnitude. If there exists a vector $\boldsymbol{a}$ from previous point A to point B within an angle $\theta \leq t$ of vector $\boldsymbol{b}$ from point B to a following point C, a descriptive vector $\boldsymbol{c}$ from A to C is the descriptive vector of the points A, B and C.

In addition Laiho suggests ideas that could be used as different heuristics: continuous acceleration, continuous movement and *muV*s based on other temporally organised data besides pitch.

4.2.1 Computing musical vectors

Using vector tables we can compute all musical vectors in different ways. In algorithm 4.1 using vector table W we can find for each point p_i, where

$i \in 0 \ldots n-1$, the musical vectors by comparing the angle for each vector $\boldsymbol{v_j}$ in column i below the diagonal with each other. The comparison can be done using cosine distance, where each pair of vectors with $cosine(\boldsymbol{v_j}, \boldsymbol{v_k})$ below a threshold is considered to contain a musical vector. The cosine of two vectors with the same angle has a distance of 0. $Cosine(\boldsymbol{v_j}, \boldsymbol{v_k})$ is defined as

$$1 - \frac{\boldsymbol{v_j} \cdot \boldsymbol{v_k}}{||\boldsymbol{v_j}||\ ||\boldsymbol{v_k}||}$$

In theory an ideal musical vector will consist of two vectors that have an identical angle, but in practice for example due to granularity of the point-set notation data, or any other data where the relation of the x- and y-axis is not completely linearly related, some difference in the angles should be tolerated. A value of 0.0001 was used in the examples of this thesis.

For music analysis and visualisation uses we usually only consider the vector with the largest magnitude of the pair, but both vectors of can be considered and used for further computations. In algorithm 4.2 the largest vectors are computed for each point by traversing the column of the vector table W starting from the largest vectors and ending the computation when a match is found, thus slightly reducing the amount of comparisons required. Alternatively we could find for each point p_i all the incoming and outgoing vectors of the point and calculate the difference in the angle of the vectors. The incoming vectors used are equivalent to the inverses of the vector backwards from the point.

Algorithm 4.1: Finding all potential muVs

```
input: W, threshold ← 0.0001:
output: muv
begin:
    $n \leftarrow |W|$
    $muv \leftarrow \{\ \}$
    for $i \leftarrow 1, n-1$:
        for $j \leftarrow i+1, n$:
            for $k \leftarrow 0, i$:
                $v_{bw} = W[i][k] * -1$
                $v_{fw} = W[i][k]$
                if cosine($v_{bw}, v_{fw}$) $\leq$ threshold:
                    if $i$ not in $muv$:
                        $muv[i] \leftarrow$ set()
                    muv[i].add(($v_{fw}, i$))
                    if $k$ not in $muv$:
                        $muv[k] \leftarrow$ set()
                    muv[k].add(($v_{bw}, k$))
    return muv
end
```

Algorithm 4.2: Finding descriptive muVs

```
input: W, D, threshold ← 0.0001:
output: muv
begin:
    $n \leftarrow |W|$
    $muv \leftarrow \{\ \}$
    for $i \leftarrow 0, n-1$:
        $V_i \leftarrow [\ ]$
        for $j \leftarrow n-1, i+1$:
            $v \leftarrow nil$
            for $k \leftarrow n-1, i$:
                if $k! = j$:
                    if cosine($W[i][j], W[i][k]$) < threshold:
                        if j > k:
                            $v \leftarrow W[i][j]$
                        break
            if $v$ is not $nil$:
                redundant ← False
                for $k \leftarrow 0, i$:
                    if cosine($W[i][j], W[i][k] * -1$) < threshold:
                        redundant ← True
                        break
                if not redundant:
                    $V_i$.append($v$)
        if $i$ not in $muv$:
            $muv[i] \leftarrow$ set()
        foreach $v \in Vi$:
            $muv[i]$.add(($v[0], v[1]$))
    return $muv$
end
```

Computing vectors for smaller point-sets, i.e. short segments of music, generally works and results in vector presentations similar to what is defined in AGM. These could be directly usable as components of music analysts work, or as generative tools used by a composer as mentioned in [19]. Larger point-sets, such as longer polyphonic musical works tend to exhibit problems with the combinatorial explosion of all point to point vectors in the muV set and the tolerance for deviation with large sum-vectors mentioned earlier. Calculating all the muVs with algorithm 4.1 and descriptive muVs in algorithm 4.2 is computationally very expensive. For each point in the dataset we compare each vector pointing to each further point with each other meaning that the time complexity for algorithm 4.1 is in all cases $O(n^3)$ and for algorithm 4.2 it is as well at least the average case.

In addition there are issues with accuracy when comparing the angles of the smallest and largest vectors. The differences between the angles of small vectors compared to the difference of the angles of a small vector and a large vector might be bigger even if the large vector ends up deviating from the small vectors more significantly.

Revised versions of the muV algorithm were developed to cut down the computational cost and the large amount of vectors that it tended to result.

The final algorithm was divided into two parts. First algorithm 4.3 finds all the smallest musical vectors using W similarly to algorithm 4.2, only the smallest vector is stored for each set of vectors that have the same direction. This allows the reduction of the amount of comparisons needed to make as W is sorted and the size of vectors with the same direction increases as the algorithm iterates through the table. In addition we can set a window within which we limit the computations. The size of this window, *max_span* is defined as a time span in quarter notes.

Algorithm 4.3: Shortest muVs

```
input: W, D, tolerance ← 0.0001, merge ← 0.0001, max_span ← 8
output: muv
begin:
  $n \leftarrow |W|$
  $muv_s \leftarrow \{\ \}$
  for $i \leftarrow 1, n-1$:
    $V_i \leftarrow [\ ]$
    for $j \leftarrow i+1, n$:
      if max_span and $D[j][0] - D[i][0] >$ max_span:
        break
      $v_{fw} \leftarrow W[i][j]$
      same ← False
      if $|V_i| > 0$:
        foreach $v_i \in V_i$:
          if cosine($v_{fw}, v_i[0]$) $\leq$ merge:
            same ← True
            break
      if not same:
        $v_b \leftarrow nil$
        $k_b \leftarrow nil$
        for $k \leftarrow i-1, 0$:
          if max_span and $D[i][0] - D[k][0] >$ max_span:
            break
          $v_{bw} \leftarrow W[i][k]$
          if cosine($v_{bw} * -1, v_{fw}$) $\leq$ tolerance
            $v_b \leftarrow v_{bw}$
            $k_b \leftarrow k$
            break
          if $v_b$ not $nil$:
            $V_i$.append(($v_{fw}, v_b, k_b$))
      if !($i \in muv_s$):
        $muv_s[i] \leftarrow$ set()
      foreach $(v_j, v_k, k_b) \in V_i$:
        $muv_s[i]$.add(($v_j[0], v_j[1]$))
        $v_b \leftarrow v_k * -1$
        if !($k_b \in$ muv):
          $muv[k_b] \leftarrow$ set()
        $muv_s[k_b]$.add(($v_b, k_b$))
  return muv
end
```

These shortest musical vectors can be thought of as the components of all the longer vectors. As the musical vectors defined to be relevant in AGM were the sum vectors of all the components with *similar direction* the following algorithm 4.4, that calculates these using the results from the Shortest *muV*s -algorithm, was developed. This algorithm uses vectors from algorithm 4.3 as input. The vectors are stored in a hash table using the starting point i of each vector as a key. The algorithm iterates for each vector's starting point the column i of the vector table W starting from the largest vector at the end of the column. An optional window can be defined by setting the *max_span* parameter to larger than 0, in that case vectors outside the window are not evaluated. When a vector that is within the *merge* threshold calculated with *cosine*-distance is found, it is stored as a descriptive *muV* in muv_l and the computation for that input vector $muv_s[i]$ is completed. If larger descriptive *muV*s are not found the original vector $muv_s[i]$ is encountered and selected as the descriptive vector and the loop for that input vector is completed. As the algorithm compares for a set of input vectors, of size m all the subsequent vectors from W the worst-case time-complexity of the algorithm is $O(mn)$ where n is the size of the dataset D from which W is computed.

Algorithm 4.4: Short to long *muV*s

```
input: $muv_s$, $W$, $D$, merge ← 0.0001, max_span ← 0
output: $muv_l$
begin:
  $muv_l \leftarrow \{\}$
  $n \leftarrow |W|$
  foreach $i \in$ keys($muv_s$):
    $V_i$ = set()
    foreach $v_s \in muv_s[i]$:
      $v_l \leftarrow nil$
      for $j \leftarrow n-1, i$:
        if $v_s = W[i][j]$:
          $v_l \leftarrow W[i][j]$
          break
        else if max_span = 0 or $D[j][0] - D[i][0] \leq$ max_span:
          if cosine($v_s, W[i][j]$) < merge:
            if $||v_s|| < ||W[i][j]||$:
              $v_l \leftarrow W[i][j]$
            else:
              $v_l \leftarrow v_s$
            break
      if $v_l$:
        $V_i$.add(($v_l[0], v_l[1]$))
    $muv_l[i] = V_i$
  return $muv_l$
end
```

Figure 5 shows the difference between the outputs of the algorithm 4.1 (above), for computing all potential *muV*s, with using the combination of algorithms 4.3 and 4.4 (below). The dataset is a longer section of music, an excerpt of Tchaikovsky's Swan Lake Act 4. Overlaid on the figures is a red pitch contour.

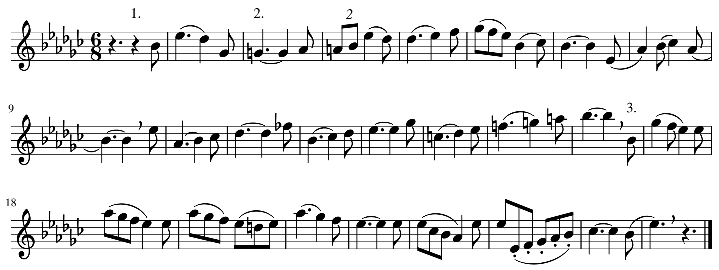


Figure 4: An excerpt of Tchaikovsky's Swan Lake Act 4.

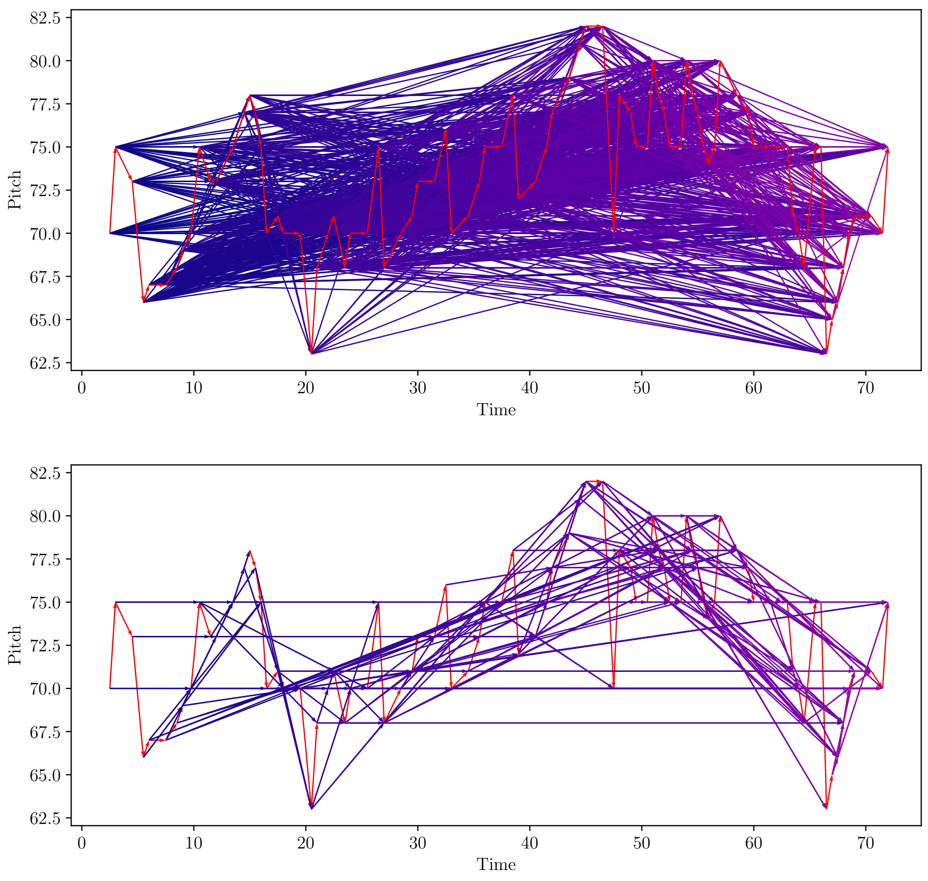


Figure 5: Comparison between all potential muVs (above) and short and descriptive muVs (below) from an excerpt of Tchaikovsky's Swan Lake Act 4.

As two of the biggest issues of the presented algorithms are their running times and the complexity and large quantity of the resulting data output by them, another approach to the concept of computing vectors similar to *muV*s was developed. It was inspired by the concept of auditory streaming from Bregman's Auditory Scene Analysis [1] which is referred to in AGM as well.

Cambouropoulos [5] has studied different views of what 'voice' means and how to describe the problem of voice separation. Listeners are thought to be capable of perceiving multiple distinct voices in music. Cambouropoulos' goal was to develop a systematic description of the cognitive task of voice separation in timbrally undifferentiated music. This was based on an examination and the usage of well-established perceptual principles of auditory streaming [1, 17] and resulted in a voice separation algorithm for a sequence of musical elements. The algorithm incorporates principles of temporal and pitch proximity with the addition of a synchronous note principle. The experimental results suggest that a single algorithm can achieve good performance in diverse musical textures, both homophonic and polyphonic, in terms of identifying perceptually relevant voices/streams. This bears similarity to the concepts of AGM and thus influenced the development of the algorithms presented here. There is no direct relation as the presented algorithms are a more simplified, 'proof of concept' version of similar ideas.

Algorithm 4.5, the streaming vectors -algorithm, only iterates through the dataset D and calculates and selects the vectors by comparing the distances between the closest points in the dataset. The data is assumed to be sorted according to the onset times of the points. For each $D[i]|i \in 0, |D| - 1$ the algorithm takes the vector to the next point $\boldsymbol{v}_i = D[i+1] - D[i]$ and in an inner loop compares it to following vectors $\boldsymbol{v}_i = D[j] - D[i]|j \in i + 2, |D|$ until an end condition is satisfied. For each vector the algorithm checks the timespan $d_j = D[j][0] - D[i][0]$ and the magnitude of the vector $m_j = ||\boldsymbol{v}_j||$. If the timespan is 0, we consider the vector as belonging to a chord and set the magnitude to *nil*. Then for every vector $\boldsymbol{v}_j$ the algorithm checks if they belong to a chord, or have a magnitude or timespan smaller than that of $\boldsymbol{v}_i$ and collects these vectors. If the timespan or magnitude is smaller, the values are stored to d_i and m_i and following vectors are compared to those. The algorithm then checks if it is at the end of the dataset or if the smallest magnitude of previous non-chord vectors is smaller than than the timespan of current vector and exits the inner loop.

Regarding the definitions of AGM the streaming vectors -algorithm deviated more from the purpose of finding *muV*s as it most likely misses a large number of musical vectors. It was more of an alternative approach to finding relations between pitches that may be perceptually connected which then could be connected to related larger *muV*s. The upside of the algorithm was it's reduced time complexity as an average case could be thought of as being $O(mn)$ where n is the size of the input and m is dependent of the number of points considered to be neighbours. The instances of the worst case $O(n^2)$ should be negligible when using musical works as input data.

Algorithm 4.5: Streaming Vectors

```
input: $D$
output: $V_S$
begin:
    $n \leftarrow |D|$
    $i \leftarrow -0$
    $V_S \leftarrow []$
    while $i < n - 1$:
        vectors $\leftarrow$ []
        chord $\leftarrow$ []
        FINISHED $\leftarrow$ False
        $j \leftarrow i + 1$
        $v_i = D[j] - D[i]$
        $d_i = D[j][0] - D[i][0]$
        if $d_i = 0$:
            chord.append($[i, j]$)
            $m_i \leftarrow nil$
        else:
            vectors.append($[v_i[0], v_i[1], i]$)
            $m_i \leftarrow ||v_i||$
        $j \leftarrow j + 1$
        while $j < n$ and not FINISHED:
            $v_j \leftarrow D[j] - D[i]$
            $m_j \leftarrow ||v_j||$
            $d_j \leftarrow D[j][0] = D[i][0]$
            if $d_j = 0$:
                chord.append($[i, j]$)
                if $m_i = nil$ or $m_j < m_i$:
                    $m_i \leftarrow m_j$
                    $j \leftarrow j + 1$
                    if $j \geq n$:
                        FINISHED $\leftarrow$ True
            else:
                if $m_i = nil$ or $m_j < m_i$:
                    $m_i \leftarrow m_j$
                    vectors.append($[v_j[0], v_j[1], i]$)
                else if $d_i \geq d_j$:
                    vectors.append($[v_j[0], v_j[1], i]$)
                if $d_i = 0$ and $d_j > 0$:
                    $d_i \leftarrow d_j$
                $j \leftarrow j + 1$
                if $j \geq n$ or $m_i < D[j][0] - D[i][0]$:
                    FINISHED $\leftarrow$ True
        if |vectors| = 0:
            vectors.append($[v_i[0], v_i[1], i]$)
        if |chord| > 0:
            chord_vectors $\leftarrow$ []
            for $c \in$ chord:
                $i \leftarrow c[0]$
                $j \leftarrow c[1]$
                $v_i \leftarrow D[i] - D[j]$
                $v_j \leftarrow D[j] - D[i]$
                chord_vectors.append($[v_i[0], v_j[0], j]$)
                chord_vectors.append($[v_j[0], v_i[0], i]$)
            $V_S \leftarrow V_S$+chord_vectors
        $V_S \leftarrow V_S$+vectors
        $i \leftarrow i + 1$
    return $V_S$
end
```

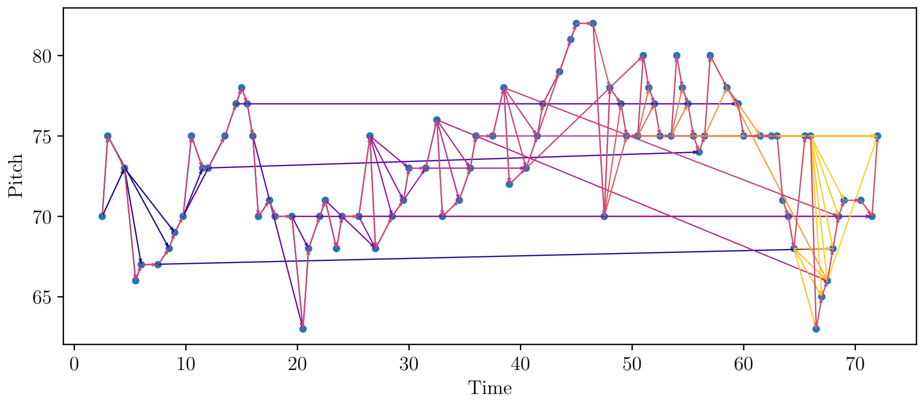


Figure 6: Streaming vectors and their muVs from an excerpt of Tchaikovsky's Swan Lake Act 4.

5 AGM software as a tool for music analysis

The algorithms developed for this thesis have been used in a set of tools developed for music analysis. They were used for the calculations for $intiC$s and muVs, plotting the vectors and $intiC$-curves, and assisting in music analysis as it is described in AGM [19].

The general idea behind the toolkit is that the user opens a dataset containing a musical work, for example from score sheets stored as files. The user then picks segments from the dataset by defining the start- and endpoints and the voice of the segment to be analysed. The parameter for threshold-values of calculating muVs as described can be adjusted and the software displays muV and $intiC$ graphs for the selected segments. Typically the analyses focus on monophonic melodies or similar material. Figures 1 and 2 demonstrated the use of $intiC$s to display the dynamic pitch 'velocities' from an excerpt of Stravinsky's Rite of Spring.

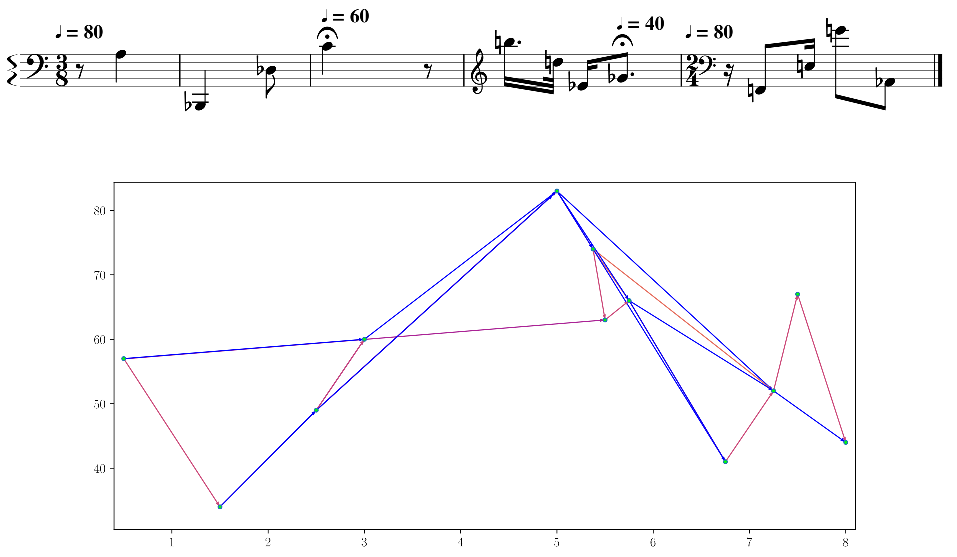


Figure 7: Example *muV*s of an excerpt from Anton Webern's Op. 30

In figure 7 the *muV*s of a segment of Webern's Op. 30 are shown. The segment contains serial music composed with series of four pitches and their variations. Here a music analyst studying AGM is especially interested in the *muV*s that coincide with the beginnings and ends of each of the series, shown in blue.

6 AGM and pattern discovery

There are several issues with computing these vectors and determining whether they are perceptually significant or interesting *musical vectors*. First of all, a large problem is the time complexity of the algorithms. In addition the resulting set of vectors with the algorithms 4.3 and 4.4 in itself still not intuitively suitable for purposes of music analysis for complete musical works. The problem might be approached by adjusting parameters that might result in smaller less dense sets of vectors. The threshold for finding *muV*s could be increased so that vectors within larger angle θ are considered redundant, or the span of x-axis in with we are searching for the vectors may be shortened. These adjustments however can result in a trade-off between finding perceptually interesting *muV*s and getting reasonably sized results

As an alternative approach to finding perceptually interesting vectors the incorporation of repeating pattern discovery with the computation of *muV*s was tried. This did not solve the issue of computational complexity, but could be beneficial both for determining which *muV*s and which patterns are perceptually interesting.

6.1 MuVs in repeating patterns

The shortest vector algorithm 4.3 finds all the musical vector components that connect the points closest to each other along a *musical vector sum*. In addition this set of short vectors M_s combined with the set of descriptive *sum vectors* M_l covers different potentially interesting sets of vectors.

$$M[i] = M_s[i] \cup M_l[i] \mid 0 \leq i \leq |M_s|$$

The repeating patterns calculated using SIATEC and the list of TECs which contains the <pattern, translator set> pairs was obtained. For each TEC a list of the muVs that match each point of each instance of a translated pattern was computed. For repeating patterns this means that there are some patterns which can be represented, or covered, by musical vectors better than others and some patterns that do not contain any musical vectors. With the point of view of the musical vectors the results may be interpreted as having vectors that appear multiple times in several overlapping patterns and vectors that don't appear in repeating patterns can be left out. The significance of a vector could be defined simply as it being found with the above algorithms. This concept can be further examined with the number of times the vector, or a pattern, is found.

Instead of storing the muVs in a list like in algorithms 4.1-4, they were stored in a hash table based dictionary, where the hash keys were calculated from tuple $p = (x, y)$, which is the starting point of the vector. This way it was simple to associate the points of the patterns with the muVs. Using musical vectors stored in a hash table muV_D, the algorithm 6.1 iterates the list of TECs and for each TEC collects the vectors whose starting and end points match points in some instance of that pattern.

Algorithm 6.1: Collecting musical vectors for patterns

```
input: $D, TEC, muV_D$
output: $muV_P$
begin:
    $n \leftarrow |TEC|$
    $M_P \leftarrow [\,]$
    for $i \leftarrow 0, n$:
        $V_P \leftarrow [\,]$
        foreach $v_t \in TEC[i][1]$:
            $P \leftarrow D[TEC[i][0] + v_t]$
            foreach $j \in TEC[i][0]$:
                $p \leftarrow D[j]$
                if $v_t + p \in M_D$:
                    foreach $v_{mu} \in muV_D[v_t + p]$:
                        $p_{mu} \leftarrow p + v_t + v_{mu}$
                        if $p_{mu} \in P$:
                            $V_P$.append($< p, v_t, v_m u >$)
        $M_P$.append($V_P$)
end
```

6.2 Inspecting patterns based on *muV*s

Collecting all *muV*s in a hash table made it possible for example to sort TECs based on how well their patterns matched with the *muV*s. Different sorting schemes for the organisation of patterns were considered. Rating the patterns based on

$$muV\text{s per TEC} = \frac{|M_P[i]|}{|\text{TEC}[i][0]||\text{TEC}[i][1]|}$$

gave the highest score to the pattern that contains the most *muV*s in its instances. These could be thought of as the largest perceptually significant patterns regardless of other factors such as their overlap. When inspecting single TECs these seem to be the most interesting ones as they may cover most of the dataset and the high scoring patterns seem relate to musical structures such as repetition of whole separate parts of the dataset, canon and repetition in harmonies.

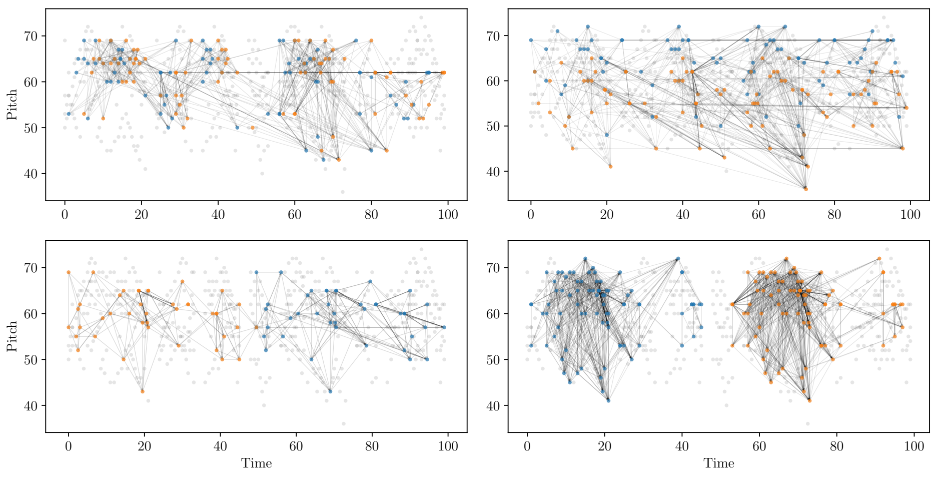


Figure 8: Bach:BWV425, patterns 0, 1, 6, 7, scored with '*muV*s per TEC', from 12477 TECs

A selection of patterns is displayed in figure 8. In this case the music was represented with chromatic pitch numbers to help discern between the notes in the patterns when inspecting the output. This might have resulted in a less repeating patterns in the results. In the top left corner, the highest scoring pattern corresponds to the repetitive quality of the chorale where same sequences of notes are repeated 4 quarter notes, or one bar apart from each other throughout the score. The second highest scoring pattern on top right shows vocal harmonies, as the pattern instances are seven semitones, i.e. a fifth, apart from each other. In addition the patterns contain multiple *muV*s that show a stable repetition of pitch numbers 45 and 69, or pitches

A2 and A4, and muVs that from the beginning to the end of the score point from 55 to 54 (G3, F#3), and from 60 to 61 (C4 to C#4). Patterns 2-5 which are not shown, overlapped with patterns 0 an 1, showing similar qualities. The patterns 6 and 7 in the bottom of the figure display repetition of different sections of the chorale.

The formula

$$muV\text{s per instance} = \frac{|M_P[i]|}{|\text{TEC}[i][1]|}$$

gave the highest score to the TECs that had the highest ratio of muVs to points in a pattern. Some of the results of this scoring are shown in figures 9 and 10, where two different musical works are displayed with different number of 'vectorised' patterns at a time. The patterns are overlaid so that the number of datapoints and vectors in pattern instances are emphasised with the colour intensity of the graphs, different hues of the colours are arbitrarily assigned to different patterns. These patterns are often small and compact. The issue with this rating is that there is a lot of redundancy. There are cases where numerous TECs cover exactly the same set of points with different variations of the patterns. This is where the selection of the patterns most differs from previous work on selecting significant patterns [24, 23, 8]. On the other hand this could be thought of as a positive feature, as because of the scoring of the TECs is the same, or they have scores closest to each other they may be grouped together.

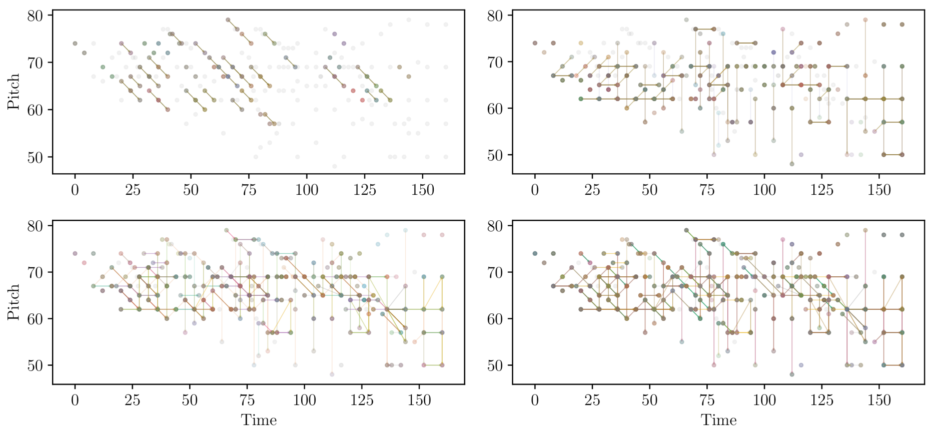


Figure 9: Palestrina:Gloria 15h, overlaid patterns 0-9, 10-39, 40-79, 0-79, muVs per instance

When examining several of the differing, highest scoring TECs as a whole, an interesting property becomes noticeable. The muVs of all the instances of the patterns in the TECs start to cover the dataset in a *compact* and almost *minimally overlapping* manner. That is to say there are only few sets of muVs that cross each other, at least compared to the muVs that appear multiple times in different patterns. Many of the highest scoring patterns in this scheme consist of short repetitive sequences of temporally adjacent notes within small intervals. The intervals and sequence sizes increased as the ordered sets of patterns were traversed. Further along the scheme, larger but still compact non-overlapping patterns can be found. The results from this scheme seem somewhat related to the concept of *milieu-territorial assemblages* of AGM.

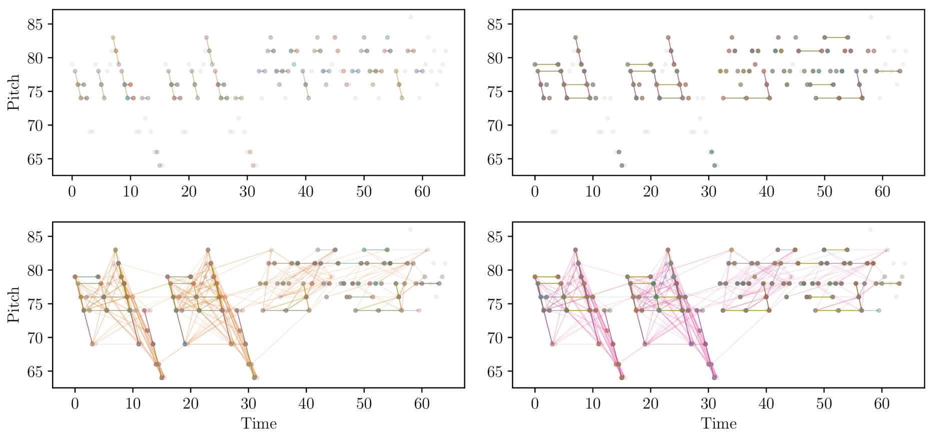


Figure 10: Ryan's Mammoth Collection: Merry Lads Of Ayer Reel, overlaid patterns 0-9, 10-39, 40-79, 0-79, muVs per instance

Another alternative when inspecting the patterns in addition to the scoring, could be to filter them out based on pattern length and the number of occurrences.

6.3 Inspecting muVs using patterns

Because of the abundance of output of the musical vector algorithms, there was a motivation to use pattern discovery to highlight and select vectors most important to the whole dataset — a complete piece of musical work. This was an alternate approach to the same issue as the selection of patterns based on their musical vector content. Instead of determining the value of the patterns based on vectors, the value of the vectors is determined based on the patterns they appear in.

The output array of algorithm 6.1, M_P, was iterated and for each muV tuple $m \in M_P[i] | i \in |M_P|$ the starting point $p = m[0] + m[1]$ was stored as a key in a hierarchical dictionary M_C for counting the instances of the vectors in different patterns. For each $M_C[p]$ the muV in the tuple $\boldsymbol{v}_m = m[2]$ was again used as a key for a sub-dictionary to count the number of times the muV appears in the dataset and it was stored in $M_C[p][\boldsymbol{v}_m]$. To emphasise the effect of the patterns the appearances of the vectors were counted by adding the number of instances of each pattern: $M_C[p][\boldsymbol{v}_m] \leftarrow M_C[p][\boldsymbol{v}_m] + |TEC[i][1]|$.

The results were then sorted based on the scoring they received from the above algorithm and the best scoring vectors were selected. This is illustrated in figure 11 where a selection of 1000 muVs (above) covers the dataset almost completely. In addition all possible muVs are displayed coloured based on their score (below). Vectors pointing down and forwards may for example be thought to correspond to descending melodic passages and the numerous horizontal vectors to the continuous repeating harmonies of the chorale. This selection was indifferent to whether the vectors are 'descriptive muVs' or their component vectors. It could be informative regarding the study of AGM to only take into account the descriptive vectors.

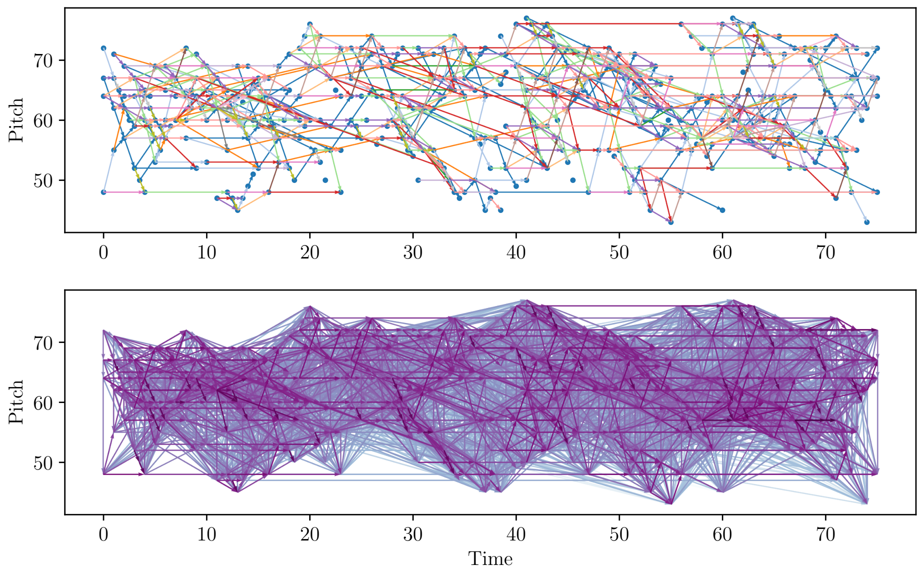


Figure 11: J.S. Bach, BWV340: muVs selected and organised based on their appearance in patterns

6.4 Filtering vector table based on muVs

Both SIATEC and the algorithms for computing musical vectors are computationally expensive and we are interested in an intersection of their results — the repeating patterns that contain muVs. That is why a logical step in the system utilising the algorithms could be to do computations on data that is only comprised of these patterns. One solution would be to compute muVs of the point-sets of the resulting patterns of SIATEC as opposed to computing the vectors of the original dataset, but this way we would not be able to use vectors that are related to points outside of the patterns themselves. Another approach is to eliminate the patterns that do not have any vectors in them before running the SIATEC-algorithm. This can be done by removing from the vector table V any patterns that do not have muVs associated with them. In this option too we will loose pattern instances that do not have muVs in them as some patterns might have a number of muVs in one instance and none in other instances. In some cases it might be useful to include such instances in the results.

As all the points in a pattern instance are next to each other in V, for each pattern instance P we can check whether its points are in the muV dictionary M_D. If a point $p \in P$ is a key in M_D, for each vector $\boldsymbol{v}$ in $M_D[p]$ we can check whether the end point of $\boldsymbol{v}$ is included in the rest of P.

It is likely that a set of patterns that contains muVs does not contain all, or possibly even some of the most perceptually interesting patterns, especially if the computation of muVs is limited by using the shortest vectors or streaming vectors. This can be illustrated by comparing the outputs of musical vector and streaming vector algorithms. Many of the vectors between points close to each other in figure 6 are not present in either of the musical vector graphs in figure 5. Further examples of different sets of vectorised patterns are included in the results and shown in appendix B. It could be a subject of further work to inspect whether this kind of filtering would be of use outside the study of muVs and related patterns. The outputs of the presented algorithms could be compared with the outputs of algorithms such as COSIATEC, that are known to output perceptually significant patterns, or with the patterns selected with the empirically validated formula in [8].

7 Results

The main issue of the muV algorithms was running time. The time complexity of the naive version of the algorithm that collects all muVs by traversing the vector table was $O(n^3)$ in any case. This was addressed by implementing an algorithm 4.3 that first finds shortest muVs inside a timespan window. The vectors of the output were then used as input for a second algorithm 4.4 which collected the descriptive muVs for each input vector. The time complexity of the shortest vector algorithm was $O(wn^2)$, where w is the average number of datapoints in a window. The vertical density of the dataset could be thought of as affecting the complexity in this case. In the worst-case if all the data points are within the window, for example they happen at the same time, the complexity is actually $O(n^3)$. However as the algorithm is supposed to be used for data where the datapoints are spread somewhat evenly on the horizontal axis and the amount of datapoints within the window is not dependant on n. The $O(wn^2)$ should be thought of as the average-case complexity and for sane inputs is the relevant case. A window that only contains a fixed amount of points was considered but this version was selected as it was thought to be more intuitive and did not affect the actual performance of the algorithm negatively. The time complexity of algorithm 4.4 was $O(mn)$ where m is the size of the output of algorithm 4.3.

Due to the fact that on larger musical works the performance of the muV-algorithms started to scale better than SIATEC, and because the end goal was to select patterns that contained muVs, it was relevant to try to help shorten the computation time of SIATEC by first trying to remove the patterns that did not contain any muVs. In an experimental setup first SIA and muV-algorithms were run, the patterns that did not contain muVs were removed from the vector table V and finally SIATEC was run on the 'filtered' V. This shortened the combined time of running the algorithms. However this is slightly problematic as it is not known if a ratio of muVs for each pattern should be used to discard patterns, and it is not yet clear what parameters for computing muVs yields best results. When using these algorithms together it would be important to take care in determining these parameters. In addition there is no guarantee that perceptually significant patterns are not lost when doing this.

The performance of the algorithms for different use cases was evaluated. For analysing sections of music, such as melodies and themes, a sampling of 100 works from a corpus of folk songs, reels and jigs from *'Ryan's Mammoth Collection of Fiddle Tunes'* included with the *music21* software toolkit [13] was used. The contents of the dataset were as described in the title and contained short monophonic pieces of music. This sample corpus is referred to as '*Ryan's'* in the results. The size of these sample datasets ranged from 53 to 203 datapoints, which was larger than any melody, motif or theme should be. The average running time of the SIATEC algorithm for this dataset was $0.72s$. Computing the musical vectors using the combination of algorithms 4.3 and 4.4 took $1.28s$ and the all muVs algorithm 4.1 on average $5.60s$. Due to the exponential nature of the algorithms the all muVs algorithm was deemed unfeasible to use on larger datasets.

An examination of larger polyphonic works from the *music21* database revealed that when the number of TECs rose high enough, for example over 10 000 TECs, the shortest muV algorithms started being noticeably faster than the SIATEC algorithm. A random selection of 15 chorale works by Bach and Palestrina was used as a secondary dataset and it is referred to as '*Chorales'*. As further examples of using the algorithms with larger musical works Prelude and Fugue in C Major by Johann Sebastian Bach were included.

Running times, ranges of dataset sizes and resulting numbers of TECs are shown in tables 1 and 2. For the datasets '*Ryan's'* and '*Chorales'* the average running times are included. The times are labelled as t_{TEC} for SIATEC, t^{f}_{TEC} for SIATEC on a filtered set of patterns and t_{muV} for musical vectors. The muVs were computed for a window of 8 quarter notes and with a threshold parameter of 0.0001. These parameters were selected to try to maximise the amount of vectors within a feasible computation time. For the filtered sets of patterns only patterns that contained at least one vector for each point in the pattern were selected. The datasets were projected to use the diatonic pitch number scheme so that the results would be more related previous work on pattern discovery where the similar morphetic pitch [22] is often used. In the first table the short and long muVs are used to filter the vector table, showing that the running time t^f_{TEC} is slightly shorter for each of the datasets. In the latter table the streaming vectors algorithm was used and it performs clearly faster. However the resulting selection of TECs is different for each algorithm. As the filtering of patterns was strict it is likely that at least for the smaller datasets interesting patterns were left out.

Table 1: Results of computing short muVs and related TECs.

	Ryan's	Chorales	Prelude in C	Fugue in C
n	53–204	134–597	614	790
TECs	242–3997	1453–8824	11779	20000
filtered TECs	13–565	209–3195	5765	13247
t_{TEC}	0.72 s	26s	109 s	229 s
t^f_{TEC}	0.24 s	17s	85 s	223 s
t_{muV}	1.3 s	12s	46 s	159 s

Table 2: Results of computing streaming muVs and related TECs.

	Ryan's	Chorales	Prelude in C	Fugue in C
filtered TECs	3–81	158–3057	1853	1152
t^f_{TEC}	0.31s	14s	46s	27s
t_{muV}	0.20s	7.3s	9.6s	41s

To further illustrate the selection of patterns using different schemes, graphs using Bach's Prelude and Fugue in C Major are included in appendix B. The colourisation of the patterns in the figures is arbitrary as there are too many patterns plotted over each other for a more accurate visual differentiation. Vectors belonging to patterns in same TECs are in the same colour and points that are in the same pattern instances are in the same colour, but same colours are also used for multiple different TECs and pattern instances.

In figure 12, using the 'muVs per instance' formula, 100 highest rated distinct patterns and their muVs are shown covering the whole dataset. Notably the prominence of broken arpeggiated chords and transpositions between them are highlighted with this selection of patterns. Some obvious repetitions are not shown with this scheme. This is seen in the first repeating arpeggios where there are discontinuities with vectors that are along the same line. These sections could be covered either by including more distinct patterns, or by including the descriptive *muV*s where their components are found. For the sake of clarity the descriptive *muV*s of each of their components were not included in the examination of the outputs of the algorithms. In figure 13 the highest rated pattern using the '*muV*s per TEC' formula is shown. As the composition consists mostly of arpeggios, they can be seen repeating exactly throughout this pattern that covers most of the piece. Following high scoring patterns are variations of this. Patterns containing streaming vectors and their *muV*s are seen in figure 14 where the streaming vectors slightly better match the structure of the arpeggiated chords than in figure 12.

For the 'Fugue in C Major', in figure 15 specifically, the patterns match parts of the subject and the answers of the fugue. In figures 16 and 18 more patterns within the themes and patterns along *muV*s between themes close to each other are shown. In figure 18 the *movement* ascribed to some of the *muV*s towards the end of the piece seems to be directed towards the final high note. In figure 17 the large pattern of descending notes throughout the piece matches melodies of the answers of the fugue.

The combination of the streaming vectors and their descriptive *muV*s resulted in a different set of patterns from the purely *muV* based approach. These are shown in appendix B, figures 14 and 19. The resulting sets of patterns selected based on streaming vectors and component *muV*s, both with the addition of their descriptive *muV*s differed. The former set was more similar to what is described in AGM as the movement, or velocities, of pitches and the latter to different perceptible units such as chord intervals and notes comprising melodic passages. In addition the *muV*s computed from streaming vectors often included those that coincided with repetitions along the harmonies or the scale of the musical work.

8 Discussion and future work

The goals of this thesis were to implement algorithms for the music theoretical concepts of AGM [19] and to study their usage in the context of selecting perceptually interesting patterns output by pattern discovery algorithms [24, 21]. Implementations of computing *intiC*-values and *muV*s were made for music analysis tools and other computer aided music analytic tasks. The usage of *muV*s to discover perceptually interesting patterns was an experimental task as the approach was different from previous work on the subject. The result was a novel approach to organising and selecting patterns. The intersection of patterns and *muV*s was used for the organisation of *muV*s as well.

The usefulness of the *muV* algorithms on the outlined tasks was evaluated based on their performance regarding running time and by examining the sorted outputs regarding the discovery of musically interesting patterns. Apart from this manual inspection, which included graphs such as those shown in this thesis, there was no further evaluation of the outputs produced by the pattern selection as it was deemed outside the scope of this thesis. There was no well established methodology for evaluating the significance of the patterns as their selection differed from the aim of other such evaluations. In other works it is usually desired that the patterns cover the dataset with minimal overlap and the algorithms presented here were indifferent of that. In addition previous work has focused on finding specifically patterns such as melodies, themes and motifs that would be correlated with traditional music analysis. However approaching such an analysis could be feasible if either a method of removing overlapping sets of the patterns was added, or if an algorithm for matching the patterns with the output of an algorithm such as COSIATEC or patterns selected with the formula by Collins et al. [8] was used.

Further additions to mitigate the time complexity issue could be developed. The vertical range of the musical datasets can be thought to be limited to lowest and highest notes of the dataset. As vectors, directed upwards or downwards at an angle from the x-axis, at some point of time or distance on the x-axis, can not reach any other point in the set, we should stop trying to find matching vectors when that point is reached. Or in the case of searching for the longest vectors that would be the ideal point to start the search. This could be done by indexing or partitioning the dataset along the time axis to access that furthest point in the dataset a vector can reach.

The algorithms were implemented with Python as they are presented in this thesis and the evaluations were run on a 4.2GHz Intel i5 processor. Regarding their performance, writing the algorithms in a lower-level language such as C or with methods more suitable for iterating arrays containing numerical data should be considered.

Observing the results of the sorted patterns and vectors did show that the approach could be useful. The first formula for selecting the patterns, 'muVs per TEC', resulted in large patterns, which corresponded to perceptually plausible musical phenomenon such as fugue and canon melodies, harmonies and large grouping structures, scoring highest. The best scoring patterns of the second formula, 'muVs per instance', on the other hand could be described to resemble a hierarchical structure of perceptual connections, where at first we have adjacent notes and intervals of chords, and later full chords, motifs and phrases, which with a limited number of TECs cover most of the musical work. This approach to structure could be useful in developing novel methods of grouping analysis. In addition it resembled the concept of *milieu-territorial assemblages* of AGM.

Similarly inspecting vectors by scoring them based on their appearance in repeated patterns resulted in their hierarchical organisation. This could be thought of a more accurate representation of the 'assemblages' as is it was a definition of the organisation of the vectors. A selection of *muV*s computed with an algorithm of choice presented in this thesis, with or without giving them scores, could be added to a model of music in the same manner as the translational vectors and patterns were used in [9] to retain additional large scale structures corresponding to different musical phenomenon.

What was completely left out regarding AGM was the generative aspect. For *intiC*s this could consist of the usage of *intiC*-curves for the generation or modification of musical material. In the case of *muV*s their trajectories could be used to predict unheard notes and thus generate new musical material.

Similarities with the concept of *intiC*, and by association with *muV*, could be found in existing works of computer aided musicology [3, 26]. This is promising regarding the usage of the methods of AGM in the development of such work. The algorithms presented in this thesis provide a way to compute these context dependent temporal and pitch based relations in music. Notably many of the cases, where low-level concepts similar to *intiC* were used, were limited to monophonic music and replacing them with a vector based approach could make it possible to expand these approaches to polyphonic music represented with multidimensional point-set data.

References

[1] Bregman, Albert S: *Auditory scene analysis: The perceptual organization of sound.* MIT press, 1994.

[2] Cambouropoulos, Emilios: *Towards a general computational theory of musical structure.* PhD thesis, 1998.

[3] Cambouropoulos, Emilios: *The local boundary detection model (lbdm) and its application in the study of expressive timing.* In *ICMC*, 2001.

[4] Cambouropoulos, Emilios: *Musical parallelism and melodic segmentation.* Music Perception: An Interdisciplinary Journal, 23(3):249–268, 2006.

[5] Cambouropoulos, Emilios: *Voice and stream: Perceptual and computational modeling of voice separation.* Music Perception: An Interdisciplinary Journal, 26(1):75–94, 2008.

[6] Collins, Tom: *Improved methods for pattern discovery in music, with applications in automated stylistic composition.* PhD thesis, The Open University, 2011.

[7] Collins, Tom, Arzt, Andreas, Flossmann, Sebastian, and Widmer, Gerhard: *Siarct-cfp: Improving precision and the discovery of inexact musical patterns in point-set representations.* In *ISMIR*, pages 549–554, 2013.

[8] Collins, Tom, Laney, Robin, Willis, Alistair, and Garthwaite, Paul H: *Modeling pattern importance in chopin's mazurkas.* Music Perception: An Interdisciplinary Journal, 28(4):387–414, 2011.

[9] Collins, Tom, Laney, Robin, Willis, Alistair, and Garthwaite, Paul H: *Developing and evaluating computational models of musical style.* AI EDAM, 30(1):16–43, 2016.

[10] Collins, Tom, Thurlow, Jeremy, Laney, Robin, Willis, Alistair, and Garthwaite, Paul: *A comparative evaluation of algorithms for discovering translational patterns in baroque keyboard works.* 2010.

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

[12] Cope, David: *Experiments in musical intelligence*, volume 1. AR editions, 1996.

[13] Cuthbert, Michael Scott and Ariza, Christopher: *music21: A toolkit for computer-aided musicology and symbolic music data.* 2010.

[14] Ferrand, Miguel, Nelson, Peter, and Wiggins, Geraint: *Memory and melodic density: a model for melody segmentation.* In *Proceedings of the XIV Colloquium on Musical Informatics (XIV CIM 2003)*, pages 95–98, 2003.

[15] Forth, Jamie and Wiggins, Geraint A: *An approach for identifying salient repetition in multidimensional representations of polyphonic music.* 2009.

[16] Hamanaka, Masatoshi, Hirata, Keiji, and Tojo, Satoshi: *Implementing "a generative theory of tonal music".* Journal of New Music Research, 35(4):249–277, 2006.

[17] Huron, David: *Tone and voice: A derivation of the rules of voice-leading from perceptual principles.* Music Perception: An Interdisciplinary Journal, 19(1):1–64, 2001.

[18] Janssen, Berit, De Haas, W Bas, Volk, Anja, and Van Kranenburg, Peter: *Finding repeated patterns in music: State of knowledge, challenges, perspectives.* In *International Symposium on Computer Music Modeling and Retrieval*, pages 277–297. Springer, 2013.

[19] Laiho, Timo *et al.*: *Perception, time and music analysis: An introduction to analytic-generative methodology (agm).* Studia musicologica Universitatis Helsingiensis 23, 2013.

[20] Lerdahl, Fred and Jackendoff, Ray: *A generative theory of tonal music.* MIT press, 1985.

[21] Meredith, David: *Point-set algorithms for pattern discovery and pattern matching in music.* In *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2006.

[22] Meredith, David: *The ps13 pitch spelling algorithm.* Journal of New Music Research, 35(2):121–159, 2006.

[23] Meredith, David: *Analysing music with point-set compression algorithms.* In *Computational Music Analysis*, pages 335–366. Springer, 2016.

[24] Meredith, David, Lemström, Kjell, and Wiggins, Geraint A: *Algorithms for discovering repeated patterns in multidimensional representations of polyphonic music.* Journal of New Music Research, 31(4):321–345, 2002.

[25] Pearce, Marcus, Müllensiefen, Daniel, and Wiggins, Geraint A: *A comparison of statistical and rule-based models of melodic segmentation.* In *ISMIR*, pages 89–94, 2008.

[26] Pearce, Marcus T, Müllensiefen, Daniel, and Wiggins, Geraint A: *The role of expectation and probabilistic learning in auditory boundary perception: A model comparison.* Perception, 39(10):1367–1391, 2010.

[27] Pearce, Marcus T and Wiggins, Geraint A: *Auditory expectation: The information dynamics of music perception and cognition.* Topics in cognitive science, 4(4):625–652, 2012.

[28] Pearce, Marcus Thomas: *The construction and evaluation of statistical models of melodic structure in music perception and composition.* PhD thesis, City University London, 2005.

[29] Wiggins, G. A. & Smaill, A.: *What can artificial intelligence bring to the musician?* pages 29–46, 2000.

A Python implementation of Structure Induction Algorithms

```python
import numpy as np

""" Python implementation of SIA using numpy. """
def sia(D):
    # Compute both vector tables from W
    vt = D - D[:,np.newaxis]
    n = len(vt)
    W = np.array([np.array([[*y, i] for y in vt[i]]) for i in range(n)])
    V_indices = np.triu_indices(W.shape[0], 1)
    V = W[V_indices]
    # Sort V
    V = V[np.lexsort(np.transpose(V)[::-1])]
    return D, W, V
```

```python
""" Python implementation of SIATEC using numpy. """
def siatec(D, W, V):
    X = siatec_X(D, V)
    Y = sort_X(X)
    tecs = siatec_tecs(D, Y, V, W)
    return tecs

""" SIATEC: Computing ordered set X"""
def siatec_X(D, V):
    m = len(V)
    X = []
    i = 0
    while i <= m:
        Q = []
        j = i+1
        while j < m and np.all(V[j][:2] == V[i][:2]):
            Q.append(tuple(D[int(V[j][2])] - D[int(V[j-1][2])]))
            j = j+1
        X.append((i, tuple(Q)))
        i = j
    return X

def sort_X(X):
    return sorted(X, key = lambda y: (len(y[1]), y[0]))
```

```python
""" Algorithm for gathering the set of patterns and their translators """
def siatec_tecs(D, Y, V, W, indexes=True, min_patlen=2):
    r = len(Y)-1
    m = len(V)
    i = 0
    tecs = []
    if r > 0:
        while True:
            j = Y[i][0]
            I = []
            while j < m and (V[j][:2] == V[Y[i][0]][:2]).all():
                I.append(int(V[j][2]))
                j = j+1
            if len(I) >= min_patlen:
                pattern = tec_pattern(D, I, indexes)
                translators = tec_translators(W, D, I)
                tecs.append([pattern, translators])
            while True:
                i = i+1
                if i > r or Y[i-1][1] != Y[i][1]:
                    break
            if i > r:
                break
    return tecs

def tec_pattern(D, I, indexes=False):
    if indexes:
        return I
    else:
        p = len(I)
        pattern = [D[I[0]]]
        for k in range(1,p):
            pattern.append(D[I[k]])
        return np.array(pattern)

def tec_translators(W, D, I):
    p = len(I)
    vectors = set(tuple(x) for x in W[I[0]][:,:2])
    if p > 1:
        for i in range(1, p):
            vectors = vectors & set(tuple(x) for x in W[I[i]][:,:2])
    return vectors
```

B Examples of organised patterns and their musical vectors

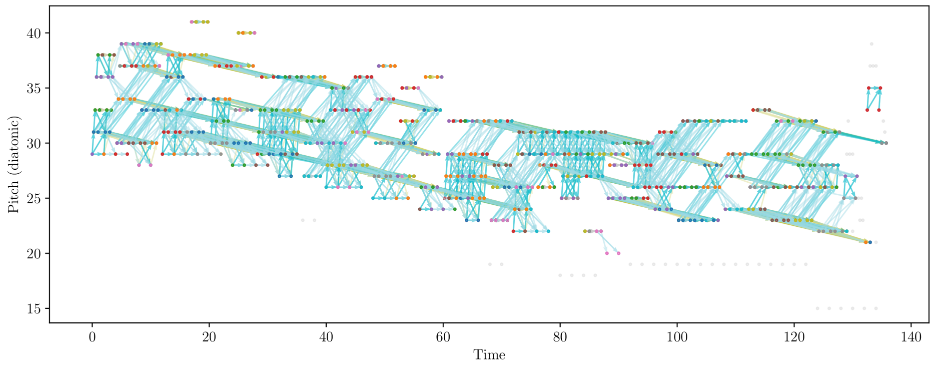


Figure 12: J.S. Bach: Prelude in C Major, 100 highest rated patterns and their vectors scored with '*muV*s per instance' formula.

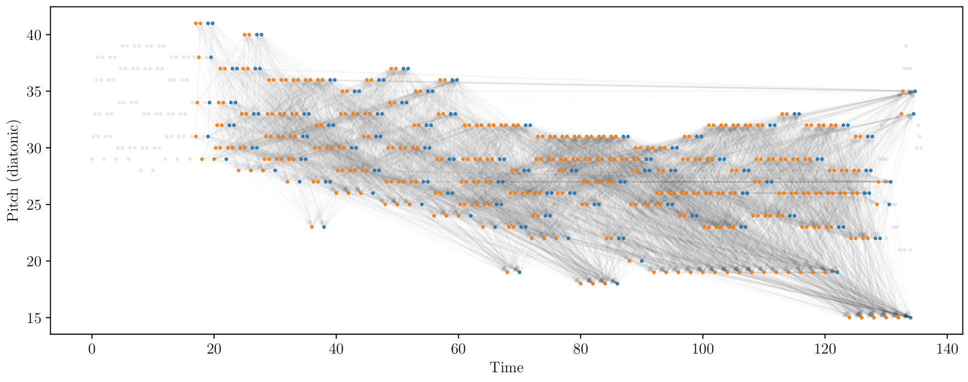


Figure 13: J.S. Bach: Prelude in C Major, highest rated pattern and its vectors with '*muV*s per TEC' formula.

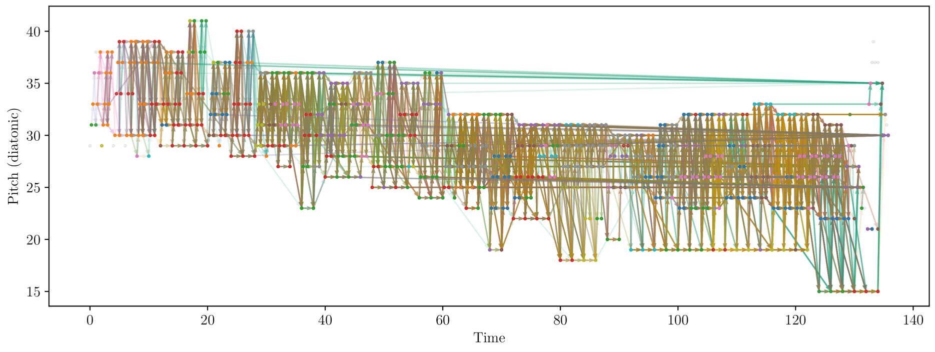


Figure 14: J.S. Bach: Prelude in C Major, 100 highest rated patterns and their vectors, based on the streaming vectors algorithm, scored with 'muVs per TEC' formula.

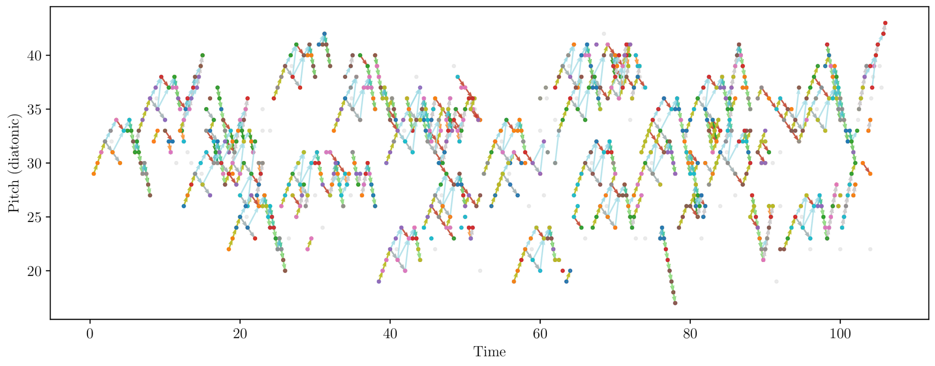


Figure 15: J.S. Bach: Fugue in C Major, 100 highest rated patterns and their vectors, scored with 'muVs per instance' formula.

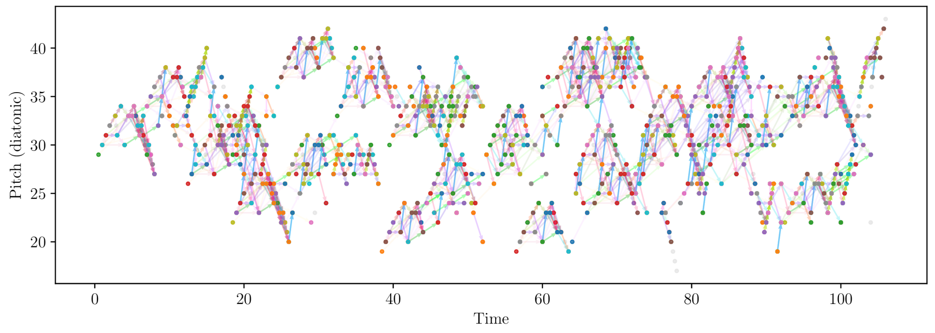


Figure 16: J.S. Bach: Fugue in C Major, patterns 100-200 and their vectors, scored with 'muVs per instance' formula.

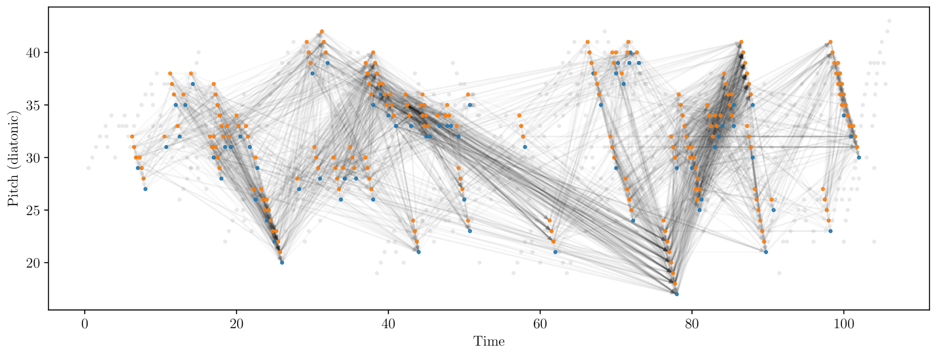


Figure 17: J.S. Bach: Fugue in C Major, highest rated pattern and its vectors, scored with ‘muVs per TEC’ formula.

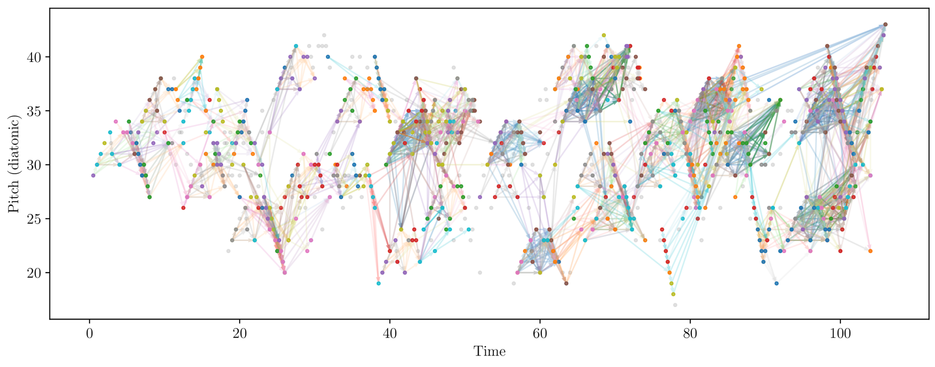


Figure 18: J.S. Bach: Fugue in C Major, 100 highest rated patterns and their vectors, with the minimum of three instances for a pattern, scored with ‘muVs per TECs’ formula.

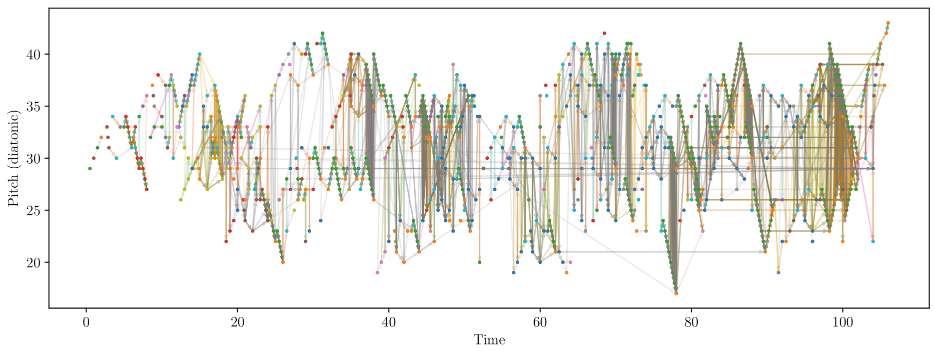


Figure 19: J.S. Bach: Fugue in C Major, 100 highest rated patterns and their vectors, based on the streaming vectors algorithm, scored with 'muVs per instance' formula.