

# Repetition sans Ennui

Human and Algorithmic Discovery of Patterns in Music

Menselijke en Algoritmische Ontdekking van Patronen in Muziek

(met een samenvatting in het Nederlands)

Proefschrift

ter verkrijging van de graad van doctor aan de  
Universiteit Utrecht  
op gezag van de  
rector magnificus, prof. dr. H.R.B.M. Kummeling,  
ingevolge het besluit van het college voor promoties  
in het openbaar te verdedigen op  
maandag 12 februari 2024 des middags te 4.15 uur

door

Iris Yuping Ren

geboren op 31 mei 1991  
te Beijing, China

**Promotoren:**

Prof. dr. Remco C. Veltkamp

Prof. dr. Gabriele Keller

Prof. dr. Anja Volk

**Copromotor:**

Dr. Wouter Swierstra

**Beoordelingscommissie:**

Dr. J.M. Calderon Trilla

Prof. dr. E. Chew

Prof. dr. J.J.E. Kursell

Prof. dr. A.A. Salah

Prof. dr. E.G.J. Wennekes



The work presented in this thesis was performed at the Department of Information and Computing Sciences, Utrecht University, Utrecht, The Netherlands. Cover artwork and design by © Iris Yuping Ren, Elma Hogeboom, 2024.



DUURZAMEDISSERTATIE.NL  
De specialist in een duurzaam proefschrift



Proudly printed on 100% recycled paper.

ISBN: 978-90-833597-1-7

© 2024 Iris Yuping Ren, The Netherlands. All rights reserved. No parts of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means without permission of the author. Alle rechten voorbehouden. Niets uit deze uitgave mag worden vermenigvuldigd, in enige vorm of op enige wijze, zonder voorafgaande schriftelijke toestemming van de auteur.

# Abstract

Finding patterns is a common act in human intellectual endeavours, and it is a complex challenge tackled by both humans and algorithms. For several decades, musical pattern discovery algorithms have been researched, and researchers have been comparing human-annotated patterns to algorithmic outputs, as well as algorithms to algorithms. However, traditional metrics have not fully captured the rich insights that these comparisons could offer. To contribute to the comparisons between musical pattern discovery mechanisms, this dissertation spans seven chapters.

Chapter 1 provides the background of the dissertation, including an overview, research approaches, contexts, scope, thesis statement, and an enumeration of contributions. Chapter 2 delves into the concept of musical patterns and explores the diverse landscape of musical pattern discovery algorithms. Our exploration reveals the complexities surrounding the definition of patterns and the multifaceted nature of these algorithms. Chapter 3 is dedicated to the collection tools for human-annotated musical patterns and the analysis of factors that influence annotations. We observe that musical background impacts annotated patterns; tool interfaces and automatic matching affect the length and frequency of annotations. Chapter 4 introduces four methods tailored for comparing musical pattern discovery algorithms. These methods provide novel insights into the discrepancies between human-annotated patterns and their algorithmically extracted counterparts. These methods provide a more comprehensive approach to comparing algorithms, aiding in the interpretation and evaluation of algorithmic outputs. Chapter 5 implements *Pattrans*, a Domain-Specific Language (DSL) in the functional language Haskell for comparing musical pattern occurrences through musical transformations. We delve into its design for uncovering the relations between pattern occurrences in a modular way. Chapter 6 employs *Pattrans* to scrutinise transformations between occurrences of musical patterns. Amongst other findings, we find that human-annotated patterns tend to have a higher proportion of exact repetitions and that different algorithms exhibit varying proportions of transformation compared to human annotations, contributing to a more nuanced view of pattern comparisons.

In summary, this dissertation not only contributes fresh perspectives to the comparison of musical patterns, but also introduces methods and tools that enrich the field of musical pattern discovery. We examine the concept of musical pattern, conduct pattern annotation experiments, and visualise and analyse human-annotated and algorithmically extracted patterns. In addition, we recognise the potential of the musical transformations that lie behind repeating and varying pattern occurrences. Using Haskell, we model the relationship between patterns and transformations. Following this, we investigate how to employ transformations to relate and classify musical pattern occurrences. Throughout our journey, we advocate for a more comprehensive approach to pattern comparison, extending beyond traditional metrics.

# Acknowledgements

Thanking everyone who supported me during this process is a nearly impossible task. My special thanks first go to my supervisors: prof. dr. Anja Volk, dr. Wouter Swierstra, prof. dr. Remco C. Veltkamp, and prof. dr. Gabriele Keller, all of whose guidance has been invaluable throughout my journey. I gratefully acknowledge the constructive comments of the anonymous referees of our papers, editor Adam Frick, and the reading committee members: prof. dr. Elaine Chew, dr. José M. Calderón T., prof. dr. Julia J.E. Kursell, prof. dr. Albert A. Salah, prof. dr. Emile G.J. Wennekes.

I extend my gratitude to my fellow PhD candidates during my PhD, dr. Dimitrios Bountouridis, dr. Hendrik Vincent Koops, Mirjam E. Visscher, Karlijn Dinnissen, dr. Victor Cacciari Miraldo, João Pizani, and dr. Alejandro Serrano Mena. To the group members, dr. Frans Wiering and dr. Peter van Kranenburg, thank you for exchanging music examples, manuscripts, and discussions. Heartfelt thanks to the greater Department of Information and Computing Sciences for indispensable support. Additionally, special appreciation goes to colleagues from various groups, departments, and institutions, both new and long-standing.

Across various research communities, my gratitude goes to dr. Matevž Pesek, dr. Berit Janssen, dr. Tom Collins, dr. David Meredith, prof. dr. Darrell Conklin, dr. Cynthia Liem, prof. dr. Geraint Wiggins, dr. Bob Sturm, dr. Donya Quick, dr. Niki Vazou, and many more for your collaborative effort and enriching discussions. Appreciation is also due to students who collaborated with me: Orestis Melkonian, Darian Tomašević, Stephan Wells, Erik Scerri, and Kevin J.J. Westerbaan, for their dedicated engagement and valuable contributions to the research.

A profound appreciation to my friends and family. I owe a world of gratitude to Saul for his support and encouragement. His belief in me has been a guiding light.

A sincere thanks to each and every one of you who has helped me in one way or another along this incredible journey!





# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Overview . . . . .	4
1.2	Approaches to studying musical pattern discovery . . . . .	8
1.3	Disciplinary contexts: a summary . . . . .	10
1.3.1	Pattern discovery in music information retrieval . . . . .	10
1.3.2	Musicology, music theory, psychology, and cognition . . . . .	10
1.3.3	Functional programming, pattern, and transformation . . . . .	11
1.4	Scope . . . . .	12
1.4.1	Restriction to monophonic data . . . . .	12
1.4.2	Restriction to symbolic data . . . . .	13
1.4.3	Datasets and generalisability . . . . .	13
1.4.4	Format of patterns . . . . .	14
1.4.5	Inter- Intra- opus and patterns . . . . .	15
1.5	Thesis statement . . . . .	15
1.6	Dissertation outline and chapter-wise main contributions . . . . .	16
<b>2</b>	<b>Patterns and Pattern Discovery</b>	<b>21</b>
2.1	Introduction . . . . .	21
2.2	Pattern as a concept . . . . .	23
2.3	Musical patterns and related concepts . . . . .	26
2.3.1	Related concepts in MIR . . . . .	26
2.3.2	Related concepts in music theory and musicology . . . . .	33
2.4	Musical pattern discovery algorithms . . . . .	44
2.5	Discussion . . . . .	50
2.6	Concluding remarks . . . . .	52
<b>3</b>	<b>Gathering Human-Annotated Musical Patterns</b>	<b>55</b>
3.1	Introduction . . . . .	55
3.2	Setup for ANOMIC and PAF . . . . .	60
3.2.1	Music material . . . . .	60
3.2.2	ANOMIC . . . . .	61

3.2.3	PAF . . . . .	63
3.2.4	Differences between the tools . . . . .	65
3.2.5	Differences between the experiments . . . . .	67
3.3	Methodology and metrics . . . . .	69
3.3.1	Agreement analysis . . . . .	70
3.3.2	Feature analysis . . . . .	71
3.4	Results . . . . .	74
3.4.1	Agreement analysis . . . . .	74
3.4.2	Feature analysis . . . . .	76
3.5	Discussion . . . . .	80
3.6	Concluding remarks . . . . .	82
<b>4</b>	<b>Comparisons of Musical Pattern Discovery Algorithms</b>	<b>87</b>
4.1	Introduction . . . . .	87
4.2	Our methods . . . . .	92
4.2.1	Summarising the use of the algorithms in our methods . . . . .	92
4.2.2	Location and Feature Visualisation (LFV) . . . . .	92
4.2.3	Pattern Polling (PP) . . . . .	102
4.2.4	Comparative Classification (CC) . . . . .	108
4.2.5	Synthetic Pattern Insertion (SPI) . . . . .	118
4.3	Discussion . . . . .	123
4.4	Concluding remarks . . . . .	125
<b>5</b>	<b>Modelling Patterns With Transformations Using Haskell</b>	<b>127</b>
5.1	Introduction . . . . .	127
5.2	Pattrans . . . . .	130
5.2.1	Basic types . . . . .	130
5.2.2	Transformation checking . . . . .	132
5.2.3	Compositions of checking transformation . . . . .	134
5.2.4	Approximation . . . . .	135
5.2.5	Compositional transformations . . . . .	140
5.3	Musical transformations in Pattrans . . . . .	141
5.4	Discussion and concluding remarks . . . . .	144
<b>6</b>	<b>Using Transformations to Understand the Relations Between Pattern Occurrences</b>	<b>149</b>
6.1	Introduction . . . . .	149
6.2	Data and background . . . . .	152
6.3	Transformations in human annotations . . . . .	154

6.4	Transformations in algorithmic output . . . . .	156
6.5	Statistical analysis . . . . .	162
6.6	A transformation profile for each pattern . . . . .	165
6.7	Discussion . . . . .	169
6.8	Concluding remarks . . . . .	171
<b>7</b>	<b>Conclusions and Future Work</b>	<b>173</b>
7.1	Summary of the chapters and their contributions . . . . .	173
7.2	Looking Ahead . . . . .	178
7.3	Bird’s eye view . . . . .	181
	<b>Appendices</b>	<b>183</b>
	<b>Appendix A Datasets</b>	<b>185</b>
A.1	JKU-PDD . . . . .	185
A.2	MTC-ANN . . . . .	186
A.3	HEMAN and its different versions . . . . .	187
	<b>Appendix B List of Publications</b>	<b>189</b>
	<b>List of Acronyms</b>	<b>191</b>
	<b>Samenvatting</b>	<b>195</b>
	<b>Curriculum Vitae</b>	<b>197</b>
	<b>References</b>	<b>199</b>



# Chapter 1 Introduction

*De patronen in muziek en alle andere kunstvormen zijn de sleutels tot het verwerken van kennis.*

– Plato (translated to Dutch by Scherder (2017))

Music is imbued with recurring patterns and structures. Musicologists study these patterns to unlock insights into how they are arranged to form pieces of music and collections of pieces, and to gain insights into the fundamental and varied roles played by patterns in music. This investigation not only broadens our understanding of music but may also help refine techniques for music creation, analysis, and education. The employment of computational methods in this domain extends our analytical capacity: specifically, these methods can potentially identify patterns and structures across a greater number of musical pieces and more complex compositions—beyond what is manageable by human cognition alone. However, the success of these computational capabilities is contingent upon their alignment with human understanding of music, necessitating a comparison between patterns identified by humans and those extracted by computational methods. Our research is poised to delve into the issue of comparing human-annotated and computationally extracted patterns, a focus that is also related to the comparison of human annotations amongst themselves and the evaluation of different algorithms against each other.

Our motivation to undertake this research is further fuelled by the developments in multiple fields, including the current state of pattern recognition and pattern discovery algorithms<sup>1</sup>. from *Music Information Retrieval (MIR)*, corpus studies in music, and the growing significance of *Functional Programming (FP)*. Specifically, corpus studies in music enable systematic analysis of large sets of musical pieces, while *FP* offers computational modularity. For example, in (Collins, 2011), it is evident that while algorithms can process pattern discovery tasks faster than humans, they "are typically not better than humans, as measured by appropriate methods".

---

<sup>1</sup>In this dissertation, we use the phrases "pattern discovery", "pattern recognition", and "pattern-finding algorithms" interchangeably. However, "pattern recognition" is often used to refer to a broad research area, "pattern discovery" is frequently applied in a musical context, while "pattern-finding" is a more general term.

Despite this discrepancy between humans and algorithms, (Forth, 2012) and others seek computational methods to assist musicological analysis by identifying patterns that may be salient in perception. Understanding the similarities and differences between human-annotated and algorithmically extracted musical patterns, an area that still holds ample room for further research, holds potential to enrich our existing methodologies in music pattern discovery.

We begin our contributions by expanding upon existing methodologies for comparing musical patterns. We venture into human-to-human comparisons of annotated patterns, a direction that is rare but provides additional insights into automated pattern discovery. Additionally, we consider and devise four methods for visualising, combining, classifying, and employing synthetic data, thereby enriching the traditional metrics used in the field.

Building on these contributions, we turn our attention to musical transformations, an area that holds largely untapped potential for both comparing and understanding musical patterns. Traditionally, in the context of evaluating pattern discovery algorithms in MIR, comparisons have relied on direct checking of correspondences between notes in the human-annotated and algorithmically extracted patterns. However, musical transformations offer a new lens for delving into the relations within and between musical patterns, enhancing the analytical power when comparing different musical pattern discovery algorithms. Therefore, the novelty of our approach in employing transformations lies in shifting the focus from pattern occurrences themselves to the transformations that interlink them, offering fresh perspectives on the comparison of musical patterns. In the later chapters of this dissertation, we aim to develop a computational system that employs these transformations to understand and exploit the transformational relations amongst musical patterns, thereby providing a deeper level of comparison. Our research also intends to explore the programming language that can effectively handle and combine these musical transformations compositionally<sup>2</sup>.

The overarching aim of this dissertation is to enhance the methodology for comparing pattern discovery in music, both between humans and algorithms and within these groups, by offering empirical evidence and developing tools focused on symbolic, monophonic data. More specifically, we ask and address the following questions:

---

<sup>2</sup>The concept of compositionality is widely used in various fields such as language, mathematics, and programming. Here, we use the term (as well as "compositionally" and "compositional") to refer to the idea that complex structures can be built by combining simpler ones. Composition in music refers to the process of creating a piece of music.

- What is the current state of research in the area of musical pattern discovery? How broad and diverse are the concepts of musical patterns and musical pattern discovery algorithms? (Chapter 2)
- How can we efficiently collect human-annotated musical patterns? Which factors have an impact on the annotations? (Chapter 3)
- How do we compare human-annotated and algorithmically extracted musical patterns, as well as amongst different algorithms, beyond existing metrics like the  $F_1$  measures? In what ways do the patterns differ the most and in what ways are they similar? (Chapter 4)
- Can we identify the transformations that explain the variation between different pattern occurrences? More concretely, how do we implement a library to automate this identification process? (Chapter 5)
- What distinctions or similarities can we see when using a system that employs musical transformations to analyse patterns? What musical transformations do we find in patterns extracted by musical pattern discovery algorithms versus those annotated by humans? (Chapter 6)

The roadmap for the remainder of this chapter is depicted in Figure 1.1. The figure lays out the chapter’s structure, starting with an overview and background, moving on to approaches for musical pattern discovery, providing disciplinary context, clarifying the research scope, stating the thesis, and finally enumerating contributions.

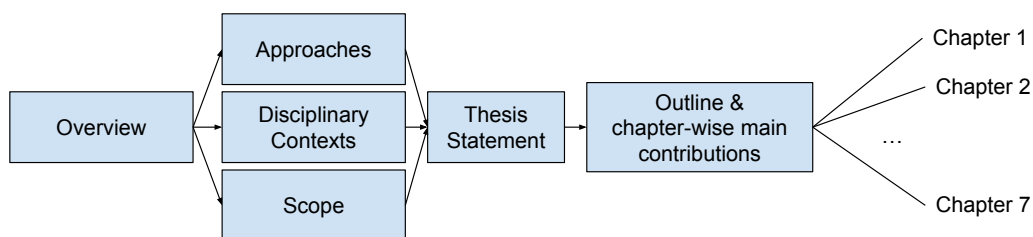


Figure 1.1: Roadmap for Chapter 1. Arrows depict the directional flow that shows associations between sections. We begin with an overview, sketching out the background and key elements of this dissertation. We then introduce the approaches we employed for studying musical pattern discovery. We also offer a summary of the disciplinary contexts we will encounter in subsequent chapters. With these diverse contexts, we distil and detail the scope of our research. Following this, we crystallise our thesis statement. Lastly, we enumerate our contributions chapter by chapter.

## 1.1 Overview

### Patterns in music: some examples

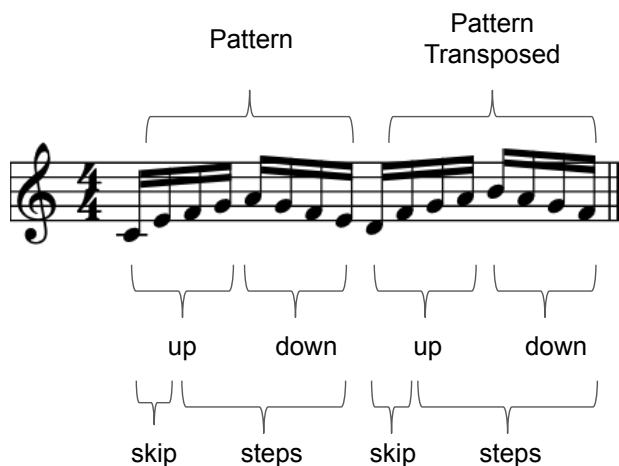


Figure 1.2: Different patterns in a piano étude by Charles-Louis Hanon (*The Virtuoso Pianist in 60 Exercises, Part 1, No. 1*). We can identify patterns by grouping notes based on transposition, repeated ascending and descending motions, or different intervallic combinations, as shown in the horizontal brackets.

For centuries, music scholars have been analysing musical scores, using concepts such as motif, theme, and unit to refer to annotated note collections, some of which have been referred to as "patterns" in MIR literature. Music scholars annotate as one way to better understand, interpret, and teach the intricate structures and thematic elements of musical pieces and corpora, thereby enriching our appreciation and knowledge of music. We provide three examples, to give a more intuitive and concrete idea of patterns that can be seen in music. Figure 1.2 shows the first few notes of a piano étude. There are a few ways that these notes can be grouped into patterns: for example, by grouping the first eight notes and noting that the remaining notes are transpositions of these; by grouping together the ascending and descending notes; or by grouping together the skips and the steps. Even in a simple sequence of notes, describing all the patterns that can be discerned is hardly a simple matter.

A more musically interesting example is presented in the two bars shown in Figure 1.3. This example shows how listeners can have two different interpretations of what the musical patterns are. In fact, this example has been studied: Gabriellson (1987) identified that this piece had been published in different editions with differ-





Figure 1.3: Different patterns in Mozart’s Adagio K.331. Two horizontal brackets indicate the two ways in which the music can be interpreted in terms of patterns (Gabrielsson, 1987).

ent phrasings<sup>3</sup>. Furthermore, it was also examined through the lens of the Generative Theory of Tonal Music by Lerdahl *and* Jackendoff (1985). As part of this research, piano performances of the piece with differing interpretations that correspond to these different phrasings were analysed. This is a prime example of *ambiguity* in music interpretation. One may either accept both interpretations or favour one over the other.

In Figure 1.4, including again the piano étude, we describe a few more examples in plain language and in musical terms. Without relying on specific musical terms, we can describe how these pattern occurrences relate to each other as follows: in (a), the second occurrence is based on the first but higher in pitch; in (b1), the second line of melody is nearly identical to the first, with a minor change in the middle; in (b2), the entire melody is almost an exact repetition of (b1), with minor changes at the beginning and the end; in (c), the two bracketed areas are largely identical, with subtle differences in pitch.

With more precise vocabulary in music, we can describe the pattern occurrences as follows: in (a), the second occurrence is a tonal transposition of the first, shifted upward by a whole tone; in (b1), a crotchet in the first line is split into two quavers in the second; in (b2), the same rhythmic changes as in (b1) apply; in addition, the second line starts off-beat, and pitches of three notes are changed in comparison to (b1); in (c), the melodic motif of the first occurrence is transposed by a semitone in the second occurrence, and the underlying harmony changes from B flat major to F dominant seventh, and then reverts to B flat major.

<sup>3</sup>Phrasing refers to the way musical elements are grouped and articulated.

# 1 Introduction



Figure 1.4: Example musical patterns in (a) Piano étude by Charles-Louis Hanon (The Virtuoso Pianist in 60 Exercises, Part 1, No. 1) (b) Dutch folk song: Daar waren drie loze gezellen (There were three idle journeymen) (3) Classical music: Mozart's Minuet, K. 282. Pattern occurrences are indicated by brackets and bolded lines. Circles are used to highlight the differences between the occurrences.

## Patterns discovered by humans and by algorithms

The descriptions above illustrate just one possible set of relationships between pattern occurrences amongst many that may be identified by humans. In other words, given the same musical excerpts, there might be alternative or multiple interpretations of the patterns contained within them from different people. Given the diversity of musical patterns, we can therefore begin to appreciate the substantial complexity that can come with automating musical pattern discovery. This complexity includes, but is not limited to, reconciling different time scales as well as different

degrees of subjectivity and ambiguity. While we can only provide a limited number of examples here, the potential of how these examples generalise is even more interesting if one considers all the pieces of music that have been written throughout human history and all the patterns that could be extracted from them. All of these ambiguities and varying interpretations pose difficulties for algorithms to work in a satisfactory way in all contexts.

Over the past few decades, many pattern-finding algorithms have been developed. Music, rich in its diversity, structure, and cultural and artistic values, is witnessing increasing automation in its creation and analysis. This dissertation looks into algorithms that can discover patterns in music.

One persistent issue with pattern-finding algorithms is that humans and algorithms tend to find different patterns in different ways. For humans to recognise patterns, a number of cognitive capabilities are typically necessary, such as the ability to conceptualise and keep the patterns in their minds, as well as the ability to recognise novel developments and maintain the goal of pattern identification throughout the process. With a whole host of other differences (Von Neumann *et al.*, 1958), such as energy efficiency, data efficiency, and complexity, the patterns found by mechanic algorithms can be perplexing and divergent from the patterns perceived by humans. For example, we often see the complexity of algorithmic output exhibited in the sheer number of the patterns discovered by machines.

Pattern recognition research aims to bridge the gaps. The gradually increasing resemblance, as well as some persistent dissimilarities of machine intelligence to that of humans, prompts comparisons between the two. This comparison, given the diversity of methods in pattern discovery, is not an easy task. A precise division between what is recognised as a pattern and what is not is rare, even amongst humans, who often disagree based on differing schools of thought and perspectives on pattern recognition. When there are multiple algorithmic results and multiple human judgements, a careful comparison would be needed.

## **Implementation and functional programming**

The common practice of separating the development of high-level utilities from the intricacies of low-level coding often results in the occasional oversight of the relevance of software design and programming languages. While the need for alignment between the design of implementations and their higher level goals—especially in terms of capturing appropriate abstractions—is broadly recognised amongst algorithm designers, our work seeks to underscore this point. In this dissertation, we illustrate this point by detailing the considerations behind our

decision to implement our tool in the functional programming language, Haskell (Peyton Jones, 2003).

Functional programming languages, as the name implies, are focused on using functions (in the mathematical sense) and their compositions. For our purpose of working with musical patterns and investigating how pattern occurrences relate to each other, a functional programming language is an advantageous option—we can then use functions to encode the musical transformations between pattern occurrences and make use of functional programming features to devise a more natural and expressive way to work with patterns and transformations.

### The big picture of musical pattern discovery

To summarise and provide an overarching view of the musical pattern discovery process explored in this dissertation, we present a schematic diagram in Figure 1.5. From the agent level (where agents are the algorithms or human annotators), to the corpus level (the collection of musical content), to the intermediate levels (such as a symphony, a tune, a tune family, and a movement of a piece), to the patterns in the piece, to the occurrences within these patterns, and eventually, to how these occurrences relate to each other, we will look into these relations between the pattern occurrences and infer back to the piece, the corpus, and the agent levels.

## 1.2 Approaches to studying musical pattern discovery

With many crucial references to musical pattern discovery, [Music Information Retrieval \(MIR\)](#) is the basis for this research. We harness various computational methods from MIR and draw on knowledge and insights from the fields of musicology, music theory, and music perception. Specifically, we incorporate musical examples and theoretical concepts from the literature in these areas. Modelling, category theory, and software engineering perspectives from the research area of functional programming are also considered. More concretely, we employ the following relevant approaches:

- Statistical analysis: used to compare between sets of patterns.
- Machine learning: classification in analysing differences between sets of patterns, synthetic data, with the emphasis on interpretability and explainability.
- Functional programming: using Haskell programs to model the relations between musical pattern occurrences.

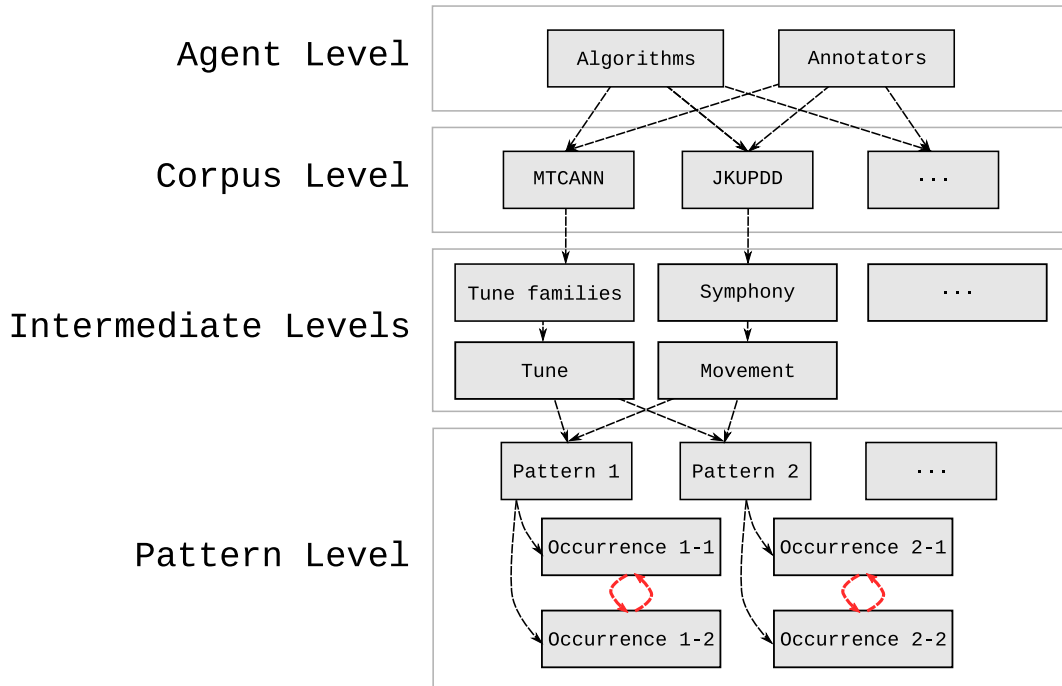


Figure 1.5: Schematic diagram illustrating musical pattern discovery on different levels. Starting from the top, we have different pattern discovery agents that operate on different corpora, which have different intermediate levels. Within these intermediate levels, we have musical patterns and their occurrences. The red arrows denote the relations between occurrences, which will be one of the foci of this dissertation, particularly in Chapters 5 and 6.

- Category theory: gives a mathematical framework behind the implementation of our library/*Domain Specific Language (DSL)* in Haskell.

We draw from recent advances and traditional methods in these areas to develop multiple kinds of analysis for musical patterns. We try to recognise the relevant discourse within various disciplines in the process.

Towards the end of the dissertation, we propose an approach involving the use of musical transformations. To study the differences and commonalities between different annotators and algorithms, we employ a set of transformations to compare different pattern occurrences by grouping these occurrences according to the transformations between them. The transformations we consider are musically meaningful and have been employed as techniques in composing music (Schoenberg, 1967), such as transposition in pitch and time. Using transformations, we compare patterns from different annotators and algorithms based on their underlying relations, as described through musical transformations, which sheds light on the potential and effective criteria these annotators and algorithms may have used to discover the patterns.

## 1.3 Disciplinary contexts: a summary

In this section, we provide a brief summary of the disciplinary contexts underpinning this research. We only offer a high-level summary of the topics here, and each chapter provides a more comprehensive overview of relevant work. Furthermore, Chapter 2 is devoted to establishing the academic backgrounds and contexts of our work.

### 1.3.1 Pattern discovery in music information retrieval

Established in 2005, the *The Music Information Retrieval Evaluation eXchange (MIREX)* initiative gathers tasks such as chord estimation and key detection to facilitate comparison of algorithms within MIR. The pattern discovery task in MIREX started in 2014. The full name of the MIREX pattern discovery task is the "Discovery of repeated themes & sections" task. Other typical MIREX tasks include segmentation, beat tracking, and chord labelling. Evaluation metrics that have been widely used are cross-entropy, accuracy, precision, recall, and F1-scores. We make use of the dataset and algorithms submitted for this "Discovery of repeated themes & sections" MIREX task. As the patterns and algorithms we compare are closely related to this task, we will provide more information about this task in subsequent chapters.

### 1.3.2 Musicology, music theory, psychology, and cognition

In this dissertation, we delve into the organised aspect of music, as defined by Edgar Varèse who described music as "organized sound" in the 1940s (Varèse, 1940). Our exploration, specifically revolving around musical patterns, draws inspiration from several areas of research such as musicology, music theory, psychology, and cognitive science.

Varèse's description separates out two aspects of music: the abstract structure (organisation) and the physical vibration (sound). While concise and potent, this definition is not neutral or without controversy, and it reflects Varèse's particular aesthetic vision for music. Speech, for example, can be seen as organised sound, but not as music in the traditional sense. Furthermore, some of the organisations employed by composers do not have to be audible by listeners. More generally, it is difficult to be exact with the meaning of the words "organised" and "sound", because they are concerned with human cognition and perception. In the broadest sense, cognitive science is the analytical study of the human mind (Lent, 2017). Perception is mostly concerned with sensory input, but it may also be considered as a form of a cognitive

process. Both perceptual and cognitive aspects are studied under the psychology of music (Deutsch, 2019). Music is fundamentally psychological, as it can hardly be considered music if it does not involve human listening experience.

Music cognition and perception are interwoven with music theory and musicology. Ockelford (2017) provides a summary of the diversity of music theory and analysis:

"...music theory and analysis take many different forms, and various attempts at classification have been made. Nicholas Cook, for example, divides analysis into 'traditional methods', such as those of Donald Tovey and Charles Rosen; 'Schenkerian analysis'; so-called 'psychological approaches' (principally those of Leonard Meyer and Rudolph Réti); 'formal approaches' (including the work of Allen Forte and Jean-Jacques Nattiez); and 'comparative analysis' (through Charles Adams's classification of melodic contours, for example, and Alan Lomax's 'cantometrics')."

While it is possible to use specialised music-theoretic terminology for each genre and individual composer, our aim is to explore computational modelling that can model symbolic music more generally. From this general aspect, we are motivated by the importance of repetition and variation in music theory and psychology. In fact, repetition and variation are vitally important in music research (Huron, 2006; Margulis, 2014; Ockelford, 2017; Temperley, 2014; Volk *et al.*, 2012; Zbikowski, 2002), and they are one of the key themes of this dissertation, as reflected in its title.

### 1.3.3 **Functional programming, pattern, and transformation**

To realise any computational idea from the ground up, we can either write programs in a specific programming language or rely on pre-existing software that we are familiar with and trust. Although writing code or using software is not always considered a scientific contribution in and of itself, they are often the very things upon which we base our analysis.

For our analysis, facing the choice between different software or programming languages, we opted for a functional programming language. Several advantages of using the functional programming language Haskell include the support for compositionality and modularity, ease of using higher-order functions<sup>4</sup>, and a powerful type system<sup>5</sup>. More concretely, while we do not use Haskell's advanced type-level

---

<sup>4</sup>Higher-order functions are essentially functions that take other functions as input and return functions as output.

<sup>5</sup>Type systems aid in managing types in programs. Broadly speaking, types are properties that are assigned to terms, such as variables and functions.

features such as type families and data kinds, and any other strongly-typed functional programming language could suffice, we nevertheless employ typeclasses<sup>6</sup>, contravariant bifunctors<sup>7</sup>, and define higher-order functions in our Haskell implementation, thereby enhancing the compositionality and modularity when using musical transformations to examine musical patterns.

Combining the thinking of functional programming and pattern discovery, we arrive at the conclusion that patterns can be examined through transformations, which can be encoded as functions and used compositionally in Haskell. Musical transformations are common in composition, analysis, and psychology, and can be manipulated in Haskell with ease.

### 1.4 Scope

In this section, we explain the concrete scope of the dissertation. Our analyses are scoped by using monophonic, (semi)symbolic MIDI data, MIREX algorithms and format of musical patterns, with a focus on intra-pattern and inter-occurrence analysis. We believe that within this well-defined scope, albeit small in contrast to the complexities of such large topics as patterns and music, we can make direct and deep contributions to the field. We expand the discussion on our choice of scope below.

#### 1.4.1 Restriction to monophonic data

We concentrate on and use monophonic data in this dissertation because monophonic data removes the complexity of polyphony, paraphony<sup>8</sup>, or heterophony<sup>9</sup>. Pattern discovery in monophonic music is not a trivial problem either, as there are many intriguing melodic designs in folk music, jazz, pop, classical, and plenty of research focuses solely on monophonic music. We use monophonic data as the first step toward greater complexity.

However, we do bear in mind that we will miss the cross-voice patterns, harmonic patterns, and abstractions to chords. This underlying harmony is often a more efficient representation, and using monophonic data might be extra challenging in

---

<sup>6</sup>In Haskell, a type class allows functions to operate on multiple types that share certain behaviour (for more details, see Chapter 5).

<sup>7</sup>A contravariant bifunctor in functional programming is a type constructor that is contravariant in one argument and covariant in another (for more details, see Chapter 5).

<sup>8</sup>Paraphony is when multiple voices or instruments play the same melody while sharing some elements, such as playing in intervals of fifths.

<sup>9</sup>Heterophony is a musical texture where multiple voices or instruments perform variations of the same melody simultaneously.



some cases if we do not consider the underlying harmony. Some of the monophonic data we use are in fact derived from polyphonic music using the clipped-skyline approach (detailed in Appendix A.1) to select notes from the polyphonic pieces.

### 1.4.2 Restriction to symbolic data

We focus on (semi)symbolic data because it provides us with more directly interpretable information which, when compared to acoustic signals, is less obscured by extraneous elements such as the recording environment and devices. Converting from acoustic information to symbolic representation is known as automatic transcription, which is a research topic of its own.

More specifically, we opted for *Musical Instrument Digital Interface (MIDI)* as the main input format, primarily due to its ease of playback and its status as a standard format for many software applications. MIDI is sometimes also known as (semi)symbolic rather than symbolic because it lacks the power to express certain constructs in sheet music, such as pitch spelling. With certain algorithms, we use other types of input, the MIREX Lisp and CSV formats, which added in morphetic pitch<sup>10</sup> (Meredith, 2006).

### 1.4.3 Datasets and generalisability

The size of datasets matters as it is related to the problem of overfitting, which we will explore more in Chapters 3 and 4. With the recent investigation into the biases in music theory (Ewell, 2020), there has been an outcry that we must confront continued reliance on small, unrepresentative corpora. In music theory and musicology books, some typical numbers of pieces and examples analysed are approximately 40 pieces for Schenker, 288 examples for Caplin, and 552 examples for Hepokoski Darcy (London, 2021). Amongst these, 25/40, 288/288, and 383/552 are from the composers Bach, Haydn, Mozart, and Beethoven (London, 2021). It also was argued that a lack of diversity could have significant effects on how we think about musical structure (how we "do music theory") and on how we think about how we hear and understand music (how we "do music cognition"). We, therefore, strive to include a diverse range of music data, but some of the available datasets we use do have a bias towards these classical Western composers as well.

<sup>10</sup>Morphetic pitch refers to the musical pitch of a note in relation to its function within a specific tonal context. Please refer to (Meredith, 2006) for more detail.

An appendix at the end of this dissertation provides detailed information about the datasets used in the various chapters. We also provide a summary of the dataset characteristics when used in each chapter.

#### 1.4.4 Format of patterns

The main format we use to encode patterns and their occurrences is the MIREX format. Algorithms submitted to the MIREX platform all produce this type of encoding as shown in the example after this paragraph of text. The format always starts with `pattern1` on the first line, `occurrence1` on the second line, followed by the timestamp and MIDI pitch value of the notes consist of the patterns on the following lines. Each note takes a line, until the end of the pattern occurrences, where the next heading of `pattern i` or `occurrence i` occupies the next line. The encodings we use are widely and currently used in the research of pattern discovery in music (de Reuse & Fujinaga, 2019; Meredith, 2019).

```
pattern1
occurrence1
17.00000, 74.00000
17.50000, 74.00000
18.00000, 74.00000
21.50000, 74.00000
22.00000, 72.00000
occurrence2
26.00000, 69.00000
26.50000, 69.00000
27.00000, 69.00000
30.50000, 69.00000
31.00000, 67.00000
occurrence3
46.00000, 71.00000
46.50000, 71.00000
47.00000, 71.00000
50.50000, 71.00000
51.00000, 69.00000
occurrence4
...
occurrence19
pattern2
```

occurrence1

...

pattern45

...

### 1.4.5 Inter- Intra- opus and patterns

Conklin (2010) presents two forms of pattern discovery: inter-opus (discovering patterns recurring across a number of pieces in a corpus) and intra-opus (discovering patterns repeating within a single piece). To simulate intra-opus from inter-opus, one can concatenate multiple pieces into a single file. However, this approach does not render them exactly equivalent, as patterns may be discovered at the boundaries of the concatenated pieces. In addition, the order of the concatenation becomes important, and the memory requirements for the two tasks differ substantially.

The human-annotated patterns in *The Meertens Tune Collections: The Annotated Corpus (MTC-ANN)* are inter-opus, while other annotations are intra-opus; thus, we do have a focus on intra-opus pattern discovery. We also use the simulation method described in the previous paragraph.

This distinction between inter- and intra- can be applied to the analysis and patterns, too. Inter-pattern analysis would focus on analysing patterns in the broader context of multiple patterns, whereas intra-pattern analysis would focus on the relations between occurrences within a single pattern. Our transformation-based approach is very much suited to the intra-pattern analysis.

## 1.5 Thesis statement

Now that we have defined the scope, inspected disciplinary contexts, considered the approaches and the background overview, we can encapsulate a thesis statement for this dissertation. In this dissertation, we strive to make the following argument:

Human-perceived musical patterns, a type of highly subjective and ubiquitous abstraction, have important connections to musical transformations, which, when explored compositionally through a functional programming language, informs both human-to-algorithm and algorithm-to-algorithm comparisons of discovered musical patterns.

We unpack this sentence into three main aspects: understanding patterns, establishing connections to transformations, and employing a functional programming language. Chapter 2 focuses on the understanding of patterns and existing musi-

cal pattern discovery algorithms. Chapter 3 examines human-annotated patterns and potential tools to facilitate the annotation process. Chapter 4 compares musical pattern discovery algorithms, introducing four methods for comparison and revealing the critical role of rhythmic features. Chapter 5 presents our implementation of the musical transformations in Haskell, the functional programming language of our choice. Chapter 6 delves into the analysis of musical patterns using our Haskell implementation, along with an array of other computational methods. A more extensive description of the outline of this dissertation is provided in the ensuing section.

### 1.6 Dissertation outline and chapter-wise main contributions

The overall structure of this dissertation takes the form of seven chapters. We will go through the chapters, stating their connections to our publications (also see Appendix B) and highlighting their contents and aims, while noting the challenges encountered.

#### Chapter 1 Introduction

This chapter provides the reader with a bird's eye view of the dissertation. We describe the approaches used in this dissertation. We also go over the research landscape and the scope we take. We put forward our thesis statement and the outline of the dissertation to substantiate the thesis.

#### Chapter 2 Patterns and Pattern Discovery

In this chapter, we introduce relevant work centred around the concept of musical patterns. By examining the literature in the field of musicology, music theory, and pattern discovery algorithms, we demonstrate the breadth and diversity of the field of musical pattern discovery. We create a list of related concepts to establish their relevance (or lack thereof) to this dissertation. We also introduce algorithms that discover musical patterns.

Through this process, we grapple with the following challenges: the concept is studied in various fields such as MIR, cognition, and musicology, each with their own terminologies and concepts—in other words, no agreed-upon definition exists for what constitutes a "pattern". Patterns are also central to music and are intertwined with other crucial concepts like similarity, boundaries, and segments, which

leads to them being studied within these different contexts for various purposes. Amidst this, a plethora of pattern discovery algorithms exist, involving different methods, and which would work best in what context remains unclear. This chapter aims to provide an overview of this complex landscape and to highlight the need for comprehensive methods of algorithm comparisons to advance the field of pattern discovery in music.

### Chapter 3 Gathering Human-Annotated Musical Patterns

This chapter is based on the following published papers (Ren, Koops, *et al.*, 2018; Tomašević *et al.*, 2021)<sup>11</sup>. In this chapter, we discuss issues regarding human-annotated musical patterns, including how annotation tools and the annotators' musical backgrounds affect the annotated patterns. Our contribution includes the development, comparison, and discussion of two annotation tools and experiments. We consider their functionality, the datasets they generate, and the consequential impact that using such reference data has on the evaluation process of pattern discovery algorithms.

Through this process, we encounter the following challenges: there is a lack of annotated musical patterns datasets to serve as reference data for evaluating the algorithms in the field. As a result, human-algorithm comparisons are often based on a small pool of annotators, leaving us little understanding of how much annotators agree or disagree, which is not a good base for evaluating the algorithms. Therefore, this chapter aims to develop digital tools for data gathering and to analyse their impact on both the annotation process and the annotated patterns.

### Chapter 4 Comparisons of Musical Pattern Discovery Algorithms

This chapter is based on the following published papers (Ren *et al.*, 2017; Ren, Volk, *et al.*, 2018). We examine up to thirteen musical pattern discovery algorithms. To compare them, we propose four methods as our main contribution of this chapter:

- **Location and Feature Visualisation, a method we devised (LFV)**: We devise new visualisation methods as well as using existing ones to examine a large number of patterns.
- **Pattern Polling, a method we devised (PP)**: We devise PP to extract musical patterns based on consensus from various algorithms. We demonstrate that while there are encouraging correspondences between our results and human

---

<sup>11</sup>Tomašević was the primary contributor to the implementation of the annotation tool, PAF, while Ren contributed to the development of the methodologies and provided input for the analysis, as well as participated in subsequent revisions of the paper.

annotations, discrepancies between the outputs from different algorithms pose a major barrier to achieving significant improvements in accuracy.

- **Comparative Classification, a method we devised (CC)**: We leverage the discriminative power of classification algorithms to compare features extracted from algorithmically discovered, hand-annotated, and randomly selected excerpts. In this process of distinguishing between musical patterns of different origins, we discover that rhythmic features play the most important role.
- **Synthetic Pattern Insertion, a method we devised (SPI)**: We examine various algorithms using synthetic data (random data with musical patterns planted within it) to better understand the behaviour of algorithms. We discover that some algorithms perform as expected and retrieve the planted patterns while others do not.

Through this process, the complexity and diversity of musical pattern discovery algorithms pose challenges, particularly when algorithms employ black-box optimisations or other types of complex mechanisms that produce a potentially large amount of difficult-to-interpret results—we encounter difficulties in selecting, explaining, and understanding the algorithms. Existing evaluation methods, such as numerical comparisons in MIREX that focus on pitch-and-onset-content, offer limited insight into the specific strengths and weaknesses of individual algorithms. Moreover, human-annotated and algorithmically extracted musical patterns tend to differ, which warrants systematic investigation for improving the algorithms. This chapter aims to extend existing approaches by developing LFV, PP, CC, and SPI.

### Chapter 5 Modelling Patterns with Transformations using Haskell

This chapter is based on previously published work of (Melkonian *et al.*, 2019)<sup>12</sup>. Using transformations, we create a framework for comparing musical patterns by describing the relations between pattern occurrences, drawing upon well-known abstractions from category theory. We also provide a Haskell implementation of the model, *Pattrans*, in the form of an embedded DSL, which is the main contribution of this chapter.

Through this process, we face the continuing challenge that existing comparison methods are largely centred on the individual elements of pitch-and-onset content within patterns. However, we believe that assessing the usefulness of the extracted patterns for specific contexts hinges upon capturing the relations between the occurrences that make up these patterns. To realise this perspective, this chapter aims

---

<sup>12</sup>Melkonian was the main contributor to the code base, while Ren focused on developing the conceptual framework and providing the necessary data.

to develop an efficient tool to articulate these relations in terms of musical transformations.

### **Chapter 6 Using Transformations to Understand the Relations between Pattern Occurrences**

By applying Pattrans to both human-annotated and algorithmically extracted patterns, we classify pattern occurrences according to the transformations that relate them to the prototype pattern. Subsequently, we compute the relative percentages of specific transformations and compare these across different annotators and algorithms. Using this data, we explore how to use musical transformations to relate and classify musical pattern occurrences, thereby demonstrating the efficacy of this approach in examining musical patterns. This chapter contributes to uncovering how a large proportion of the pattern occurrence relations in human annotations can be explained by a limited number of transformations we have implemented in Pattrans, such as exact repetition and chromatic transposition. We also demonstrate that human-annotated patterns and several algorithmically extracted patterns contain radically different proportions of musical transformations in certain datasets.

Through this process, we further confront the challenges of comparing pattern discovery algorithms, particularly in terms of the algorithms' large output size and the difficulty of relating these output patterns back to meaningful musical concepts. Therefore, we aim to use Pattrans to describe the relations between occurrences of patterns through transformations as musically meaningful concepts, as well as to group the pattern occurrences using musical transformations, thereby managing the large output size.

### **Chapter 7 Conclusions and Future Work**

This chapter summarises the dissertation, encapsulating the main arguments, methodologies, and findings. In addition, this chapter discusses potential applications of this research, outlines its limitations, and suggests avenues for future work.





# Chapter 2 Patterns and Pattern Discovery

*Humans are pattern-seeking story-telling animals, and we are quite adept at telling stories about patterns, whether they exist or not.*

– Michael Shermer

## 2.1 Introduction

### Contribution of this chapter

In this literature review chapter, we examine an array of definitions of the term "musical pattern" and its related concepts in MIR and music theory, as well as a selection of musical pattern discovery algorithms to address the following questions:

- What is the relevant work in this area of research?
- How broad and diverse are musical patterns and musical pattern discovery algorithms?

By examining the literature, we demonstrate the breadth and diversity of the field of musical pattern discovery. We will also discuss the potential challenges posed by the field's breadth and diversity.

### Musical patterns

When speaking of musical patterns, a diverse range of examples might come to mind, such as the purposeful patterns in études, the slick patterns in jazz, the elegant patterns in minimalist music, the textbook examples of fugues, a few notes from folk music, and themes and motifs from classical music. If one has a stronger affinity for music theory, constructs such as schemata, leitmotif, subject, and counter-subject might come to mind.

Musical patterns have been described in different ways, too. Assuming motifs are a type of pattern, the following are several different descriptions of a motif. Ac-

According to Webern, they are "the smallest independent particle in a musical idea" (Webern, 1963). For Whitehead, they are a "structural unit possessing thematic identity" (Whitehead & Price, 2001). Schoenberg commented from a different point of view that it is "a unit which contains one or more features of interval and rhythm [whose] presence is maintained in constant use throughout a piece" (Cambell, 2010). Webern, Whitehead, and Schoenberg used an array of words such as structure, theme, segment, unit, and many more to describe this one type of pattern. What would we use to connect the terms?

We make the connection that all kinds of patterns share a certain degree of repetition and variation. Given the importance and relevance of repetition and variation in music, in this dissertation, we take the studies behind musical repetitions (Huron, 2006; Margulis, 2014; Ockelford, 2017; Temperley, 2014; Volk *et al.*, 2012; Zbikowski, 2002) as one of the inspirations for this research.

### Patterns and their repetition in music

There are two approaches to reaching a deeper understanding of the patterns in existing music: one can ask listeners to point them out (a data-driven, bottom-up approach), or one can theorise and define *a priori* the interesting patterns to find (a top-down, rule-based approach). For the first approach, one needs to conduct general listener annotation experiments, which will be the topic of the next chapter. For the second approach, one needs to examine different theories of music in further detail, which is what this chapter will do.

Musicians have been found to agree to a degree on the importance of patterns in music, although this agreement is not always perfect (Collins, 2011; Giraud *et al.*, 2016; Ren, Koops, *et al.*, 2018). Several contentious questions include the following:

- When does a pattern begin and end?
- How important must a passage be in order for it to be called a pattern?
- What is the minimum length of a pattern?
- Have all of the occurrences of a pattern been discovered?

We will discuss later in the chapter how patterns connect to other concepts and features such as boundaries and length of a pattern.

### Musical pattern discovery algorithms

The diversity of musical patterns we address in this chapter is something to keep in mind when automating the pattern discovery process. If we would like to create a generalisable algorithm to discover and reuse the patterns in our data, we will

also run into the question of how to compare patterns discovered by humans and machines, which we will address in Chapter 4. In this chapter, we will make some connections between the concept of musical pattern and musical pattern discovery algorithms, as well as introduce a selection of algorithms, including those that will be used in our experiments in subsequent chapters (Chapter 4 and 6).

## 2.2 Pattern as a concept

This section is dedicated to providing multiple viewpoints on the concept of pattern. In our context, having these concepts of what a pattern is can be important in terms of what it might entail for the design of the annotation experiments, algorithms, and their evaluation.

### Definition from previous works

We can see different definitions of pattern in the field of MIR:

- "A melodic pattern is defined by a set of either identical or 'equipollent' (i.e. significantly similar) sequence segments." (Rolland, 1999)
- "The term 'pattern' here basically refers to N-grams, i.e. subsequences, of melodic abstractions." (Pfleiderer *et al.*, 2019)
- "A pattern is defined as a set of ontime-pitch pairs that occurs at least twice (i.e. is repeated at least once) in a piece of music" (Collins, 2011) and (Collins, 2014), which is also the definition used in the [MIREX](#) task.

These definitions seem to agree that a pattern is made up of a few notes that repeat exactly or inexactly. This commonality leads to two issues, at least. The first is to determine when an inexact repetition ceases to be a repetition. The second is to distinguish patterns that are trivial or created by chance from patterns that are more worthwhile and created by design.

Both issues have also been discussed in previous works of musical pattern discovery algorithms. Regarding the distinction between exact and inexact repetition, we see in (Forth & Wiggins, 2009) that "...repetition may exist in many forms beyond the exact repetition of musical events in sequence... In the context of computational analysis, therefore, careful consideration must be given to the notion of pattern equality". Regarding the difficulty to find more meaningful repetition, it was mentioned in (Meredith *et al.*, 2002) that "...the identification of perceptually significant repetitions is an essential step in the process by which an expert listener inter-

prets a musical work... However, the vast majority of exact repetitions in music are not perceptually significant".

We agree with the above views. We also note that, more specifically, a short sequence is likely to occur often in all kinds of music by chance; for example, two successive notes differing by a whole step. Therefore, we would naturally doubt whether such two-note combinations are meaningful patterns where a composer thought through it as a distinctive figure worthy of repetition and development. Longer note sequences that occur repeatedly are more likely to be a product of design than of chance, but they are more likely to have complex changes between occurrences that render the inexact repetition cease to be a repetition. Throughout the chapters, we have these two issues as a central theme of our discussion.

### **Intensional and extensional definitions; necessary and sufficient conditions**

Having reviewed a series of definitions, let us distinguish two common approaches to giving definitions. Intensional definitions are given by necessary and sufficient conditions. Extensional definitions are given by a compilation of things that may come under the definition. So far, the majority of what we have seen is intensional. An extensional definition stems from a dataset, something like the figure examples given in Chapter 1, in affinity to the data-driven approach.

These two types of definitions are also referred to as intensional specifications and extensional realisations. We introduce them here, not to have a full ontological and epistemological debate, but to acknowledge that *pattern* can be defined in at least these two different ways.

For intensional definitions, the next question to ask is what are the typical variables in the necessary and sufficient conditions for a passage in the music to be identified as a pattern. Sufficient conditions provide the predicates of the purposes of patterns, showing possibly some rules and algorithms to extract the patterns. Necessary conditions are characteristics, more likely to be inferred from statistical data analysis. Necessary conditions provide clues as to what is more likely to be musical patterns, and they are also filters as to what is preferred not to be regarded as patterns in a certain type of corpus or for a specific purpose. For example, there have been filters based on length, frequency, spacing, and similarity (Janssen, 2018). When a condition is both sufficient and necessary, we arrive at an "if and only if" condition, which we have not come across in the literature we examined.

To give some idea, a sufficient condition for a part in music to be a pattern could be as follows: if passage  $A = \text{passage } B$ , then  $A$  and  $B$  are the two passages that form the musical pattern. In other words, only when two passages are an exact repetition

of each other do they belong to a pattern. Another example might be that, if  $A = \text{Transposition}(B)$ ,  $A$  and  $B$  are the two passages that form the musical pattern. In other words, up to transposition, two passages are considered as belonging to a pattern.

A very loose necessary condition could be the following: passage  $A$  and passage  $B$  have to have at least  $n$  intervals in common to be considered as belonging to a pattern. We may call this type of pattern intervallic patterns. A stricter necessary condition could be that passage  $A$  and passage  $B$  must at least have the same rhythm to be considered as belonging to the same pattern. We may call this type of pattern metric patterns. Some datasets and algorithms impose the necessary condition that passages have to be of length  $l$  to be considered (belonging to) a pattern. We believe these type of conditions can be useful in certain scenarios, but when used without attention to context and with unsuitable parameters, results can be limited in their fruitfulness.

### **Patterns that fall in the scope of interest**

As mentioned at the beginning of this chapter, we are interested in repetition and variations and their relations to patterns. In later chapters, we will see that this leads us to the concept of transformation and use it to describe how one pattern occurrence connects to another.

We are also going to take excerpts of music as musical patterns from human annotations and algorithmic output. These excerpts can be viewed as a kind of extensional definition for some humans and algorithms, and we would like to approximate them by using our intensional definitions.

### **The categorisation of patterns is not considered**

One can categorise musical patterns according to different musical dimensions (Volk *et al.*, 2012). For example, one can differentiate between temporal patterns, pitch patterns, melodic patterns, and harmonic patterns. In this dissertation, we do not make hard distinctions between these categories. The reason for this is twofold. First, as our focus is on monophonic data, we do not have a strong tie with harmonic patterns, although we acknowledge that they are of extreme importance in many applications, as discussed in Section 1.4.1. Second, there is no guarantee that no patterns lie in the overlap or the gaps between these categories. We do, however, concede that a taxonomy or ontology of musical patterns could give useful insight.

## 2.3 Musical patterns and related concepts

In previous sections, we have visited a variety of definitions of pattern. This section examines a range of terms associated with the concept of patterns, including segment, structure, motif, and so on. We will see their (dis)similarity to the word pattern, as well as create connections to some algorithms where these terms are used.

### 2.3.1 Related concepts in MIR

In this section, we introduce seven concept groups that are frequently encountered in MIR and other generic fields of related research.

#### Boundary

There are many symbols for marking boundaries in musical notation, such as the bar lines and the repeat sign. There are also boundaries created by rests in the music, analogous to the punctuation in natural language. However, there also exist boundaries in music that are not written out; for example, within a bar of dense notes without rest, it is still possible for humans to demarcate one part from another, as we have shown in previous chapters, especially in Figure 1.2.

In relation to pattern and repetition, we largely concur with the statement that "patterns are defined primarily by virtue of repetition; cues such as surrounding silence, phrase boundaries, and measure lines are secondary factors that composers and performers can exploit to highlight or hide pattern occurrences." (Collins *et al.*, n.d.). Notated boundaries and boundaries that are not written out are indeed sometimes the boundaries of musical patterns. Conversely, given musical pattern instances, the beginnings and endings of these instances may serve as boundaries in the music. There are also patterns that cross boundaries, such as in jazz music, where patterns on sheet music can go over the bar lines.

In terms of applications, finding boundaries can be an important step in automating the editing, analysis, and creation of music, if not more. When one attempts to automate boundary detection in music, one could start by examining the boundaries directly or indirectly by examining musical patterns when they are available. The above-mentioned method has been explored in (Müller & Grosche, 2012; Rodríguez López & Volk, 2015) with promising results.

### Segment and section

Both the terms "segment" and "section" in the music domain refer to a passage of music, with the conventions sometimes that sections are longer than segments. It is very common to spot segments and sections in music, just as in the visual domain: segments and sections of varying sizes can often be seen in virtually every composition of image and sound.

The segmentation task in [MIR](#) is the act of performing segmentation to break a whole into parts. As there has been no ongoing study on the "sectionalisation" task, we will focus on segmentation.

Performing segmentation in music is inextricably linked to the boundary detection and pattern discovery we discussed previously. While the boundaries of patterns may delimit their beginnings and endings, performing segmentation often does not distinguish between the beginnings and endings—the ending of a previous segment is the beginning of the next segment. There are two more differences between pattern finding and segmentation:

- Unlike segmentation, the output of a pattern discovery algorithm or process does not always cover the entire piece of music, while segmentation normally does.
- Patterns can overlap with and nest within each other, while the segments do not usually overlap.

There are commonalities between the pattern discovery and the segmentation task in terms of the challenges faced by both. The boundaries of segments are very likely to correspond to the boundaries of patterns. Like the concept of pattern, the concept of segmentation also connects to music-theoretic concepts such as figure, phrase, and section (Rodríguez López, 2016). As put in (Monelle, 2014), a pessimistic view about segmentation is that "Segmentation in music will always be ultimately based on intuition, because the relation of phonology and semantics, of expression and content, functions different in music." There have, however, been strides made on this front by the MIR community (Rodríguez López *et al.*, 2014; Wiering *et al.*, 2009). In addition, more discussion between the concept of pattern and segment can be found in previous work of musical pattern discovery algorithms such as (Lartillot, 2004; Velarde *et al.*, 2016).

### Feature and viewpoint

A feature is a measurable property or characteristic of a phenomenon being observed (Bishop, 2006). This term is widely used in the context of machine learning and pattern recognition.

A related term in computational music analysis is the *viewpoint*, which is introduced by Darrel Conklin (Conklin & Anagnostopoulou, 2001) and can be viewed as a type of musical feature. Examples include interval and duration (Conklin & Anagnostopoulou, 2001). A viewpoint may be represented as a series of symbols. We tend to use the two terms interchangeably.

The important connection to patterns is that features may be used to determine patterns. It is often the case that patterns are viewed as such due to certain features they possess. For instance, Beethoven's 5th symphony is famous for its [unison, unison, unison, -3] intervallic pattern (Zbikowski, 2002).

A feature can be computationally learned from data or engineered based on domain knowledge. One of the benefits of leveraging deep learning models is that they circumvent the need to manually engineer features. In this dissertation, we will not use learned features, but rather predetermined ones. We will use feature analysis in subsequent chapters to demonstrate further connections between patterns and features.

### Similarity, distance, and energy

Hahn *et al.* (2003) describes similarity as "a broad concept that can be seen as an explanatory construct in the research area of memory retrieval, categorisation, problem solving, learning, linguistic knowledge, and processing, reasoning, as well as social judgement". Music similarity is a research topic on its own in MIR (Cambouropoulos, 2001; Park *et al.*, 2019; Volk *et al.*, 2012; Volk & Van Kranenburg, 2012). Various distance measures are employed together to investigate similarity (Janssen *et al.*, 2017), such as edit distance and earth mover's distance (Typke *et al.*, 2007). In addition, taking inspiration from physics, one may think about the similarity and distance together with how much energy is needed to transport from one set of weighted points to another set of weighted points. These concepts relate closely to musical patterns, as we will detail below.

Reti (1951) used four degrees of similarity to describe relations between musical patterns: imitation ("literal repetition of shapes, either directly or by inversion, reversion..."), variation ("changing of shapes in a slight, well traceable manner"), transformation ("creating essentially new shapes, though preserving the original



substance"), and indirect affinity ("producing an affinity between independent shapes through contributory features"). Other variations on these relations include relative repetition, ornamentation, substantive transformation (Serafine, 1988); and recurrence, development, response, contrast (LaRue, 1992). These degrees of similarity are, however, difficult to convert into a computational language.

Let us consider the connections with repetition and variation that might help with this situation. Similarity measures may determine whether there is a repetition to a certain extent. Examples include having maximal similarity may be treated equivalently as having exact repetition, and having minimal similarity then corresponds to having heavily mutated variation. We will look at repetition and variation more closely in Section 2.3.2 in the context of music. More discussion about different aspects of similarity will come up in later chapters, as well.

Through better understanding music similarity, we may peek into the wider importance of similarity in our understanding of other concepts in the world. Here, we move on to look at similarity in a broader sense, focusing on the divide between the intuitively understood notion of similarity by humans and the difficulty in converting it to computational systems.

Similarity can help explain object recognition, category forming, conceptual knowledge structuring, behavioural prediction based on experience (Hebart *et al.*, 2020). Goldstone *and* Son (2012) listed four major classes of models that have been proposed for how humans assess similarities:

"In geometric models, entities are represented by their positions in a multidimensional space, and similarity is based on the proximity of entities in this space. In featural models, entities are described by their features, and the similarity of entities is an increasing function of their shared features and/or a decreasing function of their unique features. In alignment-based models, the similarity between two structured entities is calculated by placing the elements of their structures into correspondence. In transformational models, the similarity between two entities is conceptualized as the number of transformations required to transform one entity into the other. "

Indeed, the geometric models are linked to distances, allowing for the differences to be encoded in terms of distance. For the featural model, the features we discussed in the previous section may indeed be used to calculate similarity. For the alignment-based models, they have been computationally explored in MIR as well (Bountouridis, 2018). Transformation is something we are going to consider later in this dissertation (Chapter 5 and onwards).

### Object, Gestalt, and grouping

Drawing a connection between finding musical patterns and physical objects is not difficult. The process of identifying a new pattern can be similar to the process of identifying a certain form of grouping and subsequently recognising it as an object. Later in this chapter, we will discuss the leitmotif, which is a musical pattern that may have a one-to-one correspondence with a certain object.

There are, however, edge cases where objects and groupings can be alien, ambiguous, and fuzzy. Theorising these processes has seen many research efforts, including Gestalt psychology. There are seven main Gestalt principles and factors (Wertheimer, 1938) of grouping: proximity, similarity, continuation, closure, common fate, symmetry, and Prägnanz (simplicity/good figure). There are also three key concepts in Gestalt psychology: reification, multistability, and invariance. Given that the research was disrupted by World War II, coupled with challenges in translation and the evolution of the research, publications concerning these principles often exhibit minor differences in terminology. Consequently, although these principles are commonly taught in psychology courses and referenced in various psychology and design textbooks, there may be variations in the principles themselves, with some texts including additional or fewer principles, depending on the source.

Some of the principles have been applied in music research (Tenney & Polansky, 1980) as well as musical pattern discovery algorithms (Cambouropoulos, 2006; Lartillot, 2005; Velarde *et al.*, 2016). We will not dive into the details of each of these principles, but point out the fact that we specifically considered similarity and multistability (ambiguity); we will consider symmetry and invariance together with transformation in Chapters 5 and 6.

### Structure, hierarchy, and heterarchy

Music is well-known for having rich hierarchical structures, ranging from global forms to local phrases, from harmonic progressions to melodic patterns. Repetition and variation have important roles to play: "musical structure often derives from repetition and is one of many crucial musical elements for defining and capturing structure" (Hunt, 2020). In (Ockelford, 2017; Wiggins, 1998), it has been noted that the meaning of music is in its structure, rather than being carried by its structure.

The dictionary definition of structure points us to other concepts such as arrangement, organisation, and mutual relation of the constituent parts. Once we have the grouping, objects, and patterns, another important ingredient is the way the struc-

tures are constructed on top of them. Additionally, structures may inform what kind of patterns exist inside, too.

In music, as well as several other cases, hierarchical structures are a common type of structure to see. Some examples include, in the visual domain, the pixel, edge, tex-ton, motif, part, and object hierarchy; in the language domain, the character, word, word group, clause, sentence, and story hierarchy. Figure 2.1 shows an example of a hierarchical analysis in Beethoven's composition.

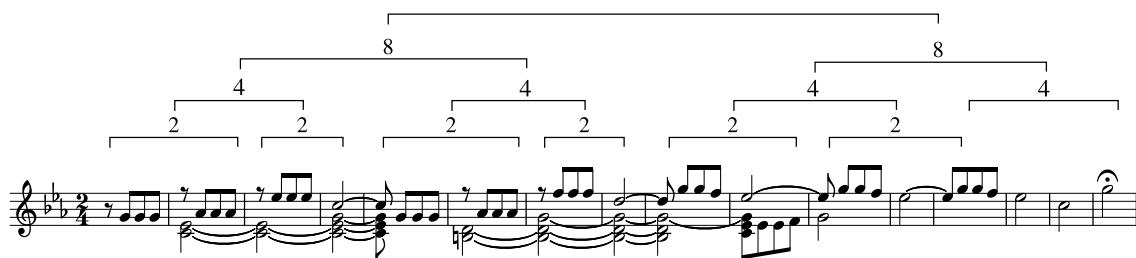


Figure 2.1: An example of hierarchical analysis of Beethoven's 5th Symphony in (Lidov, 2005).

Although the physical progression of time when listening to music is linear, the important notes in music can form hierarchies subjectively. This tendency happens to experts as well as various other kinds of listeners. The hierarchical structures allow us to examine music at different levels of detail and time scales.

There exist musical theories on this topic of the hierarchical structure of music, such as the *Generative Theory of Tonal Music (GTTM)* (Lerdahl & Jackendoff, 1985) and the Schenkerian theory of melodic reduction (Forte, 1959). The Schenkerian theory was described in an approachable way in (Cook, 2000), from which we can see the importance of reducing music complexity through hierarchy and its connection to musical patterns:

"Schenker did a reverse engineering job: he reduced it to a series of basic melodic and harmonic patterns, showing how these basic patterns were elaborated in the music Beethoven actually wrote ... it enabled you to understand the music in a way you otherwise couldn't. More specifically, it explained the moments of apparent incoherence as purely superficial phenomena resulting from the elaboration of the underlying structure; it allowed you to hear 'through' the surface to what lay underneath."

Structures do not have to be hierarchical. Heterarchy, first coined by McCulloch (1945), has been used to describe cognitive structures and to understand social organisation. It expands on hierarchy and emphasises the relation of elements. In a heterarchy, the relation between elements possesses the potential for being ranked in a number of different ways or being completely unranked (Crumley, 1995). In other

words, instead of tree-like relations in the case of a hierarchy, we may also have a model from graph theory or complex network theory, where there are simply nodes and links with no hierarchy.

In MIR, structural analyses (Allegraud *et al.*, 2019; Nieto, 2015) have been made and tools have been developed for visualising structures (Müller & Jiang, 2012). Some structures are also known as musical forms, which have been subjected to algorithmic analyses (Giraud *et al.*, 2016).

More specifically, in connectionists' MIR algorithms, learning structures and meaning has been a struggle. For example, musical structures created by generative models also tend to be local (not more than a few bars long) (Kortylewski *et al.*, 2021; Wu & Yang, 2020).

### Grammar

Grammar is very closely related to structure, especially hierarchical structures, because grammar may be used to create and summarise such structures. Grammar can be thought of as a system of rules and principles. In language, Blacking (1984) argues that "Grammars are attempts to codify the regularities of structure that communities generate in order to give coherence to their communication and to enable individuals to share meanings."

In music, one seminal theory about grammar is GTTM. According to GTTM, a listener unconsciously infers four types of hierarchical structure in a musical surface, as summarised in (Pearce, 2005) and we quote:

- "first, grouping structure which corresponds to the segmentation of the musical surface into units (e.g., motifs, phrases, periods, and sections);"
- "second, metrical structure which corresponds to the pattern of periodically recurring strong and weak beats;"
- "third, time-span reduction which represents the relative structural importance of pitch events within contextually established rhythmic units;"
- "fourth and finally, prolongational reduction reflecting patterns of tension and relaxation amongst pitch events at various levels of structure."

The patterns we consider go up to at least the third level, as we previously stated that we do not consider the tension and relaxation pattern per se. We will discuss motifs and phrases in more detail later in this chapter. Metrical aspects are considered with

feature analysis in the chapters to come. However, we do not consider many other aspects of hierarchy and grammar in this dissertation. We treat musical patterns the same way regardless of where they lie in different levels of hierarchy, in order to keep ourselves within a reasonable level of complexity.

The importance of grammar and GTTM is evident in a variety of areas of research, including MIR. A series of computational methods have been developed for creating and modifying musical pieces according to GTTM (Hamanaka *et al.*, 2014, 2015). Additional forms of grammar have also been seen in the research scene at the intersection of music and computation (Melkonian, 2019; Young, 2017). In (Wiggins, 1998), another type of grammar was applied in the context of music: "pattern grammars are a specialised grammatical formalism which extend standard Chomskian grammars". A pattern in the pattern grammar is a non-empty finite string of symbols. Angluin (1988) demonstrates how its functions as follows:

"The language of a pattern  $p$ , denoted  $L(p)$ , is the set of all strings over the alphabet  $A$  obtained by substituting non-empty strings of constant symbols for the variable symbols of  $p$ . If  $p = 122x5yyx3$ , then the language of  $p$  includes the strings 12205111103 and 122001512120013, but not the strings 12253 or 1221560601113."

### 2.3.2 Related concepts in music theory and musicology

In the previous section, we have seen connections between pattern and other concepts in MIR. This section covers twelve more concepts that are more music-specific, which implies that they are primarily from the music analysis tradition, and we will see more literature from this area.

Theme, phrase, repetition, and variation are common terms not only seen in music but also in other domains. In this section, we consider them largely in the context of music. Where necessary, we also consider their connections to the algorithmic aspects.

The concepts introduced in this section are typically associated with a certain composer, style, or genre. The term pattern has been used as an umbrella term for many of them. For example, in the original MIREX pattern discovery task, the full name of the task expands the word pattern into three other terms: motif, theme, and section. The definitions of these three terms are given below by the MIREX task:

According to Drabkin (2001a), a "motif may be of any size, and is most commonly regarded as the shortest subdivision of a theme or phrase that still maintains its identity as an idea."

A theme is the "musical material on which part or all of a work is based, usually having a recognizable melody and sometimes perceivable as a complete musical expression in itself" Drabkin (2001b).

A repeated section is the "restatement of a portion of a musical composition of any length from a single bar to a whole section, or occasionally the whole piece. Since the Classical period, repeated passages have not usually been written out; instead they are enclosed within the signs  $\|$ : and  $: \|$ " (Tilmouth, 2001).

There are more works from others who used different words for analysing music. For example, Schoenberg wrote in (Schoenberg, 2006) that components of a piece include "statement, phrase, gestalt, motive, feature, grundgestalten, and figure". However, it is unclear as to how these divisions connect with one another (seven concepts generate  $7 \times 6 = 42$  relations to consider, if not considering one-to-many and many-to-many relations), and how they may aid in understanding music and devising computational methods.

We believe that a generalised concept–musical pattern–can help with unifying the separate concepts. For example, in (Sears & Widmer, 2020), the authors try to discover voice leading patterns and shed new light on specific polyphonic patterns in music. In the same work, it has been noted that "the computational music analysis community... more importantly, develop a more sophisticated theory about the organizational principles that characterize recurrent patterns in music..." We concur with this viewpoint. Although we do not attempt to develop a theory of organising principles for musical concepts, we believe that many of these specific concepts can be unified into the topic of pattern discovery. In fact, the concept of pattern has already been used to subsume other concepts in music, which is the case with the [MIREX](#) task. In addition, according to Simon and Sumner (1993), "one of the purposes of analyzing musical structure and form is to discover the patterns that are explicit or implicit in musical works", demonstrating the unification capacity of patterns once more. In the rest of this section, we start with the broad concepts of repetition and variation and progress into more specific concepts.

### Repetition and Variation

Musical patterns are closely related to repetition and variation. It is often the case that repetitions anchor patterns in musical compositions, followed by the development of patterns under variations, with the aim of creating appealing musical structures. In fact, we have mentioned repetition and variation a few times so far. Let us dive into this topic again and see how others have written about them and consider

their connections with musical patterns. We will examine ideas from a variety of scholars on this subject to reinforce these connections.

The importance of repetition and variation is widely agreed upon. Lerdahl *and* Jackendoff (1983) observed that "the importance of parallelism (repetition) in musical structure cannot be overestimated". Rahn (1993) even goes as far as to say that "all musical structure derives from repetition". For Schenker (Schenker, 1980), "only by repetition can a series of tones be characterized as something definite. Only repetition can demarcate a series of tones and its purpose". In her seminal book, Margulis (2014) conducted a thorough examination of the function, importance, perception, and many other aspects of repetition are presented. The book provides analysis of and insight into repetition in a variety of contexts, including evolution, language, music, and dance. In other contexts, Fitch (2006) *and* Margulis (2014) both suggested that the prediction of pattern repetition distinguishes music from language more than any other feature. More on the point of variation, in (Zbikowski, 2002), it was noted that "the motive undergoes continual change, as the different versions". Schoenberg (1967) commented on how repetition is not enough with variations: "Repetition alone often gives rise to monotony. Monotony can only be overcome by variation".

Repetition and variation as concepts are, however, not without their own complications of vagueness, as once put by Meyer: "I fully agree with Tovey that 'Nothing is easier than to derive any musical idea whatever from any other musical idea'" (Meyer, 1973). Another point in (Mazzola, 2018) concerns the relation between repetition and variation: "if we include variation as a kind of repetition, we must acknowledge immediately that variation clouds this distinction between abstract and concrete domains of form". Variation always include an element of repetition, but if the variation is sophisticated, its repeated aspects may be perceptible only to the listener who already knows its style.

We have also touched on these topics in previous sections. Summarising them all, in this dissertation, we treat inexact and exact repetition as different types of repetition. Inexact repetitions are also called variations. We also separate out the issue between endogenous (concerning data) and exogenous (concerning perception) variations; if two bars of music appear subtly different on sheet music but sound the same to some people, we regard these two bars of music as a variation, not as exact repetition.

### **Idea**

The concept of musical idea has been discussed extensively in books and articles (Ferguson, 1941; Hanninen, 2003; Schoenberg, 2006). Ferguson (1941) explores what a musical idea is starting from the definition that "An idea is that once an image and a valuation of experience". Brand, Hailey, *et al.* (1997) reiterate Schoenberg's view on musical idea and composition "Composing is: thinking in tones and rhythms. Every piece of music is the presentation of a musical idea (Idee, Gedanke, Einfall)". For Schoenberg, "A musical idea is sheerly musical. It is a relation between tones." (Schoenberg, 2006). We can immediately observe some misalignment in the use of the word "idea".

Hanninen (2003) introduces the concept of recontextualisation using musical ideas and their instances. An idea was defined as "a set of one or more contextual (not sonic or structural) criteria. Ideas have (and manifest in) instances." Contextual criteria here means a set of criteria that

"identifies a characteristic of a grouping with a propensity for association among groupings within a musical context under consideration. Contextual criteria are activated by repetition; they indicate equivalence or similarity in non-linear musical spaces including pitch contour, duration series,..."

This definition is more in line with what we have seen so far about repetition and musical patterns. Schoenberg has also commented on the important role played by repetition for musical ideas: "Repetition is one of the means (in presenting an idea) to promote the comprehensibility of the idea presented." (Schoenberg, 2006)

We can see from this body of literature that a musical idea is analogous to a musical pattern: both are derived from music and our subjective experiences, both are closely related to repetition, and both are associated with a set of criteria and occurrences. Next, we will examine the musical unit, figure, and other concepts that build on top of them.

### **Unit, figure, phrase, and more**

In (Goetschius, 1904), a range of concepts have been discussed, including unit, idea, figure, motif, phrase, sentence, and more:

"The smallest unit in musical composition is the single tone. The smallest cluster of successive tones (from two to four or five in number) that will convey a definite musical impression, as miniature musical idea, is called a figure. Assuming the single tone to represent the same unit of expres-



sion as a letter of the alphabet, the melodic figure would be defined as the equivalent of a complete (small) word, pursuing the comparison further, a series of figures constitutes the melodic motive, equivalent to the smallest group of words (a subject with its article and adjective, for example); and two or three motives make a Phrase, equivalent to the complete, though comparatively brief, sentence (subject, predicate, and object)."

We agree that the smallest unit is a single musical event. We also argue that other concepts are potentially different types of units, and we have a range of concepts that are composites of different units. This succession of concepts seems to get gradually longer, but without predefined numerical intervals of length. The source of inspiration seems to be natural languages, which adheres to conventions such as punctuation and capitalisation. These conventions are, however, not standardised in music. A long sequence in a short musical composition might be a short sequence in a long musical composition. One can imagine the difficulties in arguing that a certain series of notes does not constitute a figure but a motif.

Moreover, some of those definitions can be in conflict with each other from different literature. For example, a phrase is defined in ("Phrase. Oxford University Press", 2001) as

"A term adopted from linguistic syntax and used for short musical units of various lengths. A phrase is generally regarded as longer than a Motif but shorter than a Period. As a formal unit, however, it must be considered in its polyphonic entirety, like 'period', 'sentence' and even 'theme'."

Period was not mentioned in the previous quote at all when we gave the first definition from a different source in this section. Additionally, "short musical units of various lengths" ("Phrase. Oxford University Press", 2001) can hardly be equated with "two or three motives" (Goetschius, 1904). The intrinsic difficulty is that one musical unit, figure, or phrase can cease to be a musical unit, figure, or phrase in a different musical context. We will, therefore, not further distinguish between the terms in this series of concepts for this dissertation.

The same issues of context and length exist with musical patterns. The context and the relative length of the pattern is difficult to describe formally and comprehensively. We will circumvent these issues by using transformation in Chapter 5, but they remain difficult issues in and of themselves. Next, we will discuss themes and different types of motif on their own.

### Theme

We have already seen the keyword "theme" appearing in previous sections. Here, we give a few more viewpoints and relevant research on musical themes.

For Schoenberg (Schoenberg, 1967),

"The formulation of a theme assumes that there will follow 'adventures,' 'predicaments,' which ask for solution, for elaboration, for development, for contrast."

In jazz music, a theme is

"The musical material on which part or all of a work is based, usually having a recognizable melody and sometimes perceivable as a complete musical expression in itself, independent of the work to which it belongs."

(Drabkin, 2001b; "Theme (jazz). Oxford University Press", 2003)

These descriptions are very difficult to serve as specifications for computational methods. There are, however, efforts to gather and digitise data of musical themes (Zalkow *et al.*, 2020) so that perhaps a machine learning method could help with reifying what recognisability means in music.

Not unexpectedly, musical themes also have a close relation with repetition and variation: "On the one hand, the musical theme asserts itself, but on the other, the theme extends outside itself." (Kaduri, 2006). We can also take the words such as "development" and "contrast" in Schoenberg's quote and treat them as alluding to variations. From this connection with repetition and variation, in combination with how themes were included in the MIREX task, we treat themes as a type of pattern in this dissertation.

Themes are sometimes considered longer than patterns, but a hard cutoff point is difficult to establish, as we discussed in previous sections. We do acknowledge that a pattern that constitutes a theme differs from a pattern that merely provides texture (Utgoff, 2006). This finer categorisation of patterns is not included in the scope of this dissertation.

### Motif

In Figure 2.2 (Ockelford, 2018), we show an example of motifs. In (Drabkin, 2001a), a motif is defined to be

"A short musical idea, melodic, harmonic, rhythmic, or any combination of these three. A motif may be of any size, and is most commonly regarded as the shortest subdivision of a theme or phrase that still maintains its identity as an idea."

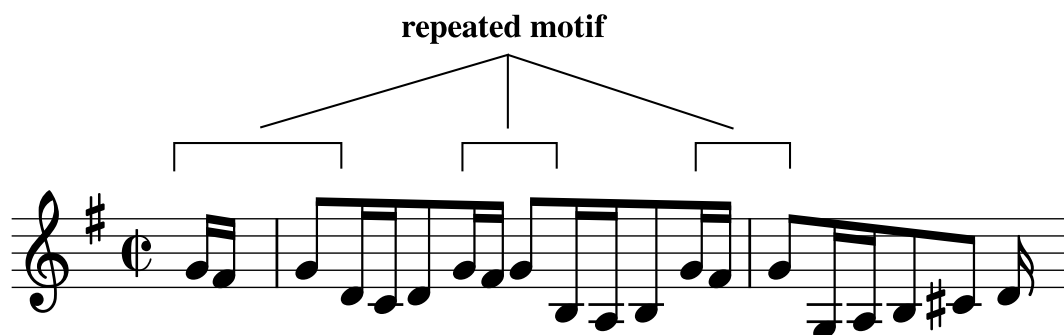


Figure 2.2: Example of a repeated motif in Bach's Brandenburg. Reproduced from (Ockelford, 2018).

Schoenberg's (Schoenberg, 1967) definition seems to agree with this and adds on to the definition by emphasising the time offset at which a motif appears:

"The motive generally appears in a characteristic and impressive manner at the beginning of a piece. The features of a motive are intervals and rhythms, combined to produce a memorable shape or contour which usually implies an inherent harmony. A motive appears constantly throughout a piece: it is repeated."

For Réti (Buteau & Mazzola, 2008), a general motif is:

"... any musical element, be it a melodic phrase or fragment or even only a rhythmical or dynamical feature which, by being constantly repeated and varied throughout a work or a section, ..."

There have been concerns about these definitions and views. For example, in (Mazzola, 2018), it has been pointed out that

"Réti asserts that he handles the words 'motif' and 'theme' without categorical distinction, but a difference between the two concepts is that a theme as a fuller group or 'period' which acquires a 'motivic' function in a composition's course."

Concerning transitivity, Buteau *and* Mazzola (2000) commented that "Réti's concept of identity of motif shapes includes relations—such as variation, which is a kind of similarity—which are not necessarily transitive."

Let us now take a look at these definitions and concerns as a whole. Through the appearance of repetition and variation in the definitions, we see a connection with the concept of pattern. Like patterns, we also observe some disagreements and concerns about how motifs are defined. Some of the arguments are closely related to what we have discussed previously in terms of context and lengths. For transitivity, we bear in mind that if one passage is varied multiple times, relating the last occurrence back to the initial occurrence could become difficult. For a definition, we

do not see a risk in treating motifs as another type of pattern. In fact, with Réti's definition, we can use pattern and motif interchangeably.

Algorithms have been devised (Ganguli *et al.*, 2017; Jiménez *et al.*, 2011) to discover motifs automatically. However, they have not been submitted to the MIREX pattern discovery task; hence, they are not our focus in this dissertation.

### **Leitmotif**

As a special type of motif, leitmotifs can be seen in the compositions by Wagner, in films, and in other background music. In (Whittall, 2001), it is defined as

"In its primary sense, a theme, or other coherent musical idea, clearly defined so as to retain its identity if modified on subsequent appearances, whose purpose is to represent or symbolize a person, object, place, idea, state of mind, supernatural force or any other ingredient in a dramatic work."

It seems this definition largely agrees with the definition in (Ockelford, 2018), which defines a leitmotif to be "A characteristic melodic, harmonic, or rhythmic idea that is associated with a particular person, place, emotion or idea." Leitmotif is a special kind of pattern since it has a semantic meaning, which most patterns do not possess.

In a computational context, datasets and algorithms on leitmotifs exist (Krause *et al.*, 2021). We do not focus on leitmotif in the dissertation, though it is a potential application direction of our research.

### **Reminiscence and head motif**

To further enrich the "motif" family, we see two more types of motif in this section. In ("Reminiscence motif. Oxford University Press", 2002), a reminiscence motif was also defined using theme, musical idea, and it has a historical connection with leitmotif:

"A theme, or other coherent musical idea, which returns more or less unaltered, as identification for the audience or to signify recollection of the past by a dramatic character. It is an important ancestor of the Leitmotif."

In (Fallows, 2001), a head motif was defined with musical idea once again. It is a more constrained version of the motif:

"A musical idea which by virtue of appearing at the beginning of each of a series of pieces or movements establishes a relationship between them."

The definitions of these particular types of motifs demonstrate once again how definitions interact and rely on one another, posing difficulties for computational methods. Admittedly, when the historical provenance and more specific descriptions are given, such as those definitions above, it is arguably more straightforward to devise an algorithm to automatically find them. For example, finding a head motif should be easier than finding motifs because the search space will be significantly reduced. Although algorithms can be adjusted and used to discover these types of motif, we do not yet know of any algorithms that discover them exclusively.

### **Ostinato**

Schnapper (2001) defines ostinato as below:

"A term used to refer to the repetition of a musical pattern many times in succession while other musical elements are generally changing."

In the same article, the importance of rhythm is also stressed: "The regular repetition of a pattern requires, as a minimum, the existence of a rhythmic structure, to which other elements may be added." (Schnapper, 2001). It is also mentioned that ostinatos can be found in oral traditions, jazz, Baroque, minimalist, and popular music, whereas a decline occurred in classical and romantic eras. Additionally, the ostinato also plays an important role in musical structure and expression.

In connection to the concept of pattern, we see that musical pattern is used to define this term and that repetition and variation also play a significant role. Ostinato and pattern share many similarities, and we deem that it is synonymous with pattern. We cannot find any algorithms targeting the discovery of ostinatos in music. Although algorithms can be adjusted and used to discover ostinato, we do not yet know of any algorithms that discover ostinato exclusively.

### **Lick**

In (Witmer, 2001), a lick is defined as

"A term used in jazz, blues and pop music to describe a short recognizable melodic motif, formula or phrase. Improvising jazz and blues musicians have at their disposal a repertory of licks, some of their own invention by which they can be identified, some borrowed from other players, and a solo may be little more than the stringing together of a number of such fragments."

This term seems to be the jazz equivalent of a motif or a pattern. We can expect, however, many specificities of patterns in jazz music may come from this different

improvisational context. There has been research concerning jazz-specific patterns and algorithms (Frieler *et al.*, 2018; Quick & Thomas, 2019). This dissertation does not have an emphasis on jazz music.

### Schemata

In a separate vein of Gallant music, we see some seminal work on schemata (Gjerdingen, 2007, 2014), which are described to be "stock musical phrases" (Gjerdingen, 2007). Figure 2.3 depicts an abstract form of a schema—Romanesca, while Figure 2.4 shows an example of this schema in actual music. Figure 2.5 shows a complete analysis of schemata in Haydn's variations (Sonata Hob. XVI, no. 27, mvt 3, Presto, 1774-76).

Although schemata are primarily concerned with polyphonic music, we provide several examples here for three reasons. First, in later chapters, we will use music from this era and will not discuss the schemata theory there. We do think that schemata are a type of musical pattern, but we do not discuss them further in this dissertation due to their polyphonic emphasis.

Second, these examples demonstrate the extra complexity of polyphonic musical patterns, which necessitates the assistance of computational methods. In fact, we will use some non-schema based polyphonic musical pattern discovery algorithms in Chapter 4. Additionally, there are efforts towards computationally modelling the schemata theory, such as in (Finkensiep *et al.*, 2020; Katsiavalos *et al.*, 2019).

Third, in Figure 2.6, we see a formulation of seeing music as a string of concatenating and overlapping beads made of schemata. The music can branch off (M, N, O, P in the figure) or choose from different possibilities (A, B, C). This is how we see music can be structured with patterns as well.

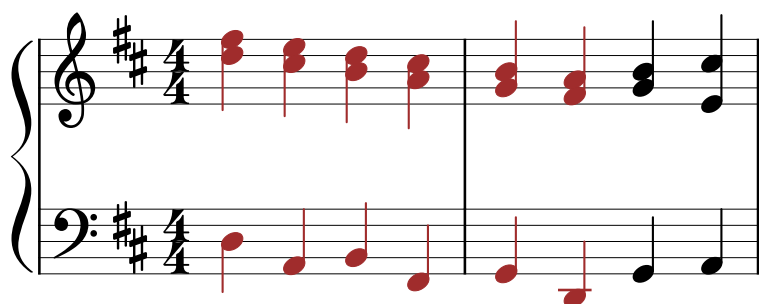


Figure 2.3: A schemata: Romanesca. Reproduced from (Gjerdingen, 2007).

### Gregorian Neume

The last related keyword of musical pattern we are going to examine is Gregorian neume. In Gregorian chants, the notation unit goes beyond a single note. For exam-

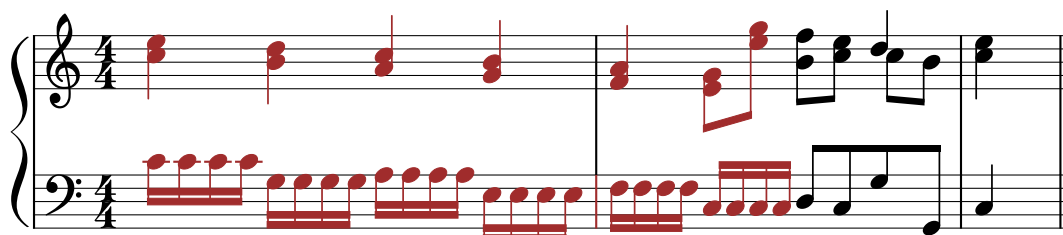


Figure 2.4: Romanesca in music with rhythmic elaboration. Reproduced from (Gjerdingen, 2007).

134 MUSIC IN THE GALANT STYLE Chapter 10 A THEME AND VARIATIONS BY HAYDN 135

Figure 2.5: Analysis on Haydn's Variations (Sonata Hob. XVI, no. 27, mvt 3, Presto, 1774-76) based on schemata theory, showing the complexity and possibility for automation. Taken from (Gjerdingen, 2007).

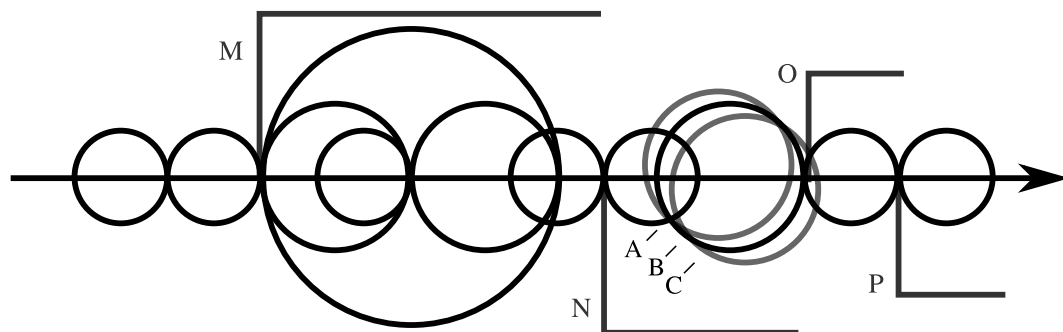


Figure 2.6: Composition of schemata. Letters A, B, and C represent schemata that are either simultaneous alternatives or superpositions. Paths M, N, O, and P represent choices not made. Reproduced from (Gjerdingen, 2007). Music can be seen as a series of "beads". They are sometimes branching and overlapping possibilities.

ple, the scandicus flexus is a neume that consists of four notes, going up and then dropping down, as in F, G, A, F (Kephart, 2017). Climacus is another neume that consists of three or more downward-moving notes.

Although we are not listing all neumes here, we can already see that these two can be referred to as musical patterns. Neumes are actually occasionally called "neume patterns". Once translated into the modern staff notation, discovering the patterns would be a reverse engineering problem, that is, converting the notes back to the neumes. Computational pattern discovery methods have been used to generate melodies for lost chants (Conklin & Maessen, 2019).

### 2.4 Musical pattern discovery algorithms

Algorithms have been devised to discover patterns using mathematical principles and *heuristics* from the music domain. They take musical pieces as input, and output a set of patterns occurring within the piece.

Leaving behind various concepts from previous sections, we focus on introducing musical pattern discovery algorithms in this section. In particular, some of the musical pattern discovery algorithms introduced here are used for our experiments later in Chapter 4.

Based on the online availability and format compatibility of the algorithms, we examine and introduce 11 musical pattern discovery algorithms, including two families of algorithms that include 7 more algorithms in total. Out of these algorithms, a few have been submitted to the MIREX task in 2013-2017. Each algorithm will be accompanied by a short description.

Readers do not have to remember all details of the algorithms' complex mechanism to interpret the results we deliver in subsequent chapters. **Our focus is to look at the output of the algorithms—the discovered patterns and pattern occurrences**—more than the mechanism of algorithms. As such, we treat algorithms as black boxes, which is not an uncommon approach (the philosophy behind MIREX is similar) and has the potential to be more generalisable. More specifically, we treat these algorithms as black boxes for two reasons. First, there are more and more black-box algorithms developed from machine learning, especially deep learning. Second, even with the white-box and grey-box algorithms, even when there are mathematical models behind these algorithms, the actual implemented operations on data may surpass our mental ability to track the application of them in real data, which also generates the need to treat the algorithms as modular, information-



hiding black boxes. For a more all-round overview of musical pattern discovery algorithms, please refer to (Janssen, 2018; Meredith, 2015).

### Wavelet-based algorithms

A wavelet is a waveform of limited duration, begins with zero, increases or decreases, and returns to zero one or more times. Wavelet transforms, more useful than Fourier transforms when it comes to non-stationary signals<sup>1</sup>, have many applications in physics and have a strong theoretical foundation in mathematics.

By convolving wavelets with symbolic melodies, we obtain correlations between the unknown symbolic melodies and the known patterns encoded in the wavelets. In (Velarde *et al.*, 2016), the resulting correlations are used for melodic segmentation and concatenation data, which is then compared, clustered, and ranked. These ranked clusters then become the melodic patterns retrieved from the piece. The two algorithms, *VM1* and *VM2*, have two Matlab implementations using several different parameters. From the original paper, we know that according to Friedman's test, VM1 and VM2 show no significant difference in the results of the three-layer F1 score, a metric in MIREX (will be discussed in Section 4.1); in addition, for discovering exact occurrences, VM1 outperforms VM2.

### Compositional hierarchical model

The *Symbolic Compositional Hierarchical Model (SYMCHM)* model was inspired by object categorisation in computer vision. As an alternative to deep learning models, in (Pesek *et al.*, 2017), the model imposes a hierarchical structure in different compositional layers and tries to provide a transparent deep architecture to discover repetitions in music. The method is based on the assumption that repetitive patterns can be characterised by the number of occurrences of their sub-patterns. As a consequence, the model learns pitch patterns from symbolic data in an unsupervised manner.

According to the original paper, the process of inference in SYMCHM is designed to be either exact or approximate. To approximate, two processes, hallucination and inhibition, are involved in finding the parts that have been changed, inserted, or deleted.

---

<sup>1</sup>Non-stationary signals are characterised by statistical properties that vary over time.

## Geometric algorithm family

The name geometric comes from the fact that the patterns in music are found by identifying their shapes in sheet music. This family of geometrically inspired pattern discovery algorithms that can identify note groups with similar shapes and therefore extract these shapes as musical patterns. In the design of these algorithms, factors contributing to the perceived importance of a pattern were discussed. The factors identified include the compactness (the notes in the pattern are contiguous) and the amount of compression that can be achieved, which has a link to repetition and variations.

For the algorithms in this geometric family, an important concept is that of *Maximal Translatable Pattern (MTP)*,

$$\text{MTP}(v, D) = \{p | p + v \in D \wedge p \in D\}$$

where  $p$  denotes pattern,  $D$  denotes a dataset, and  $v$  a vector. MTP for a vector  $v$  contains all points mapped by  $v$  onto other points in the dataset

Another important and related concept is that of *Translational Equivalence Class (TEC)*,

$$\text{TEC}(P, D) = \{Q | Q = P + v \wedge Q \in D\}$$

where  $P$  and  $Q$  denote patterns, the rest similar to those in *MTP*. TEC is the set of all translationally invariant occurrences of a pattern.

This family of algorithms typically use pitch and onset of notes and is designed with polyphonic pieces in mind. Six algorithms, *Structure Induction Algorithm (SIA)*, *Structure Induction Algorithm for TECs (SIATEC)*, *Data Compression using SIATEC (COSIATEC)*, *SIA with a sliding window of size  $r$  (SIAR)*, *SIATECCompress-Precision (SIACP)*, *SIATECCompress-Recall (SIACR)*, *SIATECCompress-F1 (SIACF1)*, *SIA for  $r$  superdiagonals and Compactness Trawler-Categorisation and Fingerprinting (SIACFP)*, and Forth, all belong to this family of geometric musical pattern discovery algorithms. A tool written in Java is available online, supporting this array of point-set compression algorithms. We provide a rough overview of this family of algorithms below; please see (Meredith, 2015) for a comprehensive review.

## SIA and SIATEC

In (Meredith *et al.*, 2002), the SIA algorithm calculates all vectors between all notes. Along with using a couple of sorting procedures, the algorithm discovers all **MTP**. As a result, SIA admits unlimited gaps in its pattern.

Proposed in the same paper as in the **SIA** algorithm, **SIATEC** computes the *Translational Equivalence Class (TEC)* based on **SIA**. The algorithm computes all the occurrences of each of the maximally repeated patterns discovered by **SIA**.

**SIA** only finds one occurrence of each **MTP** and one vector by which that **MTP** is translatable within the dataset. **SIATEC**, on the other hand, finds all the non-empty **MTPs** and then all the patterns within the dataset that are translationally equivalent to each **MTP**.

### **COSIATEC and SIATECCompress**

In (Meredith, 2013), both **COSIATEC** and **SIATECCompress** iteratively apply the **SIATEC** algorithm to compress a piece into the corresponding union of **TECs** of **MTPs**. The difference is that **COSIATEC** finds the best **TEC** and then removes its cover set from the input, while **SIATECCompress** extracts a list of **MTP TECs** and then selects a subset of the best **TECs** to cover the input. The selection criteria are based on compression ratio and compactness measures.

The **SIATECCompress** algorithm comes with three variants with different parameters. These different parameters were obtained by optimising for precision, recall, and **F<sub>1</sub> score**.

### **Forth algorithm**

Based on the geometric method **SIATEC**, the Forth algorithm (Forth, 2012) incorporates more considerations based on music theory and cognition. It employs **heuristic** measures in conjunction with compression ratio and compactness value thresholds. The algorithm also requires music to have been first parsed into voices. To list the differences between the three algorithms mentioned above:

- **COSIATEC** finds patterns that are exclusive (no overlapping patterns) and exhaustive (every note belongs to a pattern).
- **SIATECCompress** and Forth's algorithm generate patterns that can intersect, which makes them faster but less effective at compression than **COSIATEC**.

### **SIAR**

Based on the **SIA** algorithm, **SIAR** (Collins, 2011) limits the creation of vectors from

all possible note combinations by constraining them to a maximum of  $R \in \mathbb{N}$  successive notes. In this way, we might think of  $R$  as the "memory size" parameter specifying the number of notes to consider in [SIA](#).

### SIARCT-CFP

The SIARCT-CFP algorithm (Collins *et al.*, 2013) combines [SIAR](#) with a *compactness trawler*, fingerprinting, and categorisation steps. A compactness trawler removes all patterns that do not satisfy a parameterised compactness ratio  $c \in \mathbb{R}$ —the ratio between the number of notes in a pattern and the width of the bounding box of the pattern. The fingerprinting step computes the rhythmic ratios in order to allow rhythmic variations. The categorisation step ranks the discovered patterns in order of importance. [SIACFP](#) algorithm has an implementation in Matlab.

### Pattern clustering

Leaving the geometric algorithms behind, a recent pattern discovery algorithm (de Reuse & Fujinaga, 2019) uses a clustering approach. This algorithm maps passages of music onto a low-dimensional subspace using an embedding learned from human annotations of repeated patterns. The neural network used for learning the embedding is a fully connected, feedforward network with two hidden layers, each containing 100 nodes. Techniques such as dropout and batch normalisation were used, with an output layer size 5.

### MGDP

The *Maximally General Distinctive Pattern (MGDP)* algorithm (Conklin, 2010) computes the maximally general distinctive patterns in a corpus. Distinct patterns are patterns that rare or absent from a set of pieces but are frequent in another. "Maximally general distinctive" means that no subsuming pattern is also distinctive. The term has been applied to folk song melodies from three geographic regions and chord sequences from three music genres.

Prior and related to this algorithm, Conklin *and* Anagnostopoulou (2006) uses a sequential pattern mining method and works by constructing a suffix tree, which is then pruned based on the patterns' occurrences. MGDP and other related algorithm has been applied to a range of music styles, such as folk music and Gregorian chants (Conklin, 2013b; Conklin & Anagnostopoulou, 2011; Conklin & Maessen, 2019; Conklin & Weisser, 2016; Nuttall *et al.*, 2021).

## Closed pattern mining

*Closed patterns* are the longest patterns that are repeated the same number of times. Finding closed patterns is a standard technique in data mining. A closed pattern mining algorithm can be used to find patterns with pitch intervals and rhythmic ratio and have been used in (Lartillot, 2005; Ren, 2016). The algorithms can only operate on monophonic voices.

### PatMinr

The *Pattern Miner (PatMinr)* algorithm (Lartillot, 2014) considers both the concept of closed pattern, maximal pattern, and cyclic pattern. It uses an incremental one-pass approach to identify pattern occurrences, considering various music attributes such as chromatic, diatonic pitch, metrical position, and articulation. A pattern prefix tree is employed in the implementation. The algorithm works with monophonic voices and is available in Matlab as part of the *MIR* toolbox (Lartillot, 2011). This algorithm has been submitted to the MIREX task with two different sets of parameters.

### FIEXPath

This algorithm (Rolland, 1999) uses Levenshtein edit distance to compare two passages, construct a similarity graph, and then identify the passages as instances of the same pattern. It considers local and global music aspects such as absolute pitch, backward interval, crescendo, contour, time signature, tempo, and so on. The algorithm is built into the Imprology software system (Rolland, 1998).

### MotivesExtractor

MotivesExtractor (Nieto, 2015) employs perceptual grouping principles. It computes a distance matrix between pairs of potential motives and then uses a clustering algorithm to identify the instances of the same pattern. The implementation of this algorithm is in Python.

In more detail, the algorithm obtains a harmonic representation of the audio or symbolic input and extracts patterns based on a produced *self-similarity matrix* (Nieto & Farbood, 2012). Each number in the matrix denotes a similarity value of a pair of elements in the sequence, with the diagonal being identical. With some post-processing steps, pairs of high similarity are extracted and are considered to be the occurrences of the same pattern. Based on Gestalt principles, the algorithm also

filters out the patterns that contain relatively long notes or rests, or relatively large intervals.

### PAT

PAT algorithm (Cambouropoulos, 2006) employs Crochemore's set partitioning method to recursively split the melody into sets of repeating tokens. As a result, *maximal patterns* are found and filtered according to pattern lengths. The algorithm was used for segmentation based on maximal repeated patterns.

### Suffix arrays

In (Knopke & Jürgensen, 2009), suffix arrays were used to represent phrases of Palestrina masses. To extract patterns, the algorithm first uses rests as an indicator to find phrases, then inserts each phrase four times (normal order, inversion, retrograde, and retrograde inversion) into the new representation and matches the repetition. It is a fast algorithm that operates on pitch and duration of monophonic melodies.

To finish this section, we summarise the musical pattern discovery algorithms in Table 2.1, based on previous work (Janssen, 2018; Janssen, De Haas, *et al.*, 2014). We can again see the diversity of the algorithms in terms of their goals, the methods they employ, their filtering mechanisms, how music was represented in them, and how they were evaluated. From this table, in combination with Section 2.3, we can observe that the diversity and complexity of the algorithms are multidimensional, potentially complicating the process of comparison and selection amongst them.

## 2.5 Discussion

In this chapter, we examined 18 sets of pattern-related concepts and 11 musical pattern discovery algorithms. By and large, the works reviewed here provide evidence for the diversity of the concept of pattern as well as pattern discovery algorithms. We have also exhibited that it is almost impossible to define the concept of pattern precisely, comprehensively, and meaningfully, owing to the complexity of many unsolved issues in human perception, coupled with the constantly evolving field of musical analysis. This definition problem contributes to the diversity and complexity of pattern discovery algorithms, relating to the challenges we mentioned in Section 1.2. Moreover, the diversity and complexity of algorithms lead to difficulties in making useful comparisons between the algorithms. We will address this problem in Chapter 4, where we propose new methods to compare algorithms and explain

Algorithm	Study	Goal	Method	Music Representation	Filtering	Evaluation
Wavelet-based	Verlarde et al. (2016)	Discovering themes and sections; determining parent compositions	Wavelet	Pitch, duration	Similarity	Annotations
Compositional Hierarchical Model	Pesek et al. (2017)	Extract the most frequent patterns	Hierarchical Model	Pitch		Annotations
SIA and STATEC	Meredith et al. (2002)	Assisting music analysts and composers by finding repeated patterns in music	Geometric	Pitch, onset		Qualitative
COSIATEC and STATECCompress	Meredith (2013)	Classifying folk song melodies into tune families; discovering entries of subjects and countersubjects in fugues; discovering repeated themes and sections in polyphonic works	Geometric	Pitch, onset	Spacing	Classification and annotations
Forth	Forth (2012)	Pattern discovery in discrete representations of polyphonic music	Geometric	Pitch, onset	Spacing	
SIAR	Collins (2011)	Discovery of patterns in music	Geometric	Pitch, onset	Spacing	Annotations
SIARCT-CFP	Collins et al. (2013)	Discovery of motifs, themes, and sections	Geometric	Pitch, onset	Spacing, similarity	Annotations
Pattern Clustering	De Reuse and Fujinaga (2019)	Pattern discovery that models human judgement of what constitutes a significant pattern by incorporating annotations of repeated patterns, avoiding the need to design heuristics	Neural network	Symbolic features	Spacing	Annotations
MGBP	Conklin (2010)	Discovery of maximally distinctive general patterns in folk songs and chord sequences as a base for pattern-based classification methods	String-based	Viewpoint matrix	Frequency	Qualitative, classification
Closed pattern mining	Lartillot (2005) and Ren (2016)	Exhaustive but compact description of repeated motivic patterns in symbolic representations of music	String-based	Pitch, duration	Length, frequency	Qualitative
PatMinr	Lartillot (2014)	Motivic analysis for the understanding of musical compositions	String-based	Pitch, duration	Length, frequency, similarity	Annotation
FlExPat	Rolland (1999)	Salient patterns or motifs in jazz improvisation	String-based	User defined	Frequency	Qualitative
Motives Extractor	Nieto (2015)	Finding motifs in musical works	Time series	Pitch, duration	Similarity	Annotations
PAT	Cambouropoulos (2006)	Segmentation based on musical repetitions	String-based	Pitch, contour	Length and frequency	Segmentation
Suffix arrays	Knopke and Juegensen (2009)	Analysis of Palestrina's contrapuntal structure	String-based	Pitch, duration		Qualitative

Table 2.1: A table adapted from (Janssen, 2018; Janssen, De Haas, et al., 2014), summarising musical pattern discovery algorithms. Under the **Evaluation** column, "Annotations" means that there was a comparison against human-annotated patterns; "Qualitative" signifies that the evaluation contains a few selected patterns.

why it is important to compare them using those methods. In connection to other later chapters, we hope that this analysis lays the groundwork for unveiling the complexity behind the concept of pattern.

### 2.6 Concluding remarks

In this chapter, we have offered, albeit partially, a compilation of diverse views from various angles on pattern and pattern discovery. In this section, we open the discussion to several other topics that, though maybe not critical, are nevertheless intriguing. We use them to expose our limitations as well as extend and consolidate our arguments.

#### What did we not consider?

There are three points that we omitted from this chapter. Here, we either give a pointer to further discussion on them in the next chapters or explain why we did not include them.

We did not answer the following question: what makes a pattern salient in a piece of music? We pointed to a few broad concepts such as features, repetition, and structure, but we do not have the answer to this question objectively and generally. Salience is fundamentally subjective to the listener. Nevertheless, to take a statistical approach to this issue, there is converging evidence that some excerpts of music are more salient than others. For example, Burgoyne *et al.* (2013) demonstrated that different fragments of music differ in their salience in triggering musical memory. We will touch on the topic of salience again in Chapter 4, where we devise a limited kind of approximation of salience based on the number of patterns discovered at a certain spot in music.

We did not cover the relations between all concepts in Section 2.3, and it would be a Herculean task to cover them all. We introduced 18 sets of concepts in total. If one were to extend the discussion to the dyadic relations between each one of them, one would have  $18 \times 17 = 306$  relations between the concepts to go through, which would not be an easy endeavour. Moreover, this number does not take into account the fact that some of the concepts could be defined in different ways by a few different people.

We did not point out explicitly what a pattern is not. Except for the ones that are in contradiction with other definitions, we were unable to locate an entirely "false"



definition of pattern. Even the word anti-pattern<sup>2</sup> is used in other fields to refer to a pattern that should be avoided, yet this is a pattern nonetheless. This difficulty is not unexpected, however, because identifying what is not a pattern would render finding patterns much easier: one can simply define patterns to be whatever that remains from the search space that is not patterns (not considering the subtleties of the negation operation).

### Compression

Patterns, repetitions, and ambiguity, as well as a variety of other constructs we had in this chapter, seem to all point to another concept—compression. Compression is a very important concept for the existing musical pattern discovery algorithms and other adjacent fields. In natural languages, it has been shown that ambiguity is needed for the purpose of compressing information (Juba *et al.*, 2011). Once there is a pattern or repetition, it is also often the case that compression comes into the picture (Meredith, 2015).

Although we realise the importance of compression, it has been extensively studied in the context of musical patterns (Meredith, 2013), and it is not the focus of this dissertation. We will see "compression" again as a metric in Chapter 4.

### Where is the hard limit?

As humans, we possess a set of cognitive and perceptual tools to uncover and render explicit the hidden patterns underlying diverse, chaotic, random phenomena. Is there a strict limit to how far the algorithms can emulate this? At this moment in science, we do not have a precise answer for the question: "How is an incoming stream of musical information organised by the ears and brain into percepts, and how do cognitive structures develop with the experience of music?" (Collins, 2011). A direct emulation is therefore not feasible. However, another approach may be to strive for a common language for the alignment between human expectations and algorithmic output, and this is precisely what we are attempting to achieve in this dissertation.

---

<sup>2</sup>Another definition of antipattern in musical pattern discovery is the patterns that are infrequent, rare, and under-represented (Conklin, 2013a), which is also applicable here.



# Chapter 3 Gathering Human-Annotated Musical Patterns

*Data do not exist de novo.*

– Dark Data, Hand, 2020

*Humans are heterogeneous.*

– Human Compatible, Russell, 2019

## 3.1 Introduction

This chapter investigates the musical patterns annotated by different humans and the influences of different annotation tools. We mentioned in the previous chapter that musical patterns annotated by humans are an important source for automatic pattern discovery. Indeed, the study of the pattern annotation data has gained increased attention over the past few years. While human annotations are often taken as the reference or training data for developing pattern discovery algorithms, relying solely on annotations from a limited number of individuals is often insufficient to capture the complex concept of musical patterns. In addition, there exist varying levels of musical expertise, which may lead to divergent pattern discovery processes and annotator disagreement.

Asking human annotators to point out what they think of as patterns might sound simple. They just need to mark a few boundaries and tell us "those are the patterns", after all. However, the process of annotating patterns is more complex than one might assume. In continuation with some of our discussions in Chapter 2, there are many decisions that a human annotator must make when attempting to annotate patterns in music, such as when a pattern starts and ends, how to find all occurrences of a given pattern in this piece, and how similar must a variation of a pattern

be for it to be counted it as a repetition or variation of the original. During the experiments, annotators may have implicit or explicit strategies or employ conscious or subconscious filters, such as a bias toward the beginning of the music or imposing a minimum or maximum length on their annotations. As a result, when setting up a pattern annotation experiment, many factors are difficult to have perfect control over. Even with the same music material, there are factors such as different annotators' musical backgrounds, different functionality between annotation tools, and the exact procedures the annotators followed, to name a few. All of these factors may contribute to different degrees of inter-annotator agreement. In this chapter, we acquire data from multiple annotators and perform analyses to investigate the influences of different musical pattern annotators and annotation tools given the same pieces of music. In this chapter, we collect data from multiple annotators and perform analyses on the data collected to investigate the influences on the annotations made by different musical pattern annotators and annotation tools when applied to the same musical piece.

## MIREX

**MIREX** (the Music Information Retrieval Evaluation eXchange, see Chapter 1 for a detailed introduction) contains a collection of tasks, aiming to compare *State-of-the-Art (SotA)* algorithms and systems relevant for **MIR**. In MIREX, not only for the task of musical pattern discovery, a significant number of other topics currently researched in **MIR** also rely heavily on using reference or "*ground truth*", often derived from human annotations. The comparison of *SotA* algorithms on the different tasks in the yearly rounds of the **MIREX** framework has also uncovered the issue of ambiguity of musical structures for evaluating algorithms. For instance, Typke *et al.* (2006) created a new measure for tackling disagreeing perception of similarity. Flexer and Grill (2016) discovered a rather low inter-annotator agreement for the MIREX music similarity task, unveiling the problem of using a single-reference annotation for evaluating similarity algorithms. Koops *et al.* (2019) reached similar conclusions for the MIREX chord estimation task, showing low inter-annotator agreement for chord annotations between musical experts in the Chordify Annotator Subjectivity Dataset. Furthermore, Balke *et al.* (2016) concluded that the evaluation of automatic melody finding algorithms is heavily reliant on the human annotator's choice for providing ground truth.

## Relevance to pattern discovery algorithms

As discussed in the previous two chapters, the automatic discovery of musical patterns has been a longstanding research topic in computational music analysis, and the pursuit of a methodology to compare and evaluate those algorithms has garnered significant community effort, evolving into the MIREX task titled "Discovery of Repeated Themes & Sections", based heavily on work carried out in (Collins, 2011). In this task, the evaluation of newly proposed algorithms is carried out with reference annotations based on music-theoretic analyses by three experts. However, the ambiguity of musical structures and the diverse conceptualisations of the notion of patterns make the use of reference data relying on only a very small number of experts rather problematic: for the general audience, there is no clear single comprehensive definition of what constitutes a pattern, or even a repetition (see Chapter 1 and 2). Furthermore, not all recurring sequences are perceived as patterns by the listener, as this depends on the structural position of the pattern, the listener's moment-to-moment perception, and other influencing factors such as the listener's musical background or music-theoretic education. One can argue that automatically discovered patterns do not have to be perceivable, if they are useful in other contexts, such as for supporting automatic composition (Herremans & Chew, 2017) or classification (Boot *et al.*, 2016). However, in other contexts, it is unquestionably vital that automatically detected patterns are perceivable for listeners, such as in music education (Bamberger, 2000).

More extensive annotation experiments need to be carried out in order to gather reference data for the evaluation of pattern discovery algorithms that rely not only on a very restricted number of musical experts. Gathering annotations from different listeners on the same pieces, for example, could be an initial step in this direction that allows the study of differences and commonalities between listeners regarding their conceptualisation of patterns.

In (Meredith, 2015), it was noted that subjectivity could be reduced by enlisting a larger number of domain experts to reach a more universally agreed-upon ground truth. We will not have a solution for fabricating an inter-subjective **ground truth** by the end of the chapter. Rather than attempting to reduce the differences to some form of inter-subjectivity, our analysis focuses on dissecting the contributory factors of the individual variances. Such a study can pave the way for a more valid evaluation of algorithms, based on the consideration of commonalities and differences between listeners or groups of listeners.

#### HEMAN

The issue of inter-annotator agreement in pattern discovery was previously addressed through gathering multiple annotations for a single dataset from twelve annotators on six music excerpts using pen and paper (Nieto & Farbood, 2012). The data acquired from (Nieto & Farbood, 2012) initially faced the problem of reproducibility because they were not digitised. Figure 3.1 shows an example of the annotations before digitisation. In (Ren, Koops, *et al.*, 2018), the data was digitised as the *Human Estimations of Musically Agreeing Notes (HEMAN)* dataset and open-sourced<sup>1</sup>. While pen-and-paper annotation has been most commonly used for music-theoretical analysis in the past, the digitisation process afterwards is labour-intensive and error-prone. For example, in Figure 3.1, it is not immediately apparent whether to include the quaver (an eighth note) of A in bar 3 into the musical pattern. The same situation applies to the crochet in bar 14 in the annotated pattern.

For carrying out more extensive annotation experiments on musical patterns, digital tools supporting these annotations need to be developed. A digital tool can also provide functionality such as playback, enabling annotators to easily listen to the music.

#### Contribution

In this chapter, we contribute to the development of two musical pattern annotation tools and the annotation experiments conducted using these tools. *ANnOate MusIC, an annotation tool we devised (ANOMIC)* (Wells, 2019) is an annotation tool developed by a master's student at Utrecht University, and *Pattern Annotation Framework, an annotation tool we devised (PAF)* (Pesek *et al.*, 2019) by a master's student at the University of Ljubljana. The tools and experiments were developed asynchronously, but used the same musical excerpts as in *HEMAN*. As the daily supervisor, the master students received supervision from Iris Yuping Ren with the scope of this dissertation in mind.

While these digital annotation tools overcome the problems of handwritten annotation digitisation, they may influence the annotation process in a different way, such as through different music visualisations. We contribute to confirming and understanding the influence of different annotation interfaces and instructions on the discovered patterns, and study differences and commonalities between patterns discovered by annotators of different musical expertise and from different schools

---

<sup>1</sup><https://github.com/irisyupingren/HEMANanalysis>

Bach Bwv 1.6

Excerpt 1 ✓

Score Music21 Fragment Music21

The image shows a handwritten musical score for a fragment of Bach's BWV 1.6. The score is written on five staves of music in 4/4 time, with a key signature of one flat (B-flat). The music is annotated with various symbols: blue brackets and underlines group specific notes and phrases across the staves. A blue circle highlights a specific rhythmic pattern in the fourth staff. Below the score, there is a legend for the annotations: 'Motives' followed by a blue bracket symbol and the number '2', a blue wavy line symbol and the number '3', and a blue circle symbol followed by the text 'not as relevant'.

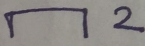
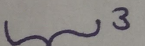
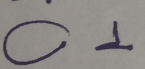
Motives  2  
 3  
 not as relevant

Figure 3.1: An example of raw paper-and-pen annotation.

of music theory. We do so by first exploring the gathered annotation datasets by performing inter-annotator agreement analysis. We then proceed to a second step, conducting a feature analysis of the patterns by looking into differences between feature distributions of musical patterns in both datasets. This feature analysis allows more detailed insight into the differences between the pattern datasets. Finally, we discuss more in detail and contribute to understanding the implications of the tools and annotators' backgrounds on the annotation process, along with the implications regarding future evaluation methods of pattern discovery algorithms based on rich annotation datasets.

In the original paper, the first author, Darian Tomašević was the primary contributor to the implementation of the annotation tool, PAF. The second author, Stephan Wells was the main contributor to the implementation of ANOMIC. The contributions of the third author, Iris Yuping Ren, include the development of the methodologies in Section 3.3, providing feedback on the experimental design, providing input for the analysis, as well as participating in the writing and subsequent revisions of the paper.

## 3.2 Setup for ANOMIC and PAF

In this section, we report the music material, the digitisation process, tools, and instructions we use for ANOMIC and PAF. ANOMIC and PAF are two musical pattern annotation tools with which two separate experiments were conducted. ANOMIC and PAF were both inspired by HEMAN and use the same music pieces to annotate. We start first by introducing the common music materials for HEMAN, ANOMIC, and PAF. Then, we introduce the tools, experiments, and the differences between them. As a side note, we use these acronyms for three purposes throughout this chapter: naming the experiments, the tools, and the generated annotations.

### 3.2.1 Music material

Throughout this chapter, we used the same music material. The collection comprises 6 music excerpts as listed below:

- Bach – Cantata BWV 1, Movement 6, Horn (20 bars)
- Bach – Cantata BWV 2, Movement 6, Soprano (15 bars)
- Beethoven – String Quartet, Op. 18, No. 1, Violin I (60 bars)
- Haydn – String Quartet, Op. 74, No. 1, Violin I (30 bars)



- Mozart – String Quartet, K. 155, Violin I (28 bars)
- Mozart – String Quartet, K. 458, Violin I (54 bars)

These pieces have been selected by Nieto *and* Farbood (2012) for pattern annotation experiments due to their different musical characteristics. For example, the Bach chorale is short and has very little rhythmic variation, while the Beethoven string quartet is long, and has clear repetitions varied in pitch. In the original paper (Nieto & Farbood, 2012), more detail was given. For long pieces, we use roughly the first page of sheet music (precise bar numbers are included in the listing).

### 3.2.2 ANOMIC

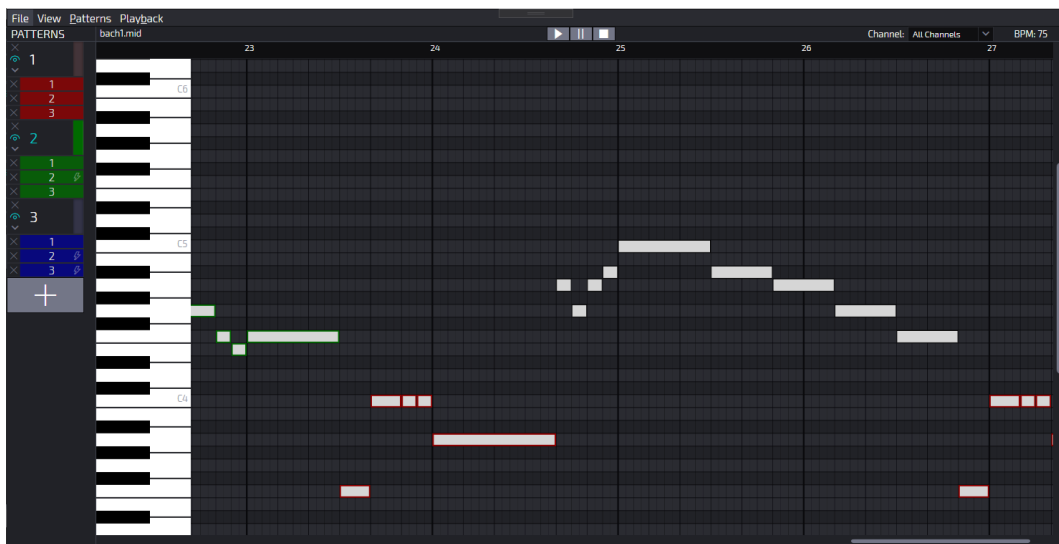


Figure 3.2: A screenshot of the interface of ANOMIC. Different colours are used to represent different patterns and their occurrences, which are also numbered on the left of the interface.

The ANOMIC pattern annotation tool was created as a stand alone application for the Microsoft Windows operating system (Wells, 2019). The main view of the tool visualises the MIDI representation of a music piece as a piano roll by plotting the musical notes as rectangles on a two-dimensional onset-time-pitch canvas. This approach is also commonly used in music editing software, as it allows for easy interaction with MIDI elements (most commonly musical notes). Figure 3.2 shows ANOMIC’s interface, where the piano roll representation is visible.

For the experiment of ANOMIC, a demo video was provided for making instructions on the pattern annotation accessible to users even with little musical training. An instruction document was also given in written form. Below, we include the most relevant instructions.

#### Instruction

With ANOMIC, a group of users installed the tool on their computers and were then asked to submit their annotations via email, along with a completed survey about their background. If we take away the instructions for installing the software, the more relevant instructions for the annotation experiment are as follows:

... Annotate repeated patterns in the music. Patterns are distinct, short musical segments or phrases that are considered to be characteristic to a given piece of music and appear multiple times throughout the piece. Be sure to listen to the music and annotate these patterns and their occurrences using the tool. The occurrences do not need to be exact matches, but they should be closely related (compare this to finding occurrences of a leitmotif in a film soundtrack, for example). For any occurrences where you are not sure whether or not they belong to a pattern, right-click on them to set their confidence level to "2" (unsure) or "1" (highly uncertain).

**Automatic Occurrence Matching** At any point during the session, the user can right-click on an occurrence that they had already found and initiate the automatic occurrence matcher for that occurrence. It will comb through the entire track and find occurrences that are exact repetitions of the occurrence in question, even if these repetitions may not be in the same time or pitch position. This implies that it is transposition-invariant and also works through polyphony.

The description of a musical pattern is similar to the instructions given in the experiment conducted by Nieto *and* Farbood (2012). Following Nieto *and* Farbood (2012), several possible interpretations as to what defines a "musical motif" were given, enabling the subjects to analyse the pieces and employ their musical intuition on what constitutes a musical pattern in the process. A new functionality was added for finding exact and chromatically transposed occurrences of annotations. We will be looking at a cross-comparison between the tools and the experiments in the following sections (Section 3.2.4 and Section 3.2.5).

#### Subjects

The ANOMIC tool was used in an annotation experiment with 26 participants of varying musical backgrounds and demographics<sup>2</sup>. In total, 788 patterns and 2,763 occurrences were gathered, annotated by 26 participants with diverse backgrounds. The participants' levels of musical expertise were assessed through a survey in-

---

<sup>2</sup>Available at <https://github.com/StephanWells/ANOMIC-dataset>

cluding 10 questions (e.g. ability to read sheet music, proficiency in playing an instrument, academic degree in music), leading to a score between 1 and 10. As a rough approximation of their musical expertise, the annotators were divided into two groups with a cutoff of 5, dubbed the musicians (14 participants) and the non-musicians (12 participants). Interested readers can refer to (Wells, 2019) for all details.

### 3.2.3 PAF

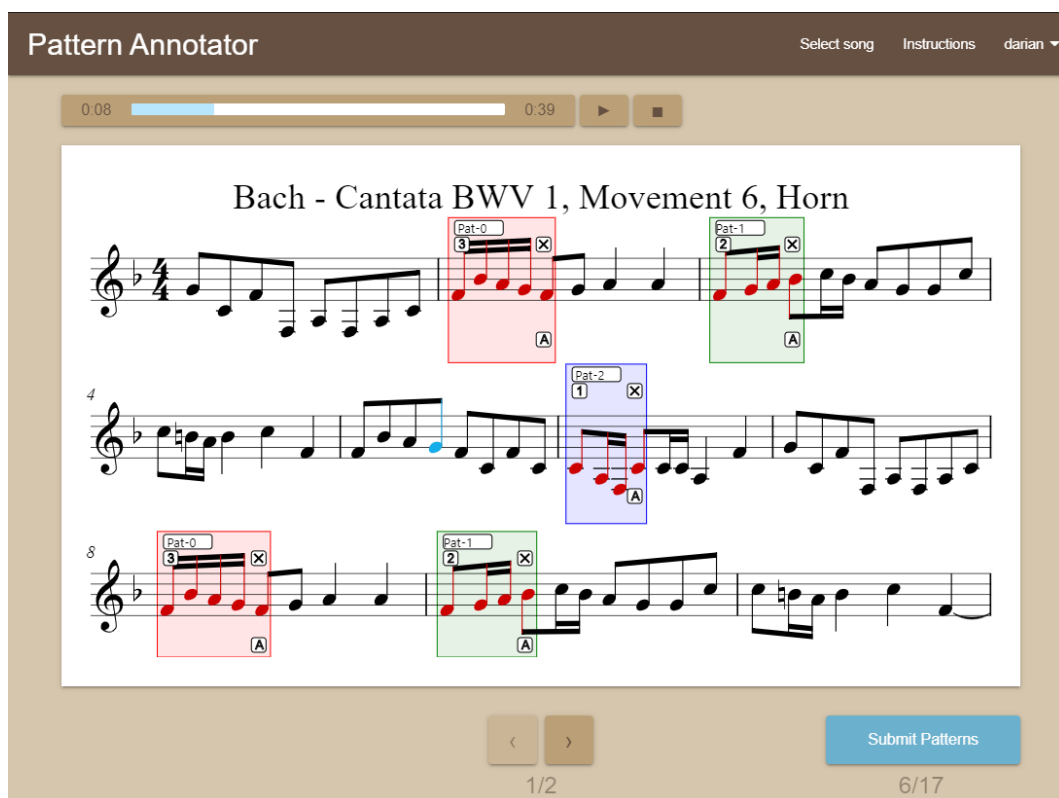


Figure 3.3: A screenshot of the interface of PAF. Different colours are used to represent various patterns and their occurrences, each of which is indexed by "Pat-n", with 'n' representing the number of a pattern. The user can also assign a relevance score to each pattern, which is displayed below the index.

The PAF tool<sup>3</sup> was developed as an online application (Pesek *et al.*, 2019), with the Django framework serving as the back-end, and with the MIDI Player and Verovio JavaScript libraries serving as the front-end visual representation of the music sheet. The tool shows the music score of a selected piece of music, as seen in Figure 3.3. Thus, the PAF tool is designed to be used mainly by individuals with a certain degree of musical expertise, namely, those who can at least read staff notation. The user can annotate music patterns by clicking on the beginning and the ending note

<sup>3</sup>Tool available at <http://framework.musiclab.si>

of the desired range in the score. The user can also set the name and the relevance (1–3) of an individual annotated pattern. To aid the annotator, the displayed score can be listened to with our synthesised MIDI files. While playing the score, the notes nearest to the current time in the music playback are highlighted. The tool also contains a feature for automatic annotation, which annotates other occurrences of the selected pattern. While developing the framework, a major focus was on the responsiveness of the layout to accommodate different screen sizes and devices (mobile, desktop). In the case of smaller display size, the score is split into multiple sheets. By open-sourcing the tool<sup>4</sup>, we hope to aid other researchers in the MIR field who are dealing with pattern-related data gathering and invite them to contribute additional features.

#### **Instruction**

The following instructions<sup>5</sup> were given to the PAF annotators:

The goal of the site is to allow users to find and submit patterns found in music.

A musical pattern or motive is defined as a short musical idea, a salient recurring figure, musical fragment, or succession of notes that has some special importance in or is characteristic of a composition<sup>6</sup>. It shouldn't be longer than a musical phrase. If you find a motive that is similar to another (or multiple versions of a motive), choose the one that you think is the most representative.

To begin, click on a note to mark it as the start of a pattern. The starting note will be coloured blue. Next, click on another note to mark the end of that pattern. All of the notes between the start and the end will be coloured red, thus representing a marked pattern<sup>7</sup>. To mark multiple patterns, simply repeat the process.

Each pattern also includes an ID field (top left corner), which can be used to change the ID of a pattern. To denote multiple pattern occurrences, the ID of a repeated pattern can be changed to match the ID of the original pattern. Alternatively, exact repetitions of a pattern can also be annotated

---

<sup>4</sup>Source code available at <https://bitbucket.org/ul-fri-lgm/patternannotationframework>

<sup>5</sup>Footnotes were not given to the annotators. They are for clarifications in the context of this dissertation.

<sup>6</sup>This description of a musical pattern is the same as the description given in the experiment conducted in (Nieto & Farbood, 2012).

<sup>7</sup>Since the description of a pattern is different between (Nieto & Farbood, 2012) and this dissertation, in the context of this dissertation, the user will be marking a pattern **occurrence** by clicking on the starting and ending point of an excerpt, not all the repetitions of a musical pattern.

automatically. By pressing the A button on a pattern (bottom right corner) all repetitions of the selected pattern will be automatically marked. Once a pattern is finalised you can also rate it from 1 (not as relevant) to 3 (highly relevant).

To delete an UNFINISHED pattern click on its starting note, which is coloured blue.

To delete an already FINISHED pattern press on the X button, which can be found in the right corner of each pattern.

To navigate the sheet music use the buttons located below the sheet music. You can also use the arrow keys on your keyboard to achieve the same result. Once you are finished click on the SUBMIT PATTERNS button and click OK. The patterns will then be submitted and the site will take you to your next task.

To zoom out or in press - or +.

To use the music player simply click on the play button. You can also navigate the song by clicking on the song progress bar in the top left.

The description of a musical pattern is the same as in (Nieto & Farbood, 2012). A range of descriptions of functionality was added to enable the user interaction with music and patterns.

### Subjects

The PAF tool was used by 13 students attending three university-level music study programmes: 4 students from the musicology masters programme at the Faculty of Arts, the University of Ljubljana (hereafter termed as MU), 3 students of the Music Academy, including music theory and composition at the University of Ljubljana (termed as TC), and 6 students from the music pedagogy programme at the Faculty of Education, the University of Maribor (termed as PE). The PAF website included a description of the tool and a summary of the tool's features. The majority of participants were students between 20–25 years old, having between 5–10 years of instrument experience, and between 8–15 years of music theory experience. In total, 373 annotations were gathered from 13 annotators over the course of a month.

### 3.2.4 Differences between the tools

We reviewed two digital pattern annotation tools that were developed concurrently—ANOMIC and PAF—and two annotation experiments conducted separately using these tools. While both tools allow for more accessible digital annotation of

### 3 *Gathering Human-Annotated Musical Patterns*

patterns, they differ in their implementation, functionality, interface, and music representation. The most glaring concerns the different visualisation of music they employ: piano roll or sheet music. While these visualisations of music may seem equivalent at first glance, they represent the same information in different ways that can in turn influence the users.

For example, the music sheet of the PAF tool can show a larger part of a music piece than the ANOMIC tool, in which the notes can appear too small when zoomed out. The visual presentation of note durations is also different. While notes are of similar sizes on the music sheet, independent of their duration, the notes on the piano roll differ tremendously in their size. Therefore, longer notes might thus attract more attention and be harder to scroll past. On the other hand, the sheet music representation can also present a better user experience on smaller screens. The music sheet can also contain additional information, apart from the actual notes, such as auxiliary signs for timbre and volume, which can influence user understanding of the music piece.

The tools also differ in how they enable the selection of patterns and the annotation of their occurrences. The controls are smoother in the ANOMIC tool, which provides the ability to use click-and-drag actions to select patterns. Conversely, annotating in the PAF tool is carried out by clicking on the starting and then ending note of a pattern. The click-and-drag approach could be perceived as more intuitive, especially considering the left-to-right piano roll visualisation, whereas demarcating the beginning and ending of a pattern makes the annotation of longer patterns faster. When the annotation starts, for both tools, a default pattern number (ID) is given at first, and if the users proceed to a different group of pattern occurrences, they can use a new number to signify this new group.

ANOMIC also has a helper function to enable users to automatically annotate exact repetitions and chromatic transpositions of already annotated patterns, as implemented by an automatic occurrence matching function (Wells, 2019). This function can ease the labour-intensive search necessary in order to annotate all occurrences of a pattern, which is not trivial for annotators (see (Volk & Van Kranenburg, 2012)). Although it was not mandatory to use this function during annotation, in a follow-up survey, we saw that the helper function was used at least 9 times by all participants, and that the function was liked by some annotators, but one participant said that it "felt like cheating". A detailed user study of ANOMIC and the helper function in particular can be found in (Wells, 2019). While PAF was later updated to include such an automatic annotation feature, it was not available in the version used for gath-

ering the data for the current analysis. Regarding other basic annotation controls, such as pattern deletion and tagging, the tools have similar functionalities.

The tools take different data gathering approaches. ANOMIC is a standalone tool and can be used offline. The gathered patterns must then be submitted via external means, which can affect the number of gathered submissions. Meanwhile, PAF is implemented as an online tool, which automatically saves the annotated patterns in a database. In addition, users must first register and provide their background information to access the tool. In comparison, the ANOMIC tool does not come with built-in registration or user information gathering functionality, so researchers must resort to online surveys. The tools also differ in the overall annotation process: PAF does not allow re-annotation of music pieces and only allows annotation of the provided music pieces, while ANOMIC allows annotation of any music piece the user provides.

Both tools allow for the playback of the presented music piece. When music is played in PAF, the currently played notes are highlighted, and the sheet pages change when the last note on the page has been played. The tool also has a separate music player with which the user can navigate through the song. The ANOMIC tool takes a different approach by providing a tracker, in the form of a vertical line that denotes the current position of the music.

These differences and similarities are still subject to change. Like many technologies nowadays, annotation tools are evolving. Future updates will most likely focus on the addition of user-friendly features, and in particular, how to enable the users to more easily modify the tools (e.g. adding helper functions) while annotating. We summarise the main differences between these two tools in Table 3.1.

#### 3.2.5 Differences between the experiments

The description of a musical pattern is similar across all experiments. For the PAF tool, the instructions for the participants were actually designed to closely follow those used in (Nieto & Farbood, 2012). For ANOMIC, the intentions were to improve upon the instructions of (Nieto & Farbood, 2012) in conjunction with the helper functions.

In summary, there are two marked differences between the instructions, those being: whether participants were asked to listen to the music and how to annotate occurrences of a pattern. While the instructions for using ANOMIC explicitly mention the importance of occurrences, PAF users were instructed to choose a representative pattern when they saw multiple similar ones. In the experiment using ANOMIC,

Tool	ANOMIC	PAF
Platform	Windows	Online
Visualisation	Piano roll	Sheet music
Background information gathering	External survey	Online registration
Musical background	Self-rated music skills	Music pedagogy and theory students
Helper functions	Auto-tagging exact rep. and chromatic transp.	None

Table 3.1: Summary of the main differences between the ANOMIC and PAF annotation tools and their experimental subjects. Here, "rep." stands for repetition, and "transp." stands for transposition.



the participants were asked to listen to the music, whereas with PAF, it was not explicitly asked.

In terms of the differences in instructions regarding whether the annotators listened to the music, we could not check whether the participants actually listened to the music for either experiment. This can be accomplished in future work by logging annotators' behaviours and interaction with the annotation tool, so that we see when and for how long the annotators have been listening to the music.

In terms of the differences in instructions regarding whether annotators annotated all occurrences, it is a more complex problem. We believe that annotating all occurrences can be more helpful for developing musical pattern discovery algorithms. However, without some helper functions, such as the ones in ANOMIC, humans might not be able to or lose patience in meticulously annotating occurrences of a pattern without errors. For PAF, although we could not conduct the experiment with the helper function, we used the newly added helper functions post-experiment to augment the collected data by identifying their repetitions. We will see that this does not entirely equalise the ANOMIC and PAF annotation experiments, but at least we add the notion of pattern occurrences in the way we can. As a result, for subsequent comparisons, we make comparisons only between ANOMIC and PAF data where the notion of pattern occurrences is present to an extent.

These discrepancies are not ideal for perfectly controlled comparisons. However, since these tools and experiments were developed separately with the same goal—gathering data for musical pattern discovery, there are values and lessons we can learn from looking at the results in parallel. We leave a more controlled experiment for future work and examine what comes when diversity is allowed during the setup of a musical pattern data collection experiment.

### 3.3 Methodology and metrics

Our analysis has three exploratory aspects: differences between annotators with different backgrounds, differences between annotating the most representative pattern and all occurrences of a pattern, and differences between annotations gathered with two different tools. All aspects will be explored by analysing inter-annotator agreement and by analysing the distributions of various pattern features, to gain deeper insights as to why annotators might have disagreed. In the following, we describe these methods in greater detail.

### 3.3.1 Agreement analysis

To measure inter-annotator agreement, an important concept is that of "matched" annotations. Given two pattern occurrences  $P_1$  and  $P_2$ , with beginnings and endings denoted as  $b_1, b_2$  and  $e_1, e_2$ , were considered to be matched when  $|b_1 - b_2| + |e_1 - e_2| \leq T$ , where  $T$  denotes a threshold value. The vertical bar notation indicates "taking absolute value" to disambiguate from taking cardinality of sets.

Given two sets of pattern occurrences  $R$  and  $C$  from two annotators, we call one set the reference ( $R$ ), and the other set the comparing set ( $C$ ). It does not matter which set is taken as the reference, because we will eventually consider the other set as the reference as well. Using  $\#$  as "the number of" sign, we then calculate the commonly used measures, namely, *precision*, *recall*, and  $F_1$  score for all possible pairings of annotators, as specified in the equations below. Each individual annotator is compared to every other annotator.

$$\text{Precision} = \frac{\#\text{matched annotations}}{\#C}$$

$$\text{Recall} = \frac{\#\text{matched annotations}}{\#R}$$

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

With a number of annotations in a piece, we can expect the precision, recall, and  $F_1$  score to give us a summary of the agreed patterns between any pairs of annotators. These measures will fit the intuition: the more far apart the different annotated pattern boundaries are, the more they disagree; the greater the number of patterns the annotators disagreed upon, the more two annotators disagree. For example, if annotator A noted that the second bar of a musical piece is a pattern, while annotator B included the last quaver (an eighth note) of the first bar and the second bar as a pattern, we have the same pattern ending, but a slightly different beginning. The threshold value gives us the flexibility to configure whether the two annotators should be considered to be in agreement (matched) or not. In the example above, if the two patterns are the only annotations in the piece, we have an  $F_1$  score of 0 if  $T < 1$  quaver (an eighth note), 1 otherwise. In this way, we can see how much disagreement there is on different scales of time resolution. The reason that we focus on the beginnings and endings of patterns is that, within the same piece of music, once the beginnings and endings are determined, the excerpt's content is the same for monophonic melodies.

In the following analysis, the starts and ends were measured in crotchets (quarter notes), and the threshold was set to 5 crotchets as the default for this chapter, following (Ren, Koops, *et al.*, 2018). In future work, other threshold values or dynamic thresholds should be investigated.

#### **In relation to MIREX's metrics**

This measure does not have direct correspondence with the occurrence or establishment scores in MIREX. MIREX measures are based on the cardinality score, which considers the actual musical events themselves (pitch and duration), and not their position in the piece (see next chapter Section 4.1 for more detail on MIREX measures). There is an indirect connection when considering that the musical events are the same given the same position in the same piece. However, the "matched annotation" for a single occurrence is a binary notion in our case, while in MIREX measures, each occurrence has a fractional value in relation to another occurrence, and then a maximum is taken. In summary, the computational process and the focus of the evaluation are both quite different between our measure and MIREX's.

#### **In relation to other inter-rater reliability measures**

Other inter-rater reliability and agreement measures exist in literature. However, it is not straightforward to convert the annotations of musical patterns to be compatible with other inter-rater reliability measures, such as Krippendorff's alpha coefficient. To calculate these measures, one usually starts with a matrix of  $n \times m$ , where  $n$  is the number of annotators, and  $m$  is the number of attribute values that will get annotated. In our case, we do not have this matrix, because each annotator might annotate a different number of patterns that need to be "matched" first, as we devised in our measure. What follows from this matching step is then counting the number of "matched" annotations and calculating corresponding percentages. As a consequence, our measures are intuitive and easy-to-understand. We do agree that our measures are simplistic and can be improved with more sophisticated statistical methods in the future.

### **3.3.2 Feature analysis**

In order to further explore the differences between annotations of the two digital tools, we compared both annotation datasets on 33 different pattern features, of which we report 7 here, for simplicity, while all 33 musical features are described

in detail in supplemental online material<sup>8</sup>. The majority of the features were inspired by the work of Collins (2011), in which musical patterns were rated based on a plethora of musical features from previous research, including (Cambouropoulos, 2006; Conklin & Bergeron, 2008; Forth & Wiggins, 2009; Meredith *et al.*, 2002; Pearce & Wiggins, 2007). Several features were also inspired by the research of Van Kranenburg *et al.* (2013), which compared global and local features of folk song melodies.

To compare features of the patterns, we take the first occurrence of each pattern annotated in ANOMIC, as PAF annotators only annotated the most representative occurrence of each pattern. We take the first occurrence because the first occurrence of a pattern in a musical piece tends to have a more significant role according to Schoenberg (1967). This taking-the-first-occurrence approach has an exception for the Occurrence feature that counts the number of occurrences of a pattern. With this Occurrences feature, namely the number of times that a pattern repeats, we use all occurrences from ANOMIC to see whether the annotators actually followed the instructions closely regarding annotating single or multiple occurrences.

We then compare features of the patterns, we computed musical features of each pattern, thus forming feature distributions for both datasets. Next, the distributions of each feature were normalised by taking the minimum and maximum values across both distributions and performing min-max feature scaling. Thus, we obtained feature distributions with a range of 0 to 1, which are visualised in Figure 3.5. The computation process of each analysed feature is described in detail in the supplemental online material<sup>8</sup>, which also includes the Python source code used for the analysis.

In Table 3.2, we list the 7 features included in this paper, namely those which we consider to be most intuitively related to pattern characteristics perceivable by users (such as the duration of the last note or the note range). Notice that the Occurrence feature is what we mentioned as an exception to the taking-the-first-occurrence approach above. Furthermore, following the comparison of local and global features in (Van Kranenburg *et al.*, 2013), we focused on local features, which are more likely to be assessed by humans when annotating patterns. We further reduced the number of important features by analysing the Spearman's and Pearson correlation coefficients between pairs of features. The highest Pearson correlation value appeared between the feature pattern duration and note range, which has a correlation of 0.64 and 0.65 for ANOMIC and PAF. Tables of the most correlated features for each dataset can be found in the supplemental online material<sup>8</sup>.

---

<sup>8</sup>Results available at [https://bitbucket.org/anonymous\\_submitter/agreement-in-musical-patterns](https://bitbucket.org/anonymous_submitter/agreement-in-musical-patterns)

Name of feature	Description
Pattern Duration	The duration of a pattern in crotchets
Occurrences	The number of times that a pattern repeats
Last Note Duration	The duration of the last note of a pattern
Note Range	The number of semitones between the lowest and highest note of a pattern
Pitch Direction Changes	The number of melodic arcs in a pattern
Intervallic Leaps	The fraction of all intervals of a pattern that are intervallic leaps ( $>$ two semitones)
Root Notes	The fraction of notes in a pattern that are root notes or their octaves

Table 3.2: Feature descriptions

Our last step to analyse pattern features is based on independent two-sample student tests (t-tests), two-sample Kolmogorov-Smirnov tests (KS-tests) and boxplot visualisations of distributions. Given our null hypothesis that the samples are drawn from the same distribution, and given that we are unsure whether our data is normally distributed, we use both parametric and non-parametric tests to verify how likely it is that the distributions actually differ. The differences are represented by high t and D statistic values in combination with low p values of the performed tests. We also considered boxplot visualisations of the distributions to better understand the shapes of the distributions and the differences between them.

## 3.4 Results

In this section, we report the results of comparisons between ANOMIC and PAF. We do not compare them together with the HEMAN data as explained in Section 3.2.5: the annotated patterns were not grouped into a pattern-occurrence relation, and therefore were difficult to compare with those that were grouped. We use the two kinds of method introduced in Section 3.3: agreement analysis and feature analysis.

### 3.4.1 Agreement analysis

In order to analyse inter-annotator agreement, we computed the  $F_1$  scores of all annotator pairs across all music pieces. We gathered these values into  $F_1$  score matrices, which allow us to visually inspect the results in a compact manner. Figure 3.4 shows these matrices, in which annotators are grouped based on the annotation tool used and their backgrounds. The analysed groups include the TC, MU and PE groups of PAF as well as the musician and the non-musician groups of ANOMIC (see Section 3.2.3 and 3.2.2 for the setup of the experiments). Based on the obtained inter-annotator agreement values, we refer to the values above 0.95 (yellow matrix values) as indicators for a strong agreement in this paper. It should be noted that the number of annotations was not split equally among music excerpts, as some annotators of the PAF tool did not annotate the last three excerpts. This was likely due to the fixed order of music excerpts and the significant time investment in the annotation process. Once this issue of the PAF tool was identified, it was reported and addressed by the developers, who randomised the ordering to improve the tool for future use.

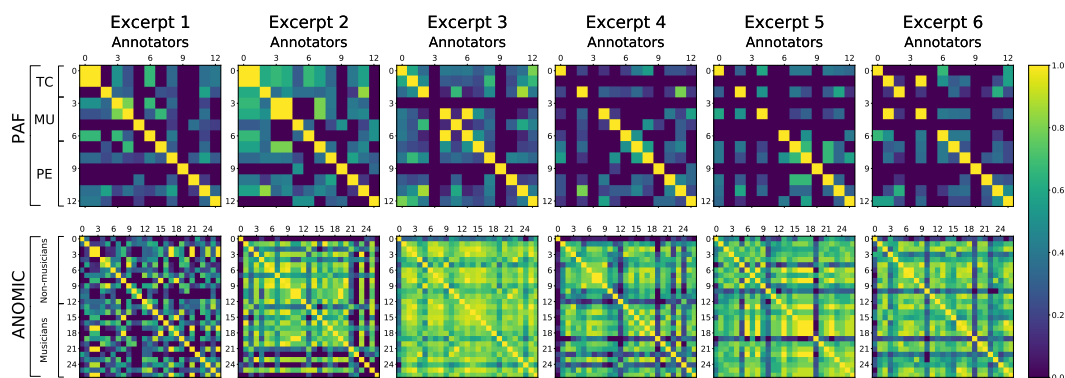


Figure 3.4:  $F_1$  score matrices, representing the pairwise agreement between annotators, with the time threshold set to 5 crotchets. Each matrix showcases results for a different music excerpt. Matrix columns and rows denote different annotators, grouped based on the annotation tool used (ANOMIC or PAF) and their musical background. Matrix cell colours correspond to the obtained pairwise agreement values, where yellow denotes high agreement and blue indicates low. Some annotators did not provide annotations for all excerpts, which can be seen along the diagonals as low agreement.

We will not make a comprehensive cross-comparison between ANOMIC and PAF using our measure introduced in 3.3.1, because of the single- or multiple-occurrence difference in the instructions of the tools. The concept of "matching annotations" is a complicated one if we compare the most representative occurrence annotation of a pattern with all the occurrences annotated for a pattern. In addition, "the most representative" and "all occurrences" are not guaranteed as the annotators can only do the best they can. We will, therefore, leave this to be explored in future work.

## PAF

The TC and MU groups show strong agreement ( $>0.95$ ) on the first three music excerpts. The results of the PE group show many weaker agreements, despite a more significant number of annotators. Contrary to the TC and MU groups, the annotators of the PE group did somewhat agree with the annotations for excerpt 4. There is only one strong agreement between annotators belonging to different groups (excerpt 6, annotators 1 and 4). Since there are several strong agreements between the annotators within individual groups, the lack of agreement between different groups could indicate the potential influence of different study programmes on the annotators' perception of the most representative musical patterns.

#### **ANOMIC**

For ANOMIC, the agreement values for the two subgroups are similar: the average inter-annotator agreement for the musician group is 0.61 and 0.63 for non-musicians. While some agreement does exist between the ANOMIC and PAF annotators, we only observe one strong agreement between two PAF annotators and one non-musician annotator of ANOMIC in excerpt 1. We do not make further comparisons between ANOMIC and PAF disagreements, as a range of factors could have contributed to their differences, such as differences in instructions.

#### **A remark on taking averages**

We have an additional note for these comparisons. In taking averages, we can compare between groups while marginalising the effects of individual differences between musical pieces. We are aware that this is not always valid because there is a varying degree of difficulty in finding patterns across different musical pieces. It is possible that a group of annotators disagree strongly on one single piece and agree perfectly with each other on the rest, which would be obscured in the average, with the music being a confounding factor. However, in Figure 3.4, we see a range of disagreement and agreement. Admittedly, excerpt 1 is more disagreed upon than others, so we also calculated the values by only using the other five excerpts, and the results and conclusions did not change. Furthermore, the computation of the average was based on the whole matrices, thus including values where users did not provide any annotations. These values were simply set to 0 and were included in the computation. We also analysed the average values if these values were ignored. Despite affecting the average values of the comparison, the changes in values were not significant since the values, based on PAF users, simply increased by around 0.01. Thus, we decided to only report the original values, which included missing annotations.

#### **3.4.2 Feature analysis**

As introduced in Section 3.3.2, we analysed, for each pattern feature, the annotations of ANOMIC annotators with musical and non-musical backgrounds and compared them to the PAF dataset, whose annotators all had a musical background. We investigated whether the difference between datasets was also present in these background subgroups to identify if the observed difference between the ANOMIC and the PAF dataset was influenced primarily by the tools or the musical backgrounds



**t-tests between pattern features**  
(t and p values of t-test)

	<b>PAF / ANOMIC</b>	<b>PAF / Mus.</b>	<b>PAF / Non-Mus.</b>	<b>Mus. / Non-Mus.</b>
Pattern Duration	7.54 ( $9.60 \times 10^{-14}$ )	4.06 ( $5.38 \times 10^{-05}$ )	7.68 ( $5.09 \times 10^{-14}$ )	4.91 ( $1.09 \times 10^{-06}$ )
Occurrences	-11.96 ( $4.35 \times 10^{-31}$ )	-16.42 ( $2.98 \times 10^{-51}$ )	-11.12 ( $1.13 \times 10^{-26}$ )	-3.1 ( $1.98 \times 10^{-03}$ )
Last Note Duration	6.75 ( $2.45 \times 10^{-11}$ )	3.93 ( $9.30 \times 10^{-05}$ )	7.36 ( $4.90 \times 10^{-13}$ )	3.49 ( $5.15 \times 10^{-04}$ )
Note Range	5.18 ( $2.57 \times 10^{-07}$ )	1.62 ( $1.05 \times 10^{-01}$ )	6.87 ( $1.36 \times 10^{-11}$ )	6.15 ( $1.23 \times 10^{-09}$ )
Pitch Direction Changes	3.53 ( $4.25 \times 10^{-04}$ )	-0.02 ( $9.86 \times 10^{-01}$ )	5.98 ( $3.55 \times 10^{-09}$ )	5.92 ( $4.79 \times 10^{-09}$ )
Intervalllic Leaps	3.89 ( $1.07 \times 10^{-04}$ )	1.95 ( $5.11 \times 10^{-02}$ )	4.7 ( $3.07 \times 10^{-06}$ )	2.75 ( $6.12 \times 10^{-03}$ )
Root Notes	4.45 ( $9.26 \times 10^{-06}$ )	3.51 ( $4.73 \times 10^{-04}$ )	3.95 ( $8.67 \times 10^{-05}$ )	0.37 ( $7.08 \times 10^{-01}$ )

Table 3.3: Table showing t and p values of t-tests with  $\alpha = 0.05$ , between the ANOMIC and PAF (musicians and non-musicians) datasets on the features of the first column. Orange cells denote tests with  $p > 0.05$ , which means that the difference between the two tested distributions is not statistically significant.

of the annotators. Additionally, based on how intuitive they are and their correlations, we narrowed the list down to 7 features: pattern duration, number of occurrences, last note duration, note range, pitch direction changes, intervallic leaps and root notes. These features are also listed in Table 3.2, along with their descriptions. Plotted distributions of these features can be seen in Figure 3.5. A list of all analysed features, all t-test and KS-test results as well as all boxplot visualisations are available online<sup>8</sup>.

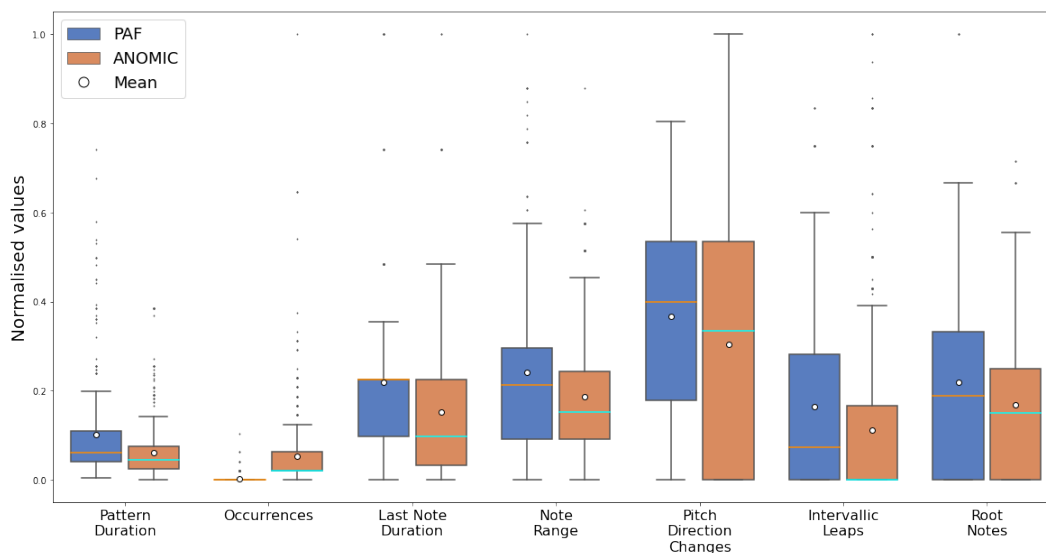


Figure 3.5: Boxplots showcasing the distributions of the analysed pattern features of the ANOMIC and the PAF datasets. For each feature, the two distributions were normalised.

#### Pattern Duration and Occurrence features

The first of the features we analysed was the **Pattern Duration** feature, which measures the length of a pattern in quarter notes (crotchets). We observed that the ANOMIC distribution had much smaller overall and interquartile ranges than the PAF distribution. The mean and median values of the distribution were also much smaller.

Next was the **Occurrences** feature, which refers to the number of times that a pattern occurs in a music excerpt, as defined by Collins (2011). From the boxplots in Figure 3.5, it is evident that the PAF distribution differs drastically from the ANOMIC distribution, which is expected due to the difference in instructions given about whether to annotate all occurrences or not. This result confirms that the annotators followed the instructions closely in this respect.

Differences between the two datasets were also seen for the **Last Note Duration** feature, which describes the duration of the last note of a pattern. Here, the

ANOMIC distribution had larger interquartile and overall ranges than the PAF distribution, while its mean and median values were lower.

The t-tests in Table 3.3 and the KS-tests (available online<sup>8</sup>) confirmed that the above-mentioned differences between distribution pairs were statistically significant. Furthermore, they revealed that the differences between the PAF dataset and the two subgroups of the ANOMIC dataset, based on musical backgrounds, were also statistically significant. Thus, we may conclude that the differences between feature distributions of pattern datasets were influenced by varying musical backgrounds but also possibly by the interfaces or the instructions of the annotation tools.

### Pitch-based features

The two annotation datasets also differed on the **Note Range** feature, which describes the number of semitones between the lowest and highest note of a pattern. Figure 3.5 shows that the PAF distribution has a much larger overall and interquartile range than the ANOMIC distribution. Its median and mean values are also higher.

Similar differences were also seen for the **Pitch Direction Changes** feature, which is defined as the number of melodic arcs in a pattern, drawing inspiration from various melodic arc features in (Van Kranenburg *et al.*, 2013). Our analysis showed that the ANOMIC distribution had a much larger overall and interquartile range. Furthermore, the distribution lacks the bottom whisker, thus being more positively skewed. The ANOMIC distribution also had significantly lower mean and median values.

We observed comparable differences among the **Intervallic Leaps** feature distributions. The feature describes the fraction of all melodic intervals of a pattern whose note range is larger than two semitones, as defined by Collins (2011). Considering the analysis results, we see that the PAF distribution has much larger overall and interquartile ranges. Its median and mean values are also significantly higher.

Finally, differences were seen for the **Root Notes** feature, which describes the fraction of notes in a pattern that are root notes or octaves of roots of the music piece. From the distributions present in Figure 3.5, we can discern that the distributions mainly differ in the overall and interquartile ranges, with the PAF distribution having much larger ranges than the ANOMIC distribution. We also note that the mean value, as well as the standard deviation of the PAF distribution, are slightly higher than those of ANOMIC.

By analysing the t-test results from Table 3.3 and the KS-test results (available online<sup>8</sup>), we confirm that the distribution differences of the first three pitch-based features between the two datasets are statistically significant. We notice that

these three features pertain to the relationships between notes in the patterns and are therefore melodically relevant. We can also discern that the differences are not statistically significant between the PAF group and the musician group of ANOMIC. However, the difference between musicians, both ANOMIC and PAF, and non-musicians of ANOMIC, is statistically significant.

## 3.5 Discussion

Based on our comparison results between ANOMIC and PAF in the last section, we draw a few conclusions in this section:

- Given the same music, familiarity with music, including the type of musical study programmes attended by the annotator, influences the annotated musical patterns.
- The interface and the automatic occurrence matching functionality of the musical pattern annotation tools influence how long and how often the musical patterns are annotated.
- Features such as note range, intervallic leaps, and pitch direction are statistically similar between ANOMIC and PAF musicians and significant different between these musicians and non-musicians in ANOMIC. Therefore, excluding the potential influence of the annotation tools, the varying musical background of the ANOMIC annotators influences the annotated musical patterns.

Please see a more in-depth discussion on each point below, as well as an outlook on human musical pattern annotations at the end of this section.

Comparing the annotations collected with the PAF tool enabled us to study three groups of annotators from different musical study programmes. We observed higher agreement between annotators of the same group when compared to annotators of different groups, indicating a potential influence of study programmes on the understanding and perception of patterns in music. Comparing the annotations of the ANOMIC annotation set, which included two groups of annotators (musicians and non-musicians), gave a similar result.

Several differences observed in the feature-based comparison of the annotation datasets point towards discrepancies in annotations caused by design differences in the tools and experiments. The PAF interface displays sheet music allowing a compact representation, with large sections of the music piece being presented to the user within a single view. By contrast, ANOMIC's piano roll representation generally displays fewer elements at once to the user in order to preserve element clarity. The difference in music visualisation might have caused users to perceive and anno-

tate patterns with durations relative to the view window size. The notes in the sheet music representation of PAF remained roughly the same size, while the piano roll elements of ANOMIC varied in their size based on the durations. Since the last note durations of the PAF dataset are on average significantly longer, differently sized elements of the ANOMIC tool could have discouraged users from picking longer notes as pattern endings. The large disparity in the occurrences feature distributions was most likely caused by the lack of automatic occurrence matching functionality in the PAF tool, as well as differences in the instructions given to the users, despite the augmentation step as described in Section 3.2.5.

Moreover, we observed differences in note range, intervallic leaps, and pitch direction changes between ANOMIC non-musicians versus ANOMIC musicians and PAF annotators. The t-test analysis showed that the differences were not statistically significant between the PAF group and the musician group of ANOMIC. However, the difference between musicians, both ANOMIC and PAF, and non-musicians of ANOMIC, was statistically significant. We, therefore, conclude the varying musical background of the ANOMIC annotators as the influential factor, and exclude the potential influence of the annotation tools. We also speculate that annotators with different musical backgrounds tend to annotate patterns with different melodic properties. Finally, for the root notes feature, we are unsure what might have caused the difference in distributions, though we believe it is not caused by the musical background based on the lack of difference between feature distributions of the ANOMIC musicians and non-musicians.

Our findings point to a major influence of the annotation tools, instructions, and the musical backgrounds of participants on the annotated patterns. As a next step, the influence of the tools should be studied in more detail using stricter controlled comparisons, including a clarification on how users should include their listening experience into the annotation process, and a controlled use of the automatic pattern matching functionality. Moreover, the analysis of the pattern datasets can be enriched by further investigations as to where annotators tend to agree, for instance, by exploring dynamic thresholds for calculating inter-annotator agreement depending on the size of the patterns. Determining in more detail different levels of granularity as to when two pattern annotations can be considered as agreeing, even if the exact beginning and ending points are not identical, can further help to identify different layers of commonality between annotators.

We believe that the widespread use of digital tools in gathering pattern annotations is inevitable in the near future. Our findings point to several directions for improvements in large-scale data collection and analysis of musical patterns. The observed

differences in annotations gathered with different tools call for further experiments and analyses for deriving technical design choices that fit the purpose of pattern annotations in an optimal way for a given context and annotator group. Establishing reference data for evaluating automatic pattern discovery algorithms from such rich annotation datasets can follow different directions. For instance, identifying subgroups of annotators that highly agree with each other can assist in establishing single-reference data based on a larger group of annotators. Establishing evaluation methods that take into account multiple reference annotations expressing different subjective interpretations of the same musical piece, can pave the way for a more adequate consideration of ambiguity and subjectivity in the evaluation of pattern discovery algorithms.

## 3.6 Concluding remarks

We have discussed several factors to consider when collecting human-annotated musical patterns for algorithms. In this section, we discuss more points of our limitations and future work. We will also touch on topics that are tangentially related to annotation experiments. We have quite a few points in this chapter in comparison to other chapters, as experiments involving humans can be more convoluted.

### **The control of variables: instruction, visualisation, and helper functions**

In this first explorative study of comparing annotation tools, we did not provide a single set of instructions for musical experts (PAF) or non-musicians (ANOMIC). Giving instructions to the annotators is one of the most difficult aspect of pattern discovery experiments to design, because we have a flexible concept of pattern and are trying to find something that we do not know ahead of time. We have not used the working definition given in Chapter 2 because we followed the instruction from previous work in this series of experiments. Additionally, we cannot verify if the annotators strictly adhered to the instructions. In the future, including more specific instructions, incorporating our definition that emphasises the importance of why a pattern is annotated, and adding user behaviour logging may allow for more controlled conditions when comparing different tools.

For instance, an instruction to first listen to music without consulting the visualisation before starting with the annotation process, might decrease an otherwise perhaps strong tendency of users to annotate patterns they can visually identify. Nevertheless, each music visualisation will influence the annotation process to a certain

extent. Musical experts may be most familiar with sheet music, but piano roll visualisations may be more accessible for annotators with less musical expertise. As our results indicate that the size of the musical excerpt that can be displayed in a single view to the user seems to influence the length of patterns annotated, this should be specifically considered when longer patterns are expected to be important for a specific corpus.

This instruction issue is related to the helper functions as well. The influence of the automatic occurrence matching functionality needs to be investigated in more depth in the future. If the goal of the annotation is to find all occurrences of a given pattern, as was the case in the ANOMIC experiment, the automatic occurrence matching functionality can alleviate the finding of the pattern occurrences for users, but it might have the side effect of pointing users to occurrences they would not have deemed important otherwise. If only the most representative pattern should be annotated, as in the PAF experiment, such a tool can assist in highlighting all other occurrences from which the user can then choose the most representative one for the annotation. Either way, this calls for a systematic investigation of using tools with and without such functionality.

The helper function embodies this cycle of improving pattern-finding algorithms and collecting annotations: the algorithm can help with the data collection process, which helps to improve the algorithm itself. In fact, algorithms are increasingly commonly used as pseudo-annotators for data collection and self-supervised learning in *Machine Learning (ML)* (Carmon *et al.*, 2019). Another possibility to explore is to create a tool that enables annotators to write their own helper functions based on their domain knowledge using a simple DSL, which is also an active strand of research (Ratner *et al.*, 2017). DSLs will be discussed more in detail from Chapter 5 onwards.

### Measures of musical background

Fairly assessing and comparing different music backgrounds can be a challenge. For most of us, musical ability exists primarily on a spectrum, and drawing a clear line between musicians and non-musicians is difficult.

In this chapter, we used either the faculty programme of the participants or a basic questionnaire to approximate level of musical expertise for our experiments. We did not collect other detailed music background of the participants, such as what music they usually listen to, whether they have perfect pitch, or whether they studied the pieces in experiments before. In future experiments, more of this information and sophisticated musical expertise indexes could be used (Müllensiefen *et al.*, 2014).

Asking about the participants' familiarity with the music material can be potentially useful in our analysis, as well.

#### **Computational modelling of experts' views**

Our participants are mostly bachelor's and master's students, and none of them are academic experts, such as professors of musicology or music theory. For developing algorithms, two specific types of humans-in-the-loop—crowd-in-the-loop or experts-in-the-loop—have different costs. The higher the expertise, the harder it is to perform annotation experiments with them due to limited availability. Experts' annotations, too, can disagree and be associated with another set of challenges. Nevertheless, a single expert's idea is important, and one may employ a different approach to computationally model an expert.

It is likely that the annotations and characterisation of musical patterns from a single expert would be small and potentially similar to what we have seen in Chapter 2, to the extent that, despite the eloquent characterisations of pattern-related concepts in previous research, they are too imprecise and unsystematic for computational approaches. Big data and machine learning approaches, except zero-, one-, n-/few-shot learning, are likely to fail when the dataset is small, too.

If there exists a well-defined, mathematically, or computationally inclined musical theory of patterns where a set of axioms and rules are explicitly given and fixed, then a different approach than the data-driven ones could be computationally implemented. In this type of top-down modelling, the evaluation would consist of testing and verification of the system in order to guarantee the correctness of the implementation, instead of comparing with annotations. However, what may deliver extra insight is to try to align these top-down theories with listening experiences that involve the bottom-up process of pattern annotation by humans.

#### **Negotiation and consultation**

In our experiments, the participants could not talk to each other and modify each other's annotations according to the discussions. It is possible that, after being in consultation with each other, the annotators may change their mind or even agree to disagree. Future experiments can consider the inclusion of consultation between annotators and the integration of chat or vote functionality within the annotation tools. We believe, however, that there are merits in looking at the first reactions from annotators without modifications.



### **Intra-annotator (dis)agreement**

In this chapter, we did not touch on the topic of intra-annotator (dis)agreement. However, this type of disagreement probably happens often in the case of musical patterns: the same annotator might disagree with their past selves. In fact, we reached out to the annotators for the HEMAN experiment five years after it took place, but we did not receive a reply. It would be interesting to consider a longitudinal study in the future.

### **In comparison with the study of disagreement in chord estimation**

Koops *et al.* (2019) investigated the inter-annotator agreements in chord estimations and was inspirational for this chapter. Here, we make a comparison between the pattern discovery task and chord estimation task in terms of the presence of multiple annotations.

In chord estimation, there is a chord or a group of chords to be assigned to specific points or periods in music. In pattern discovery, the assignments are not local but global. For every point of the music, the annotator can point out whether there is or there is no pattern. When pointing out there is a pattern, this point automatically relates to another or many other points corresponding to the pattern's occurrences in the music. Therefore, although both chord estimation and pattern discovery face inter-annotator agreement issues, the nature of the problems is different.

### **Disagreement and uncertainty**

It has been said in economics that when there is disagreement, there is uncertainty (Bomberger, 1996). Let us take a look back into the deeper implications of agreement and disagreement in terms of certainty and uncertainty.

Why do we need multiple annotators in the first place? Because there is no reason to expect that they will be in perfect agreement, and evidence shows that this is indeed not the case. We have already seen that there can be a variety of reasons as to why annotated patterns differ. Disagreement reflects inter-personal and, in some cases, intra-personal differences that give rise to uncertainties. Uncertainty makes it challenging to define patterns extensionally (see Section 2.2 where we discussed intension and extension definitions).

There are unambiguous situations where well-defined notions guarantee agreement, such as the task of simple counting or calculations, which often do not require more than a very small number of people to perform the task. The regions in be-

tween the certain/uncertain extremes consist of a spectrum of possibilities, where it is hard to tell whether a definitive answer always exists or uncertainties dominate.

In either case, facing uncertainty, we can take the intersection of the annotations, or the aggregate, or sieve through them according to different levels of importance. In the area of musical pattern discovery, since there is a lack of annotated data, the default has been to take the aggregate of data. As suggested in Section 3.5, other strategies should be investigated, too.

#### **Data-centric and model-centric views**

In the machine learning community, there has been a discussion about data-centric and model-centric views. Although the data-model interplay does not naturally bias one or the other, a majority of research puts focus on the models and algorithms, instead of the data they operate on. However, as suggested in (Northcutt, Jiang, *et al.*, 2021), a data-centric model may result in a more efficient and meaningful solution. An example would be an iterative data gathering process where the data get improved and the models are kept fixed.

While we believe that both data and models are essential, we agree that training data has become a key differentiator for the performance of algorithms (Ratner *et al.*, 2017), and that this data should not be kept fixed. In fact, as gathering data is a messy process, major errors are found in widely used datasets (Northcutt, Athalye, *et al.*, 2021). Data should not become the new "Bible" for truth, but it serves as a reference that could be investigated thoroughly and progressively.

# Chapter 4 Comparisons of Musical Pattern Discovery Algorithms

*The best programs are written so that computing machines can perform them quickly and so that human beings can understand them clearly. A programmer is ideally an essayist who works with traditional aesthetic and literary forms as well as mathematical concepts, to communicate the way that an algorithm works and to convince a reader that the results will be correct.*

– Donald E. Knuth

## 4.1 Introduction

We have seen in previous chapters that patterns are ubiquitous and diverse. These patterns, and especially patterns in music, provide unique opportunities and challenges in understanding and using algorithms. Algorithmic musical pattern discovery research aspires to uncover and extract such elements automatically.

Different algorithms can extract patterns in a myriad of ways and yield a plethora of different patterns. It is crucial to compare and understand the differences between these algorithms before we select or improve them for advancing the field of automatic pattern discovery. Comparisons have been carried out using metrics such as F1 score, but **how can we obtain more insight by using comparison methods other than utilising a few numbers from the metrics?** More specifically:

- How can we improve the comparison between the patterns extracted by different algorithms?
- How can we improve the comparison between the algorithmically extracted patterns and the human annotations?

In this chapter, we add to existing comparison techniques by proposing four methods (location and feature visualisation, pattern polling, comparative classification,

and synthetic data) to improve the comparison between musical pattern discovery algorithms and between human annotations and these algorithms. With these methods, we hope to advance the field of automatic pattern discovery more generally.

### Research landscape

Research into musical patterns and pattern discovery algorithms is of current relevance (Collins, 2011; Collins *et al.*, 2013; Conklin, 2002; Conklin & Anagnostopoulou, 2011; Forth, 2012; Hsu *et al.*, 1998; Janssen, 2018; Janssen, de Haas, *et al.*, 2014; Lartillot, 2004; Meredith, 2013, 2015; Meredith *et al.*, 2002; Nieto & Farbood, 2014; Pesek *et al.*, 2017; Ren, 2016; Ren *et al.*, 2017; Ren, Volk, *et al.*, 2018; Rolland, 1999; Velarde *et al.*, 2016; Velarde *et al.*, 2013). With recent and rapid development in research areas such as pattern recognition and machine learning, ML algorithms that are potentially suitable for automatically extracting musical patterns have become available. One motivation driving the development of musical pattern discovery is the potential for applications in areas such as genre classification (using repetitions to distinguish between genres), error correction (using the pattern occurrence happened earlier in the piece to correct subsequent corrupted occurrences) (Dixon *et al.*, 2004; Lin *et al.*, 2004) and segmentation (cutting out the repetitive patterns) (Cambouropoulos, 2006; Conklin & Anagnostopoulou, 2006) in MIR; as well as automatic composition and learning systems in *Human-Computer Interaction (HCI)* (Collins, 2011; Fowler, 1966; Kaplan, 2015). For example, given that composers employ patterns to introduce structure into their music (Hsu *et al.*, 1998), the discovered patterns can be used to provide auto-completion and inspirational suggestions; we can also highlight discovered patterns in music as a guide for attentive listening, memory anchoring, and efficient practising (Jones, 1987; Kubik, 1979); for musicologists and music theorists, facing the complexity of large corpora, the algorithmically discovered pattern candidates can provide support and evidence for categorisation and theorisation efforts (Agawu, 2014; Gjerdingen, 2007; Huron, 2006; Lerdahl & Jackendoff, 1985; Zbikowski, 2002).

### Challenges

In spite of the existence of many generic pattern discovery algorithms (Bertens *et al.*, 2016; Brand, 1999; Cooley *et al.*, 1997; Papadimitriou *et al.*, 2005; Parida, 2007; Vreeken *et al.*, 2011; Wang *et al.*, 1994) and musical pattern discovery algorithms (see (Janssen, de Haas, *et al.*, 2014) for a detailed overview), there nevertheless remain some persistent challenges. In addition to the difficulties we addressed previously about musical patterns themselves, such as the definition problem and the ambiguity and subjectivity, there are two main inter-related and algorithm-specific ones. First, there can

be a large number of output patterns, which are costly to examine manually. Second, implementation logic can be hard to comprehend, which could be caused by any of the numerous procedures that comprise the algorithm, or by a binary-only release for which we only have access to the output.

Related to these two main challenges, we identify the following subsequent challenges:

- Patterns discovered by different algorithms for the same piece differ greatly (Collins, 2017)
- The output patterns are often difficult to relate back to meaningful musical concepts (Forth, 2012)
- The output patterns do not agree well with human annotations (Boot *et al.*, 2016)
- Different application contexts for musical patterns might require different types of patterns (Janssen, 2018)

All of the challenges above contribute to the substantial difficulty of comparing between the algorithms. Musical pattern discovery algorithms are diverse, using methods inspired by geometric shapes, machine learning, and more. These algorithms were previously tested on unassociated datasets with disparate metrics (Janssen, de Haas, *et al.*, 2014). Given a domain-specific dataset with well-defined research questions, it makes sense to do so: valuable insight can be gained using one specific pattern discovery method without comparing to all other methods available (Conklin & Maessen, 2019; Neubarth *et al.*, 2018). However, a large-scale standardised comparison may deliver more insight for selecting and improving the pattern discovery algorithms available.

### State of the art: MIREX and comparing the algorithms

One attempt to standardise the evaluation of algorithms is the [MIREX](#) task we described in Chapter 1. Established in 2014, the [MIREX](#) pattern discovery task provides the [The Johannes Kepler University Patterns Development Database \(JKU-PDD\)](#) dataset and a set of metrics for comparing between pattern discovery algorithms. In the task, a *pattern* is defined as a set of time-pitch pairs that occurs at least twice in a piece of music and the human-annotated [JKU-PDD](#) dataset was introduced (Collins, 2014).

We mentioned in Section 3.3.1 that the MIREX metrics take into account musical events, and metrics such as three-layer precision, recall, and F1 score were devised to evaluate the algorithmic output. Here is a list of all the metrics used in the task:

- Establishment precision, establishment recall, and establishment F1 score

#### 4 Comparisons of Musical Pattern Discovery Algorithms

- Occurrence precision, occurrence recall, and occurrence F1 score
- Three-layer precision, three-layer recall, and three-layer F1 score
- Coverage and compression ratio
- Runtime, fifth return time, etc.
- Standard precision, recall, and F1 score
- Friedman tests

The establishment and occurrence metrics focus on two different points for pattern discovery. Establishment metrics measure whether an algorithm is capable of establishing that a pattern is repeated at least once during a piece, and are less interested in whether the algorithm can retrieve all in/exact occurrences. Occurrence metrics focus on an algorithm's ability to retrieve all occurrences. These two aspects are not completely orthogonal to each other, because occurrences help in establishing the patterns.

Three-layer metrics cross-compare all discovered occurrences with ground truth occurrences on three structural levels (Meredith, 2015). Friedman tests are chosen to investigate whether any algorithms rank consistently higher or lower than the others regarding metrics for individual pieces. Coverage and compression ratio concern with the relationship between the number of notes in the pattern discovery output compared to the set of notes in a musical work. Coverage is the percentage of notes in the musical piece that is covered by discovered patterns. Compression ratio computes how far a musical work can be more efficiently expressed in discovered patterns and their occurrences. Formal definitions are available in (Collins, 2017).

With these MIREX metrics, algorithms rarely performed consistently on all the test pieces. Patterns extracted by different algorithms also vary to a great extent given the same input. There is also significant controversy when it comes to using human annotations as ground truth and when designing an all-encompassing evaluation strategy (Janssen, de Haas, *et al.*, 2014; Meredith, 2015; Ren *et al.*, 2017; Ren *et al.*, 2020), as we explored in Chapter 3: human annotations are diverse and it is difficult to collect them.

Another pattern annotation dataset which has been used for evaluating the algorithms is the [MTC-ANN](#) dataset (Boot *et al.*, 2016; Van Kranenburg *et al.*, 2016). Algorithms' output have been compared in a classification and compression task in (Boot *et al.*, 2016) together with human annotations. Results show that the human-annotated patterns perform better than the patterns extracted by algorithms. It is difficult to derive from these metrics how the observed differences between the algorithms and human annotations can lead to improvements of the algorithms.

Most evaluation metrics rank the algorithms according to a score as the proxy for how meaningful the discovered patterns are, and this is no exception for MIREX. However, these metrics are not entirely without controversy: the performance of algorithms varies on different metrics and different musical pieces, for example, making it difficult to analyse the overall and specific strengths and limitations of algorithms. They summarise algorithms' performances by grouping together the in/correct occurrences, losing the abilities to inspect closely the different types of mistakes the algorithms are making. Problems such as these motivate us to examine the output patterns from the algorithms in more detail to advance this area of research.

### **Our methods**

To dive deeper into these challenges, we introduce visualisation approaches and computational methods for examining algorithmically discovered musical patterns. More specifically, we devise Location and Feature Visualisation methods ([LFV](#)) to visualise the locations and features of patterns. We also create Pattern Polling ([PP](#)), to combine patterns, and Comparative Classification ([CC](#)), to differentiate patterns. In addition, we propose to plant predetermined patterns into random data to generate controllable synthetic data, thereby leaving us better able to inspect, compare, validate, and select the algorithms. We provide a concrete example of using synthetic data for understanding the algorithms and expand our discussion to the potential and limitations of such an approach. Finally, we will finish the main content of the chapter with a discussion over the ground truth problem, summarise the chapter, and discuss topics related to pattern discovery algorithms in general.

Building on this prior research, our intention of comparing the algorithms with human annotations is not to claim any superiority between annotations and computed patterns, and between the algorithms, but to help users to find meaningful patterns. In other words, our intention is not to create metrics to rank the algorithms, but to introduce methods to examine their output, which could potentially inform us more about the algorithms as well as the music data being used. The ultimate purpose for comparing algorithms in reference to human data is to evaluate which algorithms are more suitable and more similar to humans in certain situations or tasks. In this dissertation, we do not produce an answer for this ultimate question, but propose methods for examining this suitability.

### **Contributions**

The take-home messages, namely our contributions, of this chapter are that:

- Visualising the locations and the features of patterns can be used to provide an overview of the discrepancies between human-annotated and algorithmically extracted patterns.
- Taking the output of different pattern discovery algorithms and "polling" together their output do not significantly improve the results of the pattern discovery task.
- Human-annotated patterns and algorithmically extracted patterns differ to such an extent that automatic classifiers can distinguish them. The important features that are responsible for the good classification results are rhythmic features.
- Synthetic data can be used to reduce the complexity of human-composed music so that we can compare more explicitly what humans expect to be extracted as patterns and what algorithms extract as patterns: we show that some algorithms extract patterns that are well-aligned with human expectation, but some algorithms produce unexpected results.

## 4.2 Our methods

### 4.2.1 Summarising the use of the algorithms in our methods

Table 4.1 summarises how we use some of the algorithms we introduced in Chapter 2 in our subsequent experiments, and whether they have been submitted to the MIREX task. The LFV, PP, CC, and Synth methods will be seen in Section 4.2.2, 4.2.3, 4.2.4, and 4.2.5, respectively.

Some algorithms are not present for all experiments because of format compatibility. Not including all algorithms does not undermine the validity of our results because we use the algorithms individually and provide our analysis mainly on a case-by-case basis, that is, we will look at algorithms in a modular fashion. We expect that our analysis and methods could still be applied in the presence or absence of any number of other algorithms.

### 4.2.2 Location and Feature Visualisation (LFV)

To understand the output of musical pattern discovery algorithms, a visual inspection and comparison is a helpful first step. In this section, we address the issue of inspecting musical patterns by comparing discovered patterns visually in two different ways. We first give a short introduction to visualisations in research in general. Fol-



Algorithm	MIREX	Vis.	PP	CC	Synth
VM1	x	x	x	x	x
VM2	x	x	x	x	x
SIAF1 (SIATECCompress - F1)	x	x	x	x	x
SIACR (SIATECCompress - Recall)	x	x	x	x	x
SIACP (SIATECCompress - Precision)	x	x	x	x	x
SC	x	x	x	x	
OL1 ( <a href="#">PatMinr</a> )	x	x	x		
OL2 ( <a href="#">PatMinr</a> )	x	x	x		
ME (Motive Extractor)	x	x	x		
SIACFP	x	x	x		x
MGDP					x
COSIATEC					x
Forth					x

Table 4.1: Table summarising the musical pattern discovery algorithms we examine. The "MIREX" column indicates whether they have been submitted to the MIREX task. "Vis." denotes visualisation. The rest of the three columns indicate whether we consider them in our three other experiments. A cross mark indicates that the algorithm is present.

lowing that, we visualise algorithmically extracted patterns and human-annotated patterns using pattern locations and then pattern features. The challenge here is to visualise a large number of discovered patterns in relation to each other. In answer to this, we abstract away different aspects in different visualisations. In later chapters, we also consider interactive visualisations where the inspector can zoom in and out of different aspects.

### Visualisation in general

"Provare per credere"—seeing is believing—is a concept perhaps not foreign to many. "Visualisations help offload cognition to perception," says Jessica Hullman (Hullman & Diakopoulos, 2011). It has been demonstrated that visualisation can assist humans in debugging complex reasoning steps (Shams *et al.*, 2018). Tukey and Wilk (1966) also stated that the effective laying open of the data to display the unanticipated is a major portion of data analysis. Visualisation is an effective tool for data storytelling, a way of making complex information relatable and different perspectives shareable. Visualisation from data enables us to straightforwardly compare our expectations and the data, which can be used as powerful as a type of pseudo-statistic model fitting and checking. By targeting the human visual cortex, we can encourage discovery with a variety of visual cues.

There are many possibilities to visualise musical patterns, including static figures, score highlighting, interactive applications, animated films, to list a few. Considering the simplicity and the static nature of most academic publications, we start with static visualisations in this chapter.

### Visualising pattern locations

As described in Chapter 2, the beginning and endings of patterns are perhaps the most important to show, because they determine the musical content when the piece of music is fixed and monophonic. Therefore, using these two values, we can concisely encode and represent a musical pattern occurrence within a musical piece. In Figure 4.1, we visualise the (non)existence of patterns by plotting a bar in between the beginning and ending points of the pattern occurrences. In this way, we can concisely show a large number of patterns and their relations to each other in location.

The music data used in this figure is the monophonic version of Chopin’s Mazurka Op.24 No.4 (the piece is made monophonic by using the clipped skyline approach<sup>1</sup>, as created in the JKU-PDD dataset). The algorithms used are summarised in Table 4.1, with a slight change in the naming convention: SIAF1, SIAP, and SIAR, in the figure are SIACF1, SIACP, and SIACR, respectively.

We can make several observations from Figure 4.1:

- Different algorithms find very different patterns—some tend to find shorter patterns, some longer, and some find many patterns, while others are more discerning.
- We have three algorithm families (SIA, VM, and OL), each of which consists of more than one algorithm. Algorithms from the same algorithm family tend to find similar patterns. Similarities observed include the number of patterns discovered, coverage of the song, and occurrence overlaps.
- The ground truth (which we take to be human annotations) is sparse in comparison to the patterns discovered by the algorithms.
- Taking an overview of the entire visualisation, we can see some correspondence and similarities between the algorithms and the ground truth patterns.

This type of visualisation might remind the reader of the orchestral graphs from (Dolan, 2013), as shown in Figure 4.2. Although the appearance is similar—with horizontal bars at different locations on the y-axis and time on the x-axis, what the

---

<sup>1</sup>Pieces in the JKU-PDD dataset are originally polyphonic. They are made to be monophonic by either the clipped skyline approach (taking prominent notes from the polyphonic stream) or the unfolding approach (concatenating different voices). See (Collins, 2017) for more details.

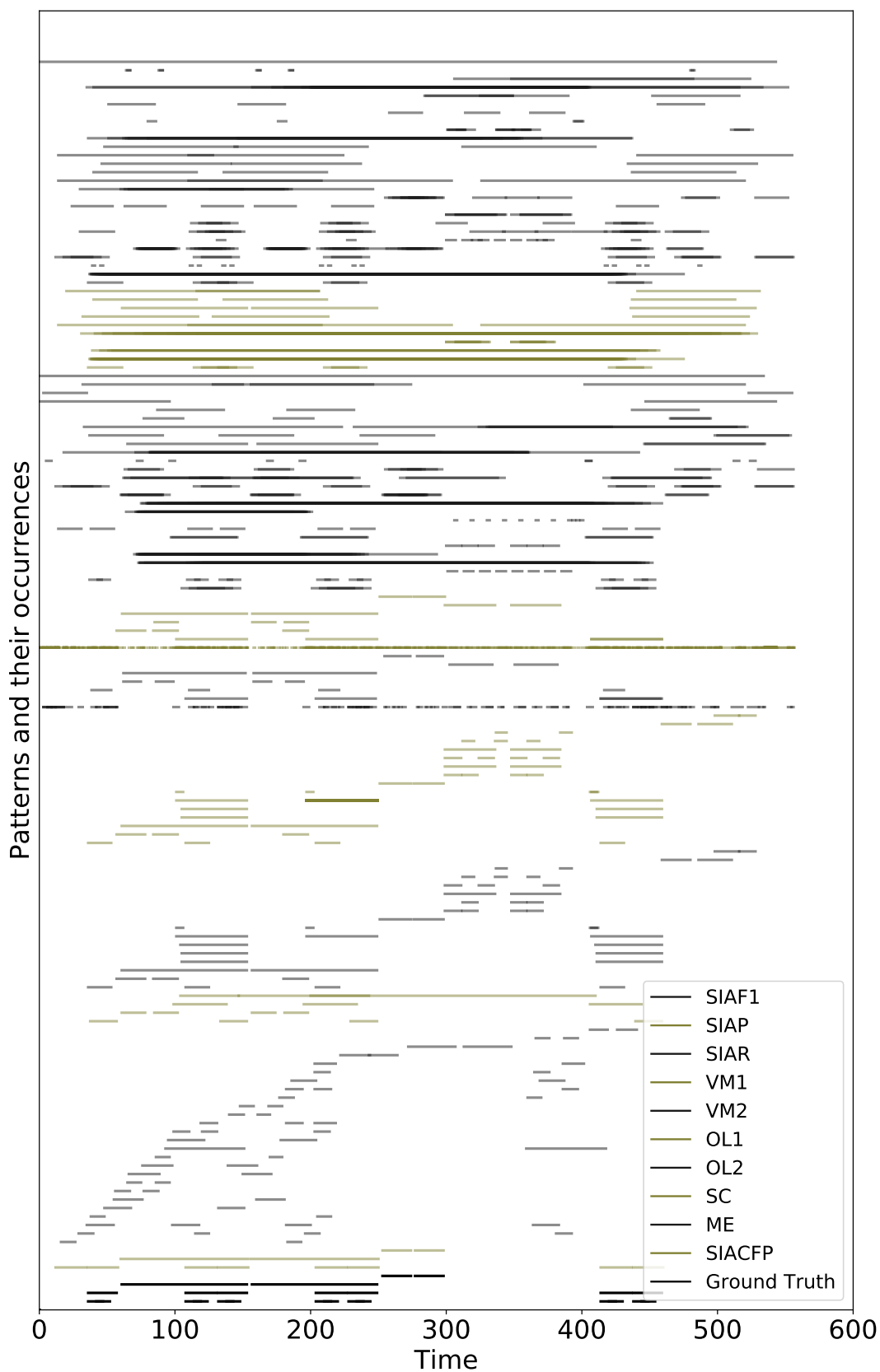


Figure 4.1: Patterns extracted by all algorithms submitted to the [MIREX](#) task 2014-2016 plus [SIACFP](#) on the monophonic Chopin's Mazurka Op.24 No.4. A horizontal bar shows the presence of a pattern. The x-axis represents the time offset in crotchet units. The names of the algorithms in the legend correspond to the order in which the bars were plotted. We can see the algorithms find different numbers of patterns and patterns of different lengths.

bars represent is quite different. In our case, each bar represents the existence of a pattern, and there can be a large number of bars; in the case of orchestral graphs, the bars represent the existence of instruments sounding in the music, and the number of bars does not exceed the number of how many types of musical instruments are present. Our visualisation is developed independently from this work.

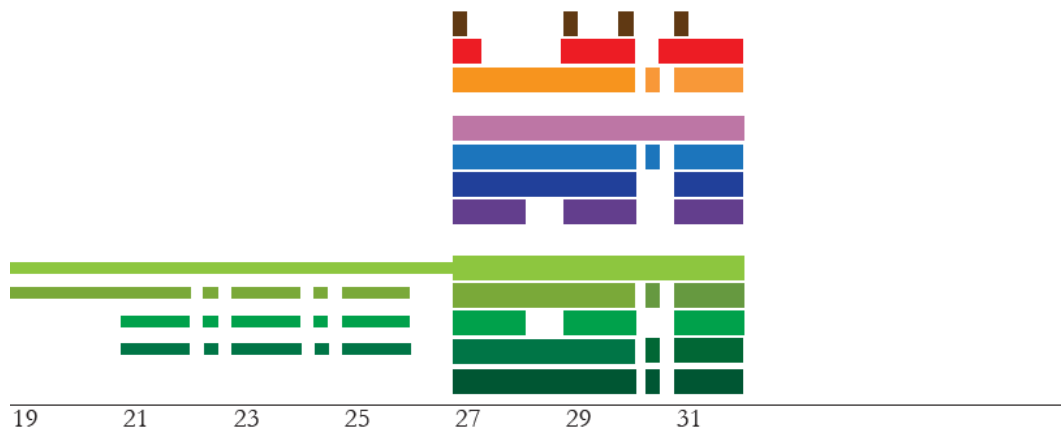


Figure 4.2: An example of orchestral graphs. Taken from Dolan, 2013.

We can also visualise patterns across different pieces. In Figure 4.3, we show the locations of human-annotated patterns in the [MTC-ANN](#) dataset<sup>2</sup> with which we plot 24 out of 26 tune families<sup>3</sup> and numerous human annotations as can be seen in the figure. Each vertical height contains the patterns from a different piece. Although this cross-piece comparison may not work on vastly different pieces of music, the comparison is fitting for the [MTC-ANN](#) dataset. We can observe that, in the same tune family, it is very likely to have patterns in similar locations.

Taking Figure 4.3 further, we attempt to show the basic pitch and rhythm information of the patterns in Figure 4.4. Each dot represents a musical event, in which the rhythm is shown by the spacings of the dots and the pitch differences are reflected in very slight vertical offsets. Related work such as (Meredith, 2016; Nikrang *et al.*, 2014) visualises musical patterns with pitch and rhythm information, too, but they do not visualise multiple tunes at the same time as we do in Figure 4.4. Although we can glean some useful information visually, such as the fact that some inserted notes can be seen in several pattern occurrences, there is a limit to this type of visualisation: the pitch differences are difficult to discern and confusing given the restricted spacing of each piece. Without an interactive zoom-in functionality, this is difficult to improve. In future work, interactive visualisation methods could

<sup>2</sup>More details about the dataset can be found in Appendix A

<sup>3</sup>A tune family is a group of tunes sharing the same ancestor in the process of oral transmission. Pieces in the same tune family share special similarity that can be characterised by musical patterns. See Appendix A or (Boot *et al.*, 2016) for more details.

be desirable where both the macro- and micro- scale of the musical content can be shown and explored. For example, (Nikrang *et al.*, 2014) is a visualisation application for musical patterns that has interactive functionality. We also show an interactive visualisation in a different context in Section 6.6.

### Visualising using *Multidimensional Scaling (MDS)*

*Principal Component Analysis (PCA)* is a well-known feature extraction and *MDS* method for visualisation. It outputs combinations of features that form orthogonal principal components. These principal components are in the same directions as the directions of the largest variances (spread) of the dataset. By examining the resulting principal components, we may gain insight into which features are of more significance in explaining the spread of the data points. By applying the principal components to perform a change of basis on the data points, we obtain a visualisation of the data points where the variances are clear. PCA has been employed and shown to be effective in a variety of MIR tasks, including hook discovery in (Van Balen, 2016).

To systematically investigate the differences between algorithmically extracted and human-annotated patterns, we visualise patterns using their features and *PCA*. We first compute the features with a common feature extraction tool: the *jSymbolic* toolbox in the *jMIR* toolset (McKay, 2010)<sup>4</sup>. The *jSymbolic* toolbox takes *MIDI* files as input and computes 155 musically meaningful features in six categories: texture, rhythm, dynamics, pitch, melody and chords<sup>5</sup>. Using the *MTC-ANN* dataset, we perform a feature selection step and retain 64 features as follows: (1) Eliminating the features which are constant across all patterns; (2) Eliminating the features which are irrelevant to the music content of time and pitch, such as the dynamics features. After this feature selection step, we perform *PCA*.

In Table 4.2.2, we report the prominent features and the weights in the first three *PCA* components: we see that there is a mixture of rhythmic and pitch related features ranked high in the two principal components; rhythmic features rank high in the third principal component. As the interpretation of the *PCA* visualisation does not require an understanding of the individual features and we will perform a more detailed feature analysis later in Section 4.2.4, we will not give a list of explanations of the features here, but some important features and their descriptions can be found in Table 4.6.

<sup>4</sup>We will also use these pattern features in Section 4.2.4 for classification. *PCA* will also be applied as pre-processing step.

<sup>5</sup>There are multiple versions of *jSymbolic*. The newest version provides more features. The version we used to perform the experiments in this chapter is *jSymbolic2.0*.

#### 4 Comparisons of Musical Pattern Discovery Algorithms



Figure 4.3: Visualisation of human-annotated patterns in [MTC-ANN](#). Different colours and segments in the figure represent different tune families. Different vertical heights represent different pieces.

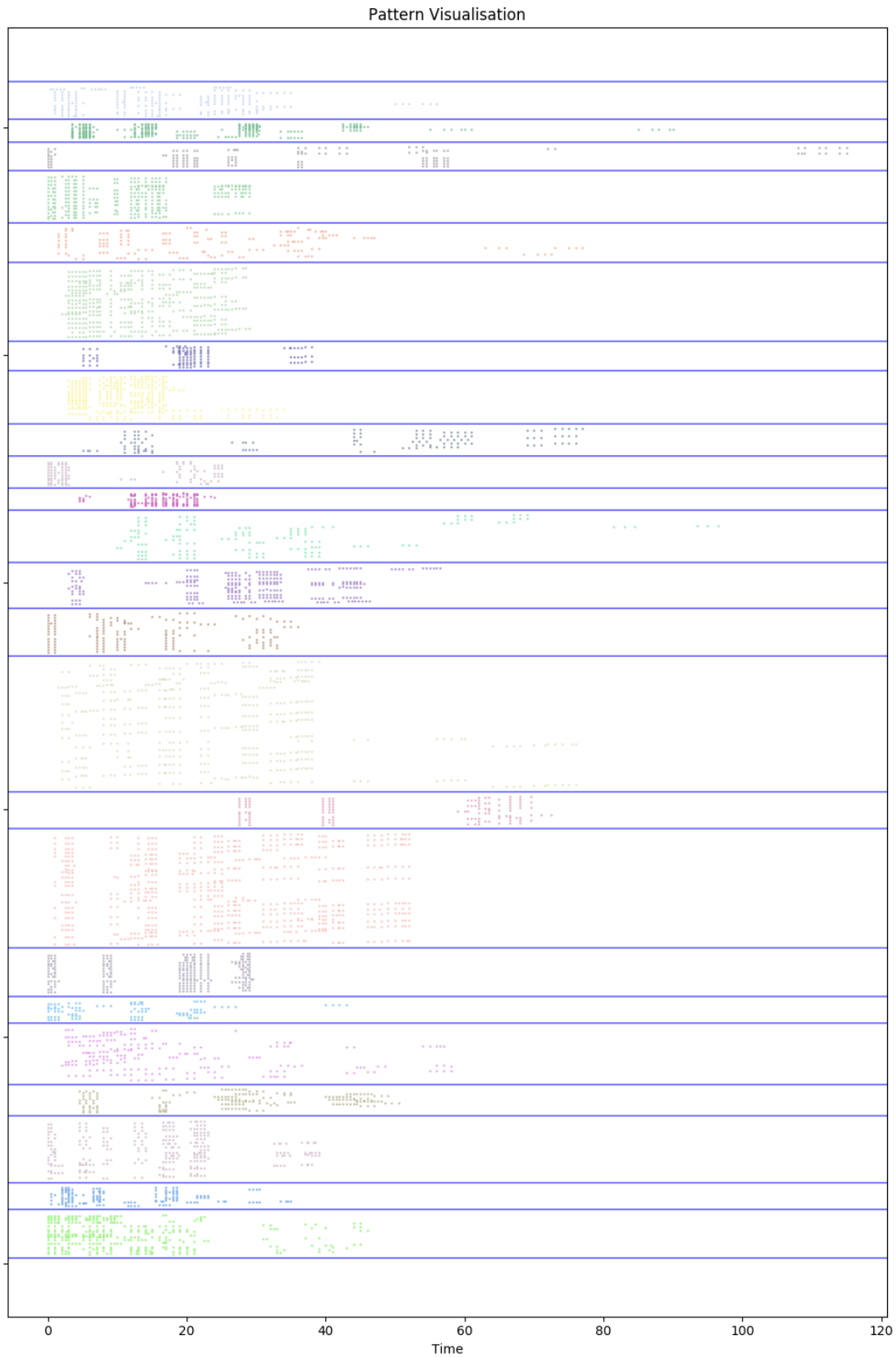


Figure 4.4: An augmented visualisation of Figure 4.3. In addition to location, pitch and rhythm can also be plotted to an extent by plotting musical events using dots. The differences between pitches are reflected in the very slight differences in the vertical positions. Without an interactive zoom-in functionality, we cannot represent every detail precisely. This limitation of static visualisation points to a more interactive visualisation method in future work.

PC (Percentage of variance explained)	Features	Weight (Percentage)
PC1 (22.51)	Number of Strong Rhythmic Pulses	5.18
	Pitch Variety	5.15
	Number of Relatively Strong Rhythmic Pulses	5.07
	Number of Common Pitches	5.07
	Number of Moderate Rhythmic Pulses	5.07
	Other Features	74.46
	Repeated Notes	8.24
	Relative Prevalence of Top Pitches	8.06
PC2 (12.42)	Relative Prevalence of Top Pitch Classes	7.58
	Prevalence of Most Common Pitch	6.32
	Prevalence of Most Common Pitch Class	5.98
	Other Features	63.82
	Combined Strength of Two Strongest Rhythmic Pulses	10.58
PC3 (8)	Polyrhythms	9.98
	Rhythmic Variability	9.27
	Strongest Rhythmic Pulse	7.26
	Strength of Strongest Rhythmic Pulse	7.14
	Other Features	55.77

Table 4.2: The first three principal components of PCA and the weights of features the components consist of. We omit the rest of  $64 - 3 = 61$  components since they do not contribute significantly ( $< 7.5\%$ ) to the variance and, for visualisation purposes, we only use the first two components.



In Figure 4.5, we plot different groups of patterns extracted from [MTC-ANN](#) in a two-dimensional PCA embedding of the feature space. We make four cross-group comparisons to show typical cases of how musical patterns are distributed in the feature space spanned by the first two components of the PCA decomposition. The visualisation is generated by first computing the PCA embedding using the annotated patterns. Then, pattern features from different algorithms<sup>6</sup> are projected onto this embedding. Finally, for baseline comparison, we extract random excerpts from the music, compute their features and add them to the PCA space. The comparison to the random baseline is of interest here because, otherwise, it would be more difficult to navigate the highly abstract PCA feature space. More specifically, the random excerpts are sampled using the following procedure:

- For each annotated pattern in MTC-ANN, we find the corresponding song where the annotation appears.
- We then pick a random starting point and take an excerpt of the same length as the pattern to construct a candidate excerpt.
- Finally, we repeat the sampling procedure five times to control for anomalous results.

From the four snapshots we take from the musical pattern PCA feature space as shown in Figure 4.5, we make several observations:

- Annotated patterns and random excerpts have extensive areas of overlap, which makes it impossible to find a linear classifier that uses the first two principal components of the annotated pattern features, which in turn makes differentiating the two groups of patterns nontrivial, as shown in the upper left subfigure.
- SIAR patterns exhibit a very different distribution from the annotated patterns and random excerpts as shown in the top right and left subfigures. Notice the annotated patterns concentrate at the top left corner in the top right subfigure. In this case, it is relatively easy to separate the long-tail area of the extracted patterns from the annotation area. By applying this observation and designing a filtering process, we could substantially improve the performance of the SIAR algorithm on MTC-ANN.
- The overlap between the annotated patterns and extracted patterns is small in the bottom left subfigure. A linear classifier can be devised to roughly separate the two groups of data using the first two principal dimensions of the annotated

---

<sup>6</sup>See Table 4.1 for a list of the algorithms with the changes of naming in the SIA family as before and VM1 to VM.

patterns. The extracted patterns of the SC algorithm have different features to the annotated patterns.

- In the bottom right subfigure, we show all the heterogeneous patterns extracted by algorithms, annotated by humans, and randomly sampled in the same PCA embedding. We can see that patterns extracted by algorithms of the same family, namely SIACP, SIACR, SIACF, and SIACFP tend to share the same long-tail property, and therefore their performance on MTC-ANN can be improved by an extra filtering step as described above.

### Results and insights into LFV

We introduce two methods of visualisation in this section. The first method, namely visualising pattern locations, gives us an overview of the large number of discovered patterns and their relations in time. The second method, namely visualising pattern features using PCA, helps us analyse different aspects of musical patterns. In combination, we refer them as LFV. More concretely, we see that the algorithmic output has discrepancies with human annotations in terms of their musical features and locations of occurrence.

After examining these two types of visualisations, we can take more concrete steps to further explore the pattern data. First, the observations made in visualising patterns (Figure 4.1) hint at the possibility of combining similar patterns from different algorithms to devise a method to combine their wisdom of these algorithms and thereby extract musical patterns based on consensus. Second, from the PCA visualisation, we must answer the question of whether patterns from different groups can be discriminated based on their features. Correspondingly, in Section 4.2.3 we will try to fuse the patterns from different algorithms, and in Section 4.2.4, we will try to classify between them.

#### 4.2.3 Pattern Polling (PP)

Ensemble methods, in which several algorithms are combined to achieve better performance, are becoming more common in various applications (Zhou, 2019). Data fusion has also gained popularity recently, where data preprocessing is performed on different modalities, then concatenated to a new representation.

In addition to the trends and success stories, why should algorithms and data become combined in the first place? In (Mitchell, 2012), it was argued that data fusion could improve system performance in four ways: representation, certainty, accuracy, and completeness. The no-free-lunch theorem (Stork *et al.*, 2001), which states that

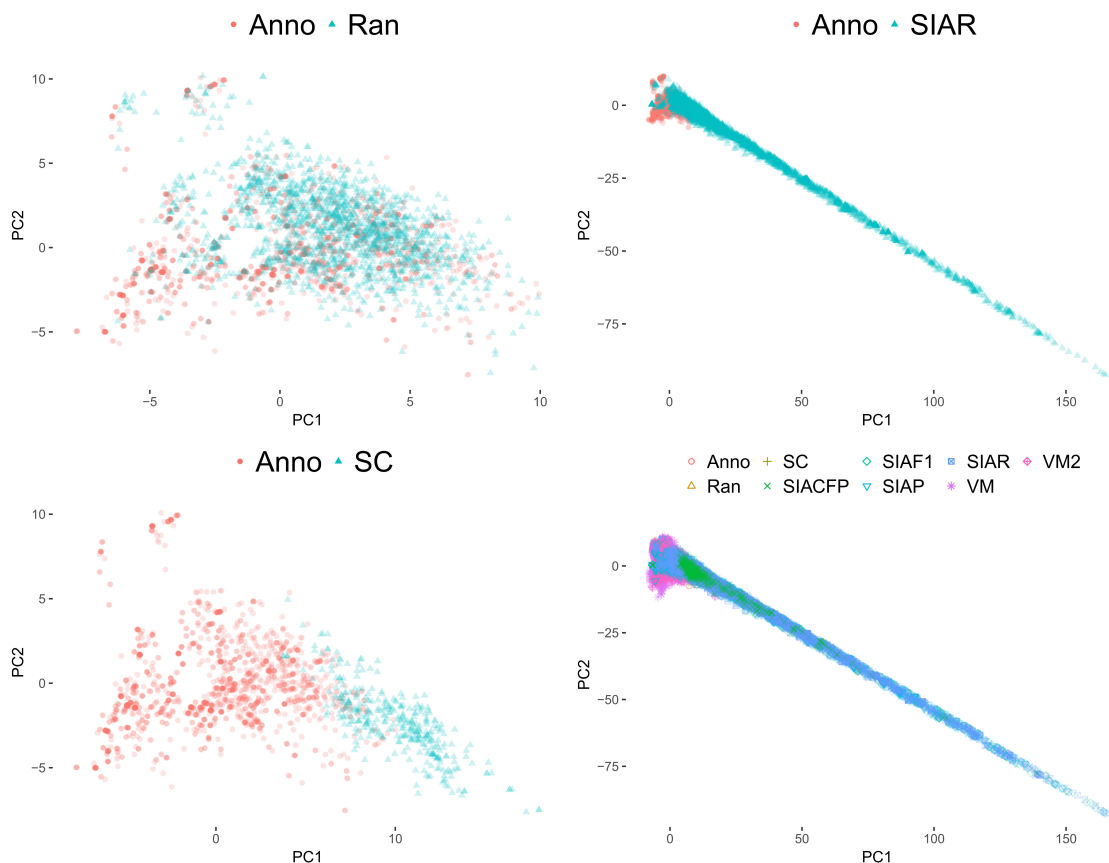


Figure 4.5: Visualisation of different groups of patterns using the space spanned by the first two principal components of the annotated pattern features in *MTC-ANN*. The legend above each subfigure denotes the correspondence between colours and algorithmic patterns/annotated patterns/random excerpts. Notice that the figures in the left column show zoomed subregions of figures in the right column. (1) Upper left: random excerpts and annotated patterns. The overlap between the two groups is large, and it is nontrivial to separate them in this two-dimensional PCA embedding. (2) Upper right: *SIACR* patterns and the annotated patterns. *SIACR* patterns exhibit a long tail which is not shared by the annotated patterns. (3) Bottom left: *SYMCHM* patterns and the annotated patterns. The overlap of the data points is small, which makes it easier to separate the two groups in this embedding. (4) Bottom right: random excerpts, annotated patterns, and patterns from all algorithms. We can see that some of the algorithmically extracted patterns are very different from the annotated patterns, and the algorithms belonging to the same family exhibit similar long tails.

any two optimisation algorithms are equivalent when their performance is averaged across all possible problems, also provides a valid reason to do so.

Integrating different algorithms using data fusion has also been shown to be a successful approach to improving overall performance in dealing with ambiguous musical data, such as automatic chord estimation (Koops *et al.*, 2016). For musical pattern discovery, according to some similar pattern locations we saw in Section 4.2.2, there is hope of finding a consensus between various algorithms to achieve an

overall better pattern discovery result. To this end, we devise Pattern Polling (PP), which takes the locations of discovered patterns from different algorithms as input, then outputs new locations of patterns based on this. We name the algorithm with "polling" because it involves measuring the level of consensus from an ensemble of algorithms.

We developed the PP algorithm based on the assumption that all pattern discovery algorithms aim to find passages containing a shared overall interest—the "patterns" in musical compositions. We view the output of each algorithm to be votes on whether a given time point participates in this pattern-salient part of the composition, i.e. is part of a musical pattern. We consider whether or not an algorithm recognises a pattern at any given time point to count as a "vote" on whether or not there is indeed a pattern present at that time offset. For the sake of convenience, we define the salience degree of a time point as the number of discovered patterns at this time offset. In essence, PP is a process in which each algorithm contributes to the salience degree of a time point based on their discovered patterns. The resulting polling curve is then taken as a base to detect new patterns with input from all algorithms.

### Polling Curve

From the voting process described above, an example polling curve using several algorithms from the MIREX task is shown in Figure 4.6 (a). To elaborate on how we calculate the polling curve, we start with discretised time points  $T = [0, 1, \dots, n]$  in the musical piece, with a resolution of one crotchet. If an algorithm finds a pattern occurrence at a given time point, we count that as one vote contributing to the pattern salience score at that time offset. We perform the same procedure for all pattern occurrences and sum the votes from all algorithms. In the end, we obtain the polling curve  $P(t)$ , which is a time series of voting counts at each time offset  $t \in T$ .

Mathematically, we have:

$$P(t) = \sum_A \sum_P \sum_O I_O^{A,P}(t) \quad (4.1)$$

where  $A$  is the set of algorithms,  $P$  is the set of patterns,  $O$  is the set of occurrences, and  $I_O^{a,p}(t)$  is the weighted indicator function of an occurrence of a pattern  $p \in P$  in the output of an algorithm  $a \in A$ :

$$I_O^{a,p}(t) = \begin{cases} \omega_a & t \in o \subseteq p \subseteq a \\ 0 & t \notin o \subseteq p \subseteq a \end{cases} \quad (4.2)$$

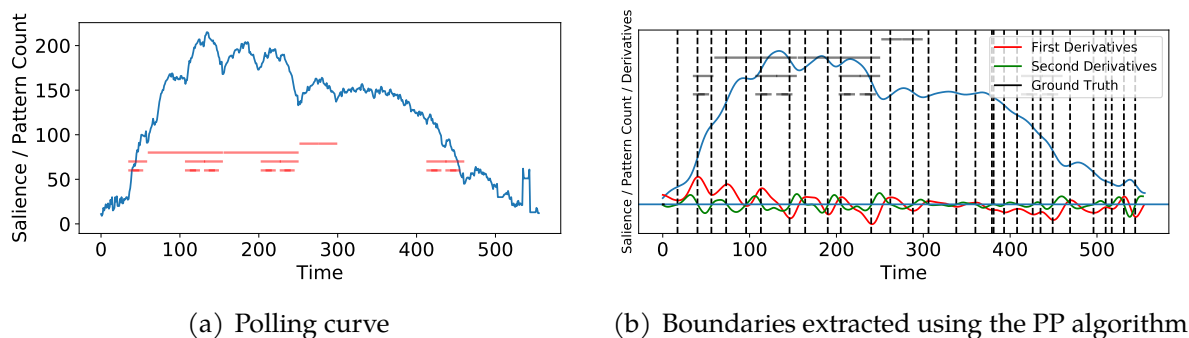


Figure 4.6: a. The polling curve of Chopin’s Mazurka Op. 24 No. 4 using algorithms from the MIREX task. The horizontal bars show where the human-annotated patterns are present. The x-axis represents the time offset in crotchet units and the y-axis represents the saliency value. We see promising correspondences between the polling curve and human annotations. b. Extracted pattern boundaries using PP indicated with dashed vertical lines. Many dashed lines are aligned with the boundaries of human annotations. We also plotted the polling curve, the ground truth human annotations, as well as both the first and second derivatives of the polling curve for reference.

where  $\omega_a$  is the weight assigned to an algorithm  $a$ . Different values of  $\omega_a$  may allow for the algorithms to be taken into account with different weights. In this dissertation, we do not explore the possibility to weight the algorithms differently, and  $\omega_a$  is always assigned to 1.

### What polling curves represent and how to extract patterns from the curve

The polling curve uses the output from all individual pattern discovery algorithms. Given a music piece and a set of algorithms, the curve provides an estimate of how likely a region in music is to contain a pattern by taking into account all algorithmic output. The curve can be used for estimating where a pattern is more likely to be present in the musical piece. In Figure 4.6, we see a promising level of agreement between this estimated likelihood and human-annotated patterns.

To extract concrete patterns, we need to identify the beginnings and endings of patterns from the curve. The critical points of the curve can be helpful in this respect. Mathematically defined as the points at which the derivatives of the curve is equal to zero, critical points indicate the prominent changes in the shape of curves. Although the polling curve is a discretised curve, we can compute the discrete derivatives and corresponding critical points. These prominent shape changes then can be regarded as pattern boundaries. To distinguish between the prominent changes and those too small to be relevant, we first perform a smoothing step on the polling curve.

### Smoothing

In our algorithm, we use the Savitzky-Golay filter (Schafer, 2011) for smoothing, which is a linear least-squares polynomial fitting filter. When we apply smoothing, we reduce the impact of small irrelevant changes in the curve at the cost of potentially losing valuable details. With different degrees of smoothing, we capture different levels of detail in the polling curve. With this in mind, we make our PP algorithm parametric on the degree of smoothness  $s$ .

### Derivative

After smoothing, to find the prominent changes in the curve, we calculate the first and second discrete derivatives and take the critical points. More formally: let  $P'(t) = P(t + 1) - P(t)$  and let  $P''(t) = P'(t + 1) - P'(t)$ ,  $t > 0, t \in T$ . We are interested in zero-crossing points  $\bar{t}$  in  $P'(t)$  and  $P''(t)$  because each zero-crossing point  $\bar{t}$  represents a change of direction in the polling curve. For example, when  $P'(t) < 0$  and  $P'(t + 1) > 0$ , we have a dipping point  $P'(\bar{t}) = 0$  on the curve. More patterns are discovered by the algorithms starting from this point, so it is likely to be the start of a pattern.

One question remains as to how strong the dipping, tipping, concave, and convex points in the curve should be if we are to pick them as boundaries. Here, we introduce a second parameter of the PP algorithm: a threshold on the steepness of the zero-crossing points  $\lambda$ . With different values of  $\lambda$ , we create a set of boundaries that consists of time offsets at which zero-crossings occur.

In Figure 4.6, we show an example of the extracted boundaries. We notice that some boundaries line up well with human-annotated pattern boundaries.

### Evaluation metrics

To evaluate the accuracy of the extracted pattern boundaries, we choose to use the ground truth of human-annotated patterns. We compare the computed boundaries with the beginnings and endings of patterns marked by humans with standard evaluation metrics of precision, recall, and F1 score. Following these standard evaluation metrics, the metrics we also used in Chapter 3, we calculate the [precision](#), [recall](#), and [F<sub>1</sub> score](#) with a degree of fuzziness, i.e. we attempt to match the boundaries with a tolerance of one crotchet note length as this is the degree of discretisation we used for creating the polling curve.

Algorithm	Precision	Recall	F1
ME	(0.125, 0.086)	(0.184, 0.077)	(0.149, 0.083)
SC	(0.396, 0.022)	(0.419, 0.068)	(0.402, 0.046)
OL1	(0.420, 0.038)	(0.565, 0.044)	(0.462, 0.023)
OL2	(0.422, 0.061)	(0.565, 0.044)	(0.483, 0.054)
SIAPF1	(0.139, 0.049)	(0.670, 0.005)	(0.228, 0.041)
SIAR	(0.213, 0.039)	(0.427, 0.000)	(0.279, 0.021)
SIAP	(0.117, 0.043)	(0.596, 0.008)	(0.195, 0.037)
VM1	(0.137, 0.035)	<b>(1.0, 0.0)</b>	(0.240, 0.029)
VM2	(0.206, 0.073)	(0.543, 0.024)	(0.296, 0.060)
SIACFP	<b>(0.819, 0.030)</b>	(0.82, 0.064)	<b>(0.815, 0.046)</b>
PP-P	<b>0.478</b>	0.206	0.249
PP-R	0.228	<b>0.867</b>	0.35
PP-F1	0.248	0.738	<b>0.360</b>

Table 4.3: A table of the (mean, variance) of the precision, recall, and  $F_1$  scores of the pattern boundaries of different algorithms. The PP-P, PP-R and PP-F1 are obtained using a 3-fold cross-validation training process optimising precision, recall, and  $F_1$  score. Because we only have one piece in the test set, there is no variance value. The best results from individual algorithms and PP are shown in bold.

## Results and insights into PP

Using the setup above, we extract patterns using the PP algorithm in a subset of JKU-PDD<sup>2</sup>. The original JKU-PDD dataset contains five pieces, and we take three pieces in the monophonic format<sup>7</sup>: Chopin’s Mazurka Op.24 No.4; Mozart’s Piano Sonata K.282, 2nd movement; and Beethoven’s Piano Sonata Op.2 No.1, 3rd movement. The other two pieces in the JKU-PDD dataset contain a concatenation of voices from the polyphonic version, which violates the assumptions of geometric algorithms—these algorithms cannot treat polyphonic music as a concatenation of voices—and are therefore excluded.

The results are shown in Table 4.3. By calculating the mean values of all algorithms, we can see that the best  $F_1$  score of PP 0.360 is slightly better than the mean of the  $F_1$  scores of individual algorithms 0.3549. When we look at the individual algorithms, the best  $F_1$  score of PP ranks fifth out of ten. The SIACFP algorithm performs the best overall. Although we also observe that PP performs slightly better than the average of the individual algorithms, we cannot yet conclude that this fusion method improves the accuracy of pattern discovery significantly.

<sup>7</sup>How polyphonic pieces are made monophonic is described in <sup>1</sup>. Here, we take the three pieces that are created using the clipped skyline approach and discard the pieces created using the unfolding approach.

### Why is there no significant improvement?

From the results, we identify some potential reasons as to why PP does not outperform individual algorithms. Firstly, the available dataset is small, and the human-annotated patterns are sparse, which is problematic for training the parameters in PP. Secondly, the algorithms disagree with each other on pattern length, pattern overlap, and the number of patterns, which may be caused by a variety of different factors, including the inherent ambiguity of music and pattern perception, the lack of a unified goal, the different target applications of the musical pattern discovery algorithms, or a combination of all of these factors. Ultimately, although we observed a promising level of consensus between algorithms in Figure 4.1 and Figure 4.6, Table 4.3 reveals that this is not yet sufficient to extract patterns that substantially agree with the human-annotated patterns.

### From location to content, from combination to discrimination

So far, in this PP experiment, we have looked at the locations of patterns yielded by different pattern extraction algorithms and explored some of the possible ways in which they might be combined. To further investigate the musical events in these patterns, we move on to the pattern features and use them to distinguish between computed patterns and human annotations.

#### 4.2.4 Comparative Classification (CC)

In the previous section, we compared algorithms using the locations of patterns. In Section 4.2.2, with PCA, we have extracted features from musical patterns and visualised them in the feature space using MTC-ANN<sup>2</sup>. In this section, using the same feature data, we perform Comparative Classification (CC) between the features of algorithmically extracted patterns, human annotations, and random excerpts, to see whether SotA classifiers can separate the patterns identified by different agents (algorithms, humans, and randomness) consistently.

We perform CC on two subtasks for two different levels of comparisons. The first classification task is to classify patterns into three groups based on how they were extracted: algorithmically, manually, or randomly. In the second task, we perform a finer level of classification on the algorithmic group from the first task by classifying patterns based on the algorithms that extracted them. In both tasks, we expect classifiers to help us discriminate between groups of patterns based on their musical features. We then explain the classification results using feature importance analysis.



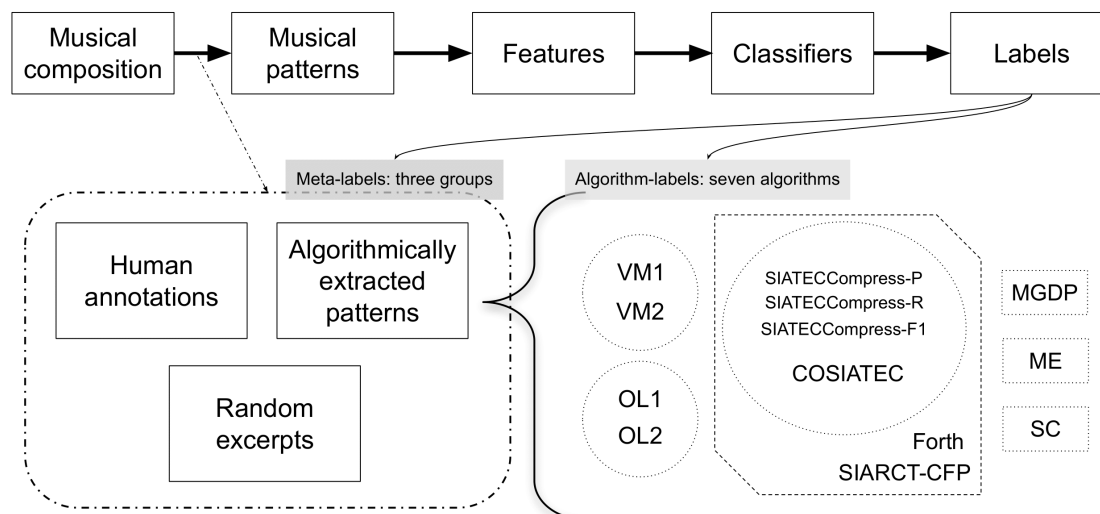


Figure 4.7: Pipeline of CC (the chain on top) and the two classification tasks using two types of labels. The meta-label classification experiment tries to use classifiers to distinguish between three groups of patterns from different origins. The algorithm-label classification experiment is performed at a finer level: we look into the algorithmically extracted patterns specifically and distinguish between seven subgroups of patterns. The seven algorithms we consider are summarised in Table 4.1. The geometric methods are in the dashed line square with cut corners. Other algorithms are also listed to be considered in the next section.

Supervised classification methods have been used extensively in MIR tasks such as genre classification and the classification of corpora from different geographic origins. Leveraging the discriminative power from classification algorithms for distinguishing the output from other algorithms is related to adversarial machine learning, which is gaining popularity (Da Fontoura Costa & Cesar Jr, 2009; Patel & Barkovich, 2002). To the best of our knowledge, the pipeline we propose in Figure 4.7 has not yet been used to compare symbolic musical pattern discovery algorithms.

Note that extracting features from patterns themselves is equivalent to analysing the patterns out-of-context. Analysing musical passages in isolation from their context might be limited in generalisability because a pattern in one musical piece might be insignificant in another piece depending on what surrounds it. Nevertheless, certain arrangements of musical events and their features could be important in signifying the differences between algorithmically extracted patterns and human annotations.

## Classifiers

To prevent the results from being classifier-specific, we use a mixture of simple and more sophisticated, linear and non-linear classifiers to perform the classification

tasks. We also use standard machine learning techniques to train and test classifiers: scaling and centring preprocessing steps are first performed on all the features. Furthermore, we create another set of features by applying PCA to the jSymbolic features to see whether the PCA features will perform better than the original ones. Additionally, to avoid overfitting, for all experiments, we use a 10-fold *cross-validation* 3-times repetition scheme. The parameter search for each classifier are performed separately on each fold. The six statistical classifiers we use are as follows:

- **Gradient Boosting Machine (GBM)** (Friedman, 2001) produces a prediction model consisting of an ensemble of decision trees. The parameters we search through are the learning rate, complexity of trees, minimum number of samples to commence splitting, and the number of iterations.
- **Linear Vector Quantisation (LVQ)** (Kohonen, 1990) applies a winner-takes-all Hebbian learning-based approach. We search through two parameters in this classifier: codebook size and number of prototypes.
- **Linear Discriminant Analysis (LDA)** (Ripley, 2007) produces a linear classifier which finds a linear combination of features that best separates different classes in datasets. This classifier is not parametric.
- **Naive Bayes (NB)** (Ng & Jordan, 2002) computes the conditional a-posterior probabilities of a categorical class variable given independent predictor variables using the Bayes rule. Three parameters are tuned for this classifier: Laplace smoothing, kernel bandwidth, and distribution type.
- **Random Forest (RF)** (Breiman, 2001) operates by constructing a multitude of decision trees. The parameter we consider is the number of variables per level.
- **Support Vector Machine (SVM)** (Suykens & Vandewalle, 1999) calculates a map from data to a new representation so that the data points of the different categories are separated by as large a gap as possible. We use the radial basis function kernel and consider two parameters: smoothing factor and weight of training examples.

### Results and Insights into CC

We mainly use accuracy and the variance of accuracy as our measures of the performance of the classifiers. To further interpret the results of the classification task, we compute confusion matrices and feature importance measures.

#### Model metrics

Figure 4.8 shows the accuracy and variance of different classifiers in the two classification tasks. We use two groups of features, the raw features and features after

the PCA step. The baseline accuracy is  $\frac{1}{\#\text{group}} = \frac{1}{3} \approx 33\%$  for the first task and  $\frac{1}{\#\text{group}} = \frac{1}{7} \approx 14\%$  for the second. We balanced the number of patterns in each group to make them the same, for the first task = 1657, and for the second task = 355.

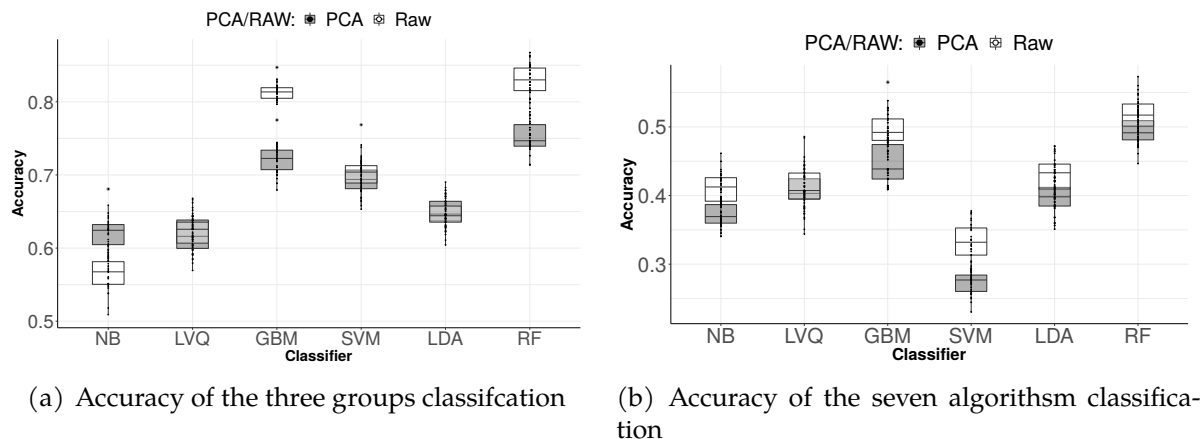


Figure 4.8: Accuracy values for classifiers in thirty experiments (10-fold cross-validation repeated three times) using six classifiers with jSymbolic features and after PCA preprocessing.

We see that all classifiers give a result higher than the baseline accuracy in both tasks. PCA improves the performance of the classifier NB in the first task. For LVQ, SVM, and LDA, using PCA or raw input does not make a significant difference. The performance of other classifiers is worse when using the PCA input. The finding that PCA has different influences on the performance of classifiers may be due to the fact that each classifier uses different internal feature transformation mechanisms. Overall, the random forest classifier gives the best results for both tasks with the raw feature input and the parameter `#variables = 32`.

The fact that classifiers can differentiate between groups of patterns with above-baseline accuracy values has a few implications. In the first task, it implies that algorithmically extracted patterns possess different properties than human-annotated patterns, which suggests that extra consideration of pattern features would be beneficial when trying to discover patterns automatically. It also shows that the algorithmically extracted patterns have different traits than random excerpts, which means that these patterns are not equivalent to randomly sampled excerpts, and are therefore potentially more useful for some applications. Lastly, it shows a difference between human-annotated patterns and randomness despite subjectivity being involved in the annotation process, which is consistent with the carefully designed annotation collection process (Van Kranenburg *et al.*, 2016) and previous findings that the annotations are useful for classifying tune families (Boot *et al.*, 2016).

In the second task, the above-baseline accuracy shows distinguishability in the patterns extracted by the algorithms, which suggests disagreement between different algorithms, reinforcing our conclusions from the first experiment. To further investigate differences between groups, we now analyse the confusion matrices.

### Confusion matrices

Table 4.4 and Table 4.5 give the confusion matrix results calculated from the classifier with the best classification results: Random Forest. We perform the repeated cross-validation experiment ten times and take the mean and variance of the resulting ten confusion matrices. The results show us that different groups of patterns are separable and dissimilar to one another according to the random forest classifier.

To read the table, we first notice that the sum of each column is roughly 500, which is the size of our test data. The row sums do not have this constraint because there are no restrictions on the group sizes as determined by the classifier. For interpreting the entries in the table, we take as an example the number 29.5 in the top right corner of the table. This number is the mean count of patterns classified as algorithmically extracted patterns but are actually human annotations. Table 4.5 is formatted similarly, with a different column sum because of the different test data size, and with algorithm names instead of group names.

In the first task, we see that the classifier can differentiate between the three groups with few instances of incorrect classification. This classification result is not the most desirable for the pattern discovery algorithms. If we had seen that the classifiers could not differentiate between the algorithmically extracted pattern group and the human-annotated pattern group, this result would suggest a level of consensus between algorithms and humans. In other words, a larger count at the last line of the first column in the confusion matrix would indicate that the patterns discovered by algorithms are harder to distinguish from the human annotations, which is more desirable if the algorithms would like to imitate the pattern discovery behaviours of human annotators. With the current results, however, we come to the conclusion that the algorithmically extracted patterns, annotated patterns, and random excerpts possess their own traits and are not similar enough for the classifiers to fail. This is in accordance with the first experiment and previous research, which shows that the algorithmically extracted patterns are not yet indistinguishable from the human annotations (Boot *et al.*, 2016; Ren *et al.*, 2017). Despite this, we at least establish that neither annotated patterns nor extracted patterns are equivalent to random data.

Original → Classified ↓	Alg	Ran	Anno
Alg	406 ( $\pm 13.5$ )	32.3 ( $\pm 7.2$ )	29.5 ( $\pm 8.3$ )
Ran	54.1 ( $\pm 8.9$ )	402 ( $\pm 15.9$ )	65 ( $\pm 14.1$ )
Anno	37 ( $\pm 8.4$ )	62.8 ( $\pm 11.3$ )	402.6 ( $\pm 14.2$ )

Table 4.4: Confusion matrix results from the ternary classification experiment using the Random Forest classifier: mean and variance (in parenthesis) of ten classification experiments. The row names indicate that the patterns are classified into the group with this name by the classifier; the column names indicate the patterns are originally from the group with this name. Three groups of data are classified with high accuracy and significant p-values  $\ll 0.05$ .

In the second task, the classifier can also differentiate patterns from different algorithms to a certain extent. The goal of this task is to examine the individual pattern discovery algorithms. Similarly to the analysis in the first subtask, distinguishing between different algorithms perfectly would indicate that the discovered patterns are very different, despite the algorithms having the same objective of "pattern discovery"; in addition, if the algorithms are in agreement with each other, we would expect the classifier to fail and the values in the confusion matrix to be more uniformly distributed. We see in Table 4.5, however, this is often not the case. The exceptions here are the patterns extracted by algorithms from the same family (SIACF, SIACP, and SIACR, for example): the classifier performs worse in distinguishing within the same family. This is an indication that we are not overfitting the classifier. More importantly, this classification result agrees with the first data fusion experiment in that the output from algorithms has a high degree of disagreement, especially when the algorithms come from different families.

### Feature importance

Figure 4.9 and 4.10 show the individual importance value of each feature used in the random forest algorithm (the best classifier in the two classification tasks), computed by using the Boruta algorithm (Kursa, Rudnicki, *et al.*, 2010). The Boruta algorithm randomly duplicates and shuffles the values in the original features as extracted by jSymbolic. The algorithm then combines some of these randomised features with the originals in order to calculate and compare the significance of different combinations of features. At the end of this process, we obtain an importance value for each feature—the Gini impurity importance value (Louppe *et al.*, 2013).

Here, in Table 4.6, we summarise the top 10 important features in the classification for Figure 4.10 (the top part of the table) and Figure 4.9 (the bottom part of the table). The descriptions are taken from the most relevant part of the developers'

Original → Classified ↓	SIACF1	SIACP	SIACR	VM2	VM1	SC	CFP
SIACF1	22.6(±9.7)	18.6(±9.9)	23.6(±12.8)	2.5(±2.3)	0.4(±0.8)	1.4(±1.5)	8.4(±4.7)
SIACP	12.8(±6.8)	34.4(±6.6)	15(±5.9)	0.5(±0.9)	1.3(±1.8)	1(±1.9)	5.1(±2.5)
SIACR	30.6(±12.1)	23(±8.7)	31.8(±9)	1.8(±2.1)	1(±1.3)	1.3(±1.9)	10.6(±5.6)
VM2	8.2(±5.1)	11.1(±4.2)	9(±4.4)	63.1(±7.2)	21.6(±7.8)	0.3(±0.6)	3.8(±2.5)
VM1	0.9(±1.4)	5.5(±2.9)	1.9(±2.2)	22.5(±6.9)	78.5(±8.8)	0(±0)	0.3(±0.9)
SC	16.2(±6.5)	6.8(±4.2)	10.3(±4.8)	7.3(±3.9)	0.2(±0.6)	93.8(±4.9)	17.1(±4.1)
CFP	15.2(±5.9)	7.2(±4)	14.9(±5.6)	8.7(±4.6)	3.6(±3.1)	8.8(±4.5)	61.1(±9.2)

Table 4.5: Confusion matrix results classifying the algorithmic output using the Random Forest classifier. The columns and rows follow the same format as in Table 4.4.

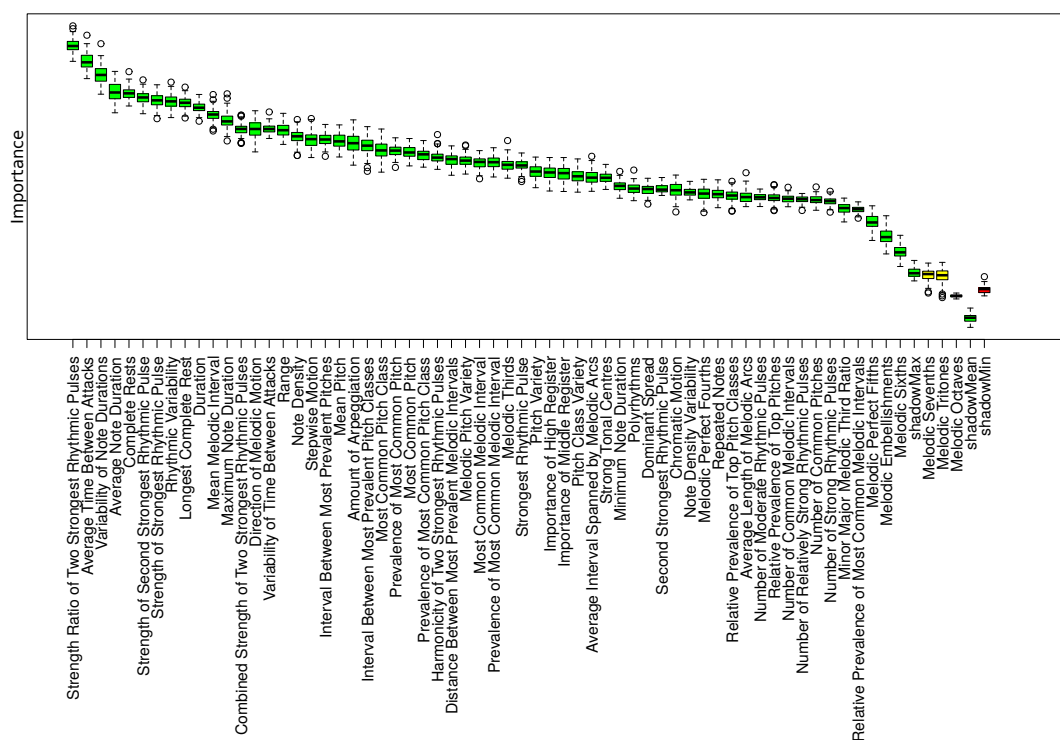


Figure 4.9: Feature importance for differentiating between annotation, algorithmic output, and random excerpts. The boxplot depicts the mean and variance (interquartile ranges) of the feature importance values (Kursa, Rudnicki, *et al.*, 2010). The features are ranked by their importance. We omit the y-axis label because the absolute importance values are not relevant for our analysis. The colour green indicates features that are more important than the randomised features and are therefore confirmed to be significant; blue entries show the performance of the random features; red and yellow indicate unimportant and tentative features, respectively.

documentation of jSymbolic. Several features in Table 4.6 use another feature: Beat Histogram. For reference, this feature was defined as follows:

"A feature vector consisting of the bin magnitudes of the beat histogram described above. The first 40 bins are not included in this feature vector... Each bin corresponds to a different beats per minute periodicity, with tempo increasing with the bin index. The magnitude of each bin is proportional to the cumulative loudness (MIDI velocity) of the notes that occur at that bin's rhythmic periodicity. The histogram is normalized."

Another convention adopted when calculating rests in the feature is that "Non-pitched (MIDI channel 10) notes are not considered in this calculation. Rests shorter than 0.1 of a quarter note are ignored".

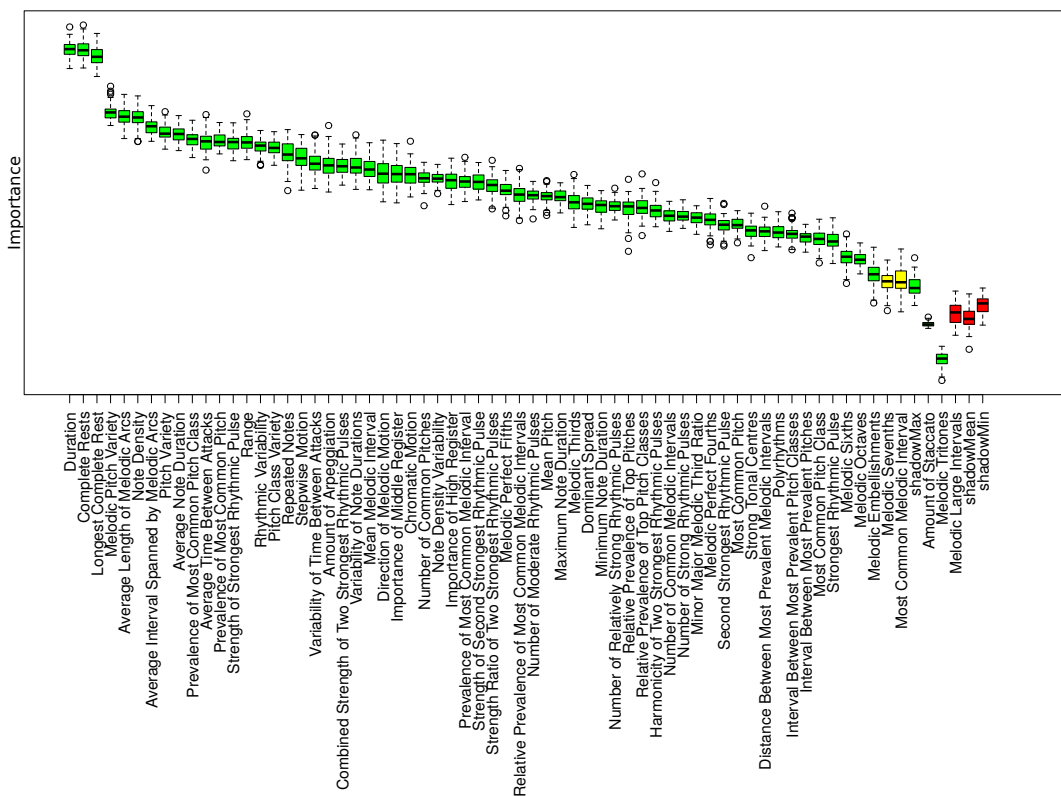


Figure 4.10: Feature importance for differentiating between output from different algorithms in the same format as Figure 4.9.

Another well-known feature extraction package, the FANTASTIC toolbox (Müllensiefen, 2009) is not used because its minimum supported input length excludes valuable annotated patterns. The jSymbolic toolbox was also not specifically designed for short excerpts, and many human-annotated and algorithmically extracted patterns are indeed short in length. Nevertheless, jSymbolic computes features in the same way for all excerpts; it does not, therefore, undermine the validity of the classification experiments.

### Implications

Although there are only 23 rhythmic features out of 63 in total in the jSymbolic features we used, the features ranked highest are rhythmic in nature. This high ranking of rhythmic features suggests that these features were more important than other features in constructing the random forest classifiers, which hints at potential improvements that could be implemented for current existing pattern discovery algorithms. String-based and data mining algorithms translate pitch and duration pairs into a list of symbols and therefore do not take into account metric structures im-



Feature name	Feature explanation
Strength Ratio of Two Strongest Rhythmic Pulses	Magnitude of the beat histogram peak with the highest magnitude divided by the magnitude of the second-highest magnitude.
Average Time Between Attacks	Average time (in seconds) between Note On events (regardless of MIDI channel).
Variability of Note Durations	Standard deviation of note durations (in seconds)
Average Note Duration	Average duration of notes (in seconds).
Complete Rests	Fraction of the music during which no pitched notes are sounding on any MIDI channel.
Strength of Second Strongest Rhythmic Pulse	Magnitude of the beat histogram peak with the second-highest magnitude.
Strength of Strongest Rhythmic Pulse	Magnitude of the beat histogram bin with the highest magnitude.
Rhythmic Variability	Standard deviation of the tempo-standardised beat histogram bin magnitudes.
Longest Complete Rest Duration	Longest amount of uninterrupted time (expressed as a fraction of the duration of a quarter note) in which no pitched notes are sounding. Total duration (in seconds) of the piece (pattern in our case).
Melodic Pitch Variety	Average number of notes that go by in a MIDI channel before a note's pitch is repeated (including the repeated note itself).
Average Length of Melodic Arcs	Average number of notes that separate melodic peaks and troughs.
Note Density	Average number of notes per second.
Average Interval Spanned by Melodic Arcs	Average melodic interval (in semitones) separating the top note of melodic peaks and the bottom note of adjacent melodic troughs.
Pitch Variety	Average number of notes that go by in a MIDI channel before a note's pitch is repeated (including the repeated note itself)
Prevalence of Most Common Pitch Class	Fraction of notes that correspond to the most common pitch class.

Table 4.6: Table of feature description. The features included are the top 10 features in Figure 4.10 (the top part of the table) and Figure 4.9 (the bottom part of the table). Only the most relevant description from the documentation of `jSymbolic` was included.

posed by musical punctuation such as bar lines. This omission is not uncommon as other algorithms also seldom explicitly consider metric features in patterns. In particular, the [SYMCHM](#) algorithm considers pitch aspects only, which explains its limited overlap with human-annotated patterns in the [PCA](#) visualisation. Generally speaking, the feature importance values we obtained suggest that in designing and evaluating pattern discovery algorithms, at least for the [MTC-ANN](#) dataset, we should take metric structures into consideration as well as repetition and pitch related features in patterns.

Our findings suggest that other jSymbolic features are almost all important for the classifiers, with the exception of three, which performed worse or at the same level as randomised features. The Boruta algorithm categorises Melodic Octaves feature as unimportant and the Melodic Sevenths and Melodic Tritones as tentative features. They are indeed nonessential features because these musical intervals rarely appear in the MTC-ANN dataset.

#### 4.2.5 Synthetic Pattern Insertion (SPI)

In previous sections, to understand where and how the algorithms disagree with each other, we computationally examined the musical patterns by visualising, combining, and classifying them. Through this algorithm-algorithm and algorithm-human comparison, we see that a diverse range of patterns can be extracted by algorithms and annotated by humans. While LFV, PP, and CC help us understand how the patterns resemble and differ from one another, predicting how algorithms would perform in the presence of new musical corpora remains a challenge.

Given the complexity of musical corpora, we propose the use of synthetic data in comparing the performance of musical pattern discovery algorithms. For example, one of the simplest and yet most fundamental questions to ask is whether pattern discovery algorithms are capable of identifying patterns in sequences such as "Pattern1" + "Pattern2" + "Random Excerpts" + "Pattern1". By artificially constructing a concatenation of musical patterns and random sequences of notes and rests, we can compare the performance of different algorithms at extracting patterns from sequences such as this—random sequences with patterns artificially inserted throughout, giving a controlled amount of regularity. This method has been widely used in other areas such as generic pattern mining and time series analysis (Bertens *et al.*, 2016; Chiu *et al.*, 2003). To the best of our knowledge, the method has not yet been used extensively to compare musical pattern discovery algorithms.

In this section, we examine seven musical pattern discovery algorithms using synthetic data. Because not all algorithms are open-sourced, we are only able to obtain

the patterns extracted by algorithms on certain datasets. Refer to Table 4.1 for the algorithms we added and removed from previous experiments in this section. We will now first describe how we created the synthetic data and then show an example piece from which different algorithms extract different patterns.

### **Creating the synthetic data**

To create the synthetic data, we randomly concatenated two predefined patterns with random excerpts. Given that we have planted the patterns artificially, we are able to compare the output of the algorithms to what is ostensibly a "ground truth". The details of the predefined patterns are given next.

One of the predefined patterns is a consecutive repetition of the notes C and G# in the fourth octave. We refer to this pattern as  $P_1$ , the repeated interval pattern. The other pattern we use is a C major scale, also known as the C Ionian scale in a modal context. We refer to this pattern as  $P_2$ , the scale pattern. Both patterns are twenty notes long. We chose these two patterns because they are of different kinds:  $P_1$  contains many local repetitions, while  $P_2$  is an example of a global pattern with a longer span. The excerpts of the patterns can be seen in Figure 4.11.

We sample the random excerpts,  $R$ , with the same note range as the scale pattern. We sample rests as well as notes. The lengths of the random excerpt are not fixed, but we constrain the total length of the random excerpts to be less than 50% of the total length of the synthetic piece.

### **Why we created the synthetic data this way**

Although this set-up is simple, it follows two rationales that we have in mind. First, we would like to reduce the challenge posed by rhythms and put more focus on investigating how pattern discovery algorithms detect pitch related patterns. Therefore, every note and rest in  $R$ ,  $P_1$ , and  $P_2$  has the note length of a crotchet. Second, the repeated interval patterns such as  $P_1$  can be used to test whether the algorithms can retrieve local repetitions of intervals, scale patterns such as  $P_2$  can be used to test whether the algorithms can identify global repetitions of an organised sequence of notes, and random excerpts such as  $R$  can be used to test whether the algorithms will recognise patterns from randomness.

### **Results and insights into SPI**

We present one piece from our set of synthetic pieces to illustrate the differences in the output produced by the algorithms. The point-set visualisation of the piece

#### 4 Comparisons of Musical Pattern Discovery Algorithms

can be seen in Figure 4.12, 4.13, and 4.2.5. Figure 4.11 visualises the presence of discovered patterns along the time axis of the synthetic piece, as introduced in Section 4.2.2. We further discuss our observations with respect to individual algorithms below.

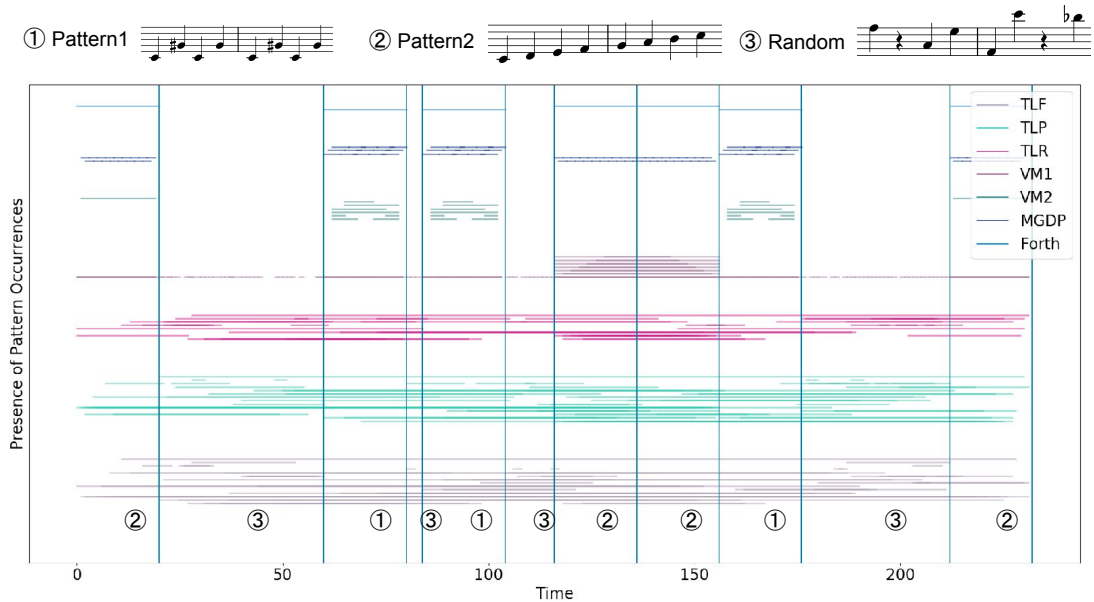


Figure 4.11: Visualisation of the presence of patterns using synthetic data.

#### VM

We observe that **VM1** consistently finds  $P_2$ , and **VM2** likewise for  $P_1$ . However, the algorithms discover multiple sub-patterns in the regions where  $P_1$  and  $P_2$  are present. Regarding the PP method introduced in Section 4.2.3, if we summarise algorithmic output by summing the counts of the discovered patterns, the resulting polling curve will give a correct indication of their presence. This tells us that the algorithms could benefit from a combined approach, though only in specific circumstances.

#### MGDP

Patterns discovered by the MGDp algorithm are short patterns. The parameters used were  $\alpha = 0.1$ ,  $\text{Min}(\text{support}) = 20$ ,  $\text{Viewpoint} = \text{Interval}$ . The discovered interval patterns are  $\{8, -8, 8\}$ ,  $\{-8, 8, -8\}$ ,  $\{2, 2, 1\}$ ,  $\{1, 2, 2\}$ ,  $\{2, 1, 2\}$ , which correspond to parts of  $P_1$  and  $P_2$ . We can see from Figure 4.11 that the trigram (three-component) interval patterns cover  $P_1$  and  $P_2$  completely, though with multiple overlapping occurrences.

## Forth

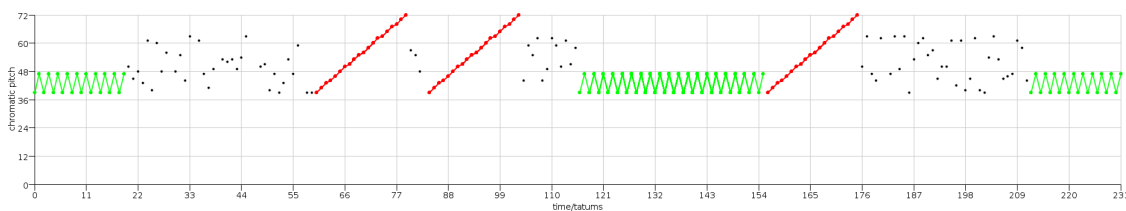


Figure 4.12: Patterns retrieved by the Forth algorithm

The Forth algorithm successfully retrieved all the planted patterns in the example piece. As we can see in Figure 4.11, by comparison with the ground truth (numbered ①-③), the two planted patterns and their occurrences are all retrieved. In Figure 4.12, we can see in more detail that the algorithms find both the repeated intervals and the scales.

## SIARCT

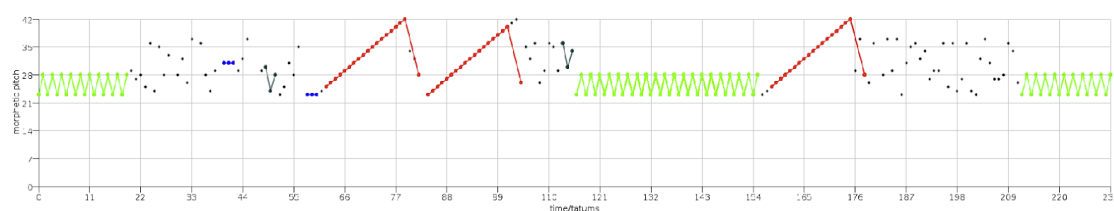


Figure 4.13: Patterns retrieved by the [SIACFP](#) algorithm.

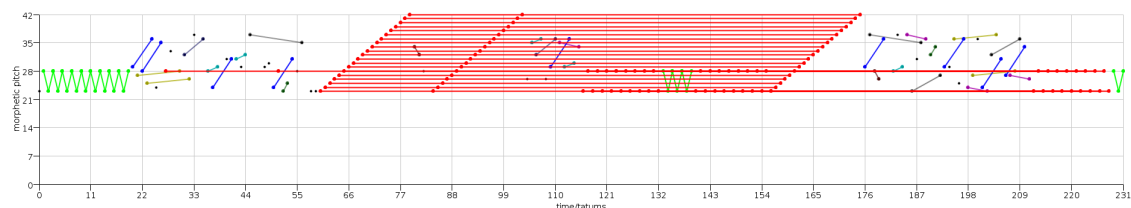
As shown in Figure 4.13, while patterns retrieved by SIARCT are indeed repeated patterns in the piece, they tend to deviate from  $P_1$  and  $P_2$ : the scale patterns are corrupted by adjacent notes in the random sequences, and the algorithm finds a small number of patterns in the random pitch sequences. Nevertheless, the approximate structure of the piece is retrieved.

## SIACPRF

Without parameter optimisation and applying out-of-the-box SIACR, SIACP, and SIACF algorithms, we find many patterns in addition to the ones that are injected into the piece in Figure 4.11. The output does not contain occurrences that are confined to within the boundaries of the planted patterns. Moreover, most of the extracted patterns cover a span of time in the piece that is much longer than the planted patterns.

## Further observations

### Patterns with long spans



[Patterns retrieved by the COSIATEC algorithm] Patterns retrieved by the COSIATEC algorithm.

Figure 4.2.5 shows long-span patterns which, with a concrete view of the example piece, can be seen to be unintuitive but not unreasonable: the injected scale  $P_2$  repeats across the piece, but the elements within the scales also repeat individually. This repetition is less intuitive because of limited human memory and the Gestalt principle of proximity in grouping notes (Lerdahl & Jackendoff, 1985; Snyder & Snyder, 2000). An audience can more plausibly identify the scale as a musical pattern than identify each individual note over a long time span. There is a duality between the locally bounded repetition (where the scale repeats as a whole) and globally identifiable repetition (notes in scale repeat separately). Depending on the subsequent applications of the extracted patterns, the algorithm should give more consideration to balancing or filtering out patterns of the desired kind. For example, if the patterns are to be used to assist with musical motif analysis, the local patterns are more valuable; if our goal is to uncover hidden structures in music, global patterns could open up more possibilities and insights.

### Comparison with ground truth

As we have discussed regarding long-span patterns, automatically evaluating the patterns can be difficult because multiple patterns can be extracted from the same constructed piece, and it is not fair to determine the correctness of the discovered patterns without application scenarios in mind. In addition, we must consider the issue of intentionality in the use and recognition of planted patterns. On the one hand, unintended patterns can emerge from the concatenation of patterns and randomness. In other words, one can inadvertently introduce inexact and exact repetitive patterns in addition to those we planted deliberately. On the other hand, the two patterns we employed are distinctive enough that a human annotator would likely be able to find the exact boundaries of both. Nevertheless, there are known cognitive phenomena such as apophenia (Steyerl, 2016), patternicity (Shermer, 2008), and hyperactive agency detection (Valdesolo & Graham, 2014). If there is a tendency to

find meaningful patterns in meaningless noise (Fyfe *et al.*, 2008; Shermer, 2008) in human subjective experience, the question of how one should evaluate the algorithmic output in relation to the ground truth human annotations remains a complex one. Therefore, despite the differences we observe in all three experiments, we cannot repudiate the value of the discovered patterns from algorithms.

### **Rhythmic features eliminated**

Notice that, in the example above, we reduced the variance in rhythmic features by using the same duration for each musical event. Rhythmic features, being the most divergence-inducing factor in CC, should give the algorithms more tendency to conform, once we control their variance by artificially increasing their regularity. Adding in more rhythmic variation could lead to greater insight into how the algorithms handle the interplay between pitch and rhythm. We defer more systematic investigations on this topic to future work. The minimal example we provided, despite being restricted to pitch, informs us of crucial aspects of the algorithms with respect to the pitch dimension. This is another advantage of using synthetic data—it gives us a higher level of control in investigating musical features.

### **Implications for the human-algorithm gap in musical pattern discovery**

By examining the patterns extracted from the synthetic data in this section, we gain new insight into how we can expect the algorithms to perform given controlled input data. For example, if we apply pattern discovery algorithms to a musical piece with sections of ostinatos ( $P_1$  being a special case) and an ornamental bridge ( $P_2$  being a special case), the algorithms might not be able to return the two sections as patterns. This might come as a surprise to human annotators who are looking for obvious repetitions in the piece. By reducing the complexity of current patterns discovery corpora to simple concatenations of preselected patterns, synthetic data may be useful for better inspecting, comparing, validating, and evaluating the algorithms by modelling the most likely behaviours of human annotators. For example, one can identify which algorithm is better suited than others for retrieving "conventional" (exact repetition) or "surprising" (regularities in randomness) kinds of patterns.

## **4.3 Discussion**

We propose to compare musical pattern discovery algorithms computationally in four ways: L<sub>CV</sub>, PP, CC, and synthetic data with planted patterns. These new meth-

ods address the two main challenges and four subsequent challenges we mentioned at the beginning of this chapter. To address the challenge posed by the large number of output patterns from the algorithms, we propose LFV to visualise the locations and features of musical patterns. To address the challenge of hard-to-understand implementations of the algorithms, we devise and experiment with PP and CC to gain insights about these algorithms from their output. We also propose synthetic data and visualisation methods to address this challenge: with planted patterns and visualisations, we can compare the pattern discovery algorithms in a simpler and more controlled manner.

For the four subsequent challenges, our methods relate to them in the following ways:

- For the differing output from algorithms, we leverage data fusion. However, we find no significant improvement of the results, because patterns from different algorithms differ too much.
- For the difficulty of relating algorithmic output back to meaningful musical concepts, we show that patterns extracted by algorithms are distinct from random patterns and are therefore not meaningless in the sense of being random. Whether being distinct from randomness implies some kind of musical meaningfulness is another question to be answered, and in Chapter 6 we further explore whether other musically meaningful concepts can be extracted by algorithms. Furthermore, by using synthetic data, we can plant musically meaningful patterns and see whether algorithms extract them as patterns.
- For the discrepancy between human annotations and algorithmically discovered patterns, we further confirm this discrepancy, and we discover that rhythmic aspects have the most influence when distinguishing between them.
- For the different application contexts of musical patterns, the visualisation and synthetic data methods we propose can be used to compare the algorithmic output with certain expectations or assumptions of the potential application contexts, and therefore determine whether an algorithm is suitable for a certain application context.

Our comparisons vary on the limited datasets available, but each of them provides a new means for investigating musical pattern discovery algorithms, namely, by visualising them, combining them, differentiating between their output, and synthesising artificial input.



## 4.4 Concluding remarks

In future work, this research could be extended in several directions. Firstly, the coverage of the algorithms could be extended. We did not perform a comprehensive comparison of all musical pattern discovery algorithms given limited availability and format compatibility. Secondly, for synthetic data, we can add more structures, possibly nested, during the data synthesis process. Adding rhythmic variations and polyphonic patterns would make the process suitable to investigate the algorithms that consider these aspects. Lastly, a natural next step is to integrate our findings into pattern discovery algorithm design. This integration could be difficult, however, because the concrete steps needed to improve an algorithm are contingent on its inner workings, which may differ from one algorithm to another. In addition, there is often not a single way to improve a given algorithm, but rather multiple strategies that can be used to address the issues we have identified. For example, an algorithm designer might add extra filters based on our classification results, consider rhythmic aspects to a greater extent, or provide pre-configured parameter sets optimised for different application scenarios.

On a more general note, a mapping between different algorithms and the application scenarios they are best suited to would be desirable, considering the diversity of pattern discovery tasks. The disagreement we found between algorithms gives support to this claim, with the differences serving as a guide for selecting algorithms based on different applications. For example, the algorithms that discover long and overlapping patterns might be more suitable for compression, and the algorithms that discover shorter patterns might be more suitable for music-theoretic and educational purposes.

One crucial link for diversifying the comparison between algorithms is to bridge the gap between the application scenario and how we formulate the strategies used in our data synthesis approach. With careful design, the data synthesis approach can potentially be tuned to simulate different application contexts. Application-driven thinking combined with the data synthesis approach also has the potential to save time when it comes to annotating the data and address issues pertaining to ambiguity in the annotated patterns. To generate these high quality synthetic data, future work may take fidelity, diversity, and generalisation performance into account, as proposed in (Alaa *et al.*, 2022).

Musical pattern discovery is an interdisciplinary area of research. Over the years, we have seen exuberant interest and considerable advancement of the [SotA](#) from contributors with diverse backgrounds. This has brought unique challenges as well as opportunities with a diverse range of tools, formulations, and evaluation meth-

#### *4 Comparisons of Musical Pattern Discovery Algorithms*

ods, including but not limited to the ones proposed in this chapter. We expect that greater coherence will be created by the feedback loop between such multifaceted evaluation results and the algorithms performing on diverse types of data. More concretely, users with access to a greater range of algorithm evaluation methodologies will be better equipped to provide feedback to algorithm designers that will ultimately be used for the design of those algorithms.

# Chapter 5 Modelling Patterns With Transformations Using Haskell

*A language that doesn't affect the way you think about programming is not worth knowing.*

– Alan J. Perlis

## 5.1 Introduction

### Summary so far about patterns and transformations

In previous chapters, we examined musical patterns annotated by humans and extracted by algorithms. We saw that many algorithms and humans disagreed on what constitutes a pattern; for example, in pattern polling and when using synthetic data, there was no clear consensus.

In this chapter, we will attempt to attain a better understanding of why certain musical fragments are considered patterns at all. An important step towards achieving this understanding is to devise a way to relate different pattern occurrences with each other. To this end, we will identify the musical transformations relating the different occurrences of a pattern. We will see that, using this methodology, we can begin to explain why these occurrences (or musical fragments) are grouped into musical patterns.

To show patterns and transformations in music, in Figure 5.1, we come back to a simple pattern and its occurrences in an étude. The same bar of notes is repeated successively, shifted upwards or downwards in the next occurrence: a pattern emerges—the occurrences are related to each other via the transposition in pitch. The second occurrence can be viewed as a variation of the first occurrence, realised by the transformation of transposition. More transformations are known to exist in music theory and can be used to characterise different types of repetitions

with variations. Similarly, Figure 5.2 shows pattern occurrences annotated by musicologists with a diverse range of transformations.

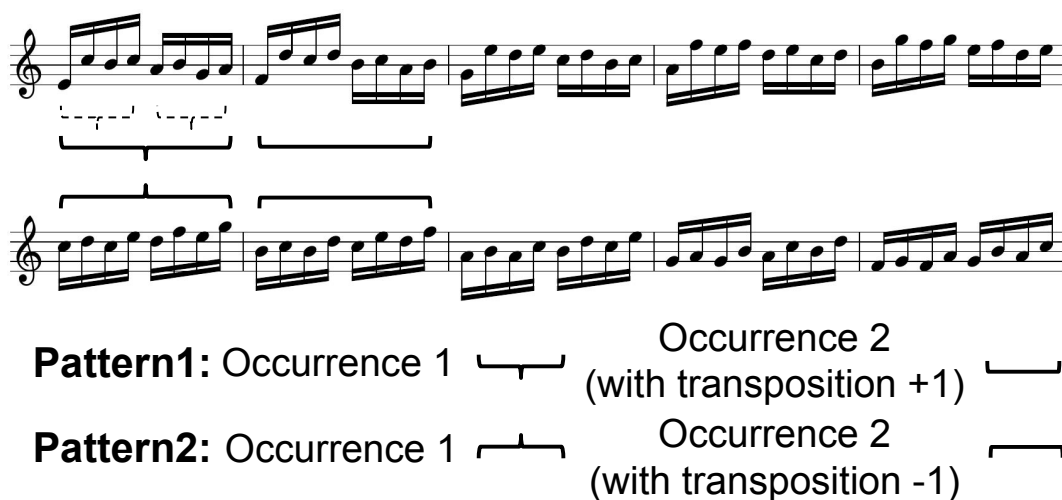


Figure 5.1: Musical pattern and transformation examples in simple études. The patterns and occurrences are marked with over-/under-braces. We only consider a two-level hierarchy of pattern-occurrence; sub-patterns as in the dotted braces are not considered.

### Research question

Given this backdrop of pattern and transformation, the research questions for this chapter are the following:

Can we identify the transformations that explain the variation between different pattern occurrences? More concretely, how do we implement a library (or a [Domain Specific Language \(DSL\)](#)) to automate this identification process?

In relation to the concrete example that was sketched above in Figure 5.1 and 5.2, given these pattern occurrences, we should be able to identify that the occurrences are related by transposition, but in general pattern occurrences can be related by many other possible transformations. Can we extend our ideas to handle transformations other than simple transpositions? We will address these questions in this chapter and the next.

### Contribution

Our contribution:

**Pattern1:** { } four annotated occurrences    
 **Pattern2:** [ ] two annotated occurrences    
 **Pattern3:** ( ) two annotated occurrences

Figure 5.2: Musical pattern and transformation examples in MTC-ANN. There are some exact repetitions. The local type of transformations—approximation—is also needed to describe the relations between the pattern occurrences.

- Implementing a library in the [Functional Programming \(FP\)](#) language Haskell (Peyton Jones, 2003), [Pattrans](#) (Melkonian *et al.*, 2019), to model musically important transformations. Using Haskell, we can model these transformations as functions. For example, one simple transformation is the chromatic transposition,

```
transpose :: Note -> Note
transpose x = x + n
```

This function transposes each note by some pitch shift  $n$ . In Figure 5.3, we schematically show the input and output of our implementation in Figure 5.2. One key insight is that functional languages are a natural fit for working with musical transformations. Throughout the library, we will use category theory<sup>1</sup> to structure the transformations in a modular and compositional fashion.

In the original paper, the first author, Orestis Melkonian was the main library implementer. The contributions of the second author, Iris Yuping Ren, include generating the idea of using musical transformations between pattern occurrences rather than the content of the occurrences, proposing the specific transformations, modifying and using the library, and significantly taking part in the writing and the re-writing of the paper and the chapter.

<sup>1</sup>A branch of mathematics with numerous applications in the research of programming languages. The connections with category theory will be explained at the end of the chapter.

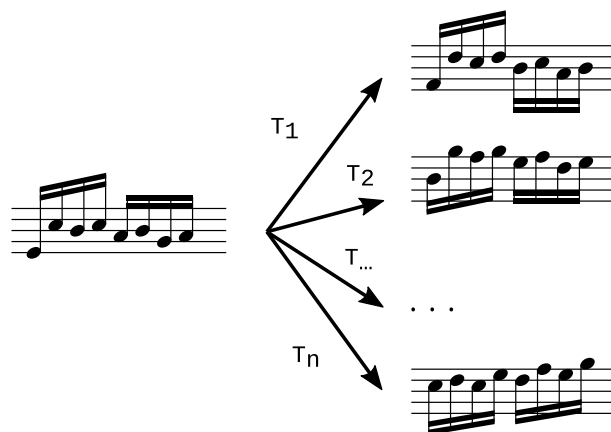


Figure 5.3: We take one set of pattern occurrences in Figure 5.1 to show the input and output of our implementation. The input is musical pattern occurrences. The output is  $T_1, T_2, \dots, T_n$ , the transformations.

## 5.2 Pattrans

The Pattrans library<sup>2</sup> has the following decoupled components: types (Section 5.2.1), transformations (Section 5.2.2), and other technical components such as import, rendering, and MIDI that we will not cover in this dissertation. In the following, we will give an overview of the main techniques we used to implement our meta-analytic framework. Although we chose Haskell for our implementation, we do not rely on any of Haskell’s advanced type-level features (e.g. type families and data kinds); thus any other strongly-typed functional programming language would suffice.

### 5.2.1 Basic types

Starting from the fundamental elements in music, we define the types of time and pitch as in Code Snippet 5.1. In our encoding, we used the simplest possible representation of musical events. Time is represented as the number of crochet beats from the start of the song, while MIDI numbers represent pitch, in the form of plain integers.<sup>3</sup> We have simplified the types of primitives for the sake of presentation. The actual implementation contains additional information about scale degree, intervals, and octaves.

We then assemble these basic types to create a basic musical event, i.e. a note, which consists of a time and pitch value, and implemented using a record type. A pattern occurrence is a list of notes. Both types are shown in Code Snippet 5.2.

<sup>2</sup>The implementation is available at <https://github.com/omelkonian/hs-pattrans>

<sup>3</sup>Here, we only consider the part of a MIDI value that represents pitch number, ignoring other features such as velocity.

```

type Time = Double
type Pitch = Integer
type Interval = Integer

```

Code Snippet 5.1: Basic types for defining musical concepts.

```

type PatternOccurrence = [Note]
data Note = Note { ontime :: Time
                  , pitch   :: Pitch
                  }

```

Code Snippet 5.2: Define pattern occurrence and note.

In accordance with the output of many pattern discovery algorithms, we define a pattern to be a sequence of pattern occurrences. In a group of pattern occurrences, we assume that there is some original material from which other occurrences stem. We refer to the original material as the *prototype pattern*. For now, we take the first occurrence of a pattern as the prototype of a pattern. We also keep some metadata (such as to which piece of music a pattern belongs), which can be used later in the analysis. The definition of the pattern in Haskell can be seen in Code Snippet 5.3. We can see that the information attached to each pattern is:

- the name of the music piece to which the pattern belongs
- whether a music expert or an algorithm discovered the pattern occurrence
- a name for the current pattern
- the pattern prototype (always taken from the first occurrence)
- all other occurrences of the prototype

```

data Pattern = Pattern
  { piece_name    :: String
  , expert_name  :: String
  , pattern_name  :: String
  , basePattern  :: PatternOccurrence
  , patterns     :: [PatternOccurrence]
  }

```

Code Snippet 5.3: A group of pattern occurrences becomes a pattern.

Notice that the difference between a pattern and an occurrence of this pattern is crucial. For example, in Figure 5.1 and Figure 5.3, the eight semi-quivers note groups constitute pattern occurrences, out of which the note group starting with the note E is the prototype pattern, and the all pattern occurrences together constitute a pattern.

## 5.2.2 Transformation checking

We now want to identify a transformation between the prototype pattern and other occurrences that are discovered either by algorithms or human annotators. To carry out this identification, we develop a library of transformations. The type signature of each such transformation takes the form of:

```
PatternOccurrence -> PatternOccurrence -> Bool
```

that checks if two pattern occurrences are related by a transformation.

Now let us consider how we can implement the transformations we want. The first thing to realise is that the definitions of our transformations will essentially be a predicate on two elements, returning `True` when they can be related by a transformation and `False` otherwise. An example of such a function is the canonical equality: just checking that the constituent notes are identical as shown in Code Snippet 5.4.

```
exactRepetition :: PatternOccurrence -> PatternOccurrence -> Bool  
exactRepetition p1 p2 = p1 == p2
```

Code Snippet 5.4: Define a function to check that two pattern occurrences are exact repetition of each other.

Although this identity "transformation", i.e. the exact repetition, looks trivial in its simplest form, the general problem that we aim to address is to write such functions compositionally. By *compositionally*, we mean that the user can write checks that only involve partial information of pattern occurrences (e.g. the pitch or the duration of a note), or try to group and combine conditions (e.g. the pitch and duration information of some notes combined) for checking the transformations. We need a library to describe such checks.

Even though we can write the checks as functions, they cannot be composed. The type of the function `exactRepetition` is very specific, demanding every note from one pattern occurrence to be compared with a corresponding note in another pattern occurrence. However, if we want to write checks compositionally, as we described above, we need to define a type that describes the checks more generally:

```
newtype Check a = Check { getCheck :: a -> a -> Bool }
```

Code Snippet 5.5: Define checker type for predicating on two elements of the same type.

The `Check` newtype in Code Snippet 5.5 is simply a wrapper to store functions with the type `a -> a -> Bool`. This can be generalised further to type `a -> b -> Bool`, as shown in Code Snippet 5.6. We also provide a convenient infix notation to check for this, as well as a type alias for *homogeneous* checkers of elements of the same type, the same as the `Check` defined in Code Snippet 5.5. The infix operator



`(<=>)` has the type signature `a -> b -> Check a b -> Bool`, which means that it takes in inputs of type `a` and type `b`, along with a checker of type `a b`; it produces a Boolean value as output. The implementation of this infix operator can be realised by the function stored in type `Check`.

```
newtype Check a b = Check { getCheck :: a -> b -> Bool }
```

```
(<=>) :: a -> b -> Check a b -> Bool
(x <=> y) p = getCheck p x y
```

```
type HomCheck a = Check a a
```

Code Snippet 5.6: Define generalised checker for predicating on two elements of two different types.

Once we have the `Checker` type, the simplest possible comparison is that of exact equality between elements of the same type, which can be defined using the `Eq` typeclass, as shown in Code Snippet 5.7.

```
equal :: Eq a => HomCheck a
equal = Check (==)
```

Code Snippet 5.7: The simplest possible `HomCheck`.

Moving on from the canonical equality example, we now show an example for checking pitch, and introduce the concept of *contravariance*. To check only for pitch, we first write a `pitch` function that extract pitches from notes, as shown in Code Snippet 5.8. A verbose way to create a function that compares pitches of pattern occurrences is also shown in Code Snippet 5.8.

```
pitch :: PatternOccurrence -> [Pitch]
pitch notes = map pitch notes
```

```
equalPitch :: PatternOccurrence -> PatternOccurrence -> Bool
equalPitch p1 p2 = (pitch p1) == (pitch p2)
```

Code Snippet 5.8: Extract pitch using `map`, and define a function to compare pitches of pattern occurrences .

To create more general checks based on viewpoints/features<sup>4</sup> such as pitch, we need the concept of contravariance. As an example of contravariance, assume you have a checker for pitches (i.e. `HomCheck [Pitch]`) and you want to use that to check the equivalence of two patterns (i.e. `HomCheck PatternOccurrence`). Hence, you need to have a function `HomCheck [Pitch] -> HomCheck PatternOccurrence`, but you cannot define it even if you have a function `[Pitch] -> PatternOccurrence`. What you, in fact, must have in your hands is a function

<sup>4</sup>Please see Section 2.3.1 for an explanation of these terms.

`PatternOccurrence -> [Pitch]` (notice the reversal in the argument order, namely, *contravariance*). The concept of *contravariance*, for our purpose, can be understood as reversing the positions of the arguments of a function. As shown in Code Snippet 5.9, we can combine `pitch` and `HomCheck` contravariantly by using a predefined operator in Haskell.

```
equalpitch' :: HomCheck [Pitch] -> HomCheck PatternOccurrence
equalpitch' = pitch >$< equal
```

```
(>$<) :: (a -> b) -> Check b -> Check a
(>$<) = contramap
```

Code Snippet 5.9: Example of contravariance using `pitch`.

In more general terms, to define more compositional operations, we first make the observation that the type parameters (e.g. `Pitch` or `Time`) occur in a *contravariant* position, since these parameters are arguments to a function. Second, in the most general case, there are two different arguments for the comparison (we need two inputs to make a comparison). With these observations, we arrive at the concept of a *contravariant bifunctor*<sup>5</sup> and define the instance for our checker type, as shown in Code Snippet 5.10.

```
class ContravariantBifunctor p where

  contraBimap :: (t -> a) -> (s -> b) -> Check a b -> Check t s
  contraBimap f g = contra1 f . contra2 g

  contra1 f = contraBimap f id

  contra2 g = contraBimap id g

instance ContravariantBifunctor Check where
  contraBimap f g p = Check $ \ x y -> (f x <=> g y) p
```

Code Snippet 5.10: Definition of contravariant bifunctor.

### 5.2.3 Compositions of checking transformation

In order to combine multiple checks in conjunction, we would like to combine multiple simple checks. These checks, for example, can be built from the `map` function such as in Code Snippet 5.8. We can extract multiple Boolean values from multiple checks and then take their conjunction. This combination of checkers can be identified with a kind of monoidal<sup>6</sup> algebraic structure.

<sup>5</sup>Contravariant bifunctors are functors with two arguments, where the map operation is contravariant.

<sup>6</sup>A monoid is a set that is closed under an associative binary operation and has an identity element. Here, we consider the set of checkers.

As an example, we show the monoidal structure with pitch and duration in Code Snippet 5.11<sup>7</sup> The first function is a verbose implementation of the check. To create more general combinations, we implement a monoidal structure and make the check more succinct: the simplified function is the second function of the same snippet.

The implementation of the monoidal structure can be seen in Code Snippet 5.12. Note that Check can be seen as a monoid in more than one way, i.e. using either conjunction or disjunction. Here, we stick to the conjunctive version (note the use of && in Code Snippet 5.12), since we do not have any use-cases for disjunctive checkers yet. If such a construct is needed in the future, we will provide newtype wrappers and the programmer would then need to manually annotate which monoid instance to use (when one uses <>).

```
exactPitchDuration :: PatternOccurrence -> PatternOccurrence ->
  ↳ Bool
exactPitchDuration p1 p2 = (pitches p1 == pitches p2) && (duration
  ↳ p1 == duration p2)

exactPitchDuration' :: Check PatternOccurrence
exactPitchDuration' = durations >$< equal
  <> pitch >$< equal
```

Code Snippet 5.11: Monoidal structure to combine pitch and duration.

```
instance Monoid (Check a b) where
  mempty = MkCheck (\ _ _ -> True)
  p <> q = MkCheck (\ x y -> (x <=> y) p
    && (x <=> y) q)
```

Code Snippet 5.12: Implementation of the monoidal structure of checkers.

## 5.2.4 Approximation

Now we can recognise exact repetitions, and in this section, we aim to model the local type of transformations in music with approximation, as we discussed with Figure 5.2 in Section 5.1. Recall in Figure 5.1 and 5.2, we showed patterns in a simple étude and folk songs. For a simple étude, we have a perfect transformation (i.e. transposition) to relate pattern occurrences. In the folk songs and more generally, however, this is rarely the case, so there is a need for "approximate matching". The problem now then is to add more flexibility by allowing inexact matches, essentially allowing the insertion and deletion of musical events.

<sup>7</sup>We assume here that there is a function "durations" that extract the durations of from musical notes. The actual implementation is given later in Code Snippet 5.16. The concept of composition is more important than the implement at this point.

This is not an easy problem to solve. In our implementation, to allow different degrees of approximation, we specify percentages for how many insertions and deletions we allow when comparing a pair of musical patterns. For example, we can match a list of pitches,  $[A, C, F, A, B]$ , and  $[A, C, G, A, B]$  with an approximation of 80%. Giving a more complex example such as  $[A, A, B, C]$ , and  $[A, B, C, D]$ , we cannot immediately line up the notes pairwise, but need to delete the first A and last D to have a match.

Formally, we extend the criteria for a "match" with the approximation of percentage  $p$ : when at least  $p$  of the prototype is in the occurrence, i.e.

$$\text{ignored} \leq (1 - p) * N \quad (5.1)$$

and when at least  $p$  of the occurrence is in the prototype, i.e.

$$\text{added} \leq (1 - p) * M \quad (5.2)$$

where  $N, M$ , are the lengths of the prototype and occurrence, respectively;  $\text{added}$  is the number of elements in the occurrence that do not appear in the pattern,  $\text{ignored}$  is the number of prototype elements that were not deleted i.e.  $M - (N - \text{added})$ . Notice that there can be multiple ways to add and ignore elements to obtain a match. For example, one can ignore all elements of the prototype occurrence and add back all elements of the other occurrence, which would not be a desirable match for approximation. We therefore aim to find the minimal number of insertions/deletions necessary.

In the first example,  $[A, C, F, A, B]$ , and  $[A, C, G, A, B]$ , we have  $N = M = 5$  and we *ignored* the prototype note F and *added* the occurrence note G (i.e.  $\text{ignored} = 1$  and  $\text{added} = 1$ ). Thus we can conclude that these two patterns are 80% equal, since the two required conditions are satisfied:

$$\text{ignored} = 1 \leq 1 = (1 - 0.8) * 5$$

and

$$\text{added} = 1 \leq 1 = (1 - 0.8) * 5$$

In implementation, we first define this approximate checker type by passing a floating point number  $p \in [0, 1]$ , representing the approximation degree as a percentage, as shown in Code Snippet 5.13. We also set a maximum look-ahead value for computational efficiency. We then approach this approximate checking in two steps. First, we define a function that computes the maximum number of allowed insertions and

deletions. Then, we consider the possible ways in which the two lists can be aligned. To compute the possible alignments, we take into account three parameters/variables (the maximum number of insertions and deletions, and the maximum-look-ahead value), try to traverse the occurrence until the prototype element is found, and consider the following cases:

- When an element of the prototype is found in the occurrence:
  - If the prototype element we are looking for is at the head of the occurrence with which we are comparing, we process onto the next prototype and occurrence element.
  - If we need to remove elements from the occurrence order to find the prototype element, we delete the elements at the head of the occurrence, and decrease the maximum insertion allowed.
- Otherwise, two cases are possible:
  - If we do not find a matching element, we delete the prototype element if possible, and we decrease the maximum deletion value allowed.
  - If we do not find the element and cannot delete it, we abort.

To find the prototype elements and update the maximum allowed insertion values, we use an auxiliary function `find` as shown in Code Snippet 5.14.

In our original paper (Melkonian *et al.*, 2019), we described the program using a count-up approach rather than the countdown approach described above. There was a problem with this count-up description. In the paper, we described the approach using a "min" function, which picks out the minimum (add, ignore) pair. The lexicographic ordering of the "min" function does not work best because it would favour (1,100) instead of (2,2) "added" and "ignored" pairs, which is not desirable in most cases. This bug was only in the presentation of the implementation, but not in the implementation itself, since we have always circumvented this problem by using an upper limit of the insertions/deletions permitted, as described here.

```
type ApproxCheck a = Float -> HomCheck a
(~~) :: ApproxCheck a -> Float -> HomCheck a
approxChecker ~~ p = approxChecker p
```

Code Snippet 5.13: Definition of approximation type and function.

The worst-case time complexity of the algorithm above is  $O(NM)$  where  $N, M$  are the lengths of the prototype and occurrence respectively, since we would need to traverse the whole occurrence for each element of the prototype. When the lengths of pattern occurrences are small, this is not a bottleneck. Nevertheless, we set a maximum look-ahead on the deletion process to facilitate faster analyses in case pattern

```

maxLookahead = 5

approx :: Eq a => ApproxCheck [a]
approx xs ys =
  let maxIgnored = (1 - p) * length xs in
      let maxAdded = (1 - p) * length ys in
          align (maxIgnored, maxAdded) xs ys

align :: [a] -> [a] -> (Int, Int) -> Bool
align ys [] (maxI, maxA) = 0 <= maxI && length ys <= maxA
align [] xs (maxI, maxA) = length xs <= maxI && 0 <= maxA
align ys (x:xs) (maxI, maxA)
  | Just (maxA', ys') <- find x ys maxA
  , maxA' >= 0
  , maxA - maxA' <= maxLookahead
  = align ys' xs (maxI, maxA')
  || (maxI > 0 && align ys xs (maxI - 1, maxA))

find :: a -> [a] -> Int -> Maybe (Int, [a])
find _ [] _ = Nothing
find x (y:ys) maxA
  | maxA < 0 = Nothing
  | x == y   = Just (maxA, ys)
  | otherwise = find x ys (maxA - 1)

```

Code Snippet 5.14: Function that search for approximations. The recursion and trivial cases are omitted.

occurrences are long (e.g. a repeating section of music), resulting in overall runtime complexity of  $O(N)$ , since we would perform  $O(1)$  computation for each of the  $N$  elements of the prototype<sup>8</sup>.

The above approximation process works fine when we compare musical notes directly. However, when we consider intervals, derived from pitches by taking the pitch difference of the consecutive notes, they score rather badly, since a simple insertion on the prototype pattern would create a lot of differences in the paired output. As a motivating example, assume our prototype consists of the notes A and B and the occurrence adds a passing note B<sup>b</sup> in between them. While first-order approximate equality works well when comparing their pitch, we get in trouble when comparing their intervals; we have lost all similarity between them, since the prototype has intervals [2] and the occurrence [1, 1].

We, therefore, consider two types of approximation, and call the one we covered above as first-order and move on to cover the second-order approximation. We call what is in Code Snippet 5.14 first-order because we only compare the sameness of the token. What we call second-order approximation also considers the intervalic equivalence between two tokens. We define it in such a way that we additionally

<sup>8</sup>Note that the faster implementation is actually an under-approximation of the proposed algorithm, i.e. there might be false negatives in the results.

allow the merging of consecutive elements in the occurrence and match them to a single entity of the prototype. Key steps of the implementation are shown in Code Snippet 5.15, where we find index of the elements in the list of notes that can make up to the sum of other elements in another list.

```

approxEq2 :: (Eq a) => ApproxCheck [a]
approxEq2 = Check $ \xs ys -> ...
  where
    ...
    align x (y:ys)
      | Just i <- findIndex 0 x (y:ys) maxLookahead, maxA >= i
      = Just (added - i, snd $ splitAt i ys)
      | otherwise = align x ys $! (added - 1)
    ...

    findIndex i 0 _ _ = Just i
    findIndex i x (y:ys) maxLookahead
      | x >= y = findIndex (i + 1) (x - y) ys (maxLookahead - 1)
      | otherwise = Nothing

```

Code Snippet 5.15: Key implementation steps of the second-order approximation.

To give an example, if we insert some intervals of "1" and "3" into a series of "1" and "2", and check whether they are 75% approximate to each other:

```
([2,1,2,2] <=> [1,1,3,2,1,1]) (approxEq2 ~~ 0.75)
```

This is true because the second-order approximate equality allows us to merge the first two and the last two intervals of the occurrence, replacing them with their sum:

```
([2,1,2,2] <=> [2,3,2,2]) (approxEq2 ~~ 0.75)
```

We now have only a single ignored interval in the prototype and a single "added" interval in the occurrence, thus it is safe to conclude that these (interval) patterns are 75% equal:

$$\text{ignored} = 1 \leq 1.0 = (1 - 0.75) * 4$$

$$\text{added} = 1 \leq 1.5 = (1 - 0.75) * 6$$

To use this package in our experiments in the next chapter, we perform multiple passes of comparisons by varying the approximation degree of 100% (exact repetition), 80%, 60%. We notate the approximated transformations by appending a single-digit number at the end of the names of the transformations and their combinations. For example, `exact8` denotes a match of exact repetition but with at most 80% approximation, i.e. setting  $p = 0.8$  in the previous equations.

## 5.2.5 Compositional transformations

With all the constructions we presented above, we can write checkers for exact repetition as shown in Code Snippet 5.16. We devised compositional operations to glue together the different features (e.g. pitch and rhythm) with the checker. In addition, we extended the notion of equality in the checker to check the approximation of a pattern.

As an example of all these combinations, in Code Snippet 5.17, we can write a checker that checks inversions with different degrees of approximation. In Code Snippet 5.18, similarly, we show the implementations of the two other checkers for transformations: checkers for chromatic transposition and rhythm-only repetition with different degrees of approximation. Chromatic transposition preserves rhythm and pitch intervals, while rhythm-only repetition only preserves rhythm. We will describe these two transformations in the context of music in the next section.

```
exactRepetitionOf :: HomCheck PatternOccurrence
exactRepetitionOf = rhythm >$< equal
                  <> pitches >$< equal

toRelative :: Num a => [a] -> [a]
toRelative = map (-) . pairs

pitches, intervals :: PatternOccurrence -> [Pitch]
pitches    = map pitch
intervals  = toRelative . pitch

onsets :: PatternOccurrence -> [Time]
onsets = sort . map ontime

durations :: PatternOccurrence -> [Time]
durations = fmap (uncurry (-)) . pairs . onsets

rhythm :: PatternOccurrence -> [Time]
rhythm = map (truncate' 2) . durations

truncate' :: Int -> Double -> Double
truncate' n x = fromIntegral (floor (x * t)) / t
  where t = 10^n
```

Code Snippet 5.16: Checker for exact repetition and the types of its components.

```
inversionOf :: ApproxCheck PatternOccurrence
inversionOf = basePitch >$< equal
            <> rhythm >$< approxEq2
            <> intervals >$< (inverse $< approxEq2)
```

Code Snippet 5.17: Checker for inversion up to approximation.

In Pattrans, we have constructed the DSL so that one pick their own sets of transformations with ease. For example, in Code Snippet 5.19, we define a default analysis



```

transpositionOf :: ApproxCheck PatternOccurrence
transpositionOf = rhythm >$< approxEq2
                 <> intervals >$< approxEq2

rhythmicOnly :: ApproxCheck PatternOccurrence
rhythmicOnly = rhythm >$< approxEq2

```

Code Snippet 5.18: Checker for chromatic transposition and rhythmic-only repetitions.

with four transformations (exact repetition, transposed, inverted, and retrograded) and an analysis called "exact" that only considers exact repetitions. Both of them are defined along with different degrees of approximation (1, 0.8, 0.6, 0.4, 0.2).

```

analyses :: [(String, Analysis)]
analyses =
  [ ( "default"
    , ( [ ("exact", (exactOf ~~))
        , ("transposed", (transpositionOf ~~))
        , ("inverted", (inversionOf ~~))
        , ("retrograded", (retrogradeOf ~~))
        ], [1,0.8..0.2] ))
    , ( "exact"
    , ( [ ("exact", (exactOf ~~))
        ], [1,0.8..0.2] ))
  ]

```

Code Snippet 5.19: One can define different sets of transformations for analysis. The transformations are then checked one by one in order.

## 5.3 Musical transformations in Pattrans

After seeing the implementation, we describe and explain the transformations we use in Pattrans. In the list below, we show our categorisation of several important musical transformations and indicate which ones are implemented in Pattrans:

- Shifting
  - in time: *exact repetition* implemented
  - in pitch: transposition, comes in different flavours (*chromatic transposition* implemented, *tonal transposition* not implemented)
- Reflecting:
  - in time: *retrograde* implemented
  - in pitch: *inversion*, comes in different flavours (chromatic inversion implemented, tonal inversion not implemented, similar to transposition)
  - composition: retrograde-inversion implemented
- Scaling:

- in time (*augmentation*, *diminution* implemented)
- (theoretically, there could also be scaling in pitch, but it has not been used in music to the best of our knowledge, not implemented)

In addition to the above, we also consider approximations—the local transformations that do not apply to all notes in a pattern, but only a few and on a local level, such as the insertion or deletion of certain notes. In the remainder of this section, we will introduce more detail of these transformations and how we group them in Pattrans.

### Primary and miscellaneous transformations

We take three prominent transformations as our primary transformations:

- **Exact repetition**: repeat an occurrence with exactly the same musical events, that is, horizontal translation preserving rhythm and pitch
- **Chromatic transposition**: move pitches by a fixed number, that is, vertical translation preserving rhythm
- **Rhythmic-only repetition**: repeat the rhythm of an occurrence while permitting any pitch transformations other than the two considered above

We devise the rhythmic-only repetition due to the influence of the feature analysis in Chapter 4, where rhythmic features were important for distinguishing between algorithmic and human-annotated patterns. We also consider a miscellaneous (misc.) transformation group, which contains the following transformations and their compositions: retrograde, augmentation, diminution, and approximation. Other potential non-primary, non-misc. transformations and relations are referred to as "unclassified" (so named because we cannot find them yet with the transformations above).

Formally speaking, we have a list of musical events in the prototype pattern occurrence,  $P = \{(t_i, p_i)\}, i \in \mathbb{N}$  (hereon, we assume  $i$ , the indices of musical events, are natural numbers), and  $P' = \{(t'_i, p'_i)\}$  the list of musical events in the pattern occurrence to compare it with, where  $t_i$  and  $t'_i$  are the onset sequences of the pattern occurrences, and  $p_i$  and  $p'_i$  the corresponding pitch sequences. Let the **Inter-Onset Intervals (IOIs)** sequences be  $\Delta t_i := t_{i+1} - t_i$  and  $\Delta t'_i := t'_{i+1} - t'_i$ . Note that, in this formulation, we do not consider the cases where  $P$  and  $P'$  are of different lengths, because the length difference breaks the criteria of being under one of the three primary transformations we consider. The checking criteria for the

three primary transformations correspond to the following:

- (1) **Exact repetition**  $\forall i(p_i = p'_i) \wedge (\Delta t_i = \Delta t'_i)$
- (2) **Chromatic transposition**  $\exists! c \in \mathbb{N}, \forall i.s.t.(p_i = p'_i + c) \wedge (\Delta t_i = \Delta t'_i)$
- (3) **Rhythmic-only repetition**:  $\forall i(\Delta t_i = \Delta t'_i)$

Intuitively, there is a gradually loosening process between the three transformations on how strictly the pitches need to match. Exact repetition requires pitch to be preserved exactly; chromatic transposition requires a constant pitch offset value; and rhythmic-only repetition does not impose any constraints on pitches. Furthermore, the last transformation covers the cases of other common music transformations: tonal transposition and inversion. We do not emphasise tonal transposition in our list of transformations because computing the one-and-only correct key (“Symbolic Key Finding Results - MIREX Wiki”, 2005) and then finding the appropriate tonal transposition given a short pattern occurrence is not always possible. These transformations have also been shown to be perceptually relevant for listeners under certain circumstances according to the discussions in (Krumhansl *et al.*, 1987; Meredith *et al.*, 2002; Temperley, 1995; Thorpe *et al.*, 2012), despite ongoing research and disagreement on the generalisability of such perceivability. For the above reasons, we group transformations into the three primary transformations and misc. transformations. Although it is possible to change this grouping in Pattrans as we will show in Section 5.2.2, this grouping will remain important in the next chapter, where we present the analytical results from Pattrans.

Though this list is not comprehensive, by using well-established and not overly complicated transformations, we equip ourselves with the power to examine musical patterns rigorously and quantitatively. For example, we can compute how many occurrences of a pattern are related to a prototype occurrence via the transformations, as well as what kind of transformations can be observed in a set of occurrences. Such data can be used to investigate different forms of musical patterns. This computation is what we will carry out in the next chapter.

### Other groups of transformations

In (Melkonian *et al.*, 2019), the paper on which this chapter is based, we chose a group of transformations as atomic transformations primarily due to their simplicity: **exact repetition**, **chromatic transposition**, **inversion**, **retrograde**, **augmentation**, **diminuation**, **retrograde inversion**, **rotation**. In this dissertation, we have re-formulated this

into primary transformations and misc. transformations, as we will be using this re-formulated grouping in the next chapter.

### Orders of transformations

When considering computationally implementing multiple transformations, the issue of order arises: given two transformations  $t_1$  and  $t_2$ , which one should be checked first? One can propose checking all transformations and their combinations against every pair of musical patterns we will be comparing with. This brute-force approach can be very computationally expensive. However, as we mentioned earlier in this section, some transformations are stricter than others, so we can check the strictest transformation first, and then the next strict one for the patterns that have not been paired.

## 5.4 Discussion and concluding remarks

In this chapter, we presented an expressive DSL to describe (monophonic) music-theoretic transformations, based on simple notions of category theory, namely monoids and contravariant functors. The transformations we used were explained and motivated.

One of the significant benefits of the transformation approach is its compositionality, meaning we can express meaningful transformations in terms of simpler ones. The FP language Haskell also gives us advantageous controls and architecture of the program. We summarise the groups of transformations and their combinations in Figure 5.4.

We do not claim that our model is comprehensive. We certainly do not touch upon the intentionality behind the musical transformations, i.e. determining whether a musical transformation was applied as a technique or accidental, which has a separate level of intricacy in itself. Instead, we use transformations to describe computations that can be used to systematically and quantitatively probe the musical patterns given by human annotators and algorithms. Using this DSL, we will see what kind of insights can be gained in the next chapter. In the remainder of this section, we will discuss remaining issues and notable points of our implementation.

### Multiple transformations

When  $t_1$  and  $t_2$  can both be used to match two pattern occurrences, we can take a pluralist view (taking both  $t_1$  and  $t_2$  as valid matches) or consider extra factors that

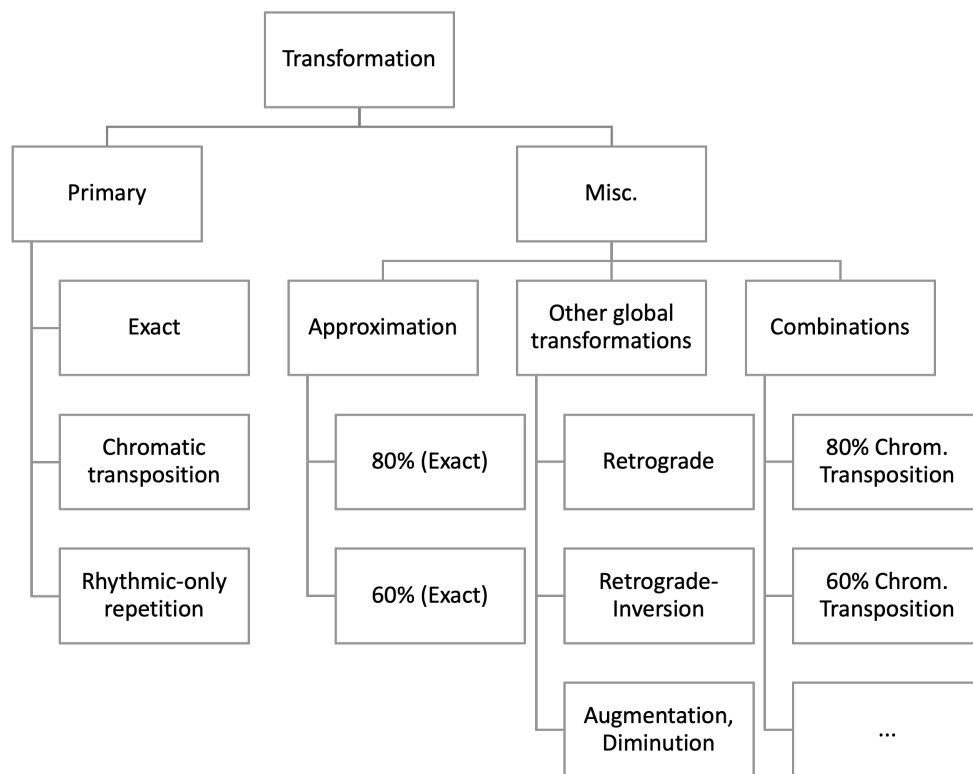


Figure 5.4: The primary and miscellaneous groups of transformations we implement in Pattrans. Chrom. is short for chromatic transposition. We do not list all the combinations such as 80% augmentation.

could be used to rank transformations, thereby enabling us to choose between  $t_1$  and  $t_2$ . In Pattrans, we take the transformation in a certain order (See Section 5.3).

If we allow ourselves to multiple matching and match pattern occurrences with the compositions of transformations, e.g.  $t_1(t_2(t_3))$ , we also need to additionally consider properties of these compositions such as associativity, commutativity, and potential equivalences between them. These are the situations we are not considering for this dissertation.

### Prototype patterns

In Pattrans, we choose the first pattern that occurred in time to be the prototype occurrence. This is not an unfounded choice: as we discussed in Chapter 2, in music, the patterns that appear earlier in time in a piece have more chance to be repeated and varied in the rest of the piece. A temporal discount factor seems to be applicable when considering how likely a musical excerpt can be a prototype pattern.

This is not always the case, however, and selecting a prototype pattern is a difficult problem: one must know which occurrence in a set is the prototype in order to test the membership of other candidate occurrences, but one must also consider

all occurrences in order to determine which is the prototype. In addition to treating the first occurrence as the prototype and all other occurrences have a flat structure, there might be a subset of prototypes, as well as different branches of how the prototypes repeat and evolve throughout the piece. Pattrans does not currently take this possibility into account.

### **Complexity in approximation and other transformations**

In addition to the computational complexity that we discussed, the codes that implement the approximation transformations are rather complex conceptually. There are hardly easy ways to communicate to the machine what we see as approximations—the local variation such as the insertions and deletions of notes, which are actually fraught with possibilities and complexities, despite their simple appearance at first sight. We do not claim that our implementation of approximation is optimal, and future research is needed to untangle this issue.

Other transformations, however, are selected with intuitiveness in mind, in the hope of reducing complexity. As we discussed in Chapter 4 and 5, there is a need for framing information coming out of computational methods in a more self-explanatory and understandable way to users. Meaningful interactions with the computations hinge on exchanging communicable results and making users less prone to misinterpretation. Our DSL, by employing modular transformations and operators to compose these transformations, is created by placing users at the centre of the design, which can be improved with multiple future iterations.

### **Transformations that were not considered**

A variety of other transformations exist, in addition to the Pattrans ones. As we have explained in this chapter and established in the previous chapter, our criteria for implementing the primary and misc. transformations include how intuitive and how common these transformations are. As an initial step that starts to look into the connections between patterns and transformations, we do not aim to be comprehensive. Instead, one of the advantages of Pattrans is that it is easy to add new transformations and combine them with existing ones. When used in combination with a variety of corpora, the list of the transformations will undoubtedly expand and even be learned from data, which is beyond the reach of this dissertation but still relevant for future study.

### **An excursion into category theory**

In *Pattrans*, with the concept of contravariant bifunctor and monoids, we enter the territory of category theory. Category theory has been used to model phenomena in many areas we have discussed in previous chapters, such as cognitive science, machine learning, and FP (Arjonilla & Ogata, 2017; Fong *et al.*, 2019). The central ideas are objects and morphisms, which can be intuitively understood as states and the processes or connections between states. This generalised formulation gives rise to an extensive network of analogies and relationships between fundamental subjects such as physics, topology, logic, and computation (Baez & Stay, 2010).

Category theory has also been used to model music structure. For example, in extending Lewin's transformational networks, Rahn (2004) provides the insight that transformational networks are essentially graphs which have arrows labelled in some semigroup, as well as nodes in some set acted on by the semigroup. In this way, the network can be extended in category theory. More work, such as (Andreatta, 2018; Giesa *et al.*, 2011; Mannone, 2018), has used category theory to model a variety of musical activities and made interdisciplinary connections. To stay inside the scope, we do not make more connections to category theory in this dissertation other than the concept of contravariant bifunctor and monoids used in *Pattrans*.





# Chapter 6 Using Transformations to Understand the Relations Between Pattern Occurrences

*Music is the pleasure the human mind experiences from counting without being aware that it is counting.*

– Gottfried Wilhelm Leibniz

## 6.1 Introduction

### Summary so far

In the earlier chapters, we have explored the significance of musical patterns and how computational algorithms are used to uncover them. Such musical patterns are widely discussed in different contexts, such as the concrete MIR tasks of compression (Meredith, 2013), classification (Lin *et al.*, 2004), and segmentation (Nieto & Farbood, 2014), or the broader contexts of psychology (Foubert *et al.*, 2017), musicology (Janssen, 2018), and education (Harkleroad, 2006). However, due to the diverse contexts in which they are used, there are various definitions, examples, and criteria related to musical patterns, as we have seen in Chapter 2. Moreover, there are discrepancies between what humans annotate as a musical pattern across different contexts, as we have seen in Chapter 3. Consequently, designing automated computational systems that can extract the desirable musical patterns is a challenging task, and comparing the output of these systems with both the output of other systems and human annotations poses significant challenges, as we have seen in Chapter 4.

More concretely, comparing the output from algorithms that automatically discover musical patterns presents the following challenges:

- For the same musical piece, patterns and their occurrences discovered by different annotators may differ, which leads to difficulties both in comparing algorithms with human annotations and in comparing the human annotations with each other.
- The size of algorithmic outputs often tends to be large, and relating these outputs back to meaningful musical concepts can be challenging, which leads to difficulties in the comparison process and the selection process of pattern discovery algorithms.

In Chapter 5, we established a connection between musical patterns and transformations. In a nutshell, we used musical transformations to account for the variation between pattern occurrences. We implemented a variety of transformation checkers in Haskell to facilitate the comparison of musical patterns. In this chapter, our aim is to address the two aforementioned challenges by using musical transformations to compare human annotations and pattern discovery algorithms.

Before we proceed, it is important to revisit and clarify the terms **pattern** and **pattern occurrences**. In Section 1.5.4, we showed the sample output of algorithms. The output is structured in two levels: patterns constitute the first level, and their occurrences form the second. Essentially, the pattern occurrences are grouped into sets that are linked to a pattern on the first level. In Chapter 4, we discussed the large number of output **patterns**. In this chapter, attention is directed towards **pattern occurrences**. A large number of pattern occurrences can be attributed to several scenarios: it may be that there are many different patterns, each with few occurrences; it may be that there are a smaller number of patterns, but each with many occurrences; or it may be a combination thereof. Depending on the different algorithms and datasets being examined, the scenario could be different. For both human annotators and algorithms, the number of patterns is always smaller than the number of pattern occurrences. In fact, the number of pattern occurrences would be at least twice the number of patterns if the definition "each pattern repeats at least twice" is adopted. In the scenario of human annotators, a lot of time and concentration is needed for human annotators to annotate all occurrences of one pattern, so annotators sometimes do not annotate all occurrences (Ren, Koops, *et al.*, 2018). If an algorithm tends to identify more relations between occurrences than another algorithm, essentially exhibiting a more liberal interpretation of repetition, it would generally lead to a greater quantity of both patterns and pattern occurrences.

### Research questions and main contributions

The thesis statement of this dissertation is as follows:

human-perceived musical patterns, a type of highly subjective and ubiquitous abstraction, have important connections to musical transformations, which, when explored compositionally through a functional programming language, informs both human-to-algorithm and algorithm-to-algorithm comparisons of discovered musical patterns.

In accordance with this thesis statement, we will use the transformations in the *Pattrans* package introduced in Chapter 5 to relate musical pattern occurrences. More specifically, we use a set of computationally well-defined and commonly used musical transformations to analyse pattern occurrences. As a result, we can retrieve and discriminate between patterns with occurrences related by different transformations. To summarise the presence of those different transformations found in a set of musical pattern occurrences, we use the proportion of each transformation found in this set of occurrences. By calculating the proportion of the transformations, we can distil the relations between pattern occurrences into a normalised vector of numbers, which can help us further inspect, classify, and thus understand pattern occurrences more numerically, from the perspective of musical transformations.

In connection to the two challenges posed at the beginning of this chapter, we aim to answer the following research questions:

- Given sets of human-annotated patterns, can we compare and group the pattern occurrences of these patterns using *Pattrans*? Which transformations are more commonly seen in human annotations than others? The answers to these questions will give us a better understanding of how human-annotated pattern occurrences are related to one another, thereby addressing the first challenge for this chapter.
- Given the potentially large output size of algorithms, are the musical transformations able to capture the occurrence relations in musical patterns? Are these relations similar or different to human-annotated patterns? The answers to these questions will aid us in better understanding what kind of transformations the algorithms discover, whether the large output from algorithms only correspond to a much smaller number of transformations, and, finally, how they compare to human annotations, thereby addressing the second challenge for this chapter.

Correspondingly, by using the *Pattrans* library described in Chapter 5, we address the two challenges posed at the beginning of this chapter, and our main contributions demonstrate the following:

- A large proportion of the pattern occurrence relations in human annotations can be explained by the transformations we consider, thereby establishing the importance of such transformations in human-annotated patterns.
- In the JKU-PDD and MTC-ANN datasets, the human annotations exhibit higher percentages of exact repetition, whereas the algorithmic ones tend to have lower percentages. Nonetheless, depending on the specific algorithm in consideration, the proportions of exact repetition may not significantly differ from those of human annotations.

### **Analysing transformation data in subsequent sections of this chapter**

In the body of this chapter, we will first introduce the music data and the pattern discovery algorithms we use. Following that, we will compare different sets of human annotations pattern occurrences using transformations. Next, we will use six pattern discovery algorithms and examine the output pattern occurrences of these algorithms, employing transformations in a similar fashion. With these steps, we will highlight the primary and misc. transformations introduced in Section 5.3. Primary transformations are exact repetition, chromatic transposition, and rhythmic-only repetition. Miscellaneous (misc.) transformations contain the following transformations and their compositions: retrograde, augmentation, diminution, and approximation. Other potential non-primary, non-misc. transformations and relations are referred to as "unclassified". The transformations will be checked in this order as we described them in this paragraph (see more discussion about the order of checking transformation in Section 5.3 and 5.4).

Following these comparisons, we will conduct statistical analysis to examine the significance of the specific transformation of exact repetition in our comparisons. In parallel to the statistical analysis, we will employ a separate process where the transformations are used to create new features. These features will then be visualised through PCA for visual examination, thereby making the proportions of transformations more examinable and yielding more insights.

## **6.2 Data and background**

We begin by introducing more in detail the two sub-problems we will be tackling, as well as the datasets we will be using. As there are overlaps in datasets across chapters, we do not reiterate all the details here, but describe each dataset in more detail in Appendix A.

## Human annotations

We have seen that datasets of human-annotated musical patterns are rare, and agreement amongst human annotators is rarer still. These disagreements can be attributed to ambiguity in the musical material, subjectivity of the annotators, different annotation protocols, and different annotation tools used. As a result, controversies remain when it comes to using human annotations for musical pattern discovery research, and other MIR tasks. In order to study the differences and commonalities between different annotators, we can now employ primary and misc. transformations to compare different pattern occurrences by grouping these according to their relations, which sheds light on the potential criteria the annotators have used. Alongside the [JKU-PDD](#) and the [MTC-ANN](#) datasets, we use the dataset collected by ANOMIC for this purpose. We will call this dataset the HEMANAll dataset in this section because the musical pieces used are identical to the initial HEMAN experiment. We divide this dataset into two sub-datasets based on the annotators' musical backgrounds: HEMANHigh (with a high self-rated musical background) and HEMANLow (with a low self-rated musical background). The datasets HEMANAll, HEMANHigh, and HEMANLow are collectively referred to as the HEMAN datasets. In comparing JKU-PDD, MTC-ANN, and the HEMAN datasets, we compare different corpora with distinct underlying annotation processes to obtain an overview of what transformations we can observe in human annotations while ignoring the annotators' backgrounds or the specific annotation procedures. In comparing within the HEMAN datasets, we compare groups of annotators with a different musical background for the same set of musical pieces and with the same annotation procedure. This comparison may give insight into the effects of annotators' different musical backgrounds on the proportions of different types of musical transformation they annotated as pattern occurrences.

In terms of the musical pieces in the datasets, JKU-PDD contains 5 musical pieces by Bach, Beethoven, Chopin, Gibbons, and Mozart; MTC-ANN is a collection of Dutch folk songs; HEMAN datasets contain 6 pieces by Bach, Beethoven, Haydn, and Mozart. Please see [Appendix A](#) for more details about each dataset.

The JKU-PDD and MTC-ANN datasets are highlighted in the comparison between algorithmically discovered patterns and human-annotated patterns. We highlight these datasets due to their widespread use in the field of musical pattern discovery. Furthermore, only the JKU-PDD dataset is used to showcase two other methods to examine the transformation proportions, i.e. through statistical testing and PCA. While other datasets may warrant closer scrutiny in future research, we currently

focus on these datasets for the examination of multiple algorithms as described below.

## Algorithms

As previously discussed, datasets of human-annotated patterns are scarce, while there is potentially an abundance of output produced by musical pattern discovery algorithms. Persistent challenges with evaluating and deploying such algorithms exist: the output size of algorithms can be large, which are costly to examine manually; implementation logic can be hard to comprehend, which could be caused by any of the numerous procedures that comprise the algorithm, or by a binary-only release for which we only have access to the output; patterns extracted by different algorithms can vary significantly given the same input, which creates controversy when it comes to using human annotations as ground truth and designing an all-encompassing evaluation strategy. The transformations provide us with a lens through which to examine the output using musically relevant concepts, therefore enabling straightforward understanding and comparison of the relations between pattern occurrences.

For investigating patterns from the algorithms, we use the six pattern discovery algorithms submitted to the [MIREX](#) task during 2014-2017<sup>1</sup>. Each algorithm has its own series of procedures to extract patterns from music, e.g. clustering, matching, and counting, and so forth. The number of pattern occurrences extracted varies across algorithms and corpora. We run the following algorithms, introduced in Chapter 2 and 4, on the JKU-PDD and MTC-ANN datasets, with the numbers in parentheses denoting the extracted pattern occurrences from each dataset, in that order: SYMCHM (59, 229), SIAF1 (794, 4486), SIACR (1590, 4683), SIACP (399, 3400), VM1 (2969, 25497), VM2 (463, 4599), and for comparison, human annotations (105, 1546).

## 6.3 Transformations in human annotations

### Observation

By employing the Pattrans package, we can effectively quantify the relations between patterns occurrences in terms of musical transformations. This quantification allows us to make the following observations.

---

<sup>1</sup>This task did not get new submissions after 2018 and changed focus to creating a new task—Patterns for Prediction.

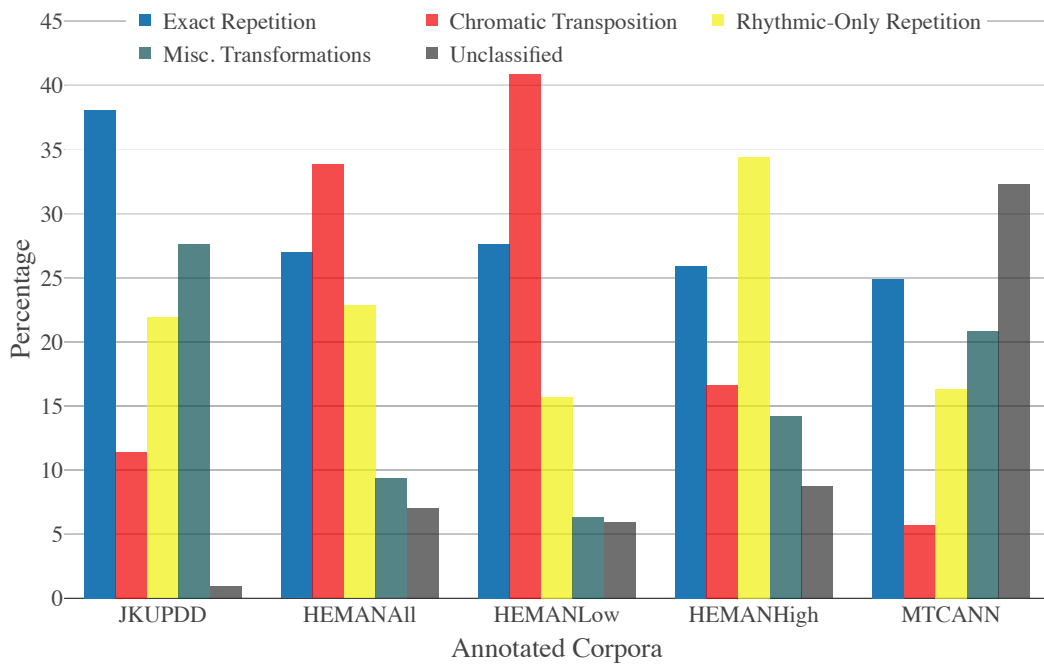


Figure 6.1: Transformations found in the human-annotated patterns in 3 datasets and 2 sub-datasets. The colours correspond to different transformations. The primary and misc. transformations are introduced in Section 5.3. Primary transformations are exact repetition, chromatic transposition, and rhythmic-only repetition. Misc. transformations contain the following transformations and their compositions: retrograde, augmentation, diminution, and approximation. All other occurrences which we could not match with any transformations nor approximations are grouped into "unclassified".

In Figure 6.1, looking across the datasets, we observe a substantial proportion of primary transformations, with exact repetition being consistently prominent. In addition, we notice varying proportions of misc. transformation and unclassified instances. Chromatic transposition and rhythmic-only repetition are notably prevalent in certain datasets.

Exact repetition, chromatic transposition, and rhythmic repetition together make up the majority (71.4% for JKU-PDD, 83.6% for HEMANAll, 87.7% for HEMANLow, 77.0% for HEMANHigh) of the pattern occurrence relations, with exception of MTC-ANN (46.9%)—although it does have a percentage close to 50%. The MTC-ANN dataset also has the most unclassified occurrence relations.

The JKU-PDD dataset exhibits the highest proportion of exact repetition amongst all datasets, with virtually no unclassified instances present. Both HEMANAll and HEMANLow feature a high proportion of chromatic transposition, while HEMANHigh is characterised by a significant proportion of rhythm-only repetition. The proportions of misc. transformations and unclassified instances are relatively low in the HEMANAll, HEMANLow, and HEMANHigh datasets.

## Implication

The fact that we see a substantial proportion of primary transformations is supporting evidence that the primary transformations are of major importance to human-annotated patterns. Additionally, as the proportion of unclassified relations (occurrences that could not be related with the transformations in Pattrans) tends to be small, and the proportions of those relations classifiable by transformations are above 70% for all datasets, this is a positively validating result for the use of our transformation approach.

In the case of MTC-ANN, we have seen that pattern occurrences within the dataset can be short, leading to a situation where even small alterations in the short patterns may cause the occurrence relation to become unidentifiable. This situation is an indication that recovering the human-annotated patterns using an algorithm in this dataset is difficult.

The annotations in the HEMAN dataset, as described in Chapter 3, are gathered using a tool with automatic exact repetition and chromatic transposition tagging. A potential side effect is that we see a higher proportion of chromatic transpositions in HEMANAll than other datasets. This raises awareness of the possible side effects caused by providing the functionality of automatic repetition tagging in annotation tools. Annotators with higher self-rated musical background scores, in HEMAN-High, had a higher proportion of rhythmic-only repetition, which is an indication that rhythmic-only repetitions are more prominent as musical patterns for the annotators with more musical backgrounds.

## 6.4 Transformations in algorithmic output

### Observation JKU-PDD

In a similar manner to human annotations, we can employ Pattrans on the pattern occurrences extracted algorithmically, and compare the output of six algorithms with the human annotations of JKU-PDD. This allows us to make the following observations.

In Figure 6.2, we observe different combinations of proportions of transformations. More specifically, we make the observation that although most of the algorithms retrieve smaller proportions of exact repetition than in the patterns annotated by humans, all algorithms extract a non-zero number of exact repetition.

SYMCHM is the only algorithm with more exact repetition and chromatic transposition in the proportion of the extracted pattern occurrence relations. However,



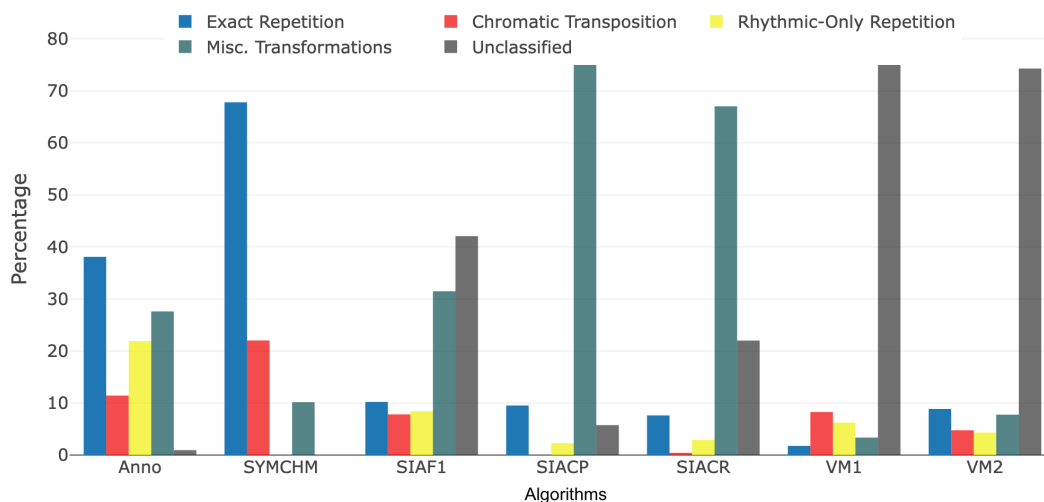


Figure 6.2: Transformations found in human-annotated and algorithmically extracted patterns in the JKU-PDD dataset. Using the same specifications as Figure 6.1. The number of pattern occurrences extracted by each algorithm is as follows: SYMCHM—59, SIAF1—794, SIACR—1590, SIACP—399, VM1—2969, VM2—463, Anno—105.

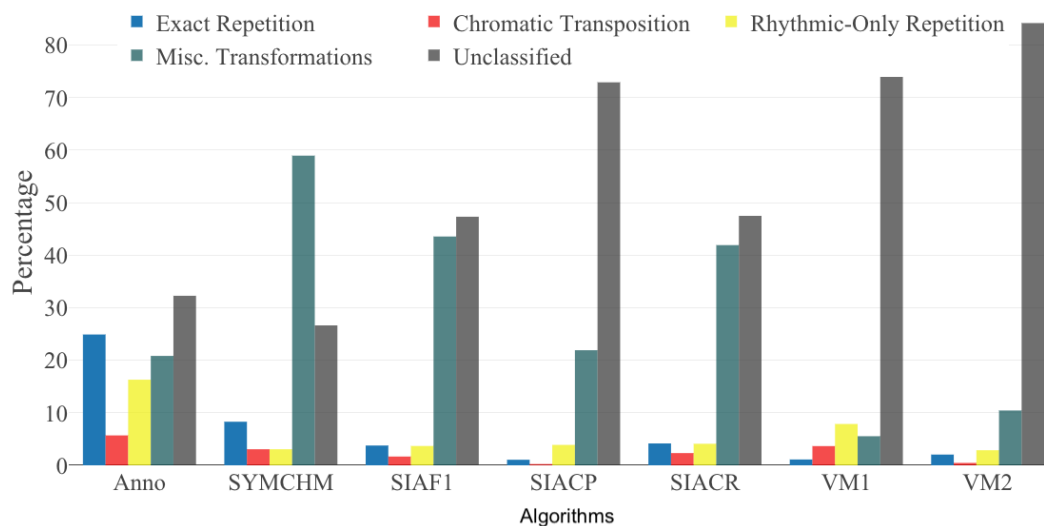


Figure 6.3: Transformations found in human-annotated and algorithmically extracted patterns in the MTC-ANN dataset. Using the same specifications as Figure 6.1. The number of pattern occurrences extracted by each algorithm is as follows: SYMCHM—229, SIAF1—4486, SIACR—4683, SIACP—3400, VM1—25497, VM2—4599, Anno—1546.

SYMCHM does not extract patterns with rhythmic repetition. VM1 and VM2, belonging to a family of wavelet-based pattern discovery algorithms, have the highest number of unclassified pattern relations. SIACP and SIACR have higher proportions of misc. transformations compared to other algorithms.

With consideration of the background on the algorithms and human annotations, we will now discuss additional observations in more detail. In the JKU-PDD dataset, human-annotated pattern occurrences are mostly related to each other by straightforward exact repetition and chromatic transposition. With the exception of SYMCHM, the algorithms extract pattern occurrences with a lower proportion of exact repetition than in human annotations. The SYMCHM algorithm employs a machine learning approach. Transformations are not explicitly encoded in the algorithm. Nevertheless, we observe that a large proportion of the output can be classified into different transformations, which indicates that the algorithm is able to learn these transformations. In this way, we interpret the output of the algorithm using domain-specific concepts (i.e. musical transformations), which have valuable potential to phenomenologically verify and interpret a complicated algorithm in a multifaceted way. The fact that SYMCHM does not extract rhythmic-only occurrences aligns with the observations in (Ren, Volk, *et al.*, 2018) and Chapter 4, which has shown that the rhythmic features of this set of pattern occurrences are considerably different from those found in other algorithms and human annotations.

In VM1 and VM2, a clustering algorithm was used as the final step to group the pattern occurrences. The large number of unclassified relations can be linked to this clustering step, whereby the distance metric used in the clustering algorithm may capture relations that cannot be matched to the transformations we considered.

SIACP, SIACR, and SIAF1 belong to the geometric family of pattern discovery algorithms. The names of the algorithms correspond to different parameter settings concerning the maximisation of the recall (SIACR), precision (SIACP), and F-1 score (SIAF1). We see that patterns discovered by the three algorithms have occurrence relations mostly in the category of misc. transformation, where there is a diverse combination of transformations and approximations. Notably, chromatic transpositions are not present in SIACP and barely present in SIACR. This presence or lack thereof could be an artefact resulting from the combined effects of the filtering steps in the algorithms.

When considering the absolute numbers of pattern occurrences, the majority of algorithms consistently identify at least three times as many pattern occurrences as humans do. The SYMCHM algorithm detects fewer pattern occurrences, approximately 60% of those identified by humans. This leads us to consider SYMCHM is

learning fewer transformations, thereby resulting in a higher proportions of detecting exact repetitions.

### Observation MTC-ANN

We extend our method to the MTC-ANN dataset—we use Pattrans on the Dutch folk songs in the MTC-ANN dataset and compare the output of six algorithms with the human annotations. Similarly to what we observe in Figure 6.3 in the previous dataset, all algorithms except SYMCHM retrieve smaller proportions of exact repetition than those found in human annotations. All algorithms extract a non-zero quantity of exact repetition. All algorithms have non-zero values for proportions of rhythmic-only repetitions, with the highest proportion found in human annotations. Fewer than 40% of the human-annotated pattern occurrences cannot be accounted for by any transformation we considered.

SYMCHM shows higher proportions of misc. and unclassified transformations compared to its results for the JKU-PDD dataset. However, SYMCHM’s proportion of exact repetition remains the highest amongst the algorithms. Small proportions of chromatic and rhythmic-only repetitions are discovered. Despite not factoring in rhythmic information, SYMCHM may have discovered other regularities in pitch that coincided with rhythmic repetition, contributing to a non-zero proportion of rhythmic-only repetitions.

VM1 and VM2 yield similar large proportions of unclassified transformations as with JKU-PDD. This could be again the effects from the clustering step of the algorithms.

SIACP, SIACR, and SIAF1 demonstrate less unclassified transformations compared to their JKU-PDD results. SIACP and SIACR exhibit a higher proportion of misc. transformations, while SIAF1 maintains a similar proportion compared to their JKU-PDD results. As discussed in Section 6.3, pattern occurrences within the MTC-ANN dataset can be short. Even minor alterations in these short patterns can render the occurrence relation unidentifiable or only relatable by transformations that permit approximation, falling into the misc. transformation category.

In terms of the absolute numbers of pattern occurrences, similar to the results obtained from the JKU-PDD dataset, most algorithms consistently detect at least twice as many pattern occurrences as humans do. The SYMCHM algorithm finds fewer pattern occurrences, roughly 15% of the pattern occurrences that humans identify. Furthermore, in light of our analysis of the results for JKU-PDD, although we are still not sure how many transformations this algorithm is learning, it undeniably extracts pattern occurrences with relations that are not covered by the transformations

we consider, and can have a lower proportion of exact repetition compared to what we observed in JKU-PDD.

In summary, Pattrans applied to the MTC-ANN dataset offers further evidence that, apart from SYMCHM, other algorithms tend to discover more patterns that contain misc. transformations, and the transformations we considered are less capable of covering their occurrence relations. We also see the distinct composition of transformations for each algorithm (family) and reveal the varying proportions of transformations within human-annotated musical patterns. Such insights serve to add further depth to the understanding of the nuanced diversity of musical pattern discovery, as we discuss further below.

### **Implication JKU-PDD and MTC-ANN, in combination with previous results**

Using transformations to detect differences between algorithms provides us with further insights about the potential applications of transformations and algorithms in varying scenarios. The transformations can serve as a filtering technique for pattern output, allowing for the removal of those patterns that are irrelevant to the desired transformations, if desired. To be more precise, one can enhance the occurrence precision score by filtering out a specific pattern occurrence when the underlying transformation is irrelevant. Similarly, by removing an entire pattern when all transformations between its occurrences are deemed irrelevant, one can improve the establishment precision score. Another potential application could involve the selection of algorithms capable of identifying patterns with varying degrees of occurrence relation proportions that resemble human annotations. With this application, it is important to acknowledge that, depending on the context and purpose, human annotations are gathered using different strategies and protocols as well, as discussed in Chapter 3 and 4. For example, annotated patterns from a musicologist to analyse the structural elements of music are likely to be different from the patterns found by a music therapist for informing their suggestions (Foubert *et al.*, 2017) or music teachers for pointing out pedagogically interesting passages. In these specific cases, having a perspective on what the relations are between the occurrences can help with choosing one or a combination of algorithms. If the goal is to reveal hidden patterns and pattern occurrence relations for the task of compression, wavelet and geometric-based algorithms are more likely to retrieve pattern occurrences with relations that have a lower percentage of exact repetition than SYMCHM. If the application's goal is to retrieve patterns for composing new music, which was attempted by Herremans and Chew (2017), the choice of the algorithms can be based upon the envisioned relations between potential pattern occurrences in the music.

Worth noting is that, although the percentages of transformations in the SYMCHM algorithm may be more similar to those identified by human annotators compared to other algorithms, the actual musical patterns may differ. Additionally, while the SYMCHM algorithm yields a manageable number of pattern occurrences in JKU-PDD and MTC-ANN datasets, it misses approximately 40% and 85% of occurrences in JKU-PDD and MTC-ANN respectively. Relying solely on the percentages of transformations may not provide a complete picture of the human-annotated and automatically discovered musical patterns. At the same time, incorporating musical transformations and Pattrans in conjunction with established measures in MIR may facilitate a more thorough comparison of musical patterns annotated by humans and extracted by algorithms. We will illustrate this final point with the following paragraph.

Building upon our discussion in Chapter 4, where we examined the extracted patterns by algorithms and datasets used in this chapter, we further discuss the connections and implications of our findings below. In Section 4.2.2, we visualised the locations of pattern occurrences. Although there were instances where the algorithmically extracted pattern occurrences overlapped with human annotations in the song from the JKU-PDD dataset, a significant number did not match human annotations. Our transformation approach allows for more insights into how these unmatched pattern occurrences relate to each other in terms of music transformations. In Section 4.2.4, we examined the pattern occurrences in their feature space, focusing on the actual notes within the patterns occurrences, as opposed to their relations with each other. This examination revealed a long-tail distribution of the pattern occurrences from VM1, VM2, SIAF1, SIACR, and SIACP. These occurrences are likely to be related by the misc. and unclassified transformations, given the large proportions of these transformations.

Further implications can be obtained by juxtaposing our findings with the results from MIREX. As introduced in Section 4.1, two key MIREX metric sets are the establishment metrics—assessing an algorithm’s ability to establish that a pattern is repeated at least once during a piece—and the occurrence metrics, gauging whether an algorithm can retrieve all occurrences of a pattern. While VM1 excels in establishment metrics and also performs well in occurrence recall, it scores low on occurrence precision. These results from MIREX reinforce the validity of our results: low precision suggests that VM1 tends to find a larger number of occurrences beyond those identified by humans, possibly those occurrences whose relations cannot be explained by the transformations we considered. MIREX results further add to our findings: VM1’s high scores in occurrence recall indicate that the algorithm detects many of the occurrences that are also identified by humans. SYMCHM scores

low on establishment metrics, which aligns with our previous observation about the low absolute number of pattern occurrences it discovers. SYMCHM's occurrence precision scores tend to outperform VM1, as SYMCHM tends not to find an excessive number of occurrences beyond those identified by humans, which corroborates our observation that the percentages of transformations are closer to those of humans. Moreover, since no algorithm achieved perfect scores in MIREX, we can alleviate our concerns about the possibility that the algorithms have covered the human-annotated patterns and their occurrences perfectly, and have added other occurrences to the established or not-yet-established patterns in addition to the perfect coverage. Without additional supporting evidence, it could be true that the algorithms discover all human annotations, given that algorithms produce a greater number of pattern occurrences; furthermore, despite lower proportions of exact repetitions, the algorithms can still cover the exact repetitions favoured by humans. However, in conjunction with the insights from the earlier sections, it becomes apparent that these algorithms do not fully recover all human annotations and append additional patterns to them.

### **Considerations on the absolute numbers of pattern occurrences**

It is fair to question the validity of comparing percentages when there is a large difference in the total number of detected pattern occurrences between human annotations and algorithms, as well as amongst algorithms themselves. Even under these circumstances, comparing percentages can offer valuable insights, as they provide a relative perspective, which can reveal insights that may not be obvious in the absolute numbers. However, the use of percentages does not always provide a complete picture. Hence, in our analysis, we also took into account the absolute numbers and incorporated insights from prior chapters to ensure the validity of our analysis.

## **6.5 Statistical analysis**

In the previous two subsections, we compared transformations in patterns with different human annotations and algorithms. In this subsection, our objective is to compare human-annotated patterns with algorithmically extracted patterns, focusing specifically on the proportions of exact repetitions as a feature. This choice is motivated by the substantial proportions of exact repetitions observed in human annotations as well as the pivotal role it holds in music, especially in the JKU-PDD dataset. Moreover, considering the scope of the dissertation, it provides an oppor-

tunity for us to thoroughly explore and analyse the intricacies associated with this specific transformation.

The Kruskal-Wallis one-way analysis of variance (Kruskal & Wallis, 1952) is selected for this comparison because it is a non-parametric statistical test. Non-parametric tests are preferable in our comparison because they do not rely on assumptions about the underlying distribution of the data or the parameters of the distribution, such as normality of residuals in the samples. Moreover, the test is designed to determine whether samples originate from the same distribution or from statistically significantly different distributions across multiple groups. Such characteristics make the Kruskal-Wallis test well-suited for our purpose.

This part of the analysis is conducted using the JKU-PDD dataset. First, we compare all human annotations to all the patterns extracted by SYMCHM, SIACP, SIACR, SIAF1, VM1, and VM2 in terms of the proportions of exact repetition and chromatic transposition between their pattern occurrences. The null hypothesis is that there is no significant group difference between the algorithmically extracted patterns and the human-annotated patterns in their proportions of exact repetition, and with the significance level  $\alpha = 0.05$ <sup>2</sup>. Subsequently, we examine the individual algorithms SYMCHM, SIAF1, and VM1<sup>3</sup> using the proportions of exact repetition, comparing each algorithm separately to the human annotations. The null hypothesis is that there is no significant difference between the algorithmically extracted patterns by SYMCHM, SIAF1, and VM1, and the human-annotated patterns, with the same significance level  $\alpha = 0.05$ .

## Observation

When comparing the proportions of exact repetitions in all human annotations to those in all algorithmically extracted patterns, we obtain a p-value  $\ll 0.05$ . Assuming the null hypothesis is true, which suggests no significant difference between the algorithmically extracted patterns and the human-annotated patterns in their proportions of exact repetition, the small p-value tells us that the probability of obtaining a difference at least as large as the one we observed in the sample data is much smaller than 0.05. Given the sufficiently small p-value, we reject the null hypothesis. This suggests that there is evidence of a significant difference between the two groups.

---

<sup>2</sup>The 0.05 significance level is a widely used threshold for determining statistical significance and is therefore used in our study.

<sup>3</sup>These algorithms are chosen because they represent different types of algorithms, and as we will see later in Figure 6.4, their distributions are also representative.

Given that the result is significant for exact repetitions, it is reasonable to further investigate the performance of each individual algorithm in terms of exact repetitions to determine if the results hold consistently across each algorithm. Figure 6.4 presents the results for exact repetition through a boxplot, showing the outcomes of the Kruskal-Wallis performed in three folds (see three p-values on the upper part of the figure: SYMCHM—0.89, SIAF1— $5.7e-10$ , VM1—0.35). These tests, similar to the one-way test performed earlier in this section, were conducted to compare the algorithmic patterns<sup>4</sup> and human annotations<sup>5</sup>. We can see that SYMCHM and VM1 have a non-significant p-value (0.89 and 0.35 respectively) while SIAF1's p-value ( $5.7e-10$ ) is significant (null hypothesis rejected). In other words, the output from SIAF1 exhibits a significantly different proportion of exact repetition compared to human annotations, while the differences between the other two algorithms (SYMCHM and VM1) and human annotations are not statistically significant when using a significance level of  $\alpha = 0.05$ . This exposes the individual differences between the algorithms.

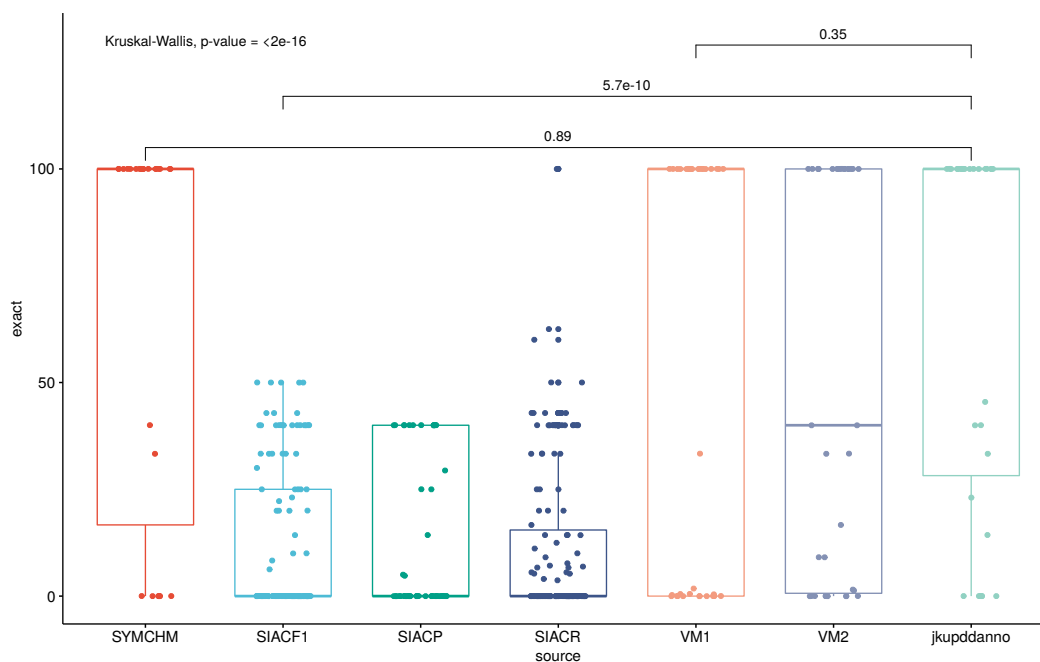


Figure 6.4: Boxplot and Kruskal-Wallis test results comparing proportions of exact repetition in human annotations (jkupddanno) and extracted by algorithms (SYMCHM, VM1, VM2, SIAF1, SIACR, SIACP) in the JKU-PDD dataset. Each point represents the percentage of exact repetition in a pattern.

<sup>4</sup>These include algorithms SYMCHM, SIAF1, and VM1. Other algorithms were not explicitly tested, as they belong to the same algorithmic family as these three and exhibit similar boxplot distributions.

<sup>5</sup>Human-annotations are indicated by the label "jkupddanno".



## Implication

The differing proportions of transformations can inform the future design of pattern discovery algorithms. If the authors of these algorithms wish to imitate the pattern discovery behaviours of human annotators, they may report first the pattern occurrences that have musicological and music-theoretic support, e.g. those that have simple musical transformations between the two occurrences. The purpose of the algorithms, however, is not confined to the replication of the reference data.

More generally, calculating and comparing the proportions of transformations can serve as an extra step to answer questions about algorithms, such as: does the algorithm discover simple patterns in a way that can be explained via the musical concepts of transformation? Or, is the algorithm trying to be more "innovative" in the patterns it discovers?

Given the importance of these transformations, in the next section, we propose a set of new features based on the proportions of transformations, which we call the *transformation profile* (TPr). We then use PCA on these TPrs to further compare how the human-annotated and algorithmically extracted patterns differ in this transformation feature space. In this way, we can use PCA to reduce the number of the transformations and examine how each pattern's occurrences compare with those of another pattern's occurrences.

## 6.6 A transformation profile for each pattern

As transformations characterise the relations between pattern occurrences, we can use the proportion of each transformation to describe patterns. More specifically, we can create a feature vector based on how often a transformation appears in the occurrence set.

For each pattern, we define a *transformation profile* (TPr) from the proportion of different transformations:

$$\text{TPr}(\text{Pattern}) = \left\{ \frac{c(t_1)}{\sum_{i=1}^n c(t_i)}, \frac{c(t_2)}{\sum_{i=1}^n c(t_i)}, \dots, \frac{c(t_n)}{\sum_{i=1}^n c(t_i)} \right\} \quad (6.1)$$

where  $t_i$  is a certain transformation indexed by  $i$ ,  $c(t_i)$  is the count of a certain transformation, and  $n$  is the number of different transformations. The denominator is

essentially the count of all occurrences if we take into account the "unclassified" as a type of unknown transformation.

The feature vector is the numeric vector representation of the bar charts presented in Section 6.3 and 6.4, that is, the constituents of occurrence relations, as defined in Eq. (6.1). By representing the patterns using TPr, we can further investigate and compare the patterns in this feature space spanned by the proportion of exact repetition, transposition, and other transformations. By doing so, we may gain new insight into the distribution of occurrences for both human-annotated and automatically extracted patterns within this feature space, which may facilitate a better understanding of the output and enable more in-depth examination of the patterns.

Each transformation corresponds to a dimension in our TPr. The more transformations in our analysis, the greater the number of dimensions the TPrs will have. To examine a set of patterns' higher-dimensional TPrs, we can use Multi-Dimensional Scaling (MDS) methods to visualise it. As we explained and used in Chapter 4, MDS projects higher-dimensional spaces onto lower-dimensional spaces spanned by the first few principal components that come from the original PCA technique or its variations. In this visual way, we can demonstrate the relative positioning of patterns based on the transformations they contain.

In Figure 6.5, we plot each individual pattern as a data point in the PCA decomposition of the TPr feature space. The PCA is calculated based on the JKU-PDD annotations, and the algorithmic patterns are projected into the space for comparison. For each pattern, we colour and size the data point differently depending on its source: algorithmically extracted or human-annotated.

### Observation

From the calculation, the first Principal Component (PC) explains 63.6% of the variance, and the second PC explains 23.6%. The fact that the main contributions to the first and second PCs come from exact repetitions and transposition highlights the importance of simple transformations. Moreover, by examining the principal components and the graphical depictions provided by the PCA projection, we are able to obtain a better view of the collective traits of the transformations in musical patterns.

In Figure 6.5, we first notice that data points form a linear relationship in this feature space. It follows that there are correlations between the axes. This correlation is expected, as the proportions of transformations sum to 1 and are therefore not independent of each other.

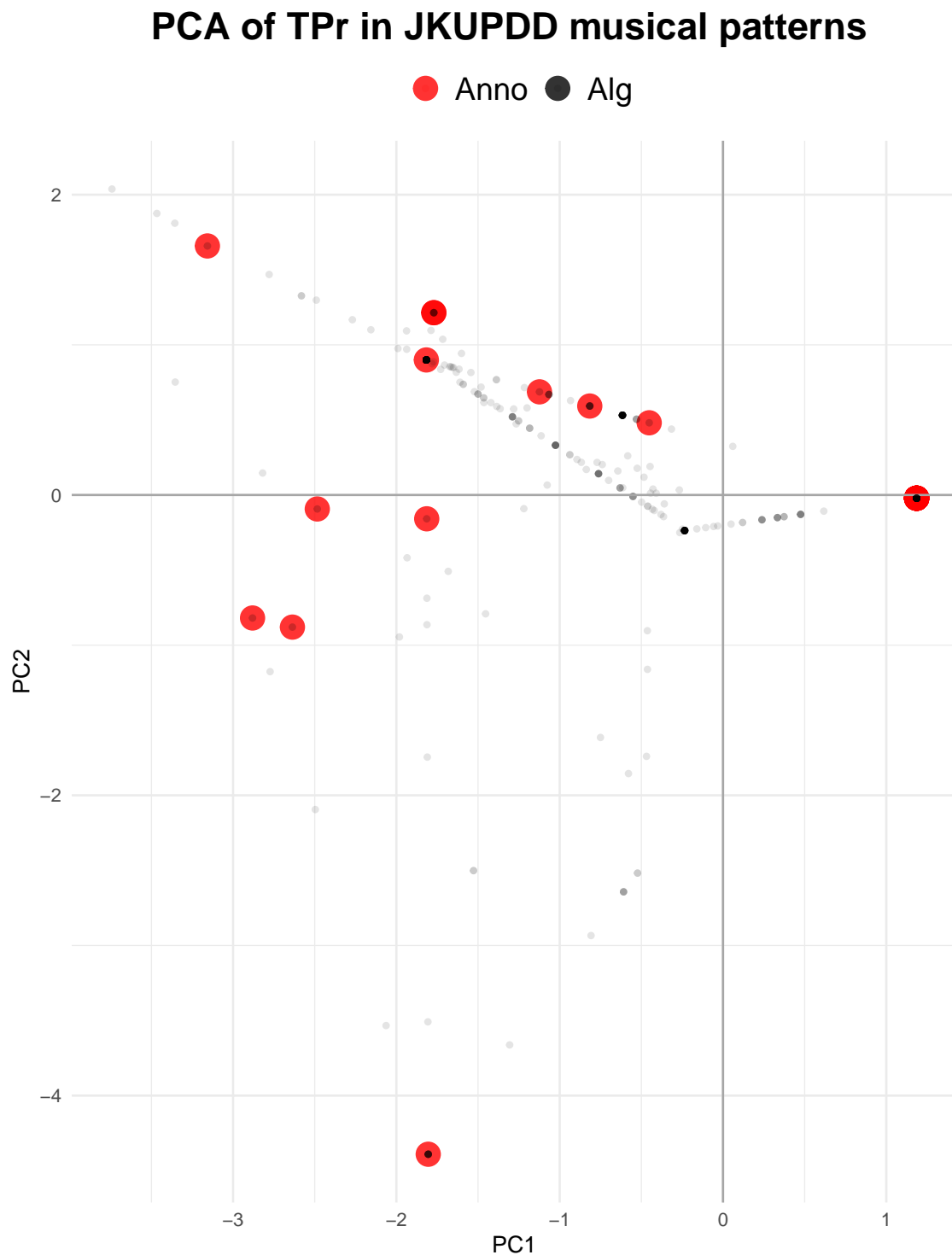


Figure 6.5: PCA visualisation of patterns' TPrs of human-annotated and algorithmic patterns in JKU-PDD. Each point represents a pattern extracted by algorithms (small black points) or annotated by humans (large red points). The highly ranked transformations in the first Principal Component (PC1) are exact repetition (37.24%), misc. transformation (29.42%), and rhythmic repetition (22.45%). For the second Principal Component (PC2), the highly ranked transformations are transposition (76.10%), and misc. transformation (15.19%).

We can further make some key observations that provide more insights for the algorithmic output, at least for the JKU-PDD dataset. Given the JKU-PDD dataset, what is immediately obvious to us is that the algorithmically discovered patterns, projected into the PCA space, tend to interpolate the anchoring human annotations. In simpler terms, the algorithmic output fills in the space between the annotations and do not form a separate cluster.

### Implication

PCA is a technique used to identify and visualise the main directions in which data points spread out. In essence, these directions, i.e. the PCs, capture the most substantial variances within the dataset. If the TPrs exhibit similar spreads, it suggests that they share similar variations.

In the context of the JKU-PDD dataset, despite many different proportions of transformations reported in Section 6.3 and 6.4, there is no separate clustering of algorithmic data points from those of human annotations in the PCA space. Therefore, the algorithmic patterns show a certain degree of alignment with human annotations in this setting—when considering individual patterns and when considering the variance of the two PCs of the transformation proportions. Note that the two main musical transformations for PC1 and PC2 are exact repetition accounting for 37.24% of PC1 and transposition accounting for 76.1% of PC2. In short, these PCA results provide support that the algorithmic output conforms to the variance of the proportions of transformations in human annotations.

Given Figure 6.2 and Figure 6.3 underscoring the differences between human and algorithmic transformation proportions, questions may arise about the seemingly contradictory results. However, there is no direct contradiction, because in Figure 6.5, we examine the patterns individually on a two-dimensional plane, as opposed to previously, where we accumulated all the patterns and compared them across five groups of transformation(s). Given the multitude of patterns and our inability to visualise in high dimensional spaces, the algorithmic points that are not overlapping with the human annotations can accumulate and contribute to the differences observed in Figure 6.2 and 6.3.

### 3D interactive visualisation

In addition to the 2D visualisation, we provide 3D interactive visualisations using multiple MDS methods such as *t-distributed Stochastic Neighbor Embedding*

(*tSNE*) and *kernel Principal Component Analysis (kPCA)*<sup>6</sup>. Figure 6.6 is a screenshot of the 3D PCA visualisation interface. For each individual pattern, we colour and mark the data point differently depending on its source: the algorithm and the musical piece. The additional third dimension allows us to have more direct access to the spread of data in the TPr feature space. By interactively engaging with the visualisation, we can examine the TPrs of individual patterns and musical pieces as desired. For instance, we can observe in Figure 6.6 that the distribution forms a shape resembling a tetrahedron, with a large range of variation along the third dimension.

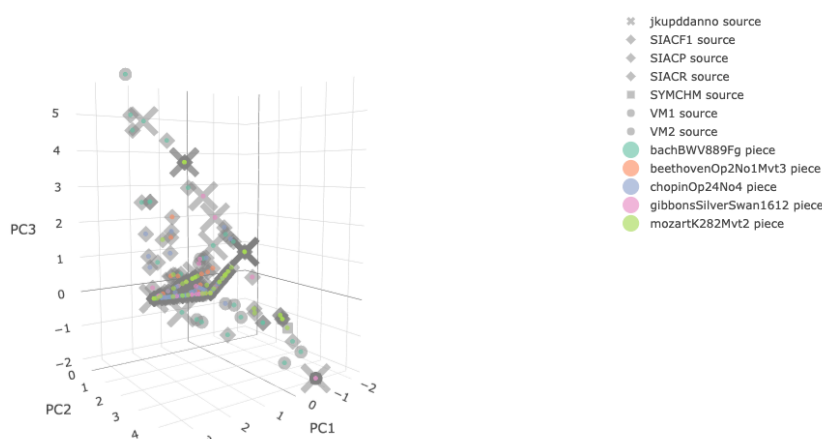


Figure 6.6: Interactive 3D PCA visualisation of the patterns' TPrs for human-annotated and algorithmic patterns in JKU-PDD.. In addition to the two dimensions in Figure 6.5, the third dimension mainly consists of rhythmic repetition (69.61%) and misc. transformation (21.03%),

## 6.7 Discussion

In this chapter, through the systematic lens of the transformations between pattern occurrences belonging to the same pattern, we compared different human annotations, and algorithmically extracted patterns. The cross-comparison of human annotations from different corpora reveals that a large proportion of the pattern occurrence relations can be explained by the transformations we consider, thereby establishing the importance of such transformations in human-annotated patterns. Each human annotation dataset exhibits unique characteristics in terms of the proportions of transformations found in their pattern occurrences: although further confirming evidence is required to draw definitive conclusions, initial findings suggest

<sup>6</sup>The visualisations can be accessed by using this URL with different endings such as *pca*, *tsne*, and *kpca*: <https://irisyupingren.github.io/research/3d/pca>

that the Dutch folk song dataset has the most misc. and unclassified transformations in proportion; the HEMANHigh and HEMANLow datasets demonstrate the highest proportions of chromatic transposition and rhythmic-only transformations, respectively.

In addition, we used six pattern discovery algorithms and showed that human annotations tend to contain higher percentages of exact repetition and other primary transformations than the algorithmically extracted patterns. More specifically, primary transformations account for a large proportion of human-annotated pattern occurrence relations; by contrast, the primary transformations only account for a small proportion of algorithmically extracted pattern occurrence relations, which tend to contain more misc. and unclassified transformations, except one algorithm (SYMCHM). We also compared the absolute numbers of pattern occurrences of algorithms and human annotations, which reveals that algorithms can also fail to retrieve the desired patterns even if the proportions of transformations are similar. While further investigation is required to establish a connection between the mechanisms of the algorithms, the human annotation process, and the proportions of transformations observed in the extracted pattern occurrences, these findings are an initial step in examining and interpreting the potentially extensive output generated by the algorithms—if a substantial percentage of the pattern occurrences can be encapsulated by a small number of transformations (e.g., exact repetition, transpositions, and rhythmic repetitions), as seen in SYMCHM and human annotations, this offers insights into the nature of these extracted and annotated occurrences in terms of the transformations. Furthermore, given the desired transformations, algorithm designers can use these transformations to filter out a specific pattern occurrence from a pattern or an entire pattern, which can improve the occurrence precision and establishment precision of the algorithms. Ultimately, if the algorithms can extract pattern occurrences with the desired transformations, an improvement in both the establishment and occurrence recall is possible.

We also utilised statistical methods to analyse the significance of the differences. One statistical test verifies the significant discrepancies between human annotations' and algorithmic output's proportions of exact repetitions in the JKU-PDD dataset. More statistical tests yield mixed results regarding the significance and non-significance of the transformations' proportions, given the different algorithms and transformations. We also devised TPr and used them in conjunction with PCA to show that, in terms of transformations, the algorithmically discovered patterns conform to human annotations. We contend that using Pattrans in tandem with statistical analysis, PCA, and potentially other established methods and measures

in MIR<sup>7</sup> lead to richer insight and a more thorough comparison between musical patterns. By doing so, our approach contributes to enhancing the evaluation step in the design of musical pattern discovery algorithms.

## 6.8 Concluding remarks

In the rest of this chapter, we will discuss a few more related concepts, implications, and limitations of our method.

### Regarding distance measures

Distance measures are of much importance in MIR research. Dual to the similarity measures we mentioned in Section 2.3.1, various distance measures have been used in MIR research: correlation distance, city block distance, Euclidean distance, Earth Mover's distance, and so on (Janssen *et al.*, 2017; Typke *et al.*, 2005).

There are interesting connections to be found between our transformation-based approach and the distance/similarity-based approach to studying musical patterns. Let  $d$  be a distance measure, and  $g$  be a transformation, and  $x$  be a musical pattern occurrence. We can then measure the distance before the transformation and after, which gives us  $d(x, g(x))$ . Depending on which distance measure is used, and whether there is a transformation involved, we may be able to use them to compare different aspects of repetition and variation to varying levels. For example, when  $d(x, g(x)) = 0$ , the distance is invariant under the transformation, or the transformation is an identity transformation— $g(x) = x$ .

### We did not provide a ranking of the algorithms

Readers might expect a final ranking of the performances of algorithms at the end of this chapter. For two reasons, we do not conclude with such a ranking. Firstly, using transformations is only one way to look at which algorithms match most closely to the human annotations, and sometimes it is even desirable for the algorithms and human annotations to have different transformations: a case-by-case analysis is more informative than an overall ranking. Secondly, we do not attempt to provide utilitarian feedback on which algorithm or transformation is more useful. Utility is context-dependent. Instead, we argue that different algorithms should be used for different purposes and in different contexts: education, MIR, and composition, just

---

<sup>7</sup>The use of Pattrans in conjunction with established measure in MIR, such as the MIREX measures, was not experimented with in this chapter due to its scope, but its use may be a valuable approach to consider in future studies.

to name a few. Instead of using a single metric such as *accuracy* or *cross-entropy*, we provide richer insight by applying more semantically meaningful considerations, i.e. musical transformations.

Admittedly, we could have used transformations to produce a numeric mapping between algorithms and different applications, such as a conclusion that algorithm A is better than algorithm B for this purpose because the transformations produced by A fit better in this context. However, we do not have enough data regarding which transformations are more desirable than others for which application.

### **Other limitations**

The focus of using the transformations is limited to a computational context to provide insight into the differences between different annotators and between different annotators and algorithms. We do not touch upon the intentionality behind the musical transformations, which is outside the scope of this work. In addition, comparing percentages of transformations performs best when the absolute numbers of pattern occurrences are similar to each other. While evidence from other chapters and sections substantiates the results of our analysis on the transformation proportions, it is important to approach with caution when analysing transformation proportions alone. We are also aware that there are almost certainly other annotations, algorithms, and corpora in the wide world that are not featured in this dissertation.



# Chapter 7 Conclusions and Future Work

In this dissertation, we have extensively investigated one of the widespread phenomena of music—patterns—as well as the connection to automating the discovery of patterns in music. In this final chapter of the dissertation, we reflect on what we have learned from the previous chapters, providing a supplementary discussion on topics related to our contributions.

## 7.1 Summary of the chapters and their contributions

In Chapter 1, we introduced the scientific topics of this dissertation, outlined the challenges regarding musical patterns and musical pattern discovery algorithms, and presented our thesis statement:

Human-perceived musical patterns, a type of highly subjective and ubiquitous abstraction, have important connections to musical transformations, which, when explored compositionally through a functional programming language, informs both human-to-algorithm and algorithm-to-algorithm comparisons of discovered musical patterns.

In Chapter 2, we investigated the concept of pattern from a range of perspectives and examined musical pattern discovery algorithms to show the breadth and diversity of the field of musical pattern discovery. Following our investigation, we have addressed the questions posed at the beginning of this dissertation concerning the concept of musical patterns and their discovery algorithms: what is the current state of research in the area of musical pattern discovery? How broad and diverse are the concepts of musical patterns and musical pattern discovery algorithms?

We started to address these questions by first introducing repetition and variation as our central themes and making connections with other related concepts. With these central themes and related concepts in mind, we examined a range of studies

relating to musical patterns and the algorithms that discover them automatically. Through this investigation, we answered our research questions, highlighting the wide-ranging nature of both the concept of pattern and the pattern discovery algorithms. Difficulties were identified when trying to provide a precise definition of "pattern", and these difficulties contribute to the diversity and complexity of research on musical patterns and pattern discovery algorithms. Furthermore, the differences in how these algorithms function—such as how they handle input, the operations they perform, and the filters they employ—complicate direct comparisons between them. Through this process, we addressed the challenges of crafting a comprehensive overview and managing diverse perspectives by selectively highlighting key studies and by categorising perspectives and concepts to streamline our analysis.

The key contribution of Chapter 2 is demonstrating the breadth and diversity of the field of musical pattern discovery through a review of relevant literature in MIR, music theory, and musicology. With this review, we shed light on the critical role repetition and variation play in understanding musical patterns. Our analysis of 18 pattern-related concepts highlights the multi-layered nature of the subject of musical pattern. We also set the stage for subsequent chapters by introducing 11 musical pattern discovery algorithms, listing their goals, methods, data representations, filtering techniques, and evaluations performed.

In Chapter 3, we designed annotation tools [ANOMIC](#) and [PAF](#), gathered human annotations, and analysed the gathered data using the features of the annotated patterns. As such, we have answered the questions we asked at the beginning of this dissertation regarding the collection of human-annotated musical patterns as ground truth data for comparing algorithms: How can we efficiently collect human-annotated musical patterns? Which factors have an impact on the annotations?

Through feature analysis, we showed that features such as note range, intervallic leaps, and pitch direction are significantly similar amongst the patterns annotated by musicians and significantly different between the patterns annotated by musicians and non-musicians. Considerations on using reference data for evaluating algorithms were also discussed, namely: the benefits of collecting more human-annotated musical patterns, enhancing the reference data by identifying subgroups of annotators that highly agree with each other, and using multiple reference annotations rather than single ones for a more comprehensive comparison of algorithms. By analysing the annotations collected by these annotation tools, our results showed that many factors should be taken into consideration when gathering

and using human-annotated musical patterns, such as the annotators' musical backgrounds, the interface of the annotation tools, and the protocol of the annotation experiments, including the influences of sheet music and piano roll formats, as well as the automatic occurrence matching functionality of the annotation tools, on the patterns being annotated. We also detailed the [MIREX](#) task, which is closely related to musical pattern discovery, discussed the [HEMAN](#) dataset, and introduced two digital annotation tools for gathering human annotations along the way. Through this process, we mitigated the challenges regarding reference data in musical pattern discovery by developing multiple annotation tools and experiments, as well as employing agreement metrics and feature analysis.

This work contributes to deepening our understanding of how different annotation interfaces and instructions influence annotated patterns. The different annotation interfaces enable a study of the disparities and commonalities amid patterns identified by annotators with varying levels of musical expertise and theoretical backgrounds, thereby providing a solid foundation for the informed use of reference data in comparing and developing pattern discovery algorithms.

In Chapter 4, we proposed four methods for comparing musical pattern discovery algorithms. Thus, we have addressed the questions that were initially raised in this dissertation about the comparison between human-annotated musical patterns and the patterns extracted by pattern discovery algorithms: how do we compare human-annotated and algorithmically extracted musical patterns in addition to existing comparison methods like the  $F_1$  measures, to deepen our understanding of musical patterns? In what ways do the human-annotated and algorithmically extracted patterns diverge the most and in what ways are they similar?

We answered these questions by designing and employing four methods, which are [LFV](#), [PP](#), [CC](#), and [SPI](#). Through this multifaceted approach, we observed discrepancies across musical dimensions when comparing the musical patterns extracted by different algorithms to those annotated by humans. These discrepancies are not only apparent in the visualisations of pattern locations and discernible through the use of classifiers and synthetic data, but they also pose challenges when attempting to harness the collective wisdom of algorithms by combining their results. Notably, the most significant differences between the features of human-annotated patterns and those extracted by algorithms arise in rhythmic features. Through this process, we addressed the challenges posed by the complexity and diversity of algorithms, as well as the limitations of current metrics, by creating and

employing a range of methods to interpret and compare algorithmically extracted musical patterns.

Our primary contributions stem from these novel methods we introduced for examining the outputs of pattern discovery algorithms. These methods offer more comprehensive approaches for comparing algorithms than existing state-of-the-art methods, which include measures such as precision, recall, and the  $F_1$  measures when directly comparing the onsets and pitches in the patterns. Specifically, visualisation aids in deciphering the large output patterns from algorithms and suggests the possibility of combining results from multiple algorithms, although the combined outcomes are not particularly impressive because the output patterns from the algorithms differ too much. Additionally, identifying key features during classification helps pinpoint the dimensions where algorithms fail to match human-annotated patterns' features. The final contribution in Chapter 4 is the use of synthetic pattern insertion, which enables us to gain new insight into how we expect the algorithms to perform given controlled input data, revealing both alignment and mismatches between human expectations and algorithmic outputs across a range of algorithms. None of these results could have been uncovered using traditional evaluation metrics. Through these novel contributions, we advocate for a more nuanced comparison between algorithmic outputs, employing a combination of methods rather than solely relying on conventional measures to pave the way for enhanced performance and interpretability in the field of musical pattern discovery.

In Chapter 5, we showcased the implementation of *Pattrans*, our own DSL for comparing musical pattern occurrences using musical transformations. Hence, we have addressed the questions presented at the beginning of this dissertation related to the application of musical transformations for matching musical pattern occurrences: can we identify the transformations that explain the variation between different pattern occurrences? More concretely, how do we implement a library to automate this identification process?

We answered the research questions by implementing *Pattrans*, which allows for multiple transformations to be applied individually or in conjunction for the purpose of examining algorithmically extracted and human-annotated musical patterns. The underlying design of *Pattrans* leverages Haskell's type system and ideas from category theory. We introduced the primary transformations—exact repetition, chromatic transposition, rhythmic-only repetition—we implemented in the library, as well as the miscellaneous transformations such as retrograde, augmentation, diminution, and approximation. Through this process, we address

the challenge that existing comparison methods are largely centred on the individual pitch-and-onset content of patterns by implementing Pattrans with musical transformations to match the relations between pattern occurrences.

Our key contribution of Chapter 5 is the implementation of a library in the functional programming language Haskell, named Pattrans, designed to model musically important transformations compositionally.

In Chapter 6, we employed Pattrans to explore the transformations between occurrences of musical patterns in both human-annotated and algorithmically extracted pattern datasets, with the aim of comparing these patterns by assessing how transformations differ or align between the pattern occurrences. In doing so, we have responded to the questions that were initially proposed in this dissertation regarding the use of musical transformations to compare musical patterns: what distinctions or similarities can be drawn when using a system that employs musical transformations to analyse patterns? What musical transformations do we find in patterns extracted by musical pattern discovery algorithms versus those annotated by humans?

The questions are answered by initially comparing the proportions of musical transformations between human-annotated pattern occurrences in the datasets of JKU-PDD, MTC-ANN, and HEMAN. By comparing the transformations between pattern occurrences quantitatively, we showed that human-annotated patterns tend to contain a higher proportion of primary transformations than the algorithmically extracted ones in both JKU-PDD and MTC-ANN datasets. We have also used statistical testing and PCA to analyse the relations revealed by musical transformations between pattern occurrences extracted by different algorithms and annotated by humans using the JKU-PDD dataset—the results offered a mixed and nuanced view, varying across different algorithms when compared to human annotations. Through this process, we tackled the challenges of handling a large number of hard-to-interpret algorithmic patterns and their comparisons by employing musical transformations and leveraging Pattrans to quantify transformation proportions.

Chapter 6 contributes to revealing that a large proportion of the pattern occurrence relations in human annotations can be explained by the transformations we consider. Furthermore, the findings on exact repetition underscore the divergence between the human-annotated patterns and algorithmic ones, and musical transformations may serve as filtering and summarising tools to narrow this gap. Statistical testing reveals that one algorithm exhibited significantly different transformation propor-

tions, while two others did not differ significantly—similar methodologies can be extended to additional algorithms and corpora for comparing patterns discovered by both algorithms and humans. PCA provides evidence supporting a conformance between human-annotated patterns and those extracted by algorithms in terms of the variance of the proportions of transformations, suggesting that, in terms of the proportion of primary transformations within each pattern, algorithms are discovering patterns similar to those recognised by humans. Through these contributions, our approach of using transformation proportions to compare musical patterns adds new tools to traditional metrics by offering a multifaceted view, specifically by applying the systematic lens of transformations to the pattern occurrences within the same pattern, enabling their better grouping and comparison.

## 7.2 Looking Ahead

### Algorithms

In the context of discovery algorithms, our approaches described in Chapter 4 and 6 possess a versatility that allows for their application beyond the specific algorithms discussed in this dissertation. By expanding their applicability to a wider range of pattern discovery algorithms in music, we could foster a more comprehensive comparison of musical patterns and pattern discovery algorithms and potentially stimulate the development of novel algorithms. We can also try to use what we learned about patterns and transformations in designing the algorithms. This could involve selecting specific transformations or using transformations as filters to reduce the algorithms' output size. Using synthetic data is another important direction going forward. There have been rapid developments and promising results in the ML field regarding using synthetic data for both training and testing data for algorithms (Lu *et al.*, 2023; Nikolenko, 2021). Expanding on the example we have presented in Section 4.2.5, synthetic data can be enhanced by incorporating musical transformations and their combinations. This approach would likely increase the diversity of the synthetic data, potentially rendering it more realistic for musical pieces that contain musical transformations. Other simulation methods can also extend the possibilities to control the structures of patterns, such as data degradation processes with different degrees of severity (McLeod *et al.*, 2020).

Beyond the scope of this dissertation, there is a multitude of existing musical pattern discovery methodologies that were not considered, as well as ample possibility for designing new algorithms employing methods that differ from those explored in this study. Although we cannot detail all these pattern discovery methodologies,

we acknowledge that some of them raise important issues about the nature of patterns and transformations, such as the role of context and attention. For instance, while we have not incorporated any Bayesian aspects when using statistics in our approaches in Chapters 3 and 6, these aspects could be explored in future studies and could provide valuable insights into how our approach deals with uncertainty and prior knowledge.

### **Functional programming**

On the functional programming front, we can further experiment with the order of the transformations, the selection of the prototype pattern, and analysis for other datasets. Varying the order of transformations also offers an intriguing area of exploration, and our implementation can be adjusted to accommodate such variations, which could reveal how different transformation sequences may influence our results. Modifying the selection of the prototype pattern could also be added as a part of the implementation, and the act of modification may impact the pattern occurrences that will be checked against the transformations. Although we anticipate that the choice of a prototype pattern would not significantly influence our analysis, a different outcome would provide insights into the role of the first occurrence of the pattern, which is our current prototype pattern. Moreover, expanding our analysis to include other datasets would not only validate and broaden the applicability of our findings but also offer valuable feedback for refining our current implementation. By undertaking an iterative process, we can ensure that our implementation continues to perform effectively and robustly across a diverse range of musical contexts.

In combination with the synthetic data mentioned above, creating an expressive DSL for generating the desired data is a promising direction as well. This could allow us to more abilities when inserting musical patterns, thereby extending the possibilities of our comparison between pattern discovery algorithms.

### **Music and annotations**

The music data we used in this dissertation is limited, but the world teems with music of different types. The scope we set ourselves for this dissertation is monophonic symbolic datasets. By limiting ourselves to monophonic MIDI datasets, we must acknowledge the exclusion of polyphonic data and music from various cultures. As described in (Cook, 2000), "in Indian and Chinese music it is often the notes between the notes, so to speak, that are responsible for the effect of the music." Moreover, polyphonic music, which features multiple and simultaneous melodic

lines, extends beyond the scope of our monophonic focus as exemplified in intricate pieces like Bach's fugues or contemporary jazz ensembles. Another example is John Cage's *4'33"*. Putting aside the debate about whether it qualifies as music, our discussion does not extend to patterns present in modern compositions like this one. Looking ahead, it is certainly feasible to extend our datasets and consider polyphonic and audio data. This would require the integration of music transcription algorithms to convert audio data into a symbolic format that the pattern discovery algorithms can process. By incorporating additional datasets, we could address a wider range of musical phenomena and capture the diversity inherent in global musical practices.

In terms of gathering annotations, we believe that the widespread adoption of digital tools for collecting pattern annotations is inevitable in the near future. The two experiments we have conducted could be refined and carried out in a more controlled environment, which would help record and eliminate variables that might influence the annotations, such as the number of times participants listen to the music or fatigue during the process. We can attract more annotators by considering gamification, introducing elements such as rewards or competitive elements, which may enhance engagement and counteract fatigue during annotation. Furthermore, annotations could cover a more diverse range of music. These annotations can also be collected with specific tasks in mind, such as tasks that could relate to compression and classification—identifying representative patterns that cover the musical piece, or patterns that distinguish one piece from another.

A challenge, however, is that while we have encountered tasks such as compression, we do not yet have a comprehensive understanding of what "all the tasks" might encompass. Some even argue that there will never be enough annotated data to train all the models for all the tasks we need to perform (Roth, 2017). If these views turn out to be true, we might need to transition from using data to supervise the training of algorithms, and instead embrace strategies from weakly-, self-, and unsupervised methodologies. Even so, we believe that annotation data would still be a valid starting point for developing algorithms.

### **Convergence of other similar research**

Independently from our work, a number of recent studies have used transformations and synthetic data for analysis and composition (Hunt *et al.*, 2019, 2020; Hunt, 2020; Silva *et al.*, 2020). Additionally, there are publications in other research areas that are also relevant to the discussions from preceding chapters. For example, in the research area of program synthesis, learning the transformations in images has



been investigated (Banburski *et al.*, 2020). Wittgenstein's perspective aligns with our approach to some extent: "understanding music consists in grasping the internal relationships between musical events" (Kaduri, 2006). Moving forward, uncovering advancements in various fields that parallel our methodologies could be beneficial for further refining our approaches.

#### **Generative AI**

Considering the ongoing advancements in AI content generation, the ability to identify patterns and plausible transformations between these pattern occurrences could be instrumental in distinguishing between human-produced content and algorithmically generated content. If there are systematic differences in the features of these patterns or the transformations between the occurrences, these differences could be employed for differentiation purposes. Moreover, the progress made in generative AI can be used to produce realistic synthetic data, which can include embedded patterns. This approach allows for an assessment of pattern recognition algorithms in terms of their effectiveness in uncovering these integrated patterns.

### **7.3 Bird's eye view**

Throughout this dissertation, we have delved into the intricacies of musical patterns and their discovery algorithms, offering a comprehensive study that propels us beyond the current state of the art. We unpacked the conceptual diversity in the field, introduced annotation tools and comparison methodologies, and constructed Pattrans to explore musical transformations. Our work goes beyond analysing existing methods and offers new tools and perspectives: whether it is identifying the nuances that separate expert musicians from non-musicians in pattern annotation or illuminating the divergences between human and algorithmically extracted patterns, our research opens new avenues for scrutinising pattern discovery algorithms. Looking forward, this research sets the stage for future innovations in the nuanced and human-centred understanding of musical patterns. In doing so, we pave the way for evolving computational methods in music research and enriching our collective understanding of musical patterns and their integral role in musical compositions.



# Appendices



# Appendix A Datasets

In this appendix, we introduce the datasets we use in this dissertation. When we use the word "music", by default, we mean these dataset we investigated, unless explicitly disambiguated for a more general discussion. As put in (Cook, 2000), "the concept of 'music' was firmly rooted in a specific corpus of musical works, and through that in a specific time and place."

## A.1 JKU-PDD

[JKU-PDD](#), as the publicly available dataset from the [MIREX](#) task, has been used in evaluating musical pattern discovery algorithms. Compiled by [MIR](#) researchers, it contains one piece each by Bach, Mozart, Beethoven, Gibbons, Chopin, and 26 patterns and 105 occurrences annotated by experts. A list of the piece and the dates of the composers can be seen in [Table A.1](#).

Composer	Dates	Piece
Orlando Gibbons	1583–1625	"The Silver Swan"
Johann Sebastian Bach	1685–1750	Fugue in A minor, BWV 889
Wolfgang Amadeus Mozart	1756–1791	Minuet from Piano Sonata in Eb major, K. 282
Ludwig van Beethoven	1770–1827	Scherzo from Piano Sonata in F minor, op. 2, no. 1
Frédéric Chopin	1810–1849	Mazurka in B minor, op. 24, no. 4

Table A.1: A summary of the pieces in the JKU-PDD dataset.

Although the pattern occurrences are not annotated exhaustively (Meredith, 2015), it has been widely used for evaluating musical pattern discovery algorithms. According to (Collins *et al.*, n.d.), the annotations are constructed from three sources: Barlow and Morgenstern 1948, Schoenberg 1967, and Bruhn 1993. Some annotations were revised with added annotations for Gibbons' "The Silver Swan". For example, because Barlow and Morgenstern 1948 is intended as a comprehensive yet concise companion for the classical music enthusiast, themes that are longer than the width of the page were curtailed for the sake of brevity. These themes are lengthened back to their musically appropriate length in the dataset.

When comparing annotations of the same piece across sources, it is quite common to find the sources in agreement. Although the sources have not gone through an inter-annotator agreement analysis, such a level of agreement is encouraging.

We use the monophonic version of this dataset, and to monophonise the music, the procedures of the clipped-skyline approach were followed as described below. From the polyphonic version, the clipped-skyline algorithm outputs the highest note at each unique onset with two scenarios. First, if the current highest note is still sounding when a new lower note begins, the new lower note is ignored. Second, if the current highest note is still sounding when a new higher note begins, the new higher note is included in the output, and the previous note's duration is clipped in time.

### A.2 MTC-ANN

**MTC-ANN** is an annotation dataset for a collection of Dutch folk songs (Van Kranenburg *et al.*, 2016). The **MTC-ANN** dataset has been used for studying oral tradition in Dutch folk songs (Van Kranenburg *et al.*, 2019), and the annotations consider inter-opus pattern occurrences. Inter-opus occurrences are not considered within one song, but between the songs within the same tune family.

Tune family is a concept in ethnomusicology that groups together tunes sharing the same ancestor in the process of oral transmission (Boot *et al.*, 2016). Oral transmission plays a significant role in folk music. Through this often imperfect communication process, certain parts of melodies remain stable, variations are created, repeated patterns emerge (Janssen, 2018). Formulated in ethnomusicological studies, the concept of tune family describes the structures in this stream of transformations: folk songs that are supposed to have a common ancestor in the process of oral transmission are grouped into a tune family (Cowdery, 1984). Local structures within the melodies—namely, characteristic motifs, or prominent, nonliterally repeated patterns—are detected to be useful in determining music similarity and classifying tune families (Cowdery, 1984). Subsequently, in an annotation study on the influence of different musical dimensions on human similarity judgements of melodies belonging to the same tune family, repeated patterns between melodies turned out to play the most important role for similarity amongst all considered musical dimensions (Volk & Van Kranenburg, 2012). Therefore, algorithms which can extract these repeated patterns automatically that would be useful for tune family classification.

During the making of [MTC-ANN](#), three experts were asked to annotate the prominent patterns in each song which best classify the song into one of 26 tune families. The dataset consists of 360 Dutch folk songs with 1,657 annotated pattern occurrences.

### A.3 HEMAN and its different versions

As we introduced in Chapter 3, we use the same music material for three experiments, with the first one that initiated the others being called [HEMAN](#). In the datasets, we observe that disagreement amongst annotators is common.

Only one of these experiments have the pattern-occurrence hierarchy, and that is the dataset gathered by [ANOMIC](#).

We therefore call this dataset gathered by ANOMIC the HEMAN dataset in Chapter 7.

This HEMAN dataset contains 2763 annotations of pattern occurrences from 26 participants in at-home self-paced listening experiments. We separate the dataset into two subsets:

- [HEMANLow](#), which is a sub-dataset of HEMAN, consists of the responses of participants with a self-rated musical background score  $< 5$  on a scale of 1-10.
- [HEMANHigh](#) is the complement of [HEMANLow](#) in HEMAN, and therefore consists of annotations of participants with a higher self-rated musical background.





## Appendix B List of Publications

Some of the work contained in this dissertation has appeared in the following publications:

- The research reported in Chapter 3 was published in:

Investigating Musical Pattern Ambiguity in a Human Annotated Dataset, IY Ren, O Nieto, HV Koops, A Volk, W Swierstra. International Conference on Music Perception and Cognition/European Society for the Cognitive Sciences of Music. Graz, Germany, 2018.

Exploring annotations for musical pattern discovery gathered with digital annotation tools. D Tomašević, S Wells, IY Ren, A Volk, M Pesek. *Journal of Mathematics and Music* 15:2, p. 194-207, 2021.

- The research reported in Chapter 4 was published in:

Feature Analysis of Repeated Patterns in Dutch Folk Songs using Principal Component Analysis. IY Ren, HV Koops, D Bountouridis, A Volk, W Swierstra, R Veltkamp. *Folk Music Analysis*, Thessaloniki, Greece, 2018.

Analysis by Classification: A Comparative Study of Annotated and Algorithmically Extracted Patterns in Symbolic Music Data. IY Ren, A Volk, W Swierstra, R Veltkamp. International Society of Music Information Retrieval, Paris, France, 2018.

In Search of the Consensus among Musical Pattern Discovery Algorithms. IY Ren, HV Koops, A Volk, W Swierstra. International Society of Music Information Retrieval, Suzhou, China, 2017.

- An earlier version of Pattrans and analysis presented in Chapter 5 and 6 was published in:

Orestis Melkonian (contribution: main library implementer), Iris Yuping Ren (contribution: ideation, writing, and application of the library), Wouter Swierstra, and Anja Volk. "What constitutes a musical pattern?". In *Proceedings of the 7th ACM SIGPLAN International Workshop on Functional Art, Music, Modeling, and Design (FARM 2019)*. ACM, New York, NY, USA, p. 95-105, 2019.



# List of Acronyms

ANOMIC	ANnOate MusIC, an annotation tool we devised .. 58, 60, 174, 187
CC	Comparative Classification, a method we devised ..... 18, 91, 123, 175
COSIATEC	Data COmpression using SIATEC..... 46, 47, 122
DSL	Domain Specific Language .... 9, 18, 83, 128, 140, 144
FP	Functional Programming..... 1, 129, 144, 147
GBM	Gradient Boosting Machine..... 110
GTTM	Generative Theory of Tonal Music ..... 31–33
HCI	Human-Computer Interaction ..... 88
HEMAN	Human Estimations of Musically Agreeing Notes .. 58, 175, 187
IOI	Inter-Onset Interval..... 142
JKU-PDD	The Johannes Kepler University Patterns Development Database ..... 89, 153, 185
kPCA	kernel Principal Component Analysis..... 169
LDA	Linear Discriminant Analysis ..... 110
LFV	Location and Feature Visualisation, a method we de- vised ..... 17, 91, 102, 123, 175
LVQ	Linear Vector Quantisation ..... 110

## List of Acronyms

MDS	Multidimensional Scaling . . . . .	97, 166
MGDP	Maximally General Distinctive Pattern . . . . .	48
MIDI	Musical Instrument Digital Interface . . . . .	13, 97, 130
MIR	Music Information Retrieval . . . . .	1, 4, 8, 10, 27, 49, 56, 88, 185
MIREX	The Music Information Retrieval Evaluation eXchange . . . . .	10, 23, 33, 34, 38, 40, 56, 89, 95, 154, 175, 185
ML	Machine Learning . . . . .	83, 88, 178
MTC-ANN	The Meertens Tune Collections: The Annotated Cor- pus . . . . .	15, 90, 96–98, 101, 103, 108, 118, 153, 186, 187
MTP	Maximal Translatable Pattern . . . . .	46, 47
NB	Naive Bayes . . . . .	110, 111
PAF	Pattern Annotation Framework, an annotation tool we devised . . . . .	58, 60, 63, 174
PatMinr	Pattern Miner . . . . .	49, 93
Pattrans	The Haskell library we devised . . . . .	18, 176
PCA	Principal Component Analysis . . . . .	97, 103, 108, 118
PP	Pattern Polling, a method we devised . . . . .	17, 91, 106, 123, 175
RF	Random Forest . . . . .	110
SIA	Structure Induction Algorithm . . . . .	46–48
SIACF1	SIATECCompress-F1 . . . . .	46
SIACFP	SIA for r superdiagonals and Compactness Trawler- Categorisation and Fingerprinting . . . . .	46, 48, 95, 121
SIACP	SIATECCompress-Precision . . . . .	46
SIACR	SIATECCompress-Recall . . . . .	46, 103
SIAR	SIA with a sliding window of size r . . . . .	46–48
SIATEC	Structure Induction Algorithm for TECs . . . . .	46, 47
SotA	State-of-the-Art . . . . .	56, 108, 125
SPI	Synthetic Pattern Insertion, a method we devised . .	18, 175

SVM	Support Vector Machine . . . . .	110
SYMCHM	Symbolic Compositional Hierarchical Model . . . . .	45, 103, 118
TEC	Translational Equivalence Class . . . . .	46, 47
tSNE	t-distributed Stochastic Neighbor Embedding . . . . .	168
VM1	Algorithm name . . . . .	45, 120
VM2	Algorithm name . . . . .	45, 120



# Samenvatting

Het vinden van patronen is een gangbare praktijk bij menselijke intellectuele inspanningen. Het is een complexe uitdaging die zowel door mensen als door algoritmen wordt uitgevoerd. Al tientallen jaren wordt er onderzoek gedaan naar algoritmen voor het ontdekken van muzikale patronen. Onderzoekers vergelijken zowel door mensen geannoteerde patronen met algoritmische output, als algoritmische output met elkaar. Traditionele uitkomstmaten hebben echter niet volledig de diepe inzichten opgeleverd die deze evaluaties zouden kunnen bieden. Dit proefschrift draagt bij aan het vergelijken van mechanismen voor het ontdekken van muzikale patronen in zeven hoofdstukken.

Hoofdstuk 1 geeft de achtergrond van het proefschrift, inclusief een overzicht van onderzoeksbenaderingen, contexten, reikwijdte, hoofdstelling, en een opsomming van de bijdragen die dit proefschrift maakt aan het onderzoeksgebied. Hoofdstuk 2 duikt in het concept van muzikale patronen, en onderzoekt het diverse landschap van algoritmen voor het ontdekken van muzikale patronen. Deze verkenning onthult de complexiteit rond de definitie van patronen en de veelzijdige aard van deze algoritmen. Hoofdstuk 3 is gewijd aan de verzamelinstrumenten voor door mensen geannoteerde muzikale patronen en de analyse van factoren die annotaties beïnvloeden. We zien dat muzikale achtergronden invloed hebben op geannoteerde patronen; toolinterfaces en automatische matching beïnvloeden de lengte en frequentie van annotaties. Hoofdstuk 4 introduceert vier methoden die zijn toegesneden op het vergelijken van algoritmen voor het ontdekken van muzikale patronen. Deze methoden bieden nieuwe inzichten in de discrepanties tussen de door mensen geannoteerde patronen en hun algoritmisch geëxtraheerde tegenhangers. Deze methoden bieden een meer alomvattende benadering voor het vergelijken van algoritmen, en helpen bij de interpretatie en evaluatie van algoritmische resultaten. Hoofdstuk 5 implementeert Pattrans, een domeinspecifieke taal (DSL) in de functionele taal Haskell, voor het vergelijken van voorkomens van muzikale patronen door middel van muzikale transformaties. We verdiepen ons in het ontwerp van deze DSL om de relaties tussen verschillende voorkomens van patronen op een modulaire manier te beschrijven en analyseren. Hoofdstuk 6 gebruikt Pattrans om transformaties tussen voorkomens van muzikale patronen

nauwkeurig te onderzoeken. We ontdekken onder meer dat door mensen geannoteerde patronen doorgaans een groter aandeel exacte herhalingen hebben, en dat verschillende algoritmen verschillende transformatieverhoudingen vertonen in vergelijking met menselijke annotaties, wat bijdraagt aan een genuanceerder beeld van patroonvergelijkingen.

Samenvattend draagt dit proefschrift niet alleen nieuwe perspectieven bij aan de vergelijking van muzikale patronen, maar introduceert het ook methoden en hulpmiddelen die het veld van de ontdekking van muzikale patronen verrijken. We onderzoeken het concept van muzikale patronen, voeren patroonannotatie-experimenten uit en visualiseren en analyseren door mensen geannoteerde en algoritmisch geëxtraheerde patronen. Bovendien zien we het potentieel van de muzikale transformaties die schuilgaan achter zich herhalende en variërende patronen en patroonvoorkomens. Met behulp van Haskell modelleren we de relatie tussen patronen en transformaties. Hierna onderzoeken we hoe we transformaties kunnen gebruiken om muzikale patroonvoorkomens te relateren en te classificeren. Tijdens ons hele traject pleiten we voor een meer alomvattende benadering van patroonvergelijking, die verder gaat dan traditionele uitkomstmaten.



# Curriculum Vitae

Iris Yuping Ren was born and grew up in Beijing, China, where she studied violin, computer science, mathematics, and astronomy through high school. Between 2009 and 2013, she completed a Bachelor of Science in the School of Mathematics and a Bachelor of Management Studies in the school of History and Culture at Shandong University, China. After taking additional modules in physics, she successfully applied to PhD programmes in mathematics and physics, but decided to start her academic career with the Erasmus Mundus programme in complex systems science. Under this programme, she then completed two Masters of Science at the University of Warwick, U.K., and École Polytechnique, France, with a thesis project on modelling symbolic music using complex network theory and analysing these networks using topological data analysis. In pursuit of her interests in computational methods for music, she entered the Electrical and Computer Engineering department at the University of Rochester. During this time in the U.S., she completed a Master's degree in Electrical Engineering, earned an Advanced Diploma in Violin at Eastman Community Music School, and participated in courses and research offered by the Department of Brain and Cognitive Sciences. Following this trajectory, in 2017, she started her PhD in computational music structure analysis using functional programming under the supervision of dr. Anja Volk, dr. Wouter Swierstra, prof. dr. Johan Jeuring, and prof. dr. Remco C. Veltkamp at the Department of Information and Computing Sciences at Utrecht University. Due to departmental changes, prof. dr. Gabriele Keller joined the supervisory team in 2019 in place of prof. dr. Johan Jeuring. During her PhD, Iris served on the Faculty Council, Departmental Advisory Committee, and as a board member of the International Society of Music Information Retrieval. Other activities she pursued during her PhD included learning about repetitive strain injury, exploring diversity and inclusion, consulting for the industry, and teaching at a local university of applied sciences.



# References

- Agawu, K. (2014). *Music as discourse: Semiotic adventures in romantic music*. Oxford University Press, Oxford, UK.
- Alaa, A., Van Breugel, B., Saveliev, E. S., & van der Schaar, M. (2022). How faithful is your synthetic data? Sample-level metrics for evaluating and auditing generative models. *International Conference on Machine Learning*, 290–306.
- Allegraud, P., Bigo, L., Feisthauer, L., Giraud, M., Groult, R., Leguy, E., & Levé, F. (2019). Learning sonata form structure on mozart’s string quartets. *Transactions of the International Society for Music Information Retrieval (TISMIR)*, 2(1), 82–96.
- Andreatta, M. (2018). From music to mathematics and backwards: Introducing algebra, topology and category theory into computational musicology. *Imagine math 6* (pp. 77–88). Springer.
- Angluin, D. (1988). Queries and concept learning. *Machine Learning*, 2(4), 319–342.
- Arjonilla, F. J., & Ogata, T. (2017). General problem solving with category theory. *arXiv preprint arXiv:1709.04825*.
- Baez, J., & Stay, M. (2010). Physics, topology, logic and computation: A rosetta stone. *New structures for physics* (pp. 95–172). Springer.
- Balke, S., Driedger, J., Abeßer, J., Dittmar, C., & Müller, M. (2016). Towards evaluating multiple predominant melody annotations in jazz recordings. *Proceedings of the 17th International Society for Music Information Retrieval (ISMIR)*, 246–252.
- Bamberger, J. S. (2000). *Developing musical intuitions: A project-based introduction to making and understanding music*. Oxford University Press, Oxford, UK.
- Banburski, A., Gandhi, A., Alford, S., Dandekar, S., Chin, S., & tomaso a poggio. (2020). Dreaming with ARC. *Learning Meets Combinatorial Algorithms at NeurIPS2020*.
- Bertens, R., Vreeken, J., & Siebes, A. (2016). Keeping it short and simple: Summarising complex event sequences with multivariate patterns. *Proceedings of the 22nd International Conference on Knowledge Discovery and Data Mining, ACM SIGKDD*, 735–744.
- Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.
- Blacking, J. (1984). *What languages do musical grammars describe?* Leo S. Olschki Editore, Florence, Italy.
- Bomberger, W. A. (1996). Disagreement as a measure of uncertainty. *Journal of Money, Credit and Banking*, 28(3), 381–392.
- Boot, P., Volk, A., & de Haas, W. B. (2016). Evaluating the role of repeated patterns in folk song classification and compression. *Journal of New Music Research*, 45(3), 223–238.

## References

- Bountouridis, D. (2018). *Music information retrieval using biologically inspired techniques* (Doctoral dissertation). Utrecht University.
- Brand, J., Hailey, C. *et al.* (1997). *Constructive dissonance: Arnold schoenberg and the transformations of twentieth-century culture*. University of California Press, Berkeley, CA, USA.
- Brand, M. (1999). Pattern discovery via entropy minimization. *Proceedings of the 7th International Workshop on Artificial Intelligence and Statistics (AISTAT)*.
- Breiman, L. (2001). Random forests. *Machine learning*, 45(1), 5–32.
- Burgoyne, J. A., Bountouridis, D., van Balen, J., Honing, H., *et al.* (2013). Hooked: A game for discovering what makes music catchy. *Proceedings of the 14th Society of Music Information Retrieval Conference (ISMIR)*, 244–250.
- Buteau, C., & Mazzola, G. (2000). From contour similarity to motivic topologies. *Musicae Scientiae*, 4(2), 125–149.
- Buteau, C., & Mazzola, G. (2008). Motivic analysis according to Rudolph Réti: Formalization by a topological model. *Journal of Mathematics and Music*, 2(3), 117–134.
- Cambell, E. (2010). *Boulez, music and philosophy*. Cambridge University Press, Cambridge, UK.
- Cambouropoulos, E. (2001). Melodic cue abstraction, similarity, and category formation: A formal model. *Music Perception*, 18(3), 347–370.
- Cambouropoulos, E. (2006). Musical parallelism and melodic segmentation. *Music Perception: An Interdisciplinary Journal*, 23(3), 249–268.
- Carmon, Y., Raghunathan, A., Schmidt, L., Duchi, J. C., & Liang, P. S. (2019). Unlabeled data improves adversarial robustness. *Advances in neural information processing systems*, 32.
- Chiu, B., Keogh, E., & Lonardi, S. (2003). Probabilistic discovery of time series motifs. *Proceedings of the 9th International Conference on Knowledge Discovery and Data Mining ACM SIGKDD*, 493–498.
- Collins, T. (2011). *Improved methods for pattern discovery in music, with applications in automated stylistic composition* (Doctoral dissertation). The Open University.
- Collins, T. (2014). *2014:Discovery of Repeated Themes & Sections Results - MIREX Wiki*. [Online; accessed 18. Jul. 2023]. [https://www.music-ir.org/mirex/wiki/2014:Discovery\\_of\\_Repeated\\_Themes\\_%26\\_Sections\\_Results](https://www.music-ir.org/mirex/wiki/2014:Discovery_of_Repeated_Themes_%26_Sections_Results)
- Collins, T. (2017). *2017:Discovery of Repeated Themes & Sections Results - MIREX Wiki*. [Online; accessed 18. Jul. 2023]. [https://www.music-ir.org/mirex/wiki/2017:Discovery\\_of\\_Repeated\\_Themes\\_%26\\_Sections\\_Results](https://www.music-ir.org/mirex/wiki/2017:Discovery_of_Repeated_Themes_%26_Sections_Results)
- Collins, T., Arzt, A., Flossmann, S., & Widmer, G. (2013). SIARCT-CFP: Improving precision and the discovery of inexact musical patterns in point-set representations. *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR)*, 549–554.
- Collins, T., Flossmann, S., Arzt, A., & Widmer, G. (n.d.). A tutorial on pattern discovery: Auditioning, formalizing, and evaluating the discovery of motives, themes, and repeated sections in symbolic and audio representations of music.

- Conklin, D. (2002). Representation and discovery of vertical patterns in music. *International Conference on Music and Artificial Intelligence*, 32–42.
- Conklin, D. (2010). Discovery of distinctive patterns in music. *Intelligent Data Analysis*, 14(5), 547–554.
- Conklin, D. (2013a). Antipattern discovery in folk tunes. *Journal of New Music Research*, 42(2), 161–169.
- Conklin, D. (2013b). Multiple viewpoint systems for music classification. *Journal of New Music Research*, 42(1), 19–26.
- Conklin, D., & Anagnostopoulou, C. (2001). Representation and discovery of multiple viewpoint patterns. *Proceedings of the 27th International Computer Music Conference (ICMC)*.
- Conklin, D., & Anagnostopoulou, C. (2006). Segmental pattern discovery in music. *INFORMS Journal on computing*, 18(3), 285–293.
- Conklin, D., & Anagnostopoulou, C. (2011). Comparative pattern analysis of cretan folk songs. *Journal of New Music Research*, 40(2), 119–125.
- Conklin, D., & Bergeron, M. (2008). Feature set patterns in music. *Computer Music Journal*, 32(1), 60–70.
- Conklin, D., & Maessen, G. (2019). Generation of melodies for the lost chant of the mozarabic rite. *Applied Sciences*, 9(20), 4285.
- Conklin, D., & Weisser, S. (2016). Pattern and antipattern discovery in ethiopian bagana songs. *Computational music analysis* (pp. 425–443). Springer.
- Cook, N. (2000). *Music: A very short introduction*. Oxford University Press, Oxford, UK.
- Cooley, R., Mobasher, B., & Srivastava, J. (1997). Web mining: Information and pattern discovery on the world wide web. *Proceedings of the 9th IEEE International Conference on Tools with Artificial Intelligence*, 558–567.
- Cowdery, J. R. (1984). A fresh look at the concept of tune family. *Ethnomusicology*, 28(3), 495–504.
- Crumley, C. L. (1995). Heterarchy and the analysis of complex societies. *Archeological Papers of the American Anthropological Association*, 6(1), 1–5.
- Da Fontoura Costa, L., & Cesar Jr, R. M. (2009). *Shape classification and analysis: Theory and practice*. CRC Press, Inc., Boca Raton, FL, USA.
- de Reuse, T., & Fujinaga, I. (2019). Pattern clustering in monophonic music by learning a non-linear embedding from human annotations. *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, 761–768.
- Deutsch, D. (2019). *Psychology and music*. Psychology Press, New York, NY, USA.
- Dixon, S., Gouyon, F., & Widmer, G. (2004). Towards characterisation of music via rhythmic patterns. *Proceedings of the 5th International Society for Music Information Retrieval (ISMIR)*, 509–517.
- Dolan, E. I. (2013). *The orchestral revolution: Haydn and the technologies of timbre*. Cambridge University Press, Cambridge, UK.

## References

- Drabkin, W. (2001a). *Motif*. Oxford University Press. In Grove Music Online [Online; accessed 18. Jul. 2023]. <https://www.oxfordmusiconline.com/grovemusic/view/10.1093/gmo/9781561592630.001.0001/omo-9781561592630-e-0000019221>
- Drabkin, W. (2001b). *Theme*. Oxford University Press. In Grove Music Online [Online; accessed 18. Jul. 2023]. <https://www.oxfordmusiconline.com/grovemusic/view/10.1093/gmo/9781561592630.001.0001/omo-9781561592630-e-0000027789>
- Ewell, P. (2020). Music theory's white racial frame. *Music Theory Online*, 26(2).
- Fallows, D. (2001). *Head-motif*. In Grove Music Online [Online; accessed 18. Jul. 2023]. <https://www.oxfordmusiconline.com/grovemusic/view/10.1093/gmo/9781561592630.001.0001/omo-9781561592630-e-0000012638>
- Ferguson, D. N. (1941). What is a musical idea? *Papers of the American Musicological Society*, 43–49.
- Finkensiep, C., Déguernel, K., Neuwirth, M., & Rohrmeier, M. (2020). Voice-leading schema recognition using rhythm and pitch features. *Proceedings of the 21st International Society for Music Information Retrieval Conference (ISMIR)*.
- Fitch, W. T. (2006). The biology and evolution of music: A comparative perspective. *Cognition*, 100(1), 173–215.
- Flexer, A., & Grill, T. (2016). The problem of limited inter-rater agreement in modelling music similarity. *Journal of New Music Research*, 45(3), 239–251.
- Fong, B., Spivak, D., & Tuyéras, R. (2019). Backprop as functor: A compositional perspective on supervised learning. *Proceedings of the 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, 1–13.
- Forte, A. (1959). Schenker's conception of musical structure. *Journal of Music Theory*, 3(1), 1–30.
- Forth, J. (2012). *Cognitively-motivated geometric methods of pattern discovery and models of similarity in music* (Doctoral dissertation). Goldsmiths, University of London.
- Forth, J., & Wiggins, G. A. (2009). An approach for identifying salient repetition in multidimensional representations of polyphonic music. *College Publications*.
- Foubert, K., Collins, T., & De Backer, J. (2017). Impaired maintenance of interpersonal synchronization in musical improvisations of patients with borderline personality disorder. *Frontiers in psychology*, 8, 537, 1–17.
- Fowler, C. B. (1966). Discovery method its relevance for music education. *Journal of Research in Music Education*, 14(2), 126–134.
- Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of statistics*, 1189–1232.
- Frieler, K., Höger, F., Pfeleiderer, M., & Dixon, S. (2018). Two web applications for exploring melodic patterns in jazz solos. *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR*, 777–783.
- Fyfe, S., Williams, C., Mason, O. J., & Pickup, G. J. (2008). Apophenia, theory of mind and schizotypy: Perceiving meaning and intentionality in randomness. *Cortex*, 44(10), 1316–1325.

- Gabrielsson, A. (1987). Once again: The theme from mozart's piano sonata in a major. *Action and Perception in Rhythm and Music*, 81–103.
- Ganguli, K. K., Lele, A., Pinjani, S., Rao, P., Srinivasamurthy, A., & Gulati, S. (2017). Melodic shape stylization for robust and efficient motif detection in hindustani vocal music. *Proceedings of the 23rd National Conference on Communications (NCC)*, 1–6.
- Giesa, T., Spivak, D. I., & Buehler, M. J. (2011). Reoccurring patterns in hierarchical protein materials and music: The power of analogies. *BioNanoScience*, 1(4), 153–161.
- Giraud, M., Groult, R., & Levé, F. (2016). Computational analysis of musical form. *Computational music analysis* (pp. 113–136). Springer.
- Gjerdingen, R. (2007). *Music in the galant style*. Oxford University Press, Oxford, UK.
- Gjerdingen, R. (2014). "Historically informed" corpus studies. *Music Perception: An Interdisciplinary Journal*, 31, 192–204.
- Goetschius, P. (1904). *Lessons in music form: A manual of analysis of all the structural factors and designs employed in musical composition* (Vol. 1). Library of Alexandria, Alexandria, Egypt.
- Goldstone, R. L., & Son, J. Y. (2012). *Similarity*. Oxford University Press, Oxford, UK.
- Hahn, U., Chater, N., & Richardson, L. B. (2003). Similarity as transformation. *Cognition*, 87(1), 1–32.
- Hamanaka, M., Hirata, K., & Tojo, S. (2014). Musical structural analysis database based on gttm. *Proceedings of the 15th International Society for Music Information Retrieval (ISMIR)*, 325–331.
- Hamanaka, M., Hirata, K., & Tojo, S. (2015).  $\sigma$  Gttm III: Learning-based time-span tree generator based on pcfg. *International Symposium on Computer Music Multidisciplinary Research*, 387–404.
- Hand, D. J. (2020). *Dark data: Why what you don't know matters*. Princeton University Press, Princeton, NJ, USA.
- Hanninen, D. A. (2003). A theory of recontextualization in music: Analyzing phenomenal transformations of repetition. *Music Theory Spectrum*, 25(1), 59–97.
- Harkleroad, L. (2006). *The math behind the music*. Cambridge University Press, Cambridge, UK.
- Hebart, M. N., Zheng, C. Y., Pereira, F., & Baker, C. I. (2020). Revealing the multidimensional mental representations of natural objects underlying human similarity judgements. *Nature Human Behaviour*, 4(11), 1173–1185.
- Herremans, D., & Chew, E. (2017). Morpheus: Generating structured music with constrained patterns and tension. *IEEE Transactions on Affective Computing*, 10(4), 510–523.
- Hsu, J.-L., Chen, A. L., & Liu, C.-C. (1998). Efficient repeating pattern finding in music databases. *Proceedings of the 7th International Conference on Information and Knowledge Management*, 281–288.
- Hullman, J., & Diakopoulos, N. (2011). Visualization rhetoric: Framing effects in narrative visualization. *IEEE transactions on visualization and computer graphics*, 17(12), 2231–2240.

## References

- Hunt, S., Mitchell, T., & Nash, C. (2019). Automating algorithmic representations of musical structure using IGME: The Interactive Generative Music Environment. *Innovation in music*, 1, 1–18.
- Hunt, S., Mitchell, T., & Nash, C. (2020). Composing computer generated music, an observational study using IGME: the Interactive Generative Music Environment. *Proceedings of 20th International Conference on New Interfaces for Musical Expression (NIME)*, 1–6.
- Hunt, S. (2020). An analysis of repetition in video game music. *Proceedings of the 3rd Joint Conference on AI Music Creativity*, 1–7.
- Huron, D. B. (2006). *Sweet anticipation: Music and the psychology of expectation*. MIT press, Cambridge, MA, USA.
- Janssen, B. (2018). *Retained or lost in transmission?* (Doctoral dissertation). University of Amsterdam.
- Janssen, B., De Haas, W. B., Volk, A., & Van Kranenburg, P. (2014). Finding repeated patterns in music: State of knowledge, challenges, perspectives. *Sound, Music, and Motion: 10th International Symposium, CMMR 2013, Marseille, France, October 15-18, 2013. Revised Selected Papers 10*, 277–297.
- Janssen, B., de Haas, W. B., Volk, A., & Van Kranenburg, P. (2014). Finding repeated patterns in music: State of knowledge, challenges, perspectives. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8905, 277–297.
- Janssen, B., Van Kranenburg, P., & Volk, A. (2017). Finding occurrences of melodic segments in folk songs employing symbolic similarity measures. *Journal of New Music Research*, 46(2), 118–134.
- Jiménez, A., Molina-Solana, M., Berzal, F., & Fajardo, W. (2011). Mining transposed motifs in music. *Journal of Intelligent Information Systems*, 36(1), 99–115.
- Jones, M. R. (1987). Dynamic pattern structure in music: Recent theory and research. *Perception & Psychophysics*, 41(6), 621–634.
- Juba, B., Kalai, A. T., Khanna, S., & Sudan, M. (2011). Compression without a common prior: An information-theoretic justification for ambiguity in language. *Proceedings of the 2nd Symposium on Innovations in Computer Science (ICS)*, 79–84.
- Kaduri, Y. (2006). Wittgenstein and Haydn on understanding music. *Contemporary Aesthetics*, 4(1), 15.
- Kaplan, F. (2015). A map for big data research in digital humanities. *Frontiers in Digital Humanities*, 2, 1.
- Katsiavalos, A., Collins, T., & Battey, B. (2019). An initial computational model for musical schemata theory. *Proceedings of the 20th International Society for Music Information Retrieval (ISMIR)*, 166–172.
- Kephart, R. (2017). *Gregorian Chant Notation*. [Online; accessed 18. Jul. 2023]. <http://www.lphrc.org/Chant>
- Knopke, I., & Jürgensen, F. (2009). A system for identifying common melodic phrases in the masses of palestrina. *Journal of New Music Research*, 38(2), 171–181.



- Kohonen, T. (1990). Improved versions of learning vector quantization. *International Joint Conference on Neural Networks (IJCNN)*, 545–550.
- Koops, H. V., de Haas, W. B., Bountouridis, D., & Volk, A. (2016). Integration and quality assessment of heterogeneous chord sequences using data fusion. *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, 178–184.
- Koops, H. V., de Haas, W. B., Burgoyne, J. A., Bransen, J., Kent-Muller, A., & Volk, A. (2019). Annotator subjectivity in harmony annotations of popular music. *Journal of New Music Research*, 48(3), 232–252.
- Kortylewski, A., He, J., Liu, Q., Cosgrove, C., Yang, C., & Yuille, A. L. (2021). Compositional generative networks and robustness to perceptible image changes. *Proceedings of 55th Annual Conference on Information Sciences and Systems (CISS)*, 1–8.
- Krause, M., Müller, M., & Weiß, C. (2021). Towards leitmotif activity detection in opera recordings. *Transactions of the International Society for Music Information Retrieval*, 4(1), 127–141.
- Krumhansl, C. L., Sandell, G. J., & Sergeant, D. C. (1987). The perception of tone hierarchies and mirror forms in twelve-tone serial music. *Music Perception: An Interdisciplinary Journal*, 5(1), 31–77.
- Kruskal, W. H., & Wallis, W. A. (1952). Use of ranks in one-criterion variance analysis. *Journal of the American statistical Association*, 47(260), 583–621.
- Kubik, G. (1979). Pattern perception and recognition in African music. *The Performing Arts: Music and Dance*, 221–49.
- Kursa, M. B., Rudnicki, W. R. *et al.* (2010). Feature selection with the Boruta package. *J Stat Software*, 36(11), 1–13.
- Lartillot, O. (2004). A musical pattern discovery system founded on a modeling of listening strategies. *Computer Music Journal*, 28(3), 53–67.
- Lartillot, O. (2005). Efficient extraction of closed motivic patterns in multi-dimensional symbolic representations of music. *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, 229–235.
- Lartillot, O. (2011). MIRtoolbox 1.3. 4 user’s manual. *Finnish Centre of Excellence in Interdisciplinary Music Research, University of Jyväskylä, Finland*.
- Lartillot, O. (2014). *PatMinr: In-depth motivic analysis of symbolic monophonic sequences*. Music Information Retrieval Evaluation eXchange, MIREX.
- LaRue, J. (1992). *Guidelines for style analysis*. Harmonie Park Press, Warren, MI, US.
- Lent, J. (2017). *The patterning instinct*. Prometheus Books, Amherst, NY, USA.
- Lerdahl, F., & Jackendoff, R. (1983). An overview of hierarchical structure in music. *Music Perception*, 229–252.
- Lerdahl, F., & Jackendoff, R. S. (1985). *A generative theory of tonal music*. MIT press, Cambridge, MA, USA.
- Lidov, D. (2005). *Is language a music?: Writings on musical form and signification*. Indiana University Press, Bloomington, IN, USA.

## References

- Lin, C.-R., Liu, N.-H., Wu, Y.-H., & Chen, A. L. (2004). Music classification using significant repeating patterns. *Proceedings of the International Conference on Database Systems for Advanced Applications*, 506–518.
- London, J. (2021). Music theory as junk science, and how and why we need to fix it. *Future Directions Speaker Series*.
- Louppe, G., Wehenkel, L., Sutera, A., & Geurts, P. (2013). Understanding variable importances in forests of randomized trees. *Advances in Neural Information Processing Systems*, 431–439.
- Lu, Y., Wang, H., & Wei, W. (2023). Machine learning for synthetic data generation: A review. *arXiv preprint arXiv:2302.04062*.
- Mannone, M. (2018). Introduction to gestural similarity in music: An application of category theory to the orchestra. *Journal of Mathematics and Music*, 12(2), 63–87.
- Margulis, E. H. (2014). *On repeat: How music plays the mind*. Oxford University Press, Oxford, UK.
- Mazzola, G. (2018). *The topos of music I: Theory: Geometric logic, classification, harmony, counterpoint, motives, rhythm*. Springer, Cham, Switzerland.
- McKay, C. (2010). *Automatic music classification with jMIR* (Doctoral dissertation). McGill University.
- McLeod, A., Owers, J., & Yoshii, K. (2020). The midi degradation toolkit: Symbolic music augmentation and correction. *Proceedings of the 21th International Society for Music Information Retrieval Conference (ISMIR)*, 846–852.
- Melkonian, O. (2019). Music as language: Putting probabilistic temporal graph grammars to good use. *Proceedings of the 7th ACM SIGPLAN International Workshop on Functional Art, Music, Modeling, and Design (FARM)*, 1–10.
- Melkonian, O., Ren, I. Y., Swierstra, W., & Volk, A. (2019). What constitutes a musical pattern? *Proceedings of the 7th ACM SIGPLAN International Workshop on Functional Art, Music, Modeling, and Design (FARM)*, 95–105.
- Meredith, D. (2006). The ps13 pitch spelling algorithm. *Journal of New Music Research*, 35(2), 121–159.
- Meredith, D. (2013). COSIATEC and SIATECCompress: Pattern discovery by geometric compression. *Music Information Retrieval Evaluation Exchange, MIREX*.
- Meredith, D. (2015). Music analysis and point-set compression. *Journal of New Music Research*, 44(3), 245–270.
- Meredith, D. (2016). Analysing music with point-set compression algorithms. *Computational music analysis* (pp. 335–366). Springer.
- Meredith, D. (2019). RECURSIA-RRT: Recursive translatable point-set pattern discovery with removal of redundant translators. *Proceedings of the 12th International Workshop on Machine Learning and Music (MML)*.
- Meredith, D., Lemström, K., & Wiggins, G. A. (2002). Algorithms for discovering repeated patterns in multidimensional representations of polyphonic music. *Journal of New Music Research*, 31(4), 321–345.
- Meyer, L. B. (1973). *Explaining music: Essays and explorations*. University of California Press, Berkeley, CA, USA.

- Mitchell, H. B. (2012). *Data fusion: Concepts and ideas*. Springer Science & Business Media, New York, NY, USA.
- Monelle, R. (2014). *Linguistics and semiotics in music*. Routledge, New York, NY, USA.
- Müllensiefen, D. (2009). Fantastic: Feature analysis technology accessing statistics (in a corpus). *Goldsmiths University of London*.
- Müllensiefen, D., Gingras, B., Musil, J., & Stewart, L. (2014). The musicality of non-musicians: An index for assessing musical sophistication in the general population. *PloS one*, 9(2), e89642.
- Müller, M., & Grosche, P. (2012). Automated segmentation of folk song field recordings. *Speech Communication; 10. ITG Symposium*, 1–4.
- Müller, M., & Jiang, N. (2012). A scape plot representation for visualizing repetitive structures of music recordings. *Proceedings of the 13th International Society for Music Information Retrieval (ISMIR)*, 97–102.
- Neubarth, K., Shanahan, D., & Conklin, D. (2018). Supervised descriptive pattern discovery in native american music. *Journal of New Music Research*, 47(1), 1–16.
- Ng, A. Y., & Jordan, M. I. (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Advances in Neural Information Processing Systems*, 841–848.
- Nieto, O. (2015). *Discovering structure in music: Automatic approaches and perceptual evaluations* (Doctoral dissertation). New York University.
- Nieto, O., & Farbood, M. M. (2014). Identifying polyphonic patterns from audio recordings using music segmentation techniques. *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, 411–416.
- Nieto, O., & Farbood, M. M. (2012). Perceptual evaluation of automatically extracted musical motives. *Proceedings of the 12th International Conference on Music Perception and Cognition, ICMPC*, 723–727.
- Nikolenko, S. I. (2021). *Synthetic data for deep learning* (Vol. 174). Springer, Cham, Switzerland.
- Nikrang, A., Collins, T., & Widmer, G. (2014). Patternviewer: An application for exploring repetitive and tonal structure. *J.-SR Jang, M. Goto & JH Lee (Chairs), Late-Brake Demo of the Proceedings of the 15th International Society for Music Information Retrieval Conference. ISMIR, Taipei, Taiwan*.
- Northcutt, C., Jiang, L., & Chuang, I. (2021). Confident learning: Estimating uncertainty in dataset labels. *Journal of Artificial Intelligence Research*, 70, 1373–1411.
- Northcutt, C. G., Athalye, A., & Mueller, J. (2021). Pervasive label errors in test sets destabilize machine learning benchmarks. *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*.
- Nuttall, T., Casado, M. G., Ferraro, A., Conklin, D., & Caro Repetto, R. (2021). A computational exploration of melodic patterns in arab-andalusian music. *Journal of Mathematics and Music*, 15(2), 168–180.
- Ockelford, A. (2017). *Repetition in music: Theoretical and metatheoretical perspectives*. Routledge, New York, NY, USA.
- Ockelford, A. (2018). *Comparing notes*. Simon and Schuster, New York, NY, USA.

## References

- Papadimitriou, S., Sun, J., & Faloutsos, C. (2005). Streaming pattern discovery in multiple time-series. *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB)*, 697–708.
- Parida, L. (2007). *Pattern discovery in bioinformatics: Theory & algorithms*. Chapman and Hall/CRC, Boca Raton, FL, USA.
- Park, S., Kwon, T., Lee, J., Kim, J., & Nam, J. (2019). A cross-scape plot representation for visualizing symbolic melodic similarity. *Proceedings of the 20th International Society for Music Information Retrieval (ISMIR)*, 423–430.
- Patel, S., & Barkovich, A. J. (2002). Analysis and classification of cerebellar malformations. *American Journal of Neuroradiology*, 23(7), 1074–1087.
- Pearce, M. T., & Wiggins, G. A. (2007). Evaluating cognitive models of musical composition. *Proceedings of the 4th international joint workshop on computational creativity*, 73–80.
- Pearce, M. T. (2005). *The construction and evaluation of statistical models of melodic structure in music perception and composition* (Doctoral dissertation). City University London.
- Pesek, M., Tomašević, D., Ren, I. Y., & Marolt, M. An opensource web-based pattern annotation framework - PAF. In: *Late-breaking Demos of ISMIR*. 2019, 1–2.
- Pesek, M., Leonardis, A., & Marolt, M. (2017). SymCHM—an unsupervised approach for pattern discovery in symbolic music with a compositional hierarchical model. *Applied Sciences*, 7(11), 1135.
- Peyton Jones, S. (2003). *Haskell 98 language and libraries: The revised report*. Cambridge University Press, Cambridge, UK.
- Pfleiderer, M., Frieler, K., & Abersser, J. (2019). *melpat — The Jazzomat research project*. [Online; accessed 18. Jul. 2023]. [https://jazzomat.hfm-weimar.de/commandline\\_tools/melpat/melpat.html](https://jazzomat.hfm-weimar.de/commandline_tools/melpat/melpat.html)
- Phrase*. Oxford University Press. (2001). In Grove Music Online [Online; accessed 18. Jul. 2023]. <https://www.oxfordmusiconline.com/grovemusic/view/10.1093/gmo/9781561592630.001.0001/omo-9781561592630-e-0000021599>
- Quick, D., & Thomas, K. (2019). A functional model of jazz improvisation. *Proceedings of the 7th ACM SIGPLAN International Workshop on Functional Art, Music, Modeling, and Design (FARM)*, 11–21.
- Rahn, J. (1993). Repetition. *Contemporary Music Review*, 7(2), 49–57.
- Rahn, J. (2004). The swerve and the flow: Music’s relationship to mathematics. *Perspectives of New Music*, 42(1), 130–148.
- Ratner, A., Bach, S. H., Ehrenberg, H., Fries, J., Wu, S., & Ré, C. (2017). Snorkel: Rapid training data creation with weak supervision. *Proceedings of the 43th International Conference on Very Large Data Bases (VLDB)*, 11(3), 269.
- Reminiscence motif*. Oxford University Press. (2002). In Grove Music Online [Online; accessed 18. Jul. 2023]. <https://www.oxfordmusiconline.com/grovemusic/view/10.1093/gmo/9781561592630.001.0001/omo-9781561592630-e-5000009217>
- Ren, I. Y. (2016). Closed patterns in folk music and other genres. *Proceedings of the 6th International Workshop on Folk Music Analysis (FMA)*, 56–58.

- Ren, I. Y., Koops, H. V., Volk, A., & Swierstra, W. (2017). In search of the consensus among musical pattern discovery algorithms. *Proceedings of the 18th International Society for Music Information Retrieval (ISMIR)*, 671–680.
- Ren, I. Y., Koops, H. V., Volk, A., & Swierstra, W. (2018). Investigating musical pattern ambiguity in a human annotated dataset. *Proceedings of the 15th International Conference on Music Perception and Cognition and the 10th triennial conference of the European Society for the Cognitive Sciences of Music (ICMPC/ESCOM)*, 361–367.
- Ren, I. Y., Volk, A., Swierstra, W., & Veltkamp, R. C. (2018). Analysis by classification: A comparative study of annotated and algorithmically extracted patterns in symbolic music data. *Proceedings of the 19th International Society for Music Information Retrieval (ISMIR)*, 539–546.
- Ren, I. Y., Volk, A., Swierstra, W., & Veltkamp, R. C. (2020). A computational evaluation of musical pattern discovery algorithms. *arXiv preprint arXiv:2010.12325*.
- Reti, R. (1951). *The thematic process in music*. Macmillan, New York, NY, USA.
- Ripley, B. D. (2007). *Pattern recognition and neural networks*. Cambridge University Press, Cambridge, UK.
- Rodríguez López, M. E. (2016). *Automatic melody segmentation* (Doctoral dissertation). Utrecht University.
- Rodríguez López, M. E., & Volk, A. (2015). Location constraints for repetition-based segmentation of melodies. *Mathematics and Computation in Music: 5th International Conference, MCM 2015, London, UK, June 22-25, 2015*, 73–84.
- Rodríguez López, M. E., Volk, A., & Bountouridis, D. (2014). Multi-strategy segmentation of melodies. *Proceedings of the 15th Conference of the International Society for Music Information Retrieval (ISMIR)*, 207–212.
- Rolland, P.-Y. (1998). *Decouverte automatique de regularites dans les sequences et application a l'analyse musicale* (Doctoral dissertation). Paris 6.
- Rolland, P.-Y. (1999). Discovering patterns in musical sequences. *Journal of New Music Research*, 28(4), 334–350.
- Roth, D. (2017). Incidental supervision: Moving beyond supervised learning. *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, 31(1).
- Russell, S. (2019). *Human compatible: Artificial intelligence and the problem of control*. Penguin, New York, NY, USA.
- Schafer, R. W. (2011). What is a Savitzky-Golay filter? *IEEE Signal Processing Magazine*, 28(4), 111–117.
- Schenker, H. (1980). *Harmony*. University of Chicago Press, Chicago, IL, USA.
- Scherder, E. (2017). *Singing in the brain: Over de unieke samenwerking tussen muziek en de hersenen*. Singel Uitgeverijen, Amsterdam, Netherlands.
- Schnapper, L. (2001). *Ostinato*. Oxford University Press. In Grove Music Online [Online; accessed 18. Jul. 2023]. <https://www.oxfordmusiconline.com/grovemusic/view/10.1093/gmo/9781561592630.001.0001/omo-9781561592630-e-0000020547>
- Schoenberg, A. (1967). *Fundamentals of musical composition*. Faber and Faber, London, UK.

## References

- Schoenberg, A. (2006). *The musical idea and the logic, technique and art of its presentation*. Indiana University Press, Bloomington, IN, USA.
- Sears, D. R., & Widmer, G. (2020). Beneath (or beyond) the surface: Discovering voice-leading patterns with skip-grams. *Journal of Mathematics and Music*, 1–26.
- Serafine, M. L. (1988). *Music as cognition: The development of thought in sound*. Columbia University Press, New York, NY, USA.
- Shams, Z., Jamnik, M., Stapleton, G., & Sato, Y. (2018). Icon: A diagrammatic theorem prover for ontologies. *Proceedings of the 16th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, 204–208.
- Shermer, M. (2008). Patternicity: Finding meaningful patterns in meaningless noise. *Scientific American*, 299(6), 48.
- Silva, N., Fischione, C., & Turchet, L. (2020). Towards real-time detection of symbolic musical patterns: Probabilistic vs. deterministic methods. *Proceedings of the 27th Conference of Open Innovations Association (FRUCT)*, 238–246.
- Simon, H. A., & Sumner, R. K. (1993). Pattern in music. *Machine models of music*, 83–110.
- Snyder, B., & Snyder, R. (2000). *Music and memory: An introduction*. MIT press, Cambridge, MA, USA.
- Steyerl, H. (2016). A sea of data: Apophenia and pattern (mis-) recognition. *E-flux Journal*, 72, 1–14.
- Stork, D. G., Duda, R. O., Hart, P. E., & Stork, D. (2001). *Pattern classification. A Wiley-Interscience Publication*.
- Suykens, J. A., & Vandewalle, J. (1999). Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3), 293–300.
- Symbolic Key Finding Results - MIREX Wiki*. (2005). [Online; accessed 18. Jul. 2023]. [https://www.music-ir.org/mirex/wiki/2005:Symbolic\\_Key\\_Finding\\_Results](https://www.music-ir.org/mirex/wiki/2005:Symbolic_Key_Finding_Results)
- Temperley, D. (1995). Motivic perception and modularity. *Music Perception: An Interdisciplinary Journal*, 13(2), 141–169.
- Temperley, D. (2014). Information flow and repetition in music. *Journal of Music Theory*, 58(2), 155–178.
- Tenney, J., & Polansky, L. (1980). Temporal gestalt perception in music. *Journal of Music Theory*, 24(2), 205–241.
- Theme (jazz)*. Oxford University Press. (2003). In Grove Music Online [Online; accessed 18. Jul. 2023]. <https://www.oxfordmusiconline.com/grovemusic/view/10.1093/gmo/9781561592630.001.0001/omo-9781561592630-e-2000445900>
- Thorpe, M., Ockelford, A., & Aksentijevic, A. (2012). An empirical exploration of the zygonic model of expectation in music. *Psychology of Music*, 40(4), 429–470.
- Tomašević, D., Wells, S., Ren, I. Y., Volk, A., & Pesek, M. (2021). Exploring annotations for musical pattern discovery gathered with digital annotation tools. *Journal of Mathematics and Music*, 15(2), 194–207.

- Tukey, J. W., & Wilk, M. B. (1966). Data analysis and statistics: An expository overview. *Proceedings of the Fall Joint Computer Conference*, 695–709.
- Typke, R., Veltkamp, R. C., & Wiering, F. (2006). A measure for evaluating retrieval techniques based on partially ordered ground truth lists. *Proceedings of IEEE International Conference on Multimedia and Expo (ICME)*, 1793–1796.
- Typke, R., Wiering, F., & Veltkamp, R. C. (2005). A survey of music information retrieval systems. *Proceedings of the 6th international Society on Music Information Retrieval (ISMIR)*, 153–160.
- Typke, R., Wiering, F., & Veltkamp, R. C. (2007). Transportation distances and human perception of melodic similarity. *Musicae Scientiae*, 11(1), 153–181.
- Utgoff, P. (2006). Detecting motives and recurring patterns in polyphonic music. *Proceedings of the 30th International Computer Music Conference (ICMC)*, 487–494.
- Valdesolo, P., & Graham, J. (2014). Awe, uncertainty, and agency detection. *Psychological science*, 25(1), 170–178.
- Van Balen, J. (2016). *Audio description and corpus analysis of popular music* (Doctoral dissertation). University Utrecht.
- Van Kranenburg, P., De Bruin, M., & Volk, A. (2019). Documenting a song culture: The dutch song database as a resource for musicological research. *International Journal on Digital Libraries*, 20, 13–23.
- Van Kranenburg, P., Janssen, B., & Volk, A. (2016). The Meertens Tune Collections: The Annotated Corpus (MTC-ANN) versions 1.1 and 2.0.1. *Meertens Online Reports*, (1).
- Van Kranenburg, P., Volk, A., & Wiering, F. (2013). A comparison between global and local features for computational classification of folk song melodies. *Journal of New Music Research*, 42(1), 1–18.
- Varèse, E. (1940). Organized sound for the sound film. *The commonweal*, 13(8), 204–5.
- Velarde, G., Meredith, D., & Weyde, T. (2016). A wavelet-based approach to pattern discovery in melodies. *Computational Music Analysis*, 303–333.
- Velarde, G., Weyde, T., & Meredith, D. (2013). An approach to melodic segmentation and classification based on filtering with the haar-wavelet. *Journal of New Music Research*, 42(4), 325–345.
- Volk, A., de Haas, W. B., & Van Kranenburg, P. (2012). Towards modelling variation in music as foundation for similarity. *Proceedings of the 12th International Conference on Music Perception and Cognition ICMPC*, 1085–1094.
- Volk, A., & Van Kranenburg, P. (2012). Melodic similarity among folk songs: An annotation study on similarity-based categorization in music. *Musicae Scientiae*, 16(3), 317–339.
- Von Neumann, J. *et al.* (1958). *Computer and the brain*. Yale University Press, New Haven, CT, USA.
- Vreeken, J., Van Leeuwen, M., & Siebes, A. (2011). Krimp: Mining itemsets that compress. *Data Mining and Knowledge Discovery*, 23(1), 169–214.

## References

- Wang, J. T.-L., Chirn, G.-W., Marr, T. G., Shapiro, B., Shasha, D., & Zhang, K. (1994). Combinatorial pattern discovery for scientific data: Some preliminary results. *Proceedings of the International Conference on Management of Data ACM SIGMOD*, 115–125.
- Webern, A. (1963). *Cited in boulez, music and philosophy, edward cambell*. Cambridge University Press, Cambridge, UK.
- Wells, S. (2019). *Creating a tool for facilitating and researching human annotation of musical patterns* (Master's thesis). Utrecht University.
- Wertheimer, M. (1938). A source book of gestalt psychology. *Kegan Paul, Trench, Trubner & Company*, 1–11.
- Whitehead, A. N., & Price, L. (2001). *Dialogues of alfred north whitehead* (Vol. 84). David R. Godine Publisher, Boston, MA, USA.
- Whittall, A. (2001). *Leitmotif*. Oxford University Press. In Grove Music Online [Online; accessed 18. Jul. 2023]. <https://www.oxfordmusiconline.com/grovemusic/view/10.1093/gmo/9781561592630.001.0001/omo-9781561592630-e-0000016360>
- Wiering, F., de Nooijer, J., Volk, A., & Tabachneck-Schijf, H. J. (2009). Cognition-based segmentation for music information retrieval systems. *Journal of New Music Research*, 38(2), 139–154.
- Wiggins, G. A. (1998). Music, syntax, and the meaning of “meaning”. *Proceedings of the 1st Symposium on Music and Computers*, 18–23.
- Witmer, R. (2001). *Lick*. Oxford University Press. In Grove Music Online [Online; accessed 18. Jul. 2023]. <https://www.oxfordmusiconline.com/grovemusic/view/10.1093/gmo/9781561592630.001.0001/omo-9781561592630-e-0000049259>
- Wu, S.-L., & Yang, Y.-H. (2020). The jazz transformer on the front line: Exploring the shortcomings of AI-composed music through quantitative measures. *Proceedings of the 21th International Society for Music Information Retrieval Conference (ISMIR)*, 144–149.
- Young, H. (2017). A categorial grammar for music and its use in automatic melody generation. *Proceedings of the 5th ACM SIGPLAN International Workshop on Functional Art, Music, Modeling, and Design (FARM)*, 1–9.
- Zalkow, F., Balke, S., Arifi-Müller, V., & Müller, M. (2020). MTD: A multimodal dataset of musical themes for MIR research. *Transactions of the International Society for Music Information Retrieval (TISMIR)*, 3(1), 180–192.
- Zbikowski, L. (2002). *Conceptualizing music: Cognitive structure, theory, and analysis*. Oxford University Press on Demand, New York, NY, USA.
- Zhou, Z.-H. (2019). *Ensemble methods: Foundations and algorithms*. Chapman and Hall/CRC, Boca Raton, FL, USA.