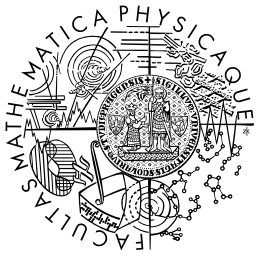


FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

DOCTORAL THESIS

Ladislav Maršík

Cover Song Identification using Music Harmony Features, Model and Complexity Analysis

Department of Software Engineering

Supervisor of the doctoral thesis: Prof. RNDr. Jaroslav Pokorný, CSc.
Study programme: Computer Science
Study branch: Software Systems

Prague 2019

I declare that I carried out this doctoral thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular, the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In Prague, 10th June 2019 signature of the author

I dedicate this thesis to my family. To my mother for love and understanding. To my father for support and for being my role model. To my grandmother Mária for care and motivation. To my sister and my brother for talks, compassion, and fun. And, to my fiancé, as she supported me in all of these aspects, ever since I moved to Prague.

This thesis would never have gotten from my mind to a paper without the support from my advisors and peers along the way. Prof. Jaroslav Pokorný is the best supervisor I could wish for, since the very first meeting on my entrance exams. He didn't hesitate for a second to take me under his guidance. In return, I wanted to bring work and results to make him proud.

I also want to thank for the administrative support and help provided by Eva Mládková. Thanks to her and Petra Novotná, the school office was a wonderful place to conduct my research.

During my studies, the most influential were my work visits and collaborations, as they have really shaped the thesis. I thus dedicate this thesis also to Pierre Hanna and Mathias Robine from LaBRI, Université Bordeaux, for facilitating my Ph.D. stage. To Martin Ilčík from ICGA TU Wien for the collaboration following my master's thesis supervision, and to the team comprising of Jan Martinovič, Kateřina Slaninová and Martin Rusek from IT4Innovations Ostrava. Last but far from the least, it was Yann Bayle from Université Bordeaux who's been my closest Ph.D. student peer, with ideas and help when I needed it the most.

This thesis was supported by several grants from multiple organizations, and could not be finished without their financial support: The Charles University in Prague, project GA UK No. 708314, GA UK No. 1580317, projects SVV-2014-260100, SVV-2016-260331, and SVV 260451. The collaboration with IT4Innovations Ostrava was supported by The internal grant agency of VŠB - Technical University of Ostrava, project no. SP2017/177 „Optimization of machine learning algorithms for the HPC platform“, by The Ministry of Education, Youth and Sports of the Czech Republic from the National Programme of Sustainability (NPU II) project „IT4Innovations excellence in science - LQ1602“, and from the Large Infrastructures for Research, Experimental Development and Innovations project „IT4Innovations National Supercomputing Center – LM2015070“. The computational part of the thesis was also supported by the project OPEN-11-47 „Deep Learning applied to Music Information Retrieval“ from IT4Innovations, granting core-hours on the computational nodes of the supercomputing center.

Title: Cover Song Identification
using Music Harmony Features, Model and Complexity Analysis

Author: Ladislav Maršík

Department: Department of Software Engineering

Supervisor: Prof. RNDr. Jaroslav Pokorný, CSc., Department of Software Engineering

Abstract: Analysis of digital music and its retrieval based on the audio features is one of the popular topics within the music information retrieval (MIR) field. Every musical piece has its characteristic harmony structure, but harmony analysis is seldom used for retrieval. Retrieval systems that do not focus on similarities in harmony progressions may consider two versions of the same song different, even though they differ only in instrumentation or a singing voice. This thesis takes various paths in exploring, how music harmony can be used in MIR, and in particular, the cover song identification (CSI) task. We first create a music harmony model based on the knowledge of music theory. We define novel concepts: a harmonic complexity of a musical piece, as well as the chord and chroma distance features. We show how these concepts can be used for retrieval, complexity analysis, and how they compare with the state-of-the-art of music harmony modeling. An extensive comparison of harmony features is then performed, using both the novel features and the traditional MIR features. Based on this comparison, the best features are proposed for the final experiments of the CSI task, with a result of 88.9% retrieval accuracy for a dataset of 2,000 songs using chroma features. The two methods used in our experiments are dynamic time warping and machine learning, for both feature comparison and our experimental results. To facilitate our research, a stand-alone application *harmony-analyser* was created and published online. Capable of music processing of WAV audio files, this application is proposed to the MIR community for feature extraction and harmony analysis. We have also published a dataset of karaoke songs *Kara1k*, which has been used for our experiments and contains a unique selection of features and annotations for future work.

Keywords: music information retrieval, music harmony analysis, cover song identification, feature extraction, chord distance, chroma vector distance, karaoke dataset, dynamic time warping, neural networks

Foreword

My years under the Charles University Ph.D. program have not exactly started with the topic of cover song identification. I entered the program as an enthusiast for music harmony and its automatic analysis, with a background in both computer science and music composition. The first Ph.D. months were spent on creating a music harmony model and defining the music complexity. A task difficult for musicologists themselves, it was a journey of trials and errors. If continued for the whole length of the Ph.D. program, this task would have easily formed a Ph.D. thesis by itself.

However, I could not avoid that I am also a software engineer. I wanted to wrap everything in a stand-alone application and deploy it online for the users. It was good that I took a software engineer approach to the musicology problem because suddenly a whole new world opened. I realized that people interested in creating such applications meet regularly at conferences and submit to interesting journals, a truly vivid and enthusiastic community of music information retrieval.

After I realized which research field I can call my own, I took an early research stage at LaBRI, Université Bordeaux. My consultant, Pierre Hanna, asked me which task do I think is the most capable of recognizing my research input. I admit that I was not sure at that time. It looked like any music-related task could benefit from having a music harmony module. Even my colleagues specializing in optical music recognition or genetic algorithms proposed that understanding music harmony could improve the accuracy of their systems.

But, the answer did not take long to appear. In our Latin music band based in Prague, we have been playing original songs and cover songs (our take on the original recordings by other authors) for years. The question „How to create a good cover song?“ has been a frequent topic in our band discussions. About the same frequent as were the questions of my friends from school and musicians inquiring about some good music detection applications. They often wanted to identify a song played live or by a different band in a different setup, and somehow their music detection apps had a hard time to do so. A simple answer to their questions was: „No, there is no fail-safe app for this yet, but my research will help to make that happen.“

Since then, I started to focus my algorithms on cover song identification. This task proved itself even more challenging than the task of a harmony model and complexity. It has a history, state-of-the-art applications, and benchmarking, but it also has inherent problems that are not solved yet. Some cover songs are so much different from the original song that even a trained musician would not be able to detect them quickly, and computer algorithms can be easily confused.

On the journey to detect cover songs using (among other techniques) a new harmony model, I hope the reader gets just about the right dose of music theory, a good understanding of the features, algorithms, and real-world results. The end should grant the reader prepared for any large-scale musical software project, as my Ph.D. program has granted me.

Contents

Introduction

„The most important in music is its harmony.“

Ilja Zeljenka, Slovak music composer

What makes the song or a musical piece stand out from another?

Most of the answers to this question would be very subjective. The melody, if it is original and distinct, can outshine the other aspects. While listening to the violin theme of the Schindler's List movie by John Williams, anyone would probably support this claim. But, is it only the melody, or also the sound of a musical instrument, played with the virtuosity of the player, that gives you the feeling of extraordinary music? The rhythm is quite simple in the movie theme, but the music originating in Africa or Latin America would certainly remind us that rhythm can be the main asset. On the other hand, while listening to Yesterday by The Beatles, the rhythm may be simple, but, arguably, it is the story and the lyrics that made the song famous. Accompanied by a masterful melody and harmony of the guitar accompaniment, of course.

The Slovak music composer Ilja Zeljenka chose harmony as the most important aspect, and his quote is often repeated by the musicians who knew him. We cannot blame them – what would become of our favorite songs, were they not accompanied by the guitars, bass or piano? Our choir experience would be „reduced“ to Gregorian chorals, with only a single voice melody.

In the following chapters, we are going to discuss algorithms that can take complex music patterns with multiple frequencies sounding together and retrieve similar patterns. Without acknowledging the presence of these multiple frequencies (the presence of harmony), our task would be impossible to solve from the very beginning. So while we cannot choose only the harmony aspect of music to judge the similarity, for certain tasks, such as cover song identification or genre detection, ignoring harmony structure would be like trying to play an instrument without proper music education. It may be doable for several pieces, but we would be missing the skill to master them all.

In this thesis, let us make a sincere attempt to acknowledge Zeljenka's quote while creating computer algorithms. If the following chapters can grant you an educated choice about harmony features, models, algorithms and possible results on musical datasets, it has fulfilled its purpose.

Using music harmony in algorithms

In a simple definition, a *harmony* is observed, when two or more tones are played simultaneously. Various fundamental frequencies included in this musical sound can be perceived by a trained musician. In this thesis, we show how a direct application of Fourier transform also grants this awareness to a computer. We also describe the possibilities of how the tones can be detected from the frequencies, making the computer aware of the musical scores. This set of routines form a low-level foundation for our research. Once we have the

tone material, a high-level music harmony features can finally be analyzed. And, from a perspective of researchers in music algorithms, this high-level understanding brings us multiple ways how to proceed. Few examples are listed below:

- It gives us new ways how to label a particular moment in a song. Musicology has given us a whole theory behind the naming of chords and harmony functions.
- It gives us plenty of ways of fingerprinting the whole song based on the harmony movements.
- It gives us a new dimension to work with. Suddenly, we can evaluate music in the „vertical“ dimension of multiple tones in a moment, rather than just a „horizontal“ dimension of the moments in time.
- One of the tones played simultaneously is usually the melody tone belonging to the singer's voice or the soloist performing a solo. We can observe how the melody tones relate to those of harmony.

For anyone interested in conducting research involving algorithms working with music, there is a field called *music information retrieval*. It contains many tasks for a researcher to study, and we present them in the subsequent chapters. Not surprisingly, many of the music information retrieval researchers have started extracting harmony features for their datasets or feeding their machine learning algorithms with them. This is to mimic the approach of a trained musician when solving tasks such as music similarity, music classification, or cover song identification.

From these tasks, we have chosen to focus on the *cover song identification*. A *cover song* is a new rendition of a previously recorded piece of music, usually performed by a different artist. The cover song identification task is thus deciding whether a song is a different version of another previous recording. The cover version can differ from the original recording in the tempo, instrumentation, artist, structure, or other aspects. It is the many ways that the cover song can differ from the original recording that makes this task very difficult (Ellis and Poliner [2007]). Still, there needs to be a handful of common aspects between the cover song and the original song (a cover pair) that the musicians preserve when reproducing the piece. Usually, it is the melody of the verse and chorus along with the underlying harmony progression.

Motivation for the thesis

There are several studies analyzing music harmony progression and trying to compare it in between different songs (De Haas et al. [2012]). Some of the progressions are complex, while some follow a simple pattern, which forms a good foundation to distinguish songs, artists, and genres. A notion of complexity has been used with the musical pieces (Knuth [1977]), but it has never been defined and is used vaguely in the literature. We have also perceived that in the literature, there is no standardized way to use music

theory models in the computation problems. The variety of models that one can use (Chew [2014], Lerdahl [2001], Lewin [1987]) can almost lead to confusion which one is the best for the task. Our initial motivation was therefore set to disambiguate the models and find (or create) one that can help us evaluate the complexity of a harmonic progression. As a result, we form a proper foundation for the coming tasks, as we believe that a thorough harmony analysis can help us detect cover songs.

The second motivation is of the real world. Analysis of harmony by an application based on a harmony model would be appreciated solely by music analysts and can hardly compete for popularity among other users. In the era of mobile applications capable of robust music detection, playlist generation, and more, users require fast solutions. We thus aim to improve the current popular applications in their weaknesses. Nowadays, the popular music detection applications, such as Shazam[1] or Soundhoud[2] are deemed to perform well for the task of music detection of a short excerpt, compared to the database of songs. The underlying algorithm is called *audio fingerprinting* (Wang [2003]), and as such, the algorithm is robust to the noise, but not to other variations between the queried excerpt and the expected match. In many real-life scenarios, such as using a live recording as the query, the audio fingerprinting would not be sufficient (Tralie and Bendich [2015]). Similarly, the pieces of classical music are always recorded in a different setup and conducted by a different conductor, and so classical music is rarely the same in two different databases. Relying on the database and a specific song can be noted as a main drawback of the current, otherwise very precise, algorithms. A more in-depth analysis of music can thus uncover such matching songs that audio fingerprinting could not.

We finally highlight that the third motivation for this work is beyond the theoretical model and the audio search. During our research, we have found out that there are many applications for this task, some of them less intuitive:

- **Plagiarism detection.** Somewhat tolerated by the music community as cover songs if the author is credited, but there is a great need for protecting the copyright in cases when the original author is taken advantage of. The infringing artist can change the lyrics keeping the same harmonies and melodies, often in the ways that would confuse audio fingerprinting approaches.

- **Optical music recognition.** As it has been pointed out by authors in the field (Rebelo et al. [2012]), the future work in optical music recognition of a music score implies the use of certain constrains to mitigate false note recognition. Such constraints can involve tempo or harmonic rules. In our work, we aim to evaluate how complex is a sequence of music notes, and such an approach can be used to determine the correct note.

- **Evolutionary algorithms.** The use of evolutionary algorithms is con-

[1] `https://www.shazam.com`
[2] `https://soundhound.com`

ditioned by the use of a fitness function to determine which individual should be removed from the population. Having a well-defined harmonic complexity is a possibility for a fitness function, e.g., if we wish to create music that is easy to listen to.

- **Music recommendation.** While collaborative filtering is a popular method to recommend music used by some of the leading platforms such as Last.fm[3], other platforms claim to use content-based approaches (Pandora[4]) where a manual input by trained musicians is used for tagging. A fully automated content-based method was pioneered by only a handful of applications to the best of our knowledge, such as Mufin[5] and the algorithms described by Schönfuss [2011]. As much as we realize that it is difficult to replace the collaborative filtering, we also are motivated by the fact that in the era of deep learning methods, one can teach the system to understand a particular aspect of music thoroughly. The user's playlist can be analyzed from different aspects such as genre, mood, melodic, or harmonic complexity; the user's preferences can be extracted and later used for recommending new songs.

Main contributions of the thesis

Without proper models and improving the basic toolset, it is hard to envision any of the above applications. In this thesis, we fill the gap where the researcher is left with no ways of evaluating music harmony, and we showcase some proofs of the benefits that music harmony features bring. Our results can be interpreted and used in multiple ways, but we consider the following as the main contributions of the thesis:

1. Developing a music harmony model useful for direct comparison of the songs and putting this model in the context of other harmony models.

2. Defining a harmonic complexity, a concept not clearly defined before although referenced frequently, to be used for various musicology or music information retrieval tasks.

3. The contribution of comparing all common harmony features in a way that was not done before (direct application of dynamic time warping as well as statistical approaches), including the newly formed features.

4. High-accuracy algorithm for cover song identification task, specializing in karaoke songs, which challenges the state-of-the-art algorithms in reducing the computation complexity of the algorithm.

5. Publishing the application *harmony-analyser* for the music analysis and the *Kara1k* dataset for cover song identification.

[3] `https://www.last.fm`
[4] `https://www.pandora.com`
[5] `https://www.mufin.com/usecase/music-recommendation`

In the thesis, we, therefore, undertake the whole path of the research, starting with the model, an overview of features, algorithms and ending with a dataset and application, without taking shortcuts or leaving much work for a third-party solution. As such, the thesis can additionally contribute as an overview of the research field.

We continue by showing how the contributions are set in the thesis and our related publications.

Published work

A substantial part of the content of the thesis has been, partially or fully, published as peer-reviewed research papers and a journal publication. The thesis can provide insights that were not included in the publications due to space limitations. On the contrary, the original and previously unpublished content includes Chapters 1 and 2, new experiments within Chapters 3 and 4, and multiple sections within other chapters.

The list of all publications is shown in Appendix 6.5, „List of Publications“. The initial research and partial works on the music harmony model and complexity were presented on four international workshops (two of them indexed in the Scopus database). The combined results were then gathered into the publication for the ACM SRC competition[6] (Maršík [2017]) within the SIGAPP SAC 2017 conference, where it was awarded the first prize out of 50 contestants. The comparison of harmony features via dynamic time warping was published in the IFIP conference CISIM 2017 published by Springer (Maršík et al. [2017]). A large part of our experiments and results, a combination of dynamic time warping experiments and statistical experiments for the musical features, was compiled in a journal article for International Journal of Semantic Computing, Vol.12, No.4 (Maršík et al. [2018]). The first presentation of our application *harmony-analyser* took place earlier, in a publication for 2016 Joint WOCMAT – IRCAM Forum Conference (Maršík [2016]. This international forum featured music technology innovations, included workshops by IRCAM[7] members, and also was featured as an ISMIR[8] satellite event. Together with attending the tutorials on ISMIR 2015, the presentation on the forum helped us to present our *harmony-analyser*[9] application to the MIR community (ISMIR being the largest MIR community event, IRCAM a world-renowned center for musical research). The karaoke dataset Kara1k (available online[10]) and cover song identification results were then presented within the Best Papers section on IEEE International Symposium on Multimedia (Bayle et al. [2017]) and was awarded a Honorary mention for the 6 Best Papers, which selected the topic for the extended journal publication. The research from this thesis was also presented on the Joseph Fourier Prize competition[11] and won the prize of the jury ranking as one of the top 4 Ph.D. topics from the Czech Republic in computer science in 2019.

[6] `https://src.acm.org`
[7] `https://www.ircam.fr`
[8] `https://www.ismir.net`
[9] `https://www.harmony-analyser.org`
[10] `http://yannbayle.fr/karamir/kara1k.php`
[11] `https://atos.net/cs/ceska-republika/cena-josepha-fouriera`

Outline of the thesis

The thesis is structured into six chapters and a conclusion, each solving one particular topic into the world of cover song identification based on music harmony. The reader can choose the chapters selectively, as every chapter has an end-to-end narrative. However, the first chapter is not to be omitted if the reader does not have a profound knowledge of music or music research, as it sets the vocabulary.

Chapter 1 is called „Understanding the terminology“ and it serves as a tutorial for the two main topics of this thesis: music harmony and music information retrieval. Not only it defines a vocabulary, but it also gives the reader an entry level to the music research accompanied by images and interesting case studies.

Chapter 2 „Related works in music information retrieval“ introduces the reader to the music information retrieval research that is closest to our work. The referenced works put together attribute for decades of research and best results from the benchmarks. We hope the reader finds the admiration and motivation in these works as we did.

Chapter 3 „Music harmony model and harmonic complexity“takes all the music theory definitions from Chapter 1 and puts them in a domain model, with music harmony being the domain. Using algebraic and graph structures, we show how to extract interesting features and define the harmonic complexity. A comparison is also shown with the state-of-the-art music harmony models.

Chapter 4 „Music harmony features comparison and cover song identification experiments“ takes us back to music information retrieval research. Before we perform our final experiments, we need to understand each of the features. One of the contributions of this thesis is showing an extensive comparison of features using both the novel and traditional features. We use both machine learning and time warping approaches to do so. We then specify the final experiments of the thesis. First, a discussion takes place, analyzing the outcome of the feature comparison and choosing the best methods for a subsequent final experiment on a dataset of karaoke songs. Another discussion is conducted about our results, where they stand in the state-of-the-art, and how can they impact future research.

Chapter 5 „harmony-analyser application“ then uncovers the system behind all of the feature extraction and experiments. While some of the experiments were done by a joint research team of Charles Univerity, Université Bordeaux[12] and IT4Innovations Ostrava[13] (collaborations much appreciated), most of the software used in this thesis is published as the *harmony-analyser*[14] project, founded by the author during his PhD programme, and open to public. This chapter gives an overview of the software and its features.

Chapter 6 „*KaraMIR* project and *Kara1k* dataset“ then shows how the „side product“of this thesis, a dataset of 2,000 songs, was converted into a

[12] `https://www.u-bordeaux.fr`

[13] `https://www.it4i.cz`

[14] `http://harmony-analyser.org`

standalone project and product, published for the community to use[15].

We finally conclude all the above topics in „Conclusion and future work“. There is one underlying thought across the chapters, and that is aiding the music information retrieval using the music harmony. We elaborate on how this was achieved and what is to be achieved in the years to come.

[15] `http://yannbayle.fr/karamir`

1. Understanding the terminology

This chapter offers important definitions, vocabulary, and background for the following chapters. The chapter has two sections: Musicology and Music Information Retrieval. These sections are quite distinct in the vocabulary, although they are both academic studies of the music domain. When there are common concepts, they often have diverse naming and definitions in the literature. For such concepts, we decided to present the more prominent name and definition. To avoid exhausting the reader with one grand dictionary, we prefer a fluent narrative with the explanations and figures.

1.1 Musicology

1.1.1 Categorization and background of the scientific fields

Let us first present the relationship between the music-related fields of science to understand the scope and background of this thesis. *Musicology* is the scholarly study of music. It is the top umbrella term that includes all musically relevant disciplines. It is considered a social science because it studies the art creations of mankind (Móži [1994]). However, there is an important subdivision of Musicology to the more social sciences such as *historic musicology* and *ethnomusicology* on one side, and *systematic musicology* on the other side. The last mentioned contains plenty of subdisciplines, usually of an interdisciplinary character, such as *music psychology* or *music acoustics*. The findings of these subdisciplines are thus often presented in a scientific format similar to psychology or physics.

We show the diagram of the scientific fields in Figure 1.1. Systematic musicology contains multiple fields of an interdisciplinary character standing in between musicology and psychology, physics, mathematics, or computer science. From this diagram, the most important parts for this thesis are:

- **Music theory**. Music theory is a parent term for disciplines working with musical structures, forming the rules of music composition and the rules for creating melodic, rhythmic and harmonic objects and their connections (Pospíšil [1985]). Music theory is considered as old as the music itself, as the founding works have been assembled in ancient Greece by Ptolemy or Aristoxenus, while the founding stone of acoustics and ratios involved in music is attributed to Pythagoras. The subdisciplines of music theory include music notation or tonal harmony, and within music theory of 20^{th} century, several direct applications of mathematics have been studied, such as the set theory or algebraic structures, resulting in a strong connection between the two fields (Lewin [1987]). Within this thesis, we:
  - Describe the basic music theory definitions in Section 1.1.2.

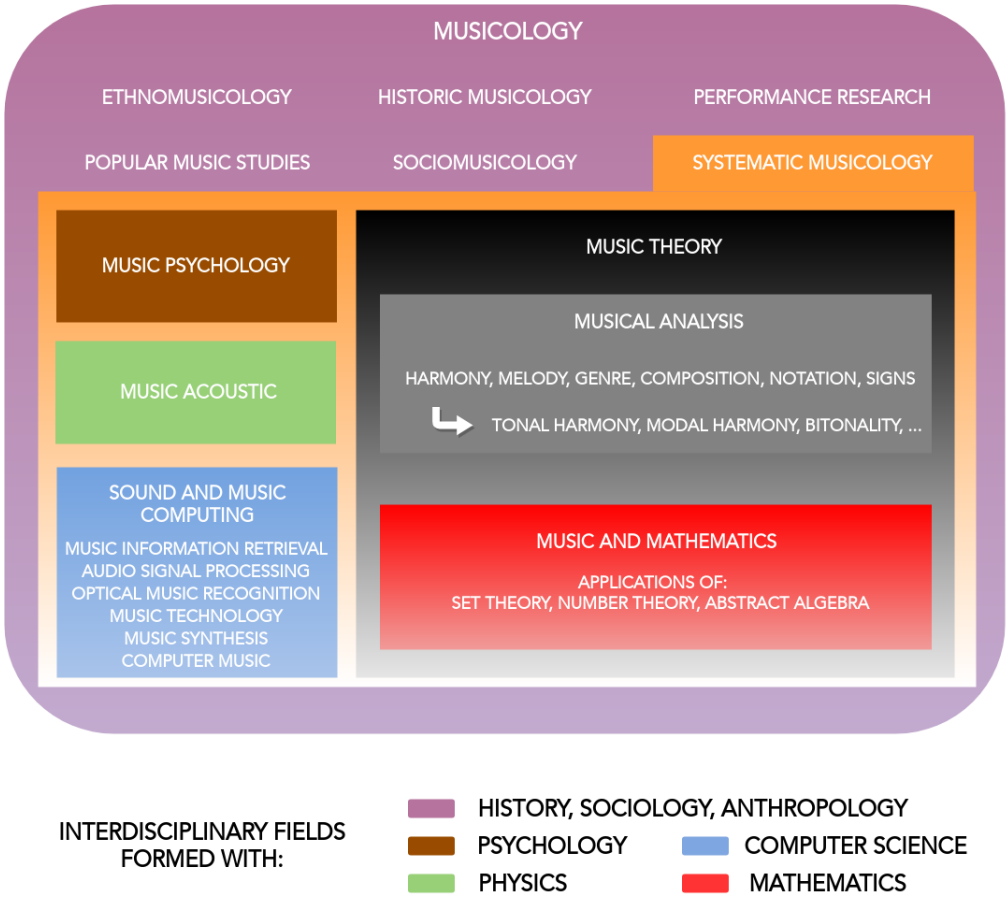


Figure 1.1: Categorization of the research fields related to musicology

- Introduce and reference several works of music theory related to our thesis in Chapter 2

.

- **Music psychology**. Although it is less obvious, music psychology contributes much of its research to music harmony modeling and thus is substantial for the thesis itself. The history of music psychology spans to the latter 19^{th} century, and the research areas include music cognition, perception, or psychoacoustics, among many. This thesis aims at creating a new music harmony model useful for music applications, and although such model serves other purposes than those of cognition and perception, the sophisticated models of music psychology are referenced and compared often throughout the work (Lerdahl [2001], Krumhansl [2004]).

- **Tonal harmony**. Tonal harmony is a musical system within music theory originating in the late Baroque in 18^{th} century, for western music. It defines the notion of musical keys, and the system is built on a principle that every part of a musical piece belongs to a particular key. After reaching its peak in music Romanticism in 19^{th} century, many composers have utilized novel approaches to music, moving outside the keys and breaching the tonal harmony rules, which were before unmatched. Nevertheless, the rules of tonal harmony still apply to a vast majority of music today. Notably, as is pointed out by Krumhansl [2005], music theory benefited greatly from the development of tonal harmony, and although the music theory continues to evolve after passing of the tonal harmony „doctrine“, it has kept the basic tonal harmony rules and is commonly being used to teach the basics of harmony. The aspects of tonal harmony important for this work are described in Section 1.1.3.

- **Sound and music computing** and **music information retrieval**. We could look back as far as 1874 for the dawn of electronic music technology when prof. Elisha Grey invented the first music synthesizer, „The music telegraph“ (Hounshell [1975]). The inventor who was racing for the telephone patent with Alexander Graham Bell founded a whole new industry, unknowingly. Nowadays, there is a number of new research fields between music and technology, and the term *sound and music computing* serves merely as a parent term for those fields that would include computer science or computation, some of them more tied to musicology, and some of them less. It contains both traditional fields (music technology, audio signal processing) and emerging new fields (music information retrieval, optical music recognition). The *audio signal processing* field has been developed for decades, and contributed a strong foundation for the further analysis sound, both non-musical and musical (we describe these foundations in Section 1.2.3), In the 21^{st} century, it has become a frequent need to search for music, automatically classify music, analyze and visualize music,

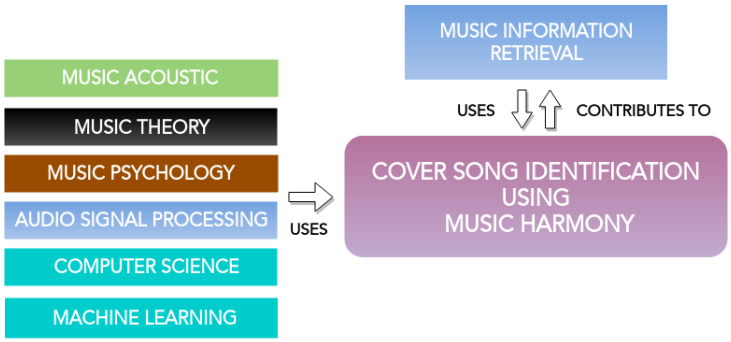


Figure 1.2: Interdisciplinary research of the thesis demonstrated. The direct results of the fields on the left are presented and used throughout the thesis, while the thesis itself contributes to the music information retrieval field.

> or, recommend music. As a result, a new field emerged, called *music information retrieval*. In one of the premiere works, Fujishima has compared the chroma vectors to a chord dictionary and obtained chord progressions in 1999 (Fujishima [1999]). Since then, almost every year a new task is formed based on users' needs and the scientists' race to solve them in the benchmarking competitions, such as MIREX[1]. In Section 1.2 we show the main music information retrieval definitions and in the related works Section 2 we show the main achievements for harmony analysis and cover song identification task.

This thesis is based on taking the knowledge from music acoustics, music theory, and music psychology to understand the music domain. For our algorithms, we use the foundations of audio signal processing on one side, and methods of computer science and machine learning on the other side. But, the thesis falls within the music information retrieval scope, as the final product is an algorithm to analyze and retrieve music tracks.

We wrap up the section with a statement that the thesis has an interdisciplinary nature, as can be observed in Figure 1.2, although the focus is on algorithms and computer science experiments. As such, we need to acknowledge the problem of the vast number of related works of a very different background, and theories that are not necessarily converging towards the same results (Gómez [2006]). We also face the bias towards one of the disciplines (arguably, computer science), as was pointed out well by Vos [2000] in a similar situation of clashing disciplines: *Because nobody is equally specialized in music theory, music history, psychoacoustics, music psychology, and so on, most theoretical approaches of tonality induction are biased towards one of the various scientific disciplines.*

1.1.2 Music theory

We follow by a series of definitions, referencing and respecting the original works by Riemann [1896a] and Schönberg [1922], as well as the traditional teaching methodology by Zika and Kořínek [1990] and Pospíšil [1985]. As the

[1] `http://www.music-ir.org/mirex/wiki/MIREX_HOME`

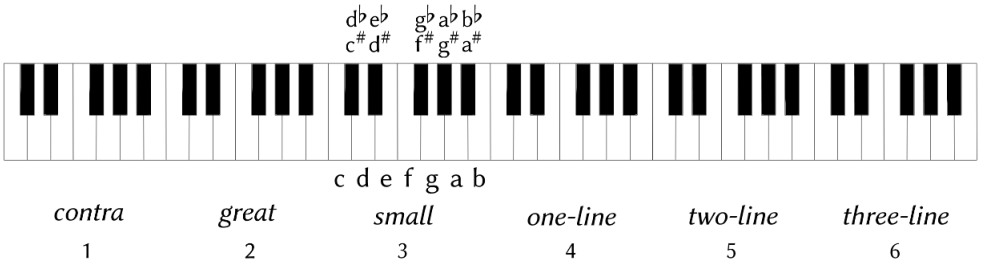


Figure 1.3: Tones arranged on the piano keyboard in six octaves, in a repeating pattern where c, d, e, f, g, a, b are the tones that are being repeated. The tones may have ambiguous naming in general, as is seen e.g. for the tone $c\#$ (alternatively $d\flat$) in between c and d tones.

majority of the thesis is technical rather than musicological, we also follow the more technical versions of definitions, as done by Krumhansl [2005] or Chew [2014]. The reader may notice that the definitions among the related works are not unified, and thus, a unique vocabulary is required for the thesis.

Basic definitions

Definition. *A* tone *is an acoustic sound that is created by the vibration of the source material, and can be described by its pitch, intensity, timbre, and duration.*

Music theory also defines the tone as the smallest element of a musical piece. The *pitch* can be quantified using a frequency, but it takes a comparison of a complex music sound to a pure tone with the sinusoidal waveform to determine the actual pitch. Therefore, the pitch should be considered as a subjective attribute of the sound. The *intensity* is commonly denoted in dB, *duration* in ms, and the *timbre* is a term specifying the different source (musical instrument or voice) originating the sound. We reference Figure 1.3 to see how the tones are arranged on a piano keyboard, within six octaves, ranging from *contra* octave to a *three-line* octave.

For a better understanding of the tones and their naming, Attachment A.1 *Tones dictionary* can serve the reader with the naming conventions and more insights about the reasons for the naming. We highly recommend to review this and later musicology attachments A.1 – A.5, especially if the reader is not experienced in music theory and needs more examples. The content of the attachments is, however, not a prerequisite for understanding the main part of the thesis. It is thus sidelined and can be used as a reference.

Definition. *An* interval *is a frequency ratio of two pitches, the simplest relationship between the two tones in music.*

The music acoustics describes a *harmonic series* as a key to understand the basic intervals and pitches in music, according to Schönberg [1922]. A harmonic series is a series of frequencies forming a musical sound. For a

source material oscillation observed as a musical sound forming a tone, it is the existence of additional oscillation nodes and partial waves that represents the harmonic series.

Figure 1.4 explains the basic idea of the harmonic series on a musical sound produced by a string. The fundamental frequency F_0 of the fundamental tone is the most prominent frequency perceived. The first overtone is formed by an additional oscillation node in the middle of the string. The first overtone and the fundamental tone have a frequency ratio $2:1$, denoted as the *perfect octave* interval. The frequencies ratio $3:2$ is denoted as the *perfect fifth* (the interval between first and second overtone). Continuing the same pattern and forming the less noticeable overtones by frequencies $3 \cdot F_0$, $4 \cdot F_0$, etc., the other intervals can be observed with the ratio $3:4$, $5:4$, and so forth:

- The frequency ratio $4:3$ is denoted as the *perfect fourth*.
- The frequency ratio $5:4$ is denoted as the *major third*.
- The frequency ratio $6:5$ is denoted as the *minor third*.

While these five intervals are of the highest importance for harmony, we encourage the reader to observe the full dictionary and examples in Attachment A.2 *Intervals dictionary* for more insights, and also to understand, why following the pattern further from the ratio $7:6$ would not be fit for defining more intervals.

Although we employed a fundamental definition of intervals by Laborecký [1997] using the frequency ratios, there is a more intuitive, practical definition of intervals as a distance between the two tones, described in most of the music theory literature (Riemann [1896a]). Considering the piano keyboard as a visual aid (Figure 1.3), the unit for measuring an interval is a *semitone*, the smallest distance between the two tones (e.g., between the tones c and $c\#$). A *whole tone* is a common name for an interval of two semitones (e.g., between the tones c and d). We can thus describe the intervals alternatively as the number of steps between the two particular tones (example: 4 semitones or 2 whole tones in between c and e). For an interested reader, the dictionary in Attachment A.2 is the best way to link the interval names with frequency ratios, semitone distances, and – same as with the tones – to understand the caveats of an ambiguous naming.

Definition. *A* scale *is a series of increasing or decreasing pitches bounded by an octave.*

There are multiple types of scales, but in our work, we focus on the *diatonic scales* which form the basis of the western music. Diatonic scales are scales created by a semitone and whole tone intervals, and they contain eight tones. We further divide two *modes* of diatonic scales. *Major scales*

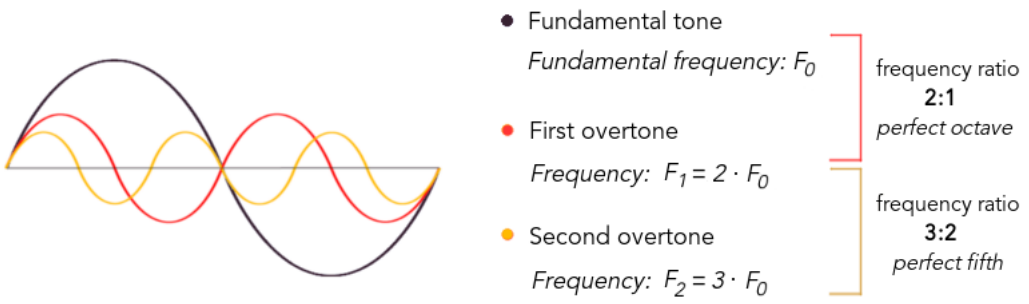


Figure 1.4: Harmonic series explained on a musical sound produced by a string, shown as a composite of a series of tones sounding together.

are the diatonic scales characterized by the presence of major thirds. The most common major scale is *C major*, formed from the basic tones shown in Figure 1.3: c, d, e, f, g, a, b, c (note the added c tone, representing the final tone, one octave up from the initial c). *Minor scales* are the diatonic scales characterized by the presence of minor thirds. The most common minor scale *a minor* is also formed from the basic tones, but from the tone a: a, b, c, d, e, f, g, a. Major scales are commonly assigned a „joyful“ character, while minor scales are commonly assigned a „sad“ character. The index of a specific tone in the scale is called the *degree* of the scale and is denoted by a roman numeral (I. degree in *C major* is the tone c, II. degree in *C major* is the tone d, etc.). We again provide Attachment A.3 *Scales dictionary* outside the main line of the text, for the reader to reference the basic scales and their degrees when necessary.

Definition. *A* chord *is a set of tones with a minimum of three tones, forming a basis of harmony. One of these tones is prominent in the harmony of the chord and is called a* chord root *of the chord.*

The other definitions of a chord also specify that the intervals between the chord tones are large enough, so that they may sound together without the feel of excessive density (Laborecký [1997]). However, for our thesis, the density and dissonance bear meaning for our analysis, so we prefer the more general definition.

The *chord root* is the tone upon which the chord can be built by stacking thirds intervals. If the chord root is indeed the bottom tone of the chord, we say that chord is in a *root position*. We can also obtain the *chord inversions* by reorganizing the tones in such manner that the root of the chord is put to the top of the chord – *first inversion* – or as the second from the top – *second inversion* – and so on. We use the term *chord tone* for each of the tones within the tone material of the chord in the context. The term *non-chord tone* denotes a tone out of the tone material of the chord. By the *tone material* we consider the tones *mapped* to one octave, i.e., taking the tone c as a chord tone if the tone c from any octave is present in the chord.

We further define a *triad* as a chord in the root position made up of three tones: the root tone (I. degree), the III. degree and the V. degree. A *Major*

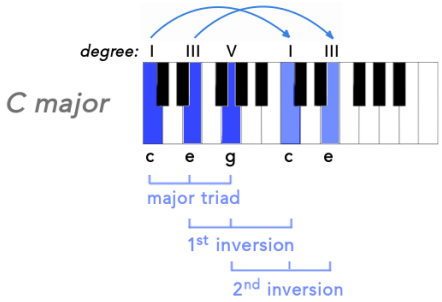


Figure 1.5: *C major* triad formed by stacking the thirds intervals on the chord root, tone *c*. By reorganizing the tones such that the I. degree is put on the top of the chord, we obtain the 1st inversion, and by putting the III. degree on the top of the chord, we obtain the 2nd inversion.

triad is formed from the tones of a major scale and a *minor triad* from the tones of a minor scale. The triad is the most referenced and used chord within the tonal harmony, as it represents the harmony of a tone with its closest overtones. The visual representation of a major triad and its inversions can be found in Figure 1.5. More information about the triads, inversions, and their naming can be found in Attachment A.4, *Chords dictionary*.

1.1.3 Tonal harmony

In the late Baroque, in the 18th century, the tonal harmony system emerged on the top of the definitions mentioned earlier. Most importantly, the diatonic scales were assigned a significant role in music composition by providing the tone material that the composers could use to create a musical piece. The scales were also extended to form broader relationship systems, called *keys*, and keys became the backbone of the *tonal harmony* system.

Definition. *A* key *is a relationship system based on a major or minor scale. There are three basic levels of relationships that define the key:*

1. *A series of tones: the major or minor scale.*
2. *A set of chords designed to build harmonies: the triads built on every degree of the scale, made out of the tones of the scale.*
3. *A basic chord series also called* harmonic cadence*, that sets apart the triads on the I., IV. and V. degree of the scale and gives them the role of* main harmonic functions*.*

Definition. Tonal harmony *is a musical system, in which:*

1. *Every part of a musical piece belongs to a major or minor key.*
2. *Every harmony has some, close or distant relationship to the center of the key, the I. degree.*

The keys are named by the scale they are based on, e.g., *C major* key, *a minor* key, etc. We refer to the tones of the key using the degree – same

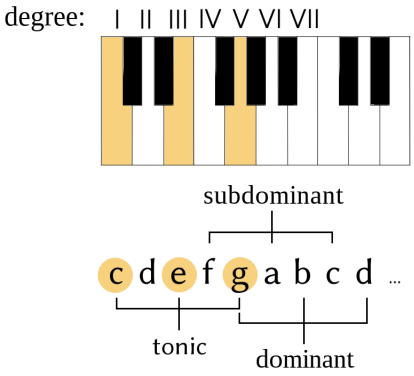


Figure 1.6: Main harmonic functions of the *C major* key

as with the scales – and we additionally allocate a special term for the I. degree: *tonic*, the base of the key. If we consider an equal-tempered tuning of the 12 semitones within the chromatic octave (discussed more in depth in Attachment A.2), there is a total of 24 scales and keys. They can be arranged on the *circle of fifths*, shown in Attachment A.3, which is the first example of a model where we can observe distances between the entities other than the basic semitone distances. Such models are a subject of our study in the Chapter 2.

We further make use of the concept of *main harmonic functions* characteristic for each key, demonstrated in Figure 1.6. The harmony in music usually starts with the *tonic* (the same name as the I. degree is also used for the triad on the I. degree) denoted by a capital T, a function of harmonic steadiness and release. Optionally, it deviates to the *subdominant* (triad on IV. degree, denoted by a capital S). Finally, the harmonic movement culminates in the *dominant* (triad on V. degree, denoted by a capital D), a function representing the maximal tension, requiring a transition back to the tonic. According to Zika and Kořínek [1990] the T – (S) – D movement is a skeleton of every music motion in musical pieces in the tonal harmony system.

In addition to the three main harmonic functions, there are also variants, or parallels, of the main harmonic functions. We provide a simplified definition of a function parallel based on the original definition by Riemann [1896a]. While this definition should suffice for the reader who has basics in tonal harmony, we refer all other readers to Attachment A.5 *Harmony functions dictionary*, where more examples and background is explained.

Definition. *A* function parallel *is a chord created from the main harmonic function either by extending the highest tone (i.e., V. degree in the tonic, I. degree in the subdominant and II. degree in the dominant) by a whole tone, or by diminishing the root tone by a semitone. We denote a parallel by adding a „P“ subscript to the function label ($T \rightarrow T_P$).*

An example parallels for the *C major* chord c, e, g are c, e, a or b, e, g.

We also add a definition from one of the modern interpretations of tonal harmony described by Volek [1988] and based on Leoš Janáček's concept of

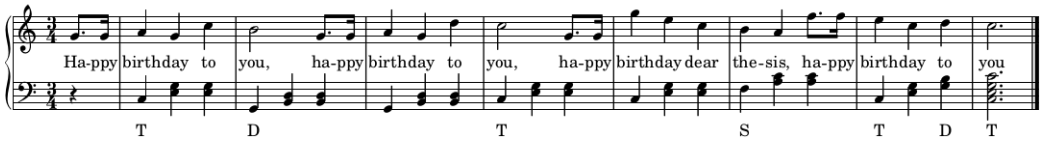


Figure 1.7: Happy birthday song in *C major* key, annotated with the main harmonic functions: T (tonic, *C major* triad), S (subdominant, *F major* triad), D (dominant, *G major* triad).

added dissonances to the chord. We commonly refer to the intervals of whole tone or a semitone (major and minor second) as *dissonant*, as opposed to, e.g., major and minor third, which are *consonant* (reference Attachment A.2 with all intervals and their types). By „adding a dissonance“ we thus mean adding a dissonant interval to some harmony.

Definition. *A* chord with an added dissonance *is a chord enriched with tones that did not originally belong to the chord (non-chord tones), thus creating a whole tone or a semitone dissonance.*

An example of a chord with an added dissonance is c, e, f, g, with the tone f added to the original *C major* chord c, e, g.

With all the main definitions of tonal harmony now in place, we show an example to illustrate the fundamental harmony analysis.

Example. Some well-known tunes have a simple harmonic structure, for example *Happy Birthday*: $T - D - T - S - T - D - T$ (Figure 1.7). However, an example of a more complex composition is Bedřich Smetana's *Vltava* from *Má vlast*, where the harmonic structure can be interpreted as: $T - S_P\ (D) - T_P\ (T) - S - T - D - T$. We can notice the use of the function parallels in a more complex musical piece. The „$S_P\ (D)$“ notation means that the harmony can be interpreted in two ways. As a subdominant parallel, or, if we consider that music has switched to a different key temporarily, the same harmony can be interpreted as a dominant in that key. Such ambiguity is common when analyzing difficult harmony progressions.

There is a possibility to dive deep into the manual harmony analysis as there are many more rules and options. However, as that would be outside the scope of this thesis, we continue with the computational approach and we refer the reader interested in more details to Zika and Kořínek [1990].

1.2 Music Information Retrieval

We now switch to the computational approach to see how the concepts of music theory can be captured in the computer systems. This section makes sure that there is no confusion among the terms of the *music information retrieval* (MIR) and related fields, and it describes the basic algorithms behind the cover song identification task.

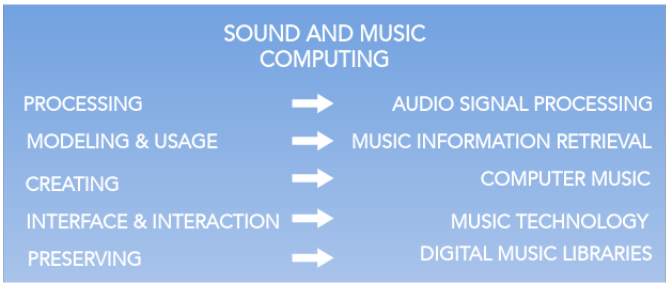


Figure 1.8: Sound and music computing as a parent discipline of multiple fields, including music information retrieval. The focus of the research is shown on the left side and a chosen representative research field on the right side.

1.2.1 MIR scope and tasks

MIR research falls within the *sound and music computing* (SMC) scope, as well as other related fields with various focus. Figure 1.8 shows the other specializations of SMC, along with their closest research disciplines. *Audio signal processing* focuses on the electronic manipulation of audio signals. MIR focuses on retrieving information from music, modeling, and using that information in the applications. Modeling is also the main topic of music cognition and related sciences. *Computer music* is applying computing in music composition to help composers create music (also known as *music synthesis*), or to let computers create music independently, e.g., via *algorithmic composition*. *Music technology* (electronic and digital) creates tools to satisfy the needs of a wide range of musicians, composers, sound engineers, or producers. Other related fields include *music interfaces* or *computer-human interaction*. Although we could continue further with more and more specializations, we consider the emerging need for preserving music libraries as a must-have in our selection, as there is trending research and annual meetings in both *optical music recognition* and *digital music libraries*.

In this thesis, we model the musical sound and use that information to search and retrieve cover songs, which clearly falls within the MIR scope. The *cover song identification* (CSI) task is one of the many tasks available within MIR. Any task that computationally retrieves or analyses music data, creates models, searches for music, or classifies music, would likely be a MIR task. Some examples of the popular tasks include *audio classification*, *audio fingerprinting*, *audio melody extraction*, *audio key detection*, *audio chord estimation*, *drum transcription*, or *music and speech detection*. These and multiple more were offered in the 2018 rendition of MIREX benchmarking[2].

1.2.2 Cover song identification

The aim of the CSI is to retrieve or identify a different version of a previously recorded musical piece. A cover version usually shares the title with the original song, but it may differ in many ways. Such variations can include a change in the singing style, key, tempo, lyrics, genre, or recording conditions.

[2] `https://www.music-ir.org/mirex/wiki/2018:Main_Page`

This implies that the audio features that solely rely on the style, key, tempo, lyrics, or genre are not sufficient to perform a CSI task. Clearly, the CSI is considered a very difficult task (Ellis [2006], Osmalskyj [2017]) because of the many ways in which the cover version can differ from the original recording.

In the CSI literature, there is no consensus on whether the query song should be the cover version or the original recording. This may be because at some times it is difficult to label one recording as the original (consider multiple versions of *Happy Birthday* song). But, as in this thesis we mostly focus on an original-to-cover song comparison, we internally define the query as the original recording made by the artist, and we perform the search for the cover song in the dataset.

There are also two different approaches to define the results of CSI. Either we wish to implement a *query-by-example* model, where for the query song, the most probable covers are returned in decreasing order. Alternatively, we wish to perform a *general classification task*: given two audio recordings, decide whether they are a cover pair or a non-cover pair.

A similar ambiguity is present also with two possible types of experiments for the CSI task. The first type is the most common in the literature and considers that for each original song, the corresponding cover song is present in the reference collection. In the second type, the corresponding cover song might not exist in the collection and requires us to decide whether there is a match. However, these issues can be tackled by a single system, if a good similarity model is introduced. For the most CSI systems, a pairwise comparison is made and a score calculated between the query track and each track of the reference collection. The collection can then be sorted, with respect to the similarity with the query. The final decision is then made based on a similarity model, which can utilize a threshold or a more complex solution.

Osmalskyj [2017] describes the cover song identification in three steps:

1. Feature extraction step
2. Feature post-processing
3. Similarity estimation step, with the possible use of a similarity model trained on a separate collection using a machine learning algorithm

Figure 1.9 shows the outline of such CSI system. Query song and reference collections are processed, and features are extracted (example: chroma features, or MFCC features, see Section 1.2.4). Usually, the features are post-processed to build a higher-level description of the audio tracks (example: simple aggregation). An optional learning phase can be conducted for the dataset after the same feature extraction is done for a learning collection, to build a more sophisticated similarity model. Finally, the similarity model (example: Random forest) decides whether the cover song is present in the reference collection, for the given query song. The definitions for the used example features and models are in Section 1.2.4.

The feature extraction and post-processing steps are necessary for all the songs, including the query song, reference collection, and the learning collection. After the initial extraction of the features from the audio signal,

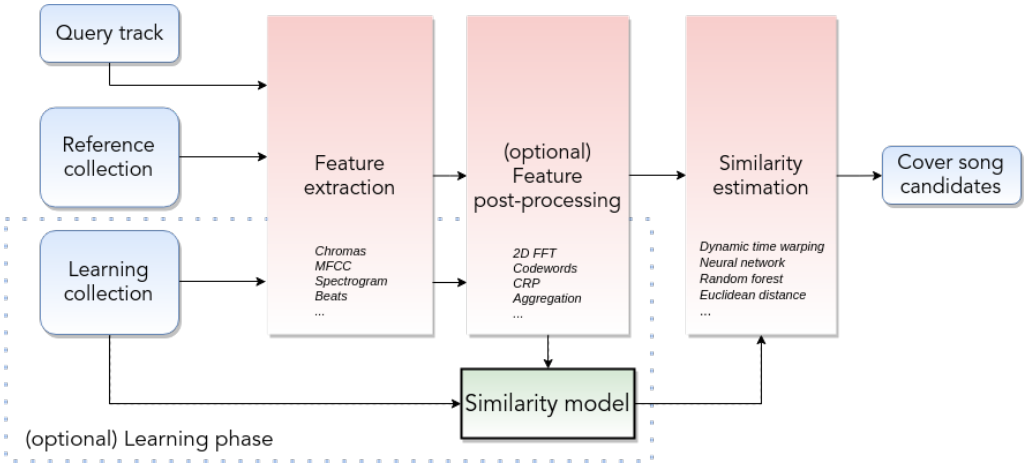


Figure 1.9: A cover song identification system outline inspired by Osmalskyj [2017]. The arrows show the order of the steps and the optional learning phase is shown in the dotted area.

post-processing algorithms are applied to build a higher-level description of the audio tracks. If we possess a ground-truth for the training collection (*is a cover pair* vs. *is not a cover pair* for each pair of the songs), we can issue a learning algorithm to build a supervised similarity model. If not, we can employ a more straightforward model – or a direct comparison of the features – based only on the immediate content of the two songs. Either way, we use this model to get the final result, determining if a cover song was found for the query song.

Note that the post-processing step may be optional, depending on the structure of the given features. For example, the spectrogram can be a direct input for the learning phase and similarity estimation, if the model can handle the spectrogram's dimensions. In reality, however, this is a rare case, and some kind of processing is useful before the similarity estimation.

1.2.3 The basics of signal processing

The typical feature extraction for any harmony feature starts with a *Discrete Fourier Transform* (DFT), shown in Equation 1.1.

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi kn/N} \qquad k = 0, \ldots, N-1 \tag{1.1}$$

The vector x represents an input signal, in the form of a series of N samples of the sound. $e^{-j2\pi kn/N}$ represents a complex sinusoid (j is the imaginary unit), which we multiply with the input signal, sample by sample. We sum the products over the number of samples of the signal, and the result is assigned to the vector X representing the frequency spectrum. k is representing the discrete frequency index and iterates over $0, \ldots, N-1$ same as n, the discrete time index. $2\pi k/N$ from the complex sinusoid represents the frequency in radians (ω_k), and if we know the sampling rate of the audio f_s, we can obtain the frequency in Hz at the index k as $f_k = f_s k/N$.

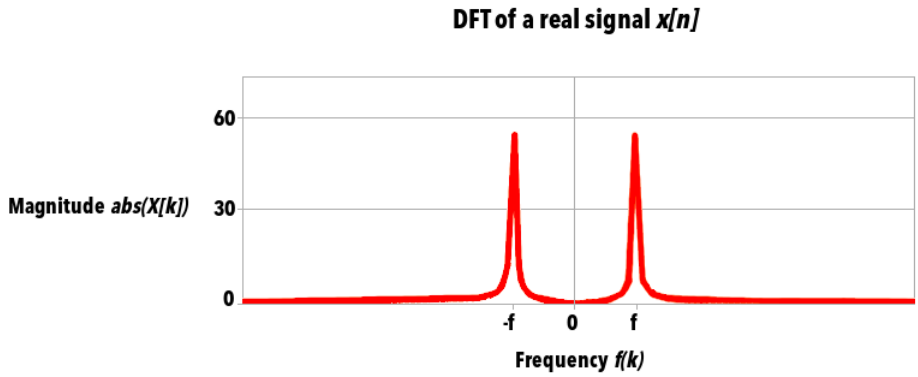


Figure 1.10: The magnitude spectrum that can be obtained from the DFT (Equation 1.1), in case that $x[n]$ is a real signal (with no imaginery part). A spectrum of a tone sounding at the frequency f is plotted.

The DFT provides us with the frequency spectrum of the signal, which can later be used to extract the tones and harmonies. One interesting aspect of the DFT is that for real signals ($x[n] \in \mathbb{R}$, no imaginary part), the spectrum is symmetric and the negative side of the spectrum is a mirrored version of the positive side (see Figure 1.10). This stems from the fact that a complex sinusoid is used in Equation 1.1 to multiply with the input signal. As a result, it is usually suitable to consider only the first half of the computation, i.e., the index k ranging from 0 to $N/2$ as we can observe in the later definitions of the features.

In its original form, the DFT takes the whole time-based signal of the time length N and outputs only one frequency-based spectrum. To capture frequencies of *each moment* in the musical track, the *Short-time Fourier Transform* (STFT) was introduced, that utilizes a sliding window of a small length to focus DFT on all the particular moments:

$$X[k] = \sum_{n=0}^{N-1} w[n]x[n + lH]e^{-j2\pi kn/N} \qquad k = 0, \ldots, N-1 \qquad (1.2)$$

The reader may notice that the definition of STFT is very similar to that of the DFT, with the addition of a function $w[n]$ along with arguments l and H next to the time index n. $w[n]$ represents the *window function* that shapes the analysis window. The analysis window is multiplied with the excerpt of the input signal to make sure that the overlapping excerpts blend into each other well in the final spectrum. There are multiple options for the window function, but *Hamming* or *Hann* window are among the ones commonly used. l represents the frame number (i.e., which iteration of the windowed DFT is being computed). H represents the hop-size, which says how much the window moves on the time axis before the next windowed DFT computation.

The Figure 1.11 shows the STFT and its windowing mechanism. The final output of STFT is a series of power spectra in time, commonly referred to as a spectrogram. As can be seen in the figure, the analysis starts with

dividing the signal into small windows. DFT is then applied, frequency spectra are computed and later visualized in the spectrogram. Figure 1.11 also shows the possibility of further processing the spectrogram to obtain more high-level features. The creation of a chromagram is depicted as an example of more high-level features, which are essential for our thesis. In the next section, we take a closer look at these features and how to create them.

1.2.4 Basic definitions

In this section, we continue with the basic definitions of high-level features used in this thesis. The terms that we define here are commonly used by MIR researchers, and it is necessary to know them to comprehend the rest of the thesis fully.

Chroma features

Chroma features are commonly referring to a series of 12-dimensional vectors of floating-point numbers, capturing the presence of each tone in a short music moment (Gómez [2006]). The concept has been proposed and studied at the beginning of the 21st century by multiple authors; we first became aware of it in the work of Fujishima [1999] and Bartsch and Wakefield [2001]. Chroma features are obtained directly from the DFT output by grouping frequencies that belong together in one frequency bin, where the bins are set by the tempered tuning of the piano scale.

A series of 12-dimensional chroma vectors can also be referred to as a *chromagram* and is often visualized as we saw in Figure 1.11, *D)*. In the literature, we can find the terms chroma features and chromagram used interchangeably, both referring to the same series of vectors. But notably, the first introduction of the term chromagram was accompanied by certain constraints for its computation specific for a musical key estimation (Pauws [2004]).

For the purposes of the thesis we employ the following general definition of a chroma vector:

Definition. Chroma vector *is a 12-dimensional vector:*

$$< c_A, c_{A\#}, c_B, c_C, c_{C\#}, c_D, c_{D\#}, c_E, c_F, c_{F\#}, c_G, c_{G\#} >$$

where $c_A \in \mathbb{R}$ *represents the presence of the A tone,* $c_{A\#} \in \mathbb{R}$ *represents the presence of the A# tone, etc.*

The value distribution of $c_A, c_{A\#}, \ldots$ depends on the algorithm used for evaluating the presence of the tones, but it is a common practice to normalize to $[0, 1]$ interval, where the loudness ratio is preserved in between the tones. This definition intentionally avoids specifying an algorithm, as there are indeed multiple possible approaches, often depending on the task at hand. The initial work of Fujishima [1999] specifies the definition of *Pitch Class Profiles* which are a good example of simple chroma features:

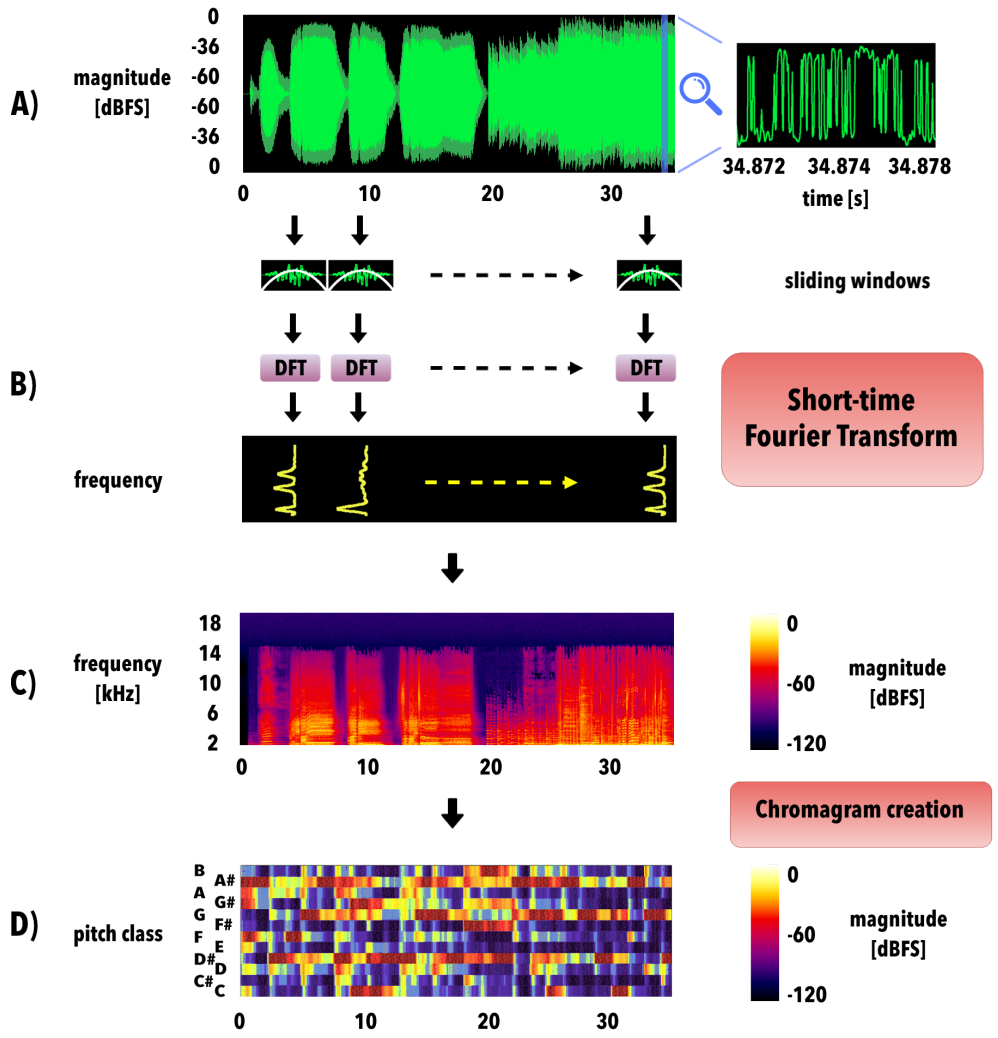


Figure 1.11: The basic audio signal processing to obtain harmony features is depicted.
A) The original sound plotted as a waveform signal in time, with the amplitude on the y axis.
B) The STFT algorithm breaks down the signal using a window function, and with the window sliding from left to right the DFT is used to compute the frequency spectrum for each moment.
C) The final result of STFT is a spectrogram, with frequency on the y axis.
D) The spectrogram can be further processed into the chromagram, with the 12 pitch classes on the y axis.

Definition. *For the frequency index k of the DFT spectrum, let us define a table $p(k)$ that maps the frequency to* 12 *bins* $0 \dots 11$*, along with one unused bin of* -1*, as:*

$$p[k] = \begin{cases} -1 & for \quad k = 0 \\ round(12 \cdot log_2(\frac{\frac{f_s \cdot k}{N}}{f_{ref}})) \mod 12 & for \quad k = 1, \dots, N/2 \end{cases} \tag{1.3}$$

where f_{ref} is the reference frequency of the bin 0 *– e.g. a frequency of the A tone,* $440Hz$ *– and f_s is a sampling rate.*

The Pitch Class Profile *(PCP) is a 12-dimensional vector that can be computed from the DFT spectrum $X[k]$ and $p = 0, 1, \dots 11$ as:*

$$PCP(p) = \sum_{k:p(k)=p} |X[k]|^2 \qquad k = 0, \dots, N/2 \tag{1.4}$$

The first part of this definition assigns every frequency to one of the 12 bins, with the exception of the first frequency $k = 0$ as the logarithm would be undefined. As we recall from the previous section on signal processing, the sampling rate f_s is related to the highest possible frequency in the DFT spectrum (the x axis spans $[-\frac{f_s}{2}, \frac{f_s}{2}]$). For the real signals, it is enough to observe the first half of the spectrum ($k = 0, \dots N/2$). Thus, the term $\frac{f_s \cdot k}{N} = f_s \cdot \frac{k}{N}$ represents the frequency of the spectrum bin k (f_k, see Section 1.2.3). And so the input to the logarithm is a ratio of f_k with the reference frequency f_{ref} and we can see that when $f_k = f_{ref}$, the bin 0 is used ($log(1) = 0$). The other frequencies f_k are assigned to the bins according to the so-called *tempered tuning* ratios (reference Attachment A.2 for more information). The second part of the definition is a simple summation of square magnitudes of the frequencies according to the bins.

In line with this definition, the term *pitch class* is used in MIR to indicate a set of all pitches that are a whole number of octaves apart, e.g., the pitch class c consists of the tones C in all octaves.

The PCP using the summed magnitudes proved to be a useful chroma feature for the first chord detection algorithms (Fujishima [1999]). The other trial chroma features followed this example and added more steps to the extraction. From the more sophisticated implementations, we reference Mauch and Dixon [2010] as a good recipe for more versatile chroma features. The algorithm starts with transforming the spectrum into a log-frequency spectrum (constant-Q) with three bins per semitone, optional tuning, and spectral whitening steps. The processed log-frequency spectrum is then used as an input for NNLS (Non-negative least squares) approximate transcription, using a dictionary of harmonic notes with geometrically decaying harmonics magnitudes.

For more information on the initial research on chroma features, we refer the reader to Bartsch and Wakefield [2001] or Jiang et al. [2011].

HPCP features

It soon became obvious that some standardization was needed for the emerging variety of chroma features. That along with several improvements, was achieved by the introduction of *Harmony Pitch Class Profiles* (HPCP) by Gómez [2006]. Same as other chroma features, these vector features measure the intensity of each of the 12 semitones. Unlike some other chroma features, the HPCP rely on locating the *spectral peaks* of the DFT spectra as a first step (simply as local maxima; the reader can recall the visible frequency peaks in Figure 1.10). The HPCP features also stand out for the following improvements:

- Higher resolution of HPCP can be used – commonly 36 instead of 12, which measures an intensity of each $\frac{1}{3}$ of a semitone.
- A weighting function $w(l, f_i)$ is added to the feature computation. Its parameters are: l, the HPCP bin index (usually $1 \ldots 36$) and a frequency f_i of the spectral peak with index i. With an oval shape resembling a windowing function, the weighting function prioritizes the magnitude of the frequency f_i to be added to the closest bin, but also to the other neighboring bins with a lesser magnitude. As can be seen in Figure 1.12, for the resolution of 36 bins, the frequency f_i contributes to four bins with the weights according to the distance from the bin. Four weighting functions closest to the frequency f_i are shown, with a cos^2 shape, specified by Gómez [2006]. As such, the function minimizes, for instance, the errors stemming from the tuning differences.
- The presence of harmonics (overtone series, see Attachment A.2) is considered for the tones in the sound. This is achieved by another weighting procedure. If n is the index of the overtone, then $w_{harm}(n) = s^{n-1}$, where $s < 1$ (s is a parameter, a recommended default value of 0.6 can be used). The weighting function decreases with the higher overtones, which simulates the diminishing amplitudes of the overtones. This function is used to determine how much the spectral peak f_i contributes to the bins other than its underlying bin, in a situation when f_i contains an overtone of another tone.

Putting it all together, the HPCP vector using 36 as the number of bins is defined by the formula below:

$$HPCP(l) = \sum_{i=0}^{nPeaks} w(l, f_i) \cdot {a_i}^2 \quad l = 1, \ldots, 36 \tag{1.5}$$

where $nPeaks$ is the number of spectral peaks that we found in the DFT, f_i is the frequency of the spectral peak at index i and a_i is the magnitude at this frequency. We can notice the use of the weighting functions $w(l, f_i)$, making sure that the square amplitude a^2 contributes to four bins. The other weighting procedure using w_{harm} is performed as a separate step, as well as other post-processing, including normalization. For a more in-depth definition and a full procedure for extracting HPCP we refer the reader to

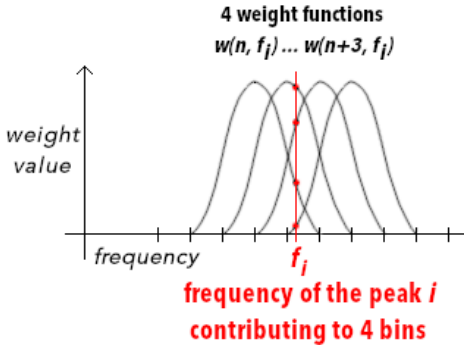


Figure 1.12: 36-bin HPCP weighting functions explained. The contribution of the peak frequency f_i to the four bins of the HPCP feature is depicted.

the original work by Gómez [2006].

We close this section on chroma-like features by pointing out, that using the standard chroma or HPCP features for CSI would fail to match the same songs that are shifted to different keys. This is a common situation in the popular music – a song initially composed for *C major*, can be adapted by another interpret or band and shifted, e.g., to *D major*. If we kept the values at indices ($0 \ldots 12$, or $0 \ldots 36$ for HPCP) intact, the comparison of two songs following the indices would not find the similarity. In multiple works (Ellis [2006], Khadkevich and Omologo [2013]), there is therefore suggested to perform a circular shift of chroma vectors of one piece, producing 12 (or 36) variations, and compare each of this variation to the target song. Another way around is to estimate the shift up front. A good standardization of HPCP features utilizing this idea is proposed by Gómez [2006]. A THPCP (*Transposed Harmonic Pitch Class Profile*) uses another parameter, *shift*, which can be computed for the song as a part of the feature extraction by key detection techniques (*shift* is assigned the index of the estimated key). This „normalizes“ each set of THPCP vectors to the key of the corresponding musical piece and makes the comparison possible. Optionally, if the key detection is not available, *shift* can be assigned for each THPCP vector separately, as the index of the bin which has the maximal value – in effect normalizing towards the tone which is the most dominant. This may lead to inaccuracies, but produces a feature invariant to transpositions. For 36 as the number of bins, we can compute the THPCP as is shown below:

$$THPCP(l) = HPCP(mod(l - shift, 36)) \quad l = 1, \ldots, 36 \qquad (1.6)$$

where *shift* is determined by either of the techniques mentioned above.

MFCC features

We continue with another fundamental definition, one very commonly spoken of in MIR works, although it found its first applications in speech processing: Mel-frequency cepstrum and MFCC features. The MFCCs were first used for MIR in the seminal work of Foote [1997]. We base our definitions on this

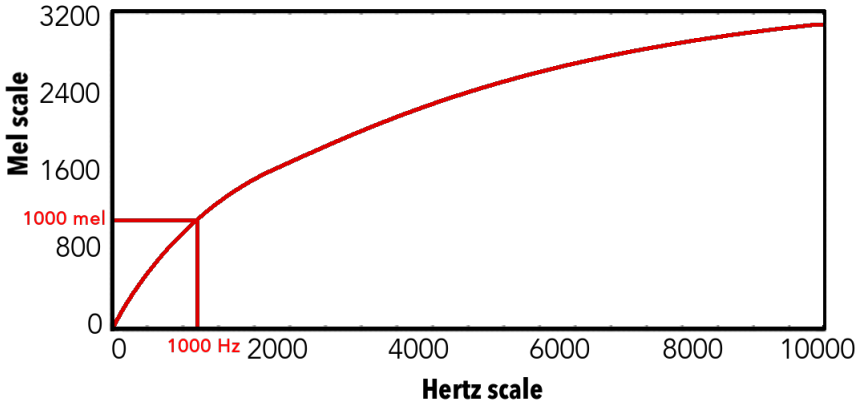


Figure 1.13: Mel scale designed to match listener's perception of pitches

work and the work of Sigurdsson et al. [2006].

For a skilled audio engineer, a common way to describe the Mel-frequency cepstrum (MFC) would be: *A representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency.* However, this is not intuitive for a reader without a signal processing background, so we continue with a step-by-step explanation with more details. The takeaway lesson from the above description is that we can map the spectrum of a standard, Hertz frequency scale, to a *mel scale*. From the listener's point of view, the standard spectrogram using a Hertz scale (shown in Figure 1.11 *C)*) does not use the best scale on the vertical axis. As we know from the previous sections, the musical intervals are mapped exponentially on the frequency scale (*perfect octave* interval represents a doubled frequency). This does not align with the listener's perception of distances, and so, for audio signal processing and speech processing, the mel scale was invented, by the experts in psychoacoustics (Stevens et al. [1937]).

First, a perceptual pitch unit of 1000 mels was assigned to be equal to 1000 Hz. Then a curve was formed by an empirical study (Figure 1.13). Up until about 500 Hz, the function is almost linear, mels rising slightly faster than Hz ($\sim$ 600 mels). Then, increasingly large intervals in Hz were judged by listeners to produce equal pitch increments in mels, resulting in the mel scale rising about two times slower than the Hertz scale above 500 Hz. This curve is commonly described by the equation below (f being the frequency in Hertz, ϕ the pitch in mels).

$$\phi = 2595 \cdot log_{10}(\frac{f}{700} + 1) \tag{1.7}$$

The introduction of the mel scale was motivated by the fact that humans are much better at differentiating small changes in pitch at low frequencies, than they are at high frequencies. The practical implication of the mel scale is thus to make the subsequent audio features match more closely what humans hear.

We now explain in steps how the MFCC features are computed, as they are referenced frequently in the thesis. In case that the reader seeks more information about MFCC and mel scale, we recommend the work of Foote [1997] as the best reference.

Definition. *The* mel-frequency cepstral coefficients *(MFCC) are coefficients that collectively make up mel-frequency cepstrum, derived from a cepstral representation of the audio clip (a nonlinear "spectrum-of-a-spectrum"). In particular, we take the following steps to obtain MFCCs:*

1. *First, the STFT is computed for an audio signal, by dividing it to the short time windows and computing a DFT, as shown in the Equation 1.2.*

2. *The magnitude spectrum from Equation 1.2 is then scaled in both frequency and magnitude.*

   - *The spectrum is scaled logarithmically using* mel filter bank*, denoted $H(k, m)$. This step is also referred to as mapping the powers of the STFT onto the mel scale, using overlapping triangular windows. This visualization, as well as the definition of the mel filter bank, is provided at the end of this definition, to remain focused on the main steps first. For this step, a constant M is chosen, as the number of filter banks (commonly $M = 12$), and a new parameter m is used, that iterates up to M: $m = 1, 2, \ldots, M$. The result of scaling is a modified mel spectrum $X'[m]$:*

   $$X'[m] = \sum_{k=0}^{N-1} |X[k]| \cdot H(k, m) \quad for\ m = 1, 2, \ldots, M \tag{1.8}$$

   - *Then, the magnitude is further scaled, by taking a logarithm, resulting in mel log spectrum $X''[m]$:*

   $$X''[m] = ln(X'[m]) \quad for\ m = 1, 2, \ldots, M \tag{1.9}$$

3. *The MFCCs are then obtained by taking the discrete cosine transform (DCT) of the mel log spectrum, as if it were a signal. We refer to the M values of the resulting spectrum as* MFCCs*:*

   $$mfcc[l] = \sum_{m=1}^{M} X''[m] cos(l \frac{\pi}{M}(m - \frac{1}{2})) \quad for\ l = 1, 2, \ldots, M \tag{1.10}$$

   The value $mfcc[l]$ is referred to as the lth MFCC.

Note the use of DCT, although by „cepstrum“ *we usually understand the application of inverse DFT (iDFT) on the logarithm of DFT* spectrum*. The use of DCT is purposeful to obtain a better result in terms of energy.*

We now go back to define the mel filter bank $H(k, m)$ used in step 2 to scale the frequency. Mel filter bank is a collection of triangular filters defined by the

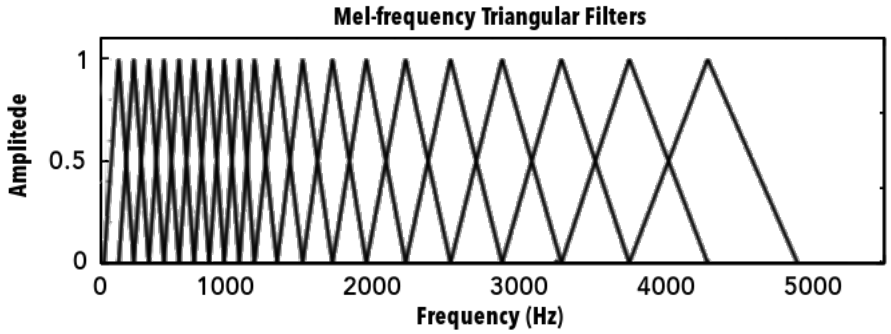


Figure 1.14: Triangular overlapping windows forming the mel filter bank, as proposed by Davis and Mermelstein [1980]. 20 windows are depicted, with $M = 20$ and the parameter $m = 1, 2, \ldots, M$ used as an index in the definition of MFCC.

center frequencies $f_c(m)$, as is shown in Equation 1.11 below and visualized in Figure 1.14. From multiple possible triangular filters functions, we show the function from the original MFCC definition by Davis and Mermelstein [1980]. The center frequencies $f_c(m)$ from the below definition specify the locations of the top vertices of each triangle.

$$H(k,m) = \begin{cases} 0 & for \quad f(k) < f_c(m-1) \\ \frac{f(k)-f_c(m-1)}{f_c(m)-f_c(m-1)} & for \quad f_c(m-1) \leq f(k) < f_c(m) \\ \frac{f(k)-f_c(m+1)}{f_c(m)-f_c(m+1)} & for \quad f_c(m) \leq f(k) < f_c(m+1) \\ 0 & for \quad f(k) \geq f_c(m+1) \end{cases} m = 1, \ldots, M \tag{1.11}$$

The center frequencies can be found on the mel scale introduced in Equation 1.7. First, we assign the desired lowest pitch of the filter banks ϕ_{min} and the highest pitch ϕ_{max} (e.g. 0 mels as ϕ_{min}, and 3200 mels corresponding to $\sim 11,000$ Hz as ϕ_{max}). According to the number of filters M we can then derive $\Delta\phi$ as:

$$\Delta\phi = \frac{(\phi_{max} - \phi_{min})}{M+1} \tag{1.12}$$

$\Delta\phi$ represents the horizontal distance between the top vertices of the neighboring triangles on the mel filter bank. The center frequencies on the mel scale are found by:

$$\phi_c(m) = m \cdot \Delta\phi \quad for\ m = 1, 2, \ldots, M \tag{1.13}$$

Finally, to get the center frequencies to the Hertz scale, we can employ an inverse equation to Equation 1.7.

$$f_c(m) = 700 \cdot (10^{\phi_c(m)/2595} - 1) \quad for\ m = 1, 2, \ldots, M \tag{1.14}$$

Features based on chords and keys

Alongside the chroma features and the MFCCs, it is common to use music theory structures, such as chords or keys, directly for MIR experiments. As the reader is already familiar with the musicology definitions of chords and keys from the earlier sections, we close this chapter with a couple of informal definitions that the MIR researchers tend to use when working with chords and keys: *chord progression*, *key progression*, and *chord distance*.

A *chord progression* (a sequence of chord labels) is a familiar concept for musicians, who often use it to play together in an unrehearsed situation. The idea of using chord progression itself as a fingerprint for large-scale music retrieval was proposed by Khadkevich and Omologo [2013], improving the state-of-the-art cover song identification results at the time. The progression can be represented as a sequence of strings (*C, F6, Gmaj7, ...*), or boolean vectors similar to chroma vectors. The same string representation can be used to show the *key progression* of the piece, capturing the key modulations occurring (e.g., *C major, A minor, C major*, simply written as *C, Ami, C*).

A *chord distance* is a concept based on the acknowledged music cognition findings: the listeners perceive the differences in chords in a way that can be predicted by a formal model. Such a model can evaluate the perceived difference, and return an integer number for a pair of chords, as their chord distance. Fred Lerdahl's *Tonal Pitch Space* (TPS) model (Lerdahl [2001]), and the *TPS Distance* for chords was proposed and backed by the empirical studies. This concept was further studied by a few MIR authors (De Haas et al. [2008], Rocher et al. [2010a], Maršík et al. [2017]), combining the cognitive and computational chord distances.

We devote a considerable part of the following chapters using these definitions and building up new ideas around them. We are interested, in particular, whether manufactured features such as chroma features or MFCCs perform better than the extracted chords or keys being used as a feature. We are also about to investigate the possible uses of chord distances. However, we first take the course to observe the related works in Chapter 2 to show the reader what was already proposed to be able to distinguish our contributions.

2. Related works

In this chapter, we provide a summary of the work most related to ours. Similar as in the previous chapter, we divide the content into the related works in musicology, and related works in music information retrieval.

2.1 Related works in musicology

The works on musicology are related to our effort to define a music harmony model to be able to compare songs from the perspective of harmony. An obstacle we need to tackle is that there are multiple models of harmony already published, but to apply them directly to a song comparison is not straightforward. These formalisms are very rarely implemented in the state-of-the-art music services, probably due to the complex and high-level approach they imply. Even a profound harmony model may not be suitable for the CSI task if there is no easy way to use it to extract features from the songs. With this in mind, we offer a review of the known models and their highlights.

2.1.1 Tonal pitch space

One way to extract features from the song is to evaluate transitions between successive chords. *Chord distances* discussed in Section 1.2.4 can be straightforwardly utilized for this task, and multiple models of harmony have their ways of defining chord distances. The first model from this category that we present is the **tonal pitch space** (TPS) model.

In his inspiring work, Lerdahl [2001] introduces a model describing distances between pitches, chords, and keys. Fred Lerdahl's findings go beyond the notion of distances in the pitch space. In his writings, a more complex *tonal tension* model is specified, using TPS as only one of the components, along with prolongational event structure, surface dissonances and a model of voice-leading attraction (Lerdahl and Krumhansl [2007]). However, for our song comparison by harmony, a sole TPS model suffices. It has been shown in earlier studies, that the TPS model can be used to compute meaningful descriptors for MIR studies (De Haas et al. [2008]).

Fred Lerdahl's model of space starts with the layer of semitones, upon which the other four layers are built. In the order of increasing stability, the five levels are chromatic level, diatonic level, triadic level, fifths level, and octave level. An arbitrary chord can be represented in this space in the context of a key (region), forming a *basic space of the chord*. To get the basic space, we first plot all chromatic pitches at the bottom (chromatic level), which would be $0 - 11$ in a numeric representation of pitches (0 represents the pitch c, see Figure 2.1). We continue by plotting the pitches belonging to the key to the diatonic level, forming the basis for the region. The chord tones are plotted in a similar fashion to the triadic level. Lerdahl assumes

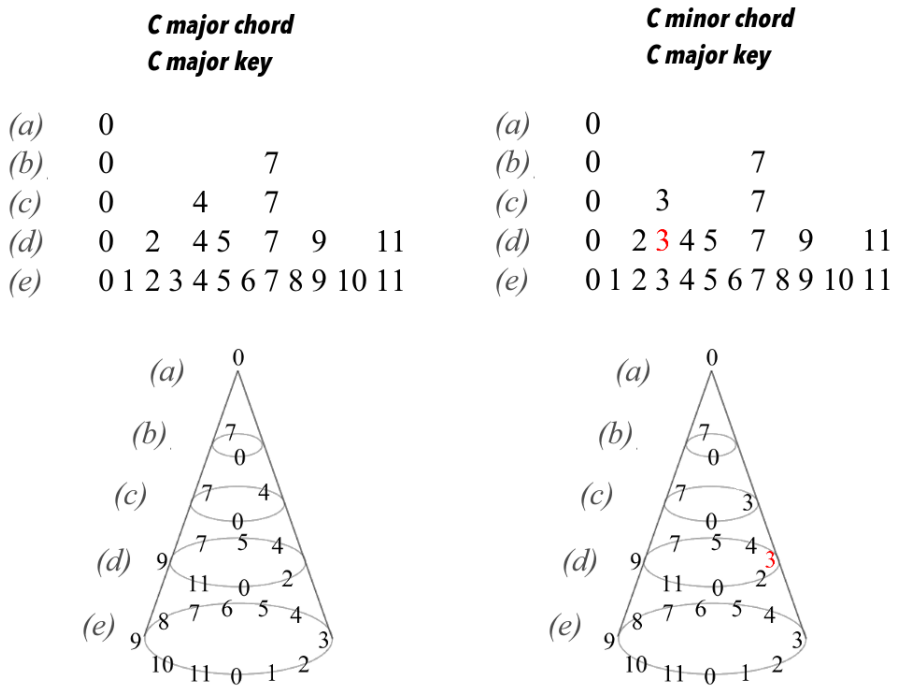


Figure 2.1: A basic space of a chord proposed by Lerdahl [2001]. The *C major* chord on the left and the *C minor* chord on the right are visualized on five levels, from (a) to (e): octave, fifths, triadic, diatonic, and chromatic level. Both are in the same context of a *C major* key, in a standard visualization (at the top) and a conic visualization (at the bottom).

the fifth as the next stable pitch, and finally, the root pitch as the most stable in accordance with the theoretical and empirical studies.

As an example, Figure 2.1 shows the basic spaces of the chords *C major* (on the left) and *C minor* (on the right) in the same context of the *C major* key. The chords share the octave and the fifths level (tones *c* and *g*). The triadic level shows a different tone for the *C minor* chord (*e♭*) which is also copied to the diatonic level to highlight the dissonance with the key. At the bottom, the basic space of a chord is presented as a pitch class cone, to highlight the periodicity of pitch classes.

Note that if a particular pitch is not present on the level below, we copy the pitch to the level below to achieve a hierarchy and highlight the dissonance (the pitch 3 representing *e♭* in *C minor* chord is present at the triadic level *(c)*, but is not a part of the *C major* key on level *(d)* and therefore is copied below).

The number of transformations of the basic space measures the distance between the chords. Starting with the basic space of the first chord, we employ the following transformations to match the second chord:

1. The circle-of-fifths regional rule: shifting the level *(d)* seven chromatic steps to the left or right to match the level *(d)* of the second chord.

2. The circle-of-fifths chordal rule: shifting the levels *(a)–(c)* four diatonic steps to the left or right to match the levels *(a)–(c)* of the second chord.

The transformations are shown in Figure 2.2. It can be observed that the two rules represent the distance between the keys and chords on the circle of fifths discussed in Attachment A.3. After the transformations, the basic spaces of the two chords may still not align perfectly. Note that in the example in Figure 2.2 (*C major* and *G7* chords), the chordal rule shift of 4 diatonic tones to the right does not align perfectly with the *G7* chord because of the added seventh note. In such cases, we determine to stop shifting when there are as few as possible distinctive pitch classes. Mainly for these cases, the third comparison is added: *the count of distinctive pitch classes*. Looking at the basic space of the second chord, we count the number of all pitch classes not present in the original chord's basic space. In Figure 2.2, we can see five non-common pitch classes, which means a longer distance between the chords. The further details of the rules are explained in the *Tonal Pitch Space* book by Lerdahl [2001].

Lerdahl then combines all these rules into the definition of a chord distance, which we reference as the *tonal pitch space distance* in this thesis. We employ the following definition, similar to the definition by Lerdahl, with the exception of using „the distance on the circle of fifths“ instead of explicitly stating „the number of applications of the circle-of-fifths rule“, which yields the same results given that we work with chords that can be found on the circle of fifths:

Definition. Tonal pitch space distance (TPS distance) *of two chords* C_1, C_2

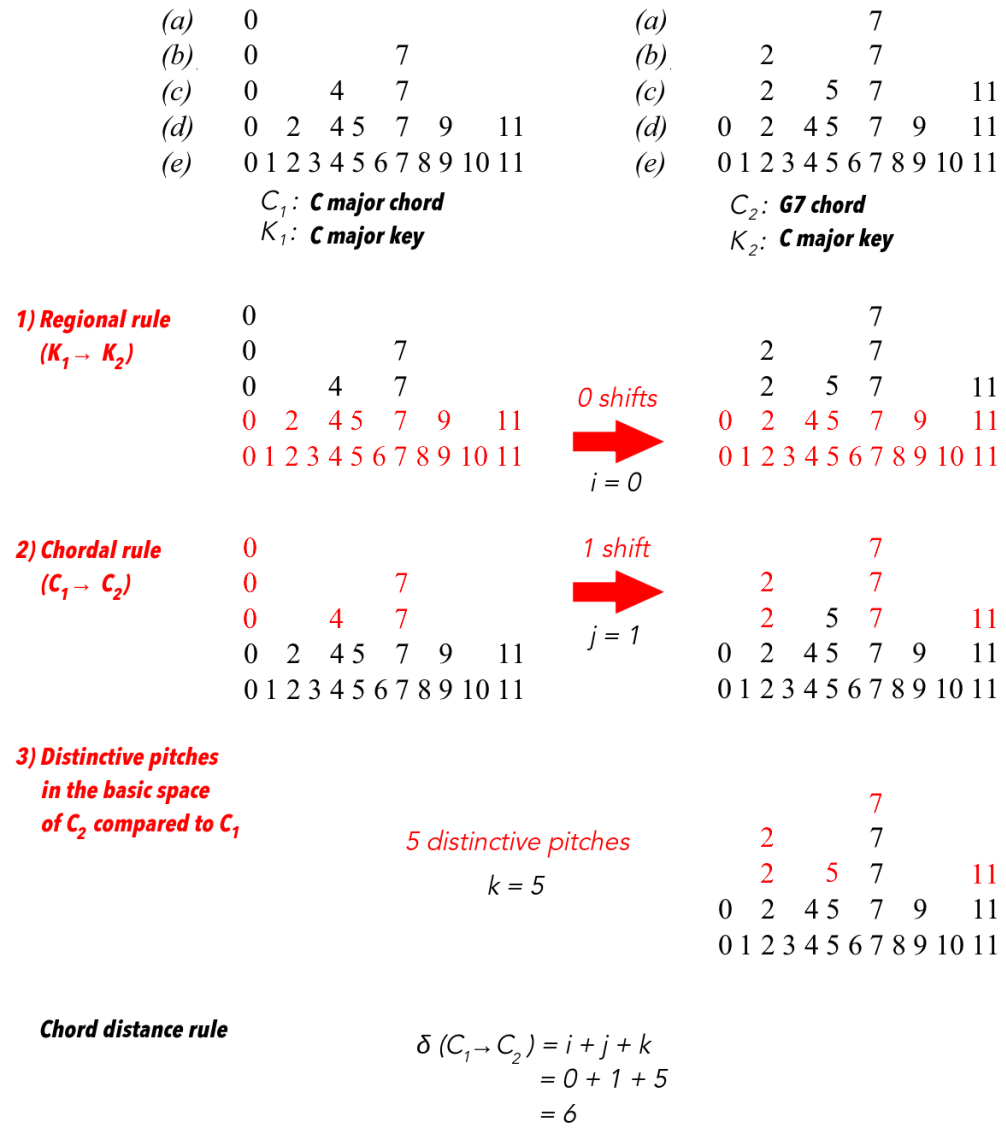


Figure 2.2: Explanation of the TPS distance as specified by Lerdahl. For the two chords C_1, C_2 in the context of the respective keys K_1, K_2:
1) First, the number of regional circle-of-fifths rule needed to shift K_1 to K_2 is assigned to the variable i.
2) Second, the number of applications of chordal circle-of-fifths rule is assigned to the variable j. Highlighted in red are the levels of the space that shift, in this case, four diatonic steps to the right, which resolves to seven chromatic steps (modulo 12).
3) Then, the number of distinctive pitches in the basic space of $C_2 + K_2$ compared to $C_1 + K_1$ is assigned to the variable k.
Finally, the chord distance is computed as the sum of $i + j + k$.

from the respective keys K_1, K_2 is calculated as:

$$\delta(C_1, C_2) = i + j + k$$

where i is the distance between the keys K_1, K_2 on the circle of fifths and j is the distance between the chords C_1 and C_2 on the circle of fifths. k is the number of non-common pitch classes in the space of the chord C_2 compared to the space of the chord C_1.

The reader should note that this distance function, as defined by Lerdahl, is non-symmetrical, because of the formulation: *„... the number of non-common pitch classes in the space of the chord C_2 compared to the space of the chord C_1“*. This was previously observed by De Haas et al. [2008], with a proposal to preserve the symmetry by calculating *„the number of non-common pitch classes within the basic spaces of C_1, C_2, together divided by two“*, instead of the original formulation. The resulting measure was called *Tonal Pitch Step Distance* by the authors. Moving forward, we clearly mark, whether we use the original definition (TPS distance), or the symmetrical definition by De Haas et al. (Tonal Pitch Step Distance).

The TPS distance (original version) was successfully used to improve chord estimations by Rocher et al. [2010b] (best chord distance out of eight candidates), and the aforementioned Tonal Pitch Step Distance was used for a music similarity task in the publication by De Haas et al. [2008]. The full potential of TPS is still not exploited for MIR tasks, which motivated us to try the model for cover version identification in Chapter 5.

2.1.2 The Harmonic Network, Tonnetz and transformations

Historically, the first important music harmony model is that of a geometric harmonic grid called Tonnetz (meaning *tone network* in German). The idea was first proposed in the 18$^{\text{th}}$ century by Euler [1739] (back then under the name *Speculum Musicum*) and it represents the tonal space as a two-dimensional pitch space grid (see Figure 2.3 *b)*). Throughout the history, the musicologists and composers have been returning to this concept (including, e.g., Arnold Schönberg), but the presentation as a triangular grid under the name *Tonnetz* is mostly attributed to Riemann [1896b]. However, it was Longuet-Higgins [1962] who presented a first completely formal model, under the name *The Harmonic Grid*. Longuet-Higgins showed that the frequency ratios of musical intervals take the form of a product of powers of the prime factors: $2^x \cdot 3^y \cdot 5^z$, where $x, y, z \in \mathbb{Z}$. This creates a three-dimensional grid of pitches around x, y, z axes, where the intervals can be presented as vectors. But, since the ratio of 2 corresponds to a *perfect octave* interval and can be considered as the same pitch class, we can simply omit that dimension and project the space to a y, z plane. That is how Longuet-Higgins specified The Harmonic Grid, shown in Figure 2.3 *a)*.

The model was further extended in the *Neo-Riemannian theory*. The Neo-Riemannian theory is a loose collection of ideas present in the writings of multiple theorists, with a central idea to relate harmonies directly to

a)

E	B	F#	C#	G#	D#	A#	E#	B#
C	G	D	A	E	B	F#	C#	G#
Ab	Eb	Bb	F	C	G	D	A	E
Fb	Cb	Gb	Db	Ab	Eb	Bb	F	C
Dbb	Abb	Ebb	Bbb	Fb	Cb	Gb	Db	Ab

b)

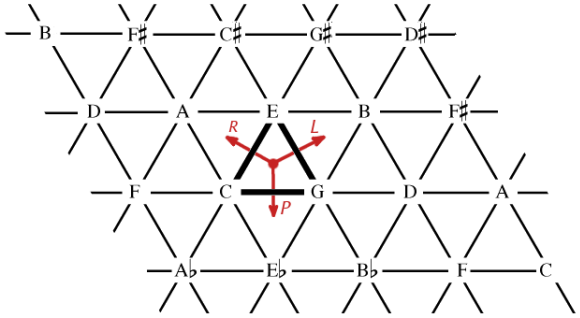


Figure 2.3: *a)* The Harmonic Network adapted from Longuet-Higgins [1962] *b)* The Tonnetz grid with basic Neo-Riemannian transformations as proposed by Lewin [1982].

each other rather than to a tonic. In the most referenced visual representation, chords are represented as triangles, and three basic transformations (labeled R, L, P) are represented as translations to one of the adjacent triangles. These three basic *Neo-Riemannian transformations* were first proposed by Lewin [1982] in the work *Generalized Musical Intervals and Transformations*, and each holds a particular meaning for music. The R transformation exchanges a triad for its *relative*, e.g. *C major* to *a minor*, the L transformation exchanges a triad for its Leading-tone exchange, e.g. *C major* to *e minor*, and the P transformation exchanges a triad for its *parallel*, e.g. *C major* to *c minor*. Note the ambiguity in the *parallel* term – here, the parallel comes from the notation commonly used in USA, and means modifying *C major* to *c minor*, whereas the parallels how we defined them in Chapter 1.1.3 and Attachment A.5, based on original Riemann's German notation, yields modifying *C major* to *a minor*, which would be called *relative* in USA (the same ambiguity is in describing the keys).

In Figure 2.3 *b)*, a transformation is shown as a translation of the triangle around one of its edges. We can thus, informally (with respect to a more thorough study behind the matter by Lewin [1987]), define a simple Neo-Riemannian chord distance using the Tonnetz grid. Note that this definition is independent of the musical key, and also that the two chords need to be selected from a simple chord dictionary (they need to be located on the Tonnetz grid).

Definition. *The* Neo-Riemannian distance *of the two chords* C_1, C_2 *that*

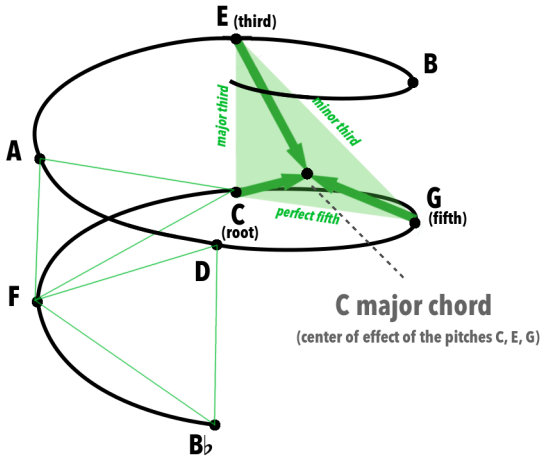


Figure 2.4: The Spiral Array (the figure adapted from Chew [2014]): The pitches are arranged in a helix structure. Highlighted is the *C major* chord represented by a triangle, but also having a point representation („center of effect“) described as a convex combination of the points representing its pitches.

can be found on the Tonnetz grid is equal to the number of Neo-Riemannian transformations on the Tonnetz grid needed to translate from C_1 *to* C_2*.*

More research was done on the subject of the Tonnetz grid, which did not fit our abbreviated survey. One example for all would be visualizing the complex chords and chord progressions in time using Tonnetz, using the so-called *isochords* visualization by Bergstrom et al. [2007].

2.1.3 Mathematical and Computational Modeling of Tonality

A very unique and versatile spiral model was introduced by Chew [2000, 2014]. With the tradition of spiral or helix models dating as far back as 1846 (see Shepard [1982], for a comprehensive study), Chew proposes a spiral model that also satisfies the arrangement of the Tonnetz grid. If the planar grid from the Figure 2.3 *a)* or *b)* is rolled up so that the same pitches line up with themselves (A coincides with the A four columns away, F coincides with F four columns away, etc.), the resulting array has a form of a helix and is called *The Spiral Array* by Chew, see Figure 2.4. In the book *Mathematical and Computational Modeling of Tonality* (Chew [2014]), it is explained how not only the pitches are arranged meaningfully, but also the chords and keys can be formed as a convex combination of points. Notice the perfect fifth intervals for the pitches adjacent horizontally along the spiral and the major third intervals for the pitches adjacent vertically in Figure 2.4. This in effect presents the simple chords such as major and minor triads as triangles on the helix.

A method for deriving a chord distance can be extracted too, as Chew

proposes *„. . . that each chord is represented spatially as a point in the interior of the triangle“*. Also, in general, *„. . . any collection of pitch representations can be appropriately weighted to generate a barycenter in the interior of the helix that represents the composite effect of the pitches called the center of effect“*. So if we weight the three pitches forming the chord equally, we can represent the chord by the center of the triangle formed by its pitches. The spiral distance of the two chords is then the geometric distance between the two 3-dimensional points, centers of effect of the chords (we take the definition in accordance with Rocher et al. [2010a]):

Definition. *The* spiral distance *of the two chords C_1, C_2 is equal to the geometric distance of their centers of effect $C_1(k_1)$, $C_2(k_2)$. $k_1, k_2 \in \mathbb{N}_0$ represent the indices of the roots of the chords C_1, C_2 on the helix (e.g. if $k_1 = 0$, the very first pitch at the bottom of the helix is taken). Let the function $P(k)$ return the coordinates of the pitch at the index k on the spiral:*

$$P(k) = \begin{bmatrix} x_k \\ y_k \\ z_k \end{bmatrix} = \begin{bmatrix} rsin(\frac{k\pi}{2}) \\ rcos(\frac{k\pi}{2}) \\ kh \end{bmatrix} \tag{2.1}$$

Then the function $C_M(k)$ returns the center of effect of pitches as a convex combination of the points that make up the chord M. For a major triad, the definition is below (similar definitions can be extracted for minor triads or more complex chords):

$$C_{majortriad}(k) = w_1 \cdot P(k) + w_2 \cdot P(k+1) + w_3 \cdot P(k+4) \tag{2.2}$$

w_1, w_2, w_3 are the weights of the pitches in the chord, $w_1 \geq w_2 \geq w_3 > 0$ and $\sum_{i=1}^{3} w_i = 1$. Considering the same proportion of the pitches in the chord, we can assign the same weight of $w_1 = w_2 = w_3 = \frac{1}{3}$ for the triad.

The final calculation of the spiral distance is achieved by calculating the geometric distance of the two points $A = C_1(k_1)$ and $B = C_2(k_2)$ as follows:

$$\sqrt{(A_x - B_x)^2 + (A_y - B_y)^2 + (A_z - B_z)^2} \tag{2.3}$$

This chord distance was experimentally studied by Rocher et al. [2010a] and yields one of the best results along with Lerdahl's distance. Chew's monograph also adds multiple experiments for key finding, music segmentation, and pitch spelling, by using the center of effects idea (Chew [2014]).

2.1.4 Other notable chord distance measures

Thanks to Rocher et al. [2010a], we have a thorough review of the aforementioned and other available chord distances, including musicology, geometric, or computational approaches. Seven chord distances are compared and tested on the chord estimation task. In this context, chord distances are used to select the correct chord candidates. In order to label the successive chords, a dynamic programming algorithm chooses the chord candidates with the smallest chord distance.

In the review of Rocher et al., next to the TPS and spiral distance, the best performing distances were the following two:

- Paiement distance (Paiement et al. [2005]) based on the strength of the harmonics compared to every note on the chromatic scale.
- Estrada distance (Mathieu [2002]) based on counting the tones of the chords, and subtracting the number of common intervals.

We conclude our review of known chord distances by these two definitions. They are unique by taking a probabilistic approach (Paiement distance) or a computational approach (Estrada distance) in evaluating the final distance.

Definition. Estrada distance *between two chords C_1, C_2 is computed as:*

$$\delta(C_1, C_2) = max(\#(C_1), \#(C_2)) - |VC_1 \cap VC_2| - 1$$

where $\#(C_1)$ and $\#(C_2)$ are the number of tones in each chord respectively, and VC_1 and VC_2 are the interval vectors of chords C_1, C_2 respectively. Negative results are set to 0. The distance values come from 0 to 11.

Definition. *For each chord C_j, let the distributed representation:*

$$l_j = j_j(n_1), \ldots, l_j(n_d) \tag{2.4}$$

correspond to the perceived strength of the harmonics related to every note n_k of the well-tempered scale, where the first d notes of this scale are relevant. A 12-dimensional vector v_j then gives a measure of the relative strength of each pitch class in a given chord. The value $v_j(0)$ is associated with the pitch class C, the value $v_j(1)$ with the pitch class $C\#$ and so on up to 11:

$$v_j(i) = \sum_{n_k : 1 \leq k \leq d, (n_k mod 12) = i} l(n_k) \tag{2.5}$$

Paiement distance *between two chords C_1, C_2 is then computed as an Euclidean distance induced by this representation.*

Chord distances and their application in a series were also studied in the work of De Haas et al. [2008, 2013b]. Contrary to Rocher et al., the authors have chosen not to compare subsequent chords, but to compare the whole chord series to a common chord profile representing the key of the piece. In this work, it is also stressed that there is surprisingly little research conducting a chord series comparison, given the popularity of chord annotations.

2.2 Related works in MIR

MIR is nowadays a field of science that has a very wide range of applications and tasks including audio classification, audio fingerprinting, audio key detection, cover song identification, and others. Our thesis focuses on the cover song identification, but also provides new research on mathematical modeling of music, harmony analysis, and music complexity, while we also take a short course on chord estimation or genre detection along the way, as our model from Chapter 3 can be used in multiple ways. That is why the range of MIR works related to ours is quite broad.

We already looked at the works on mathematical modeling of music in the previous Section 2.1, which also partly fall within MIR. We now take a look at the related works on music complexity and harmony analysis, which also share many concepts from the previous section, but this time, we observe them from the computational perspective. The remainder of the section is devoted to the related works on the cover song identification task.

2.2.1 MIR works on music complexity

The works on music complexity are actually the smallest group in our survey, mostly because the term *music complexity* is not defined per se. Intuitively, we distinguish simple music and more complex music with regards to harmony, melody, rhythm, etc. (harmonically complex music can be very straightforward in rhythm, and vice versa). Based on this intuition and a theorem that our personal preference of music should correlate with our preferred complexity of music, an initial work *towards a harmonic complexity of musical pieces* was established (Maršík [2013], Maršík et al. [2014a]). In these studies, we outlined a first system for a harmonic complexity similar to the computational complexity, and we continue in the same effort in this work.

But, were there any previous leads towards the complexity of music and its usefulness for research? We highlight the work of Zanette [2008a,b], who also tries to search for the answer, and cites Schönberg: *„Two impulses struggle with each other within man: the demand for repetition of pleasant stimuli, and the opposing desire for variety, for change, for a new stimulus.“* Zanette also highlights that a random, disordered changing should hardly qualify as complex. The variety, the unexpected changes, and the occurrence of the new stimulus in music could, therefore, be related to music complexity, with the caveat of not having a random nature.

There is another article defining music *space complexity* the same way as the computational complexity theory: *The Complexity of Songs* by Donald E. Knuth [1977]. Knuth describes the space complexity of most of the songs as linear and finds interesting results for *Old McDonald had a farm* song. A logarithmic or even constant complexity is assigned to some popular songs. Although published as an inside joke, interestingly pointing out the analogy with computational complexity, we can definitely use it as an inspiration to use computational complexity as a measure for harmonic complexity. In particular, we can quote Knuth with regards to the repetitions in the verses and choruses – leading to smaller space complexity. This should be taken into account when defining the complexity of the music.

With only a few proposals for harmonic complexity – let alone music complexity – to the best of our knowledge, there is still room for new definitions. Nonetheless, in the next section, we show that there are also other useful descriptors about music harmony, not necessarily named or associated with complexity.

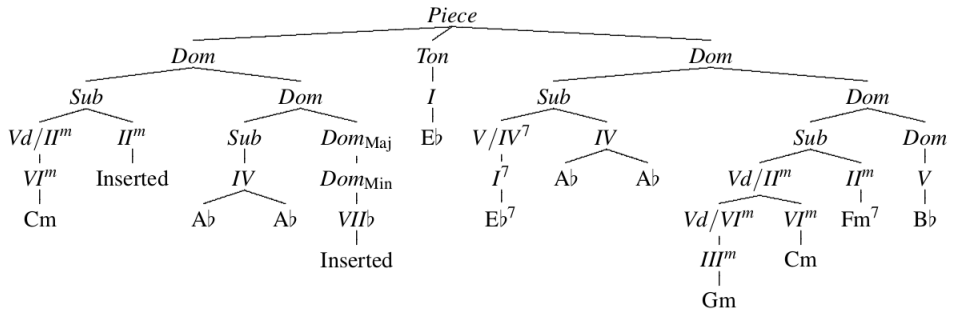


Figure 2.5: An excerpt of the HarmTrace system analysis (the figure from De Haas et al. [2012]): The song *The long and winding road* by The Beatles is analyzed.

2.2.2 MIR works on harmony analysis

Multiple MIR studies have chosen harmony analysis to be the main topic. The outcome of their work varies – some analyze harmony in order to find the best candidate for the chord or tone transcription, others try to help the musicians, e.g., by visualizing the musical piece or aid the performance.

De Haas [2012] devoted his research to develop a system for automatic functional harmonic analysis. The presented Haskell-based system HarmTrace[1] is capable of deriving a tree structure explaining the tonal function of the chords in the piece (see Figure 2.5). The first layer parses the song as the sequence of tonic (T) and dominant (D) chords. The next layer considers that a dominant may be preceded by subdominant (S). The following layers consider that a function parallel may be used. If the song chord progression does not satisfy the harmonic tree specification, a chord label can be inserted or deleted (see *Inserted* nodes). A Haskell-based program derived the harmonic tree for the input ground-truth chord annotations.

One of the possible applications is to improve chord transcription algorithms (detecting the correct chord progression for a musical piece), as is specified in De Haas et al. [2012]. The authors had found statistically significant improvement when the tonal harmony analysis was used. The number of errors in creating such a tree helps to select the chords – such chords are selected, that trigger the smallest amount of errors. Bas De Haas followed up on the initial results and also employed the system for structural segmentation of music based on harmonies (De Haas et al. [2013a]). Later on, his systems have laid a foundation for the automatic chord transcription services available to the public (Chordify[2]). Additionally, we consider the method of De Haas et al. to be a good recipe to compare musical pieces and their harmonic complexity, as was discussed in Section 2.2.1 previously.

If we switch the focus from chords to tonality and keys, it is mostly the *key finding* algorithm that draws the attention of many MIR studies (Sapp [2005], Chew [2014], Gómez [2006] to name a few). The music cognition

[1] `http://hackage.haskell.org/package/HarmTrace-2.0`
[2] `https://chordify.net`

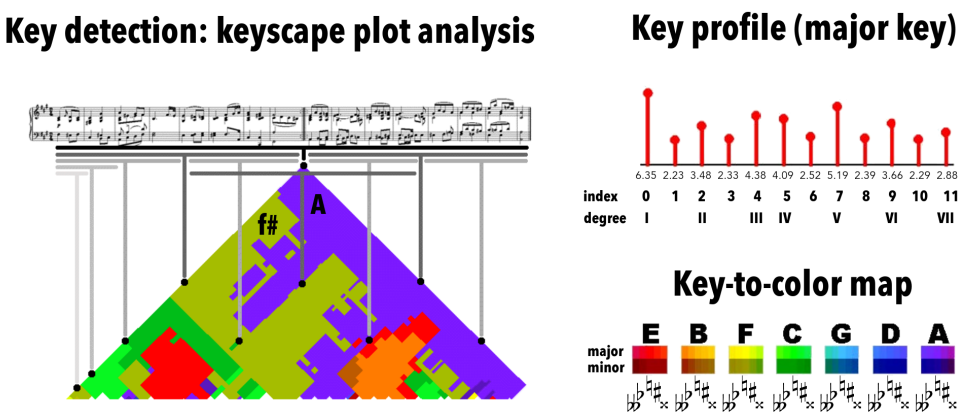


Figure 2.6: Key detection analysis example, the figure adapted from Sapp [2005]. On the top-right, the Krumhansl-Kessler probe-tone profile of a major key is shown Krumhansl and Kessler [1982]. The weights are assigned to each tone of the piano scale, given by the empirical data of listeners perception. By comparison of the tone material to the key profiles, the keyscape plot can be created.

work of Carol Krumhansl presents the basis of most of the state-of-the-art algorithms. In particular, it was the creation of key profiles based on experiments with listeners by Krumhansl and Kessler [1982]. Participants of the Krumhansl-Kessler experiment were presented a key-establishing context, followed by a probe tone. Each participant had to rate, how well the probe tone fitted into a previously presented musical context. The results were averaged, and key profiles (vectors of 12 real values) for major and minor keys were created. By comparison of the tone material of a song to such key profiles, one can estimate the key that the piece is played in (Krumhansl-Schmuckler key-finding algorithm, Krumhansl [1990]). A sample analysis from the work of Sapp [2005], also using a nice color visualization can be seen in Figure 2.6. This type of plot was nicknamed *keyscape* since it is analogous to a landscape painting. The key-finding algorithm is used at various points of time, and a plot of the found keys is visualized using the color map.

This and other studies have been looking for new results based on the Krumhansl-Schmuckler algorithm for several decades. Notable is the use of modified key profiles by Temperley [2001] based on further experimentation with a musical corpus of classical music. The results of the most recent studies are not only the detected key with frame precision (detected for every moment of the musical piece, e.g., Noland and Sandler [2007]) but also an interesting feature called *key clarity* or *key strength*, evaluating how stable is the best-matching key, or in other words how much did it outweigh the other key candidates (Bogdanov et al. [2011]).

Although there are many other MIR works on the harmony analysis, we would like to highlight those projects that go beyond „only“ transcribing the chords, or „only“ detecting the key. These tasks are, admittedly, challenging enough, but it is intriguing to have a MIR system equipped with a universal

module, so that the outcome showcases all harmony knowledge combined. An additional „magic“ happens when a single system not only analyzes a piece but also calculates a similarity between pieces in a non-trivial way. Such criteria are met in the broad scope of projects of De Haas [2012] and Chew [2014]. Gómez [2006] showed how HPCP is a universal feature for both chord and key estimation and left cues for organizing a music collection. On the other hand, Rocher et al. [2010b] showed how chords and keys, if they are evaluated concurrently by the same algorithm, yield better estimation accuracy. Within the project, the team from LaBRI also showed how the harmony of the musical piece could be represented in a tree with five levels and compared to other harmony trees (Robine et al. [2009], Rocher [2011]). These works were among the most influential for the work on this thesis.

2.2.3 MIR works on cover song identification

The first global academic attention to the CSI emerged in 2006 when this task was added to the benchmark of the Music Information Retrieval Evaluation eXchange (MIREX, Downie et al. [2005])[3]. It was the same year that Gómez has published her Ph.D. thesis *Tonal Description of Music Audio Signals* (Gómez [2006]) with a chapter devoted to the CSI using solely tonal descriptors. This certainly shared the goals with our thesis and laid a foundation for our research. We commence our review with the highlights from the MIREX benchmark results followed by the summary of the most relevant works.

CSI on the MIREX benchmarking

Before the annual MIREX challenge included *Audio Cover Song Identification* as one of its tasks, probably the closest topic to the problem of version identification was *music similarity* (Yang [2001a], Volk et al. [2016]). A few early papers drew attention to the comparison of two versions or recordings of the same piece (Purwins et al. [2000] and Yang [2001b]). They both noted that we need to consider that the pieces can differ in instrumentation, interpretation, tempo, or finally, that the whole music score can be different – modulated or otherwise adjusted. If the changes are not in the music score, spectral analysis (similar to music fingerprinting) or constant Q transform (a technique related to chroma-based features) can be applied directly. The results with spectral similarity were promising already in 2001 for Yang (up to 90% accuracy on a smaller dataset of 120 songs).

It is the more difficult cover versions where the music score changes, which add the „twist“ to the CSI task. The MIREX challenge provides the researchers each year with a set of non-trivial cover songs (Mixed Collection of 330 audio files, 30 songs each in 11 versions, plus a separate test on Mazurka collection: a search within 539 mazurkas, based on 49 music scores, each in 11 performances)[4]. Each of these tracks is used as a query, and the goal

[3] https://www.music-ir.org/mirex/wiki/MIREX_HOME

[4] https://www.music-ir.org/mirex/wiki/2019:Audio_Cover_Song_Identification

is to retrieve the other cover tracks from the same version group. Among the evaluation metrics, there is a total number of covers identified, a mean number of covers identified, a mean rank of first correctly identified cover, and mean arithmetic of average precisions (MAP, which we describe and use in Chapter 5). There are 2 – 8 algorithms submitted every year to MIREX CSI, with only 2012 being the year without competitors. As Downie et al. [2008] specify, CSI was an important addition to MIREX, as it can address the distinction between timbral/acoustic similarity and musical similarity, as the versions span multiple genres, and instrumentations, yet are considered undeniably similar. They remark that the task can motivate MIR researchers to expand their notions of similarity *„. . . to include the important idea that musical works retain their identity notwithstanding variations in style, genre, orchestration, rhythm or melodic ornamentation“*.

MIREX benchmarking results

The LabROSA system achieved the best results in the first year of the CSI benchmarking, with beat-synchronous chroma features (Ellis and Poliner [2007]). Chroma vectors were calculated for beat-length segments of the songs, where beats were extracted by preprocessing. Besides the 761 out of 3,300 total number of covers identified, Ellis also showcased the result of 34 correctly identified covers from his *covers80*[5] dataset, which was a promising start, proving that harmony features such as chroma vectors are meaningful for the CSI task.

The winning submission in the 2007 benchmarking was a system developed by Serrà et al. [2008]. Based on an extensive series of experiments, the authors have proposed improvements compared to LabROSA system, notably: the use of improved chroma features (HPCP by Gómez [2006], with the resolution of 36 bins), tackling the key transposition problem using a shift based on the global song profile, binary similarity matrix, dynamic programming and the local alignment (Smith-Waterman algorithm). A *dynamic time warping* (DTW) method with various constraints was used as a similarity measure. DTW (Müller [2007]) is a method used for determining an optimal alignment between two time series, which became an industry standard for evaluating similarities in music. Serrà and other authors were experimenting with algorithms based on DTW to discover similarities in various moments in the songs.

The best CSI results in MIREX at the time of writing this paper were achieved in 2009 – unbeaten for almost 10 years so far for the Mixed Collection dataset – by an improved version of the algorithm by Serrà et al. [2009]. The system correctly identified 2,426 out of 3,300 cover songs (73.5%). The most significant improvements were: using a cross-recurrence plot instead of the binary similarity matrix and reasoning that a maximal length of diagonal lines (L_{max}) experimentally proved to have the highest discriminative power. The final measure (Q_{max}) was a slight modification of L_{max}, accounting for the tempo changes (the curvature of the line) and gaps.

An overview of this best-scoring system is shown in Figure 2.7. Given two

[5] `https://labrosa.ee.columbia.edu/projects/coversongs/covers80`

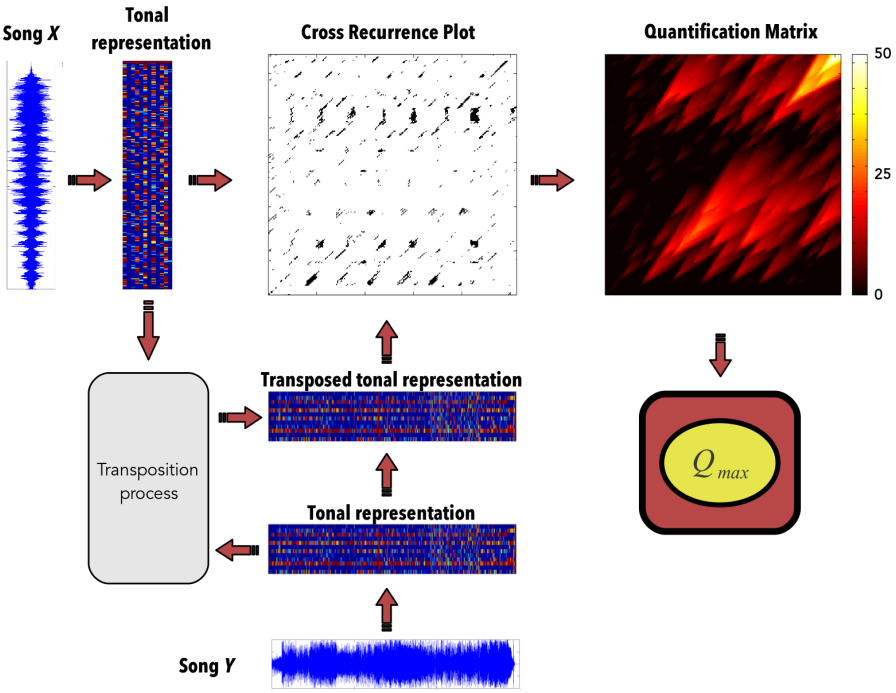


Figure 2.7: An overview of the Q_{max} algorithm, figure adopted from Serrà [2011].

songs, first a PCP time series descriptor is extracted, and one song is transposed to the main tonality of the other. Then a state space representation of the two songs using delay coordinates involving an embedding dimension is constructed. From this state space representation, a cross recurrence plot is created using a fixed maximum percentage of nearest neighbors. Subsequently, the Q_{max} algorithm is used to extract features that are sensitive to the characteristics of the song version. This algorithm is similar to the maximal length of diagonal lines L_{max} but capable of addressing the longest curved and potentially disrupted trace in the recurrence plot (tempo changes and disruptions of the tonal progression) (Serrà et al. [2009]).

Analysis of the CSI state-of-the-art results

It is remarkable how the algorithm of Serrà et al. remains the best-performing for the whole decade when applied to the popular music of Mixed Collection. Only a recent 2018 result reached a slightly better performance, but only for the Mazurka collection (14 more mazurkas retrieved, but 322 less popular songs retrieved for Mixed Collection by Chen et al. [2017]). On the one hand, it reassures the use of DTW-based methods until the present day. On the other hand, as countless other studies are using DTW methods and the one from 2009 is still the best for CSI, perhaps the full potential has been reached, and the algorithms have to be changed. Table 2.1 presents all the MIREX algorithms since 2006 that have had a chance to compete with the algorithm of Serrà et al., sorted from the oldest to the newest. The *total number of covers identified in top 10* score from MIREX is shown in the *Identified* column, from the maximum of 3,300 cover songs of the MIREX Mixed Collection.

Algorithms with at least half of these covers identified (1,650) are listed. If the same first author submitted more than one algorithm to MIREX, the best-performing algorithm is featured. It can be observed that the research teams have indeed used many new ideas throughout the years, including fusion methods or deep neural networks. The results are certainly improving, and we can expect new contestants to challenge the state-of-the-art in the next years.

Table 2.1: A review of the best CSI algorithms submitted to MIREX benchmark during the years 2006–2018.

Authors	MIREX	Identified	Short description
Egorov and Linetsky [2008]	2008	1,778	PCP feature extracted using special band-pass filters, peak analysis, and suppression of higher harmonics
Ravuri and Ellis [2009]	2009	2,046	Multi-feature approach (LabROSA 2006 and Serrà et al. 2007 similarity matrices) with a support vector machine classification
Serrà et al. [2009]	**2009**	**2,426**	**Cross-recurrence quantification, search for the maximal length of diagonal lines – the Q_{max} algorithm**
Chan et al. (Wang et al. [2014b])	2013	1,714	Chroma DCT-reduced log pitch features (Müller and Ewert [2010]) combined with cross-recurrence quantification (Serrà et al. [2009])
Chen et al. [2017]	2017	2,104	Similarity Network Fusion system (Wang et al. [2014a]) combining HPCP and melody features, Q_{max} (Serrà et al. [2009]), D_{max} (Yang and Chen [2016]), using sparse group LASSO on the similarities to obtain the final probability
Tralie [2017a]	2017	1,967	Similarity Network Fusion system (Wang et al. [2014a]) combining MFCC and HPCP
Lee et al. [2018b]	2018	2,104	Convolutional neural network (ConvNet-based) applied to cross-similarity matrices of chroma features specified in Jiang et al. [2011]

Based on the research of the last few years, the current trends in CSI are:

- Combining various features rather than trying to extract a brand new feature. This goes along with finding a way to combine the similarity measures achieved by the multiple features, rather than relying on a single similarity measure for all. One can use either number of the similarity measures (DTW score, Q_{max}, D_{max}, Shannon information by Foster et al. [2015], or others) and the final result can be achieved by database pruning and optimal rank aggregation or by a fusion model (Osmalskyj [2017], Tralie [2017a,b], Chen [2016], Chen et al. [2017]).

- The focus is shifting to the large-scale methods (Bertin-Mahieux and Ellis [2012], Humphrey et al. [2013], Correya et al. [2018]) rather than outperforming the results on the smaller datasets.

- The most recent trend is the use of Convolutional Neural Networks (CNN) (Lee et al. [2018b,a], Chang et al. [2017] or a Similarity Network Fusion (SNF). Although the CNN model has good results, it was the SNF algorithm featured in Nature Methods that has outperformed the best results for the Mazurka dataset on the MIREX benchmark[6] (Wang et al. [2014a]). The CoverNet deep neural network of Lee et al. is based on the publicly available ConvNet[7], which was itself

[6] `https://www.music-ir.org/mirex/wiki/2018:Audio_Cover_Song_Identification_Results`
[7] `http://libccv.org/doc/doc-convnet`

based on the ground-breaking work of Krizhevsky et al. [2012] (referred to as *AlexNet*). The authors also suggest ranking strategies using the representation vector outcomes of the neural network, along with the predicted probability.

Analysis of the features used for CSI

An interesting recent study was conducted by Correya et al. [2018], where metadata and lyrics were used along with the musical content. Although this information is not always available, and having them clearly makes the task easier (by including, e.g., the title), their study shows how important are lyrics and metadata compared to the musical content. The best score (MAP score of 0.531 for the *SecondHandSongs/MSD* dataset[8] compared to the previous best scores lesser than 0.15) was achieved when all these features were combined.

When it comes to the standard task when only audio is available, the chroma-based features dominate the CSI task. The reader can notice that the best algorithms, including those by Serrà et al., Lee et al. and Chen et al. all start with chroma-based features. However, there are notable uses of entirely new features: Hashed Chroma Landmarks (Bertin-Mahieux and Ellis [2011], 2D Fourier Transform Magnitude (Bertin-Mahieux and Ellis [2012]), 2D-Fourier Magnitude Coefficients (Humphrey et al. [2013]), etc. The use of MFCC vectors itself was first pioneered by Tralie and Bendich [2015] when chroma features were dominating the state-of-the-art. Nowadays, MFCCs are considered one of the features that should be taken into account for CSI.

In summary, the idea that the harmonic and melodic content should be invariant over the cover songs is a centerpiece of the majority of the CSI works. In this context, we think that there have not been many studies up to date focusing on chords, keys and more in-depth functional analysis and its impact on CSI, perhaps because of the added complexity that it requires. Lee [2006] used Hidden Markov Models to extract chords from chroma vectors and then compared the chord sequences using DTW. Bello [2007] and Robine et al. [2007] have studied string alignment methods on chord sequences. Even though the above mentioned chroma-based systems outperformed the few chord-based algorithms on MIREX, the full potential of the role of music theory has not been explored yet. Especially with the possibility of using models of music cognition, new distance measures, complexity, or functional analysis, which is our focus for Chapter 3.

[8] `https://labrosa.ee.columbia.edu/millionsong/secondhand`

3. Music harmony model and harmonic complexity

We now switch the topic back to the tonal harmony in an attempt to develop a model useful for MIR. Our effort is not only to extract features useful for CSI, but also to analyze the songs with regards to harmonic complexity, or to visualize the harmony progression of the song.

This chapter uses the strong foundations of the related works on musicology from Chapter 2, and provides original contributions. The contributions in this chapter can be summarized as follows:

1. Creating a new model of music harmony called *TSD distance model (tonic-subdominant-dominant)* and putting it in the context of other theoretical models.

2. Defining new distances between chords and chroma vectors: *Chord complexity distance* (ChordCD), *Chroma complexity difference* (ChromaCD). We propose to use these distances in a time series as a fingerprint of the harmonic content of a song. If all the values of ChordCD for a song are averaged, we get the value which we propose to name *Average Transition Complexity* (ATC), representing a harmonic complexity of that song.

3. Case studies featuring analysis of popular songs by authors such as Oasis, Queen or Beatles, as well as pieces of classical music. This part also features a proposal for visualization of important/interesting harmony sequences in music.

3.1 Motivation for a new model of harmony

The gap between music theory and recent MIR applications has been pointed out by multiple researchers, calling for more work on how music theory can help recent retrieval tasks (De Haas et al. [2012], Lewis et al. [2015]). Even if the application provides a valid result (e.g., retrieves a correct song), it may be considered a „black box“ from the user's perspective with no understanding why the result was made such. Our work is motivated by these shortcomings, and we choose to exploit the rules of music theory and provide visualizations for the end user. The analysis does not assume to be complete in the context of all music properties, including melody or rhythm. We focus on music harmony, since it is one of the most critical aspects to describe the song structure, and can also be used effectively for cover song identification, as was shown in Chapter 2.

3.1.1 Harmonic Complexity

Several tasks, such as music classification, music recommendation, or CSI, require us to think: Should we rely on the computer-extracted features or

prefer human analysis? Let us take an example use case of music recommendation for music radios. There have been approaches to do a fully automated extraction, based only on the content (Schönfuss [2011]), nowadays presented as a music recommendation service Mufin[1]. In other systems like Pandora music radio[2], the extraction of relevant features is still a domain of human analysis, mostly by the trained musicologists or experts. And finally, the collaborative approach taking into account the user preferences is very common for the majority of other solutions, including, for example, Spotify[3]. All of these approaches usually employ a machine learning method to process the extracted features.

It is difficult to say which approach is the winning approach, as each of them was validated by years of successful usage in applications. In an attempt to honor the analysis of experts and exploit the speed of computer algorithms at the same time, we propose a hybrid approach and a new feature, *harmonic complexity*. In order to obtain this feature, a process takes place which simulates the process a trained musicologist would use to analyze the musical piece. This analysis is known as a *functional analysis*, where by a function we mean the tonal functions, referencing Attachment A.5. The analysis itself is a useful by-product, but to interpret its results as a numeric value of harmonic complexity, we take further steps.

Whenever the music obeys simple tonal harmony rules, we assign it a lower value of complexity, whereas if the rules are complex, or the harmony does not obey any known rules, we assign it higher values. We employ a concept similar to the computational complexity to finally output the measure of the complexity of the harmonic movements in the piece. The model empowering us to do so is called a *TSD distance model* and it is based on formal grammars from the computer science standpoint, and on the functional harmony analysis from the musicology standpoint. We believe that such a model fills the gap as mentioned above between the formalizations of music theory and MIR works.

3.1.2 Chord and chroma vector distances

The terms of chord and chroma vector distances are not defined formally in the literature and can suggest multiple definitions. In MIR, chords are usually represented either as 12-dimensional vectors similar to chroma vectors or as a sequence of tones (e.g., CEG). If we want to obtain a distance between two chords, we can, therefore, use vector or string distances. However, various other chord distances based on music theory have been proposed, which makes the decision of choosing one distance challenging. In Chapter 1 we presented, that one of the musicology standard is to use the *circle of fifths* and in Chapter 2 we have shown related models of 20^{th} and 21^{st} century that define musicological distances.

With chroma vector distances, the situation is slightly different – there is no research project, to the best of our knowledge, trying to define the

[1] `https://www.mufin.com`

[2] `https://www.pandora.com`

[3] `https://www.spotify.com`

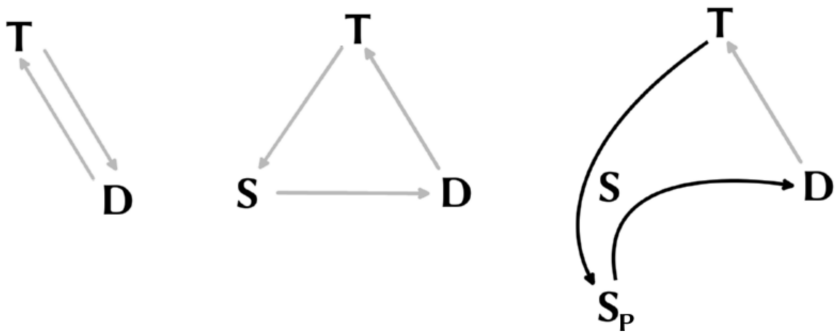


Figure 3.1: Increasing complexity of a harmonic movement: The transitions $T - D - T$ or $T - S - D - T$ are assigned zero complexity, but the transitions $T - S_P$ and $S_P - D$ are considered more complex.

distances between two chroma vectors, in some other way than the standard measures, e.g., Euclidean or Manhattan distance. Clearly, the use case for comparing two chroma vectors is typical for routines, for example, forming a self-similarity matrix. We think that for such vectors that represent musical data in time, there should be a way to derive distances other than Euclidean or Manhattan. For two musical moments described by chroma vectors, we thus focus on extracting a musicological distance, and we coin the term as the chroma vector distance.

More importantly, we see a disadvantage in using a full-length, frame-specific feature such as a 12-dimensional chroma vector. It comes with additional time overhead when processing long sequences. As we show in the following sections, having a distance measure can help to tackle the curse of dimensionality.

3.2 TSD distance model

The basic idea of our model is simple: The use of basic harmonic functions does not increase harmonic complexity. However, every single use of a parallel or added dissonances increases the complexity of a musical piece. We can recognize the use of complex harmonies by evaluating each pair of succeeding chords – evaluating their harmonic *transition*. If we shape the three basic transitions between T, S, D on a triangle, the idea of increasing complexity can be illustrated as in Figure 3.1. On the right, we see that the harmonic progression can skip the subdominant (S) and use a subdominant parallel (S_P) instead.

We base our model on the overtone series and consequent harmony rules by Schönberg [1922] and Riemann [1896a]. We further chose not to differentiate between the transitions $S - D$, or $D - S$, and the rest of the transitions between the basic harmonic functions. In the original literature, the transition $D - S$ violates the rules of the tonal harmony. However, there are some exceptions described by Zika and Kořínek [1990], and the mentioned transition is appearing more often in today's music. This is why we chose to treat all $T - S - D$ transitions as the same in complexity. Moreover, we chose the modifying of the basic harmonic functions to be our primary target in evalu-

ating the harmonic complexity. The transitions between the basic harmonic functions itself are, in principle, different from creating parallels or adding dissonances and therefore should not be considered complex in this concept.

We also merge the concepts of a *function parallel* and a *chord with an added dissonance* defined in Section 1.1.2 since both are a way to modify the basic harmonic function.

3.2.1 Formalization of transitions

We now want to evaluate the transition from the basic harmonic function to its parallel (e.g. $T \to T_P$) and in between the parallels (e.g. $T_P \to S_P$). That alone can provide us information on how much the music deviates from the simplest progression. We use a linguistic approach to evaluate these transitions.

First of all, let us consider the simplest case: $T \to T_P$. We wish to create a parallel either by adding a tone or modifying a tone. Formally, we define a *sentential form* as a type of a chord notation and two different parametric rules to modify the sentential form: *add(t)* and *alter(t,alt)*.

Definition. *A chord is written in a* sentential form $t_1t_2...t_n$ *if it consists of pitch classes* $t_1, t_2, \ldots, t_n$*, and for* $i \neq j$*:* t_i *is not in the same pitch class as* t_j*. Moreover* $t_1 \ldots t_n$ *are ordered in the order of the chromatic scale.*

The sentential form is therefore only an ordered enumeration of chord's tones, working within one octave. We further define the transition rules between the sentential forms in the same fashion as Hopcroft et al. [2001], but with notable differences. We design our own types of rules, applicable to the whole sentential form. We also try to avoid excessive formalism. In the following we assume, that the derivation starts with the sentential form representing the (original) basic harmonic function – an analogy with the start symbol of Hopcroft et al.

Definition. *The* add(t) *rule between two sentential forms* h *and* g *is defined as follows:*

$$h \xrightarrow[add(t)]{} g \Leftrightarrow (h = t_1, t_2, \ldots t_n) \wedge (g = t_1, \ldots t_j, t, t_{j+1}, \ldots t_n) \tag{3.1}$$

where t *belongs to the same key as* $t_1 \ldots t_n$*, and* $\forall i : t \neq t_i$*.*

We wish to simulate both adding dissonances and creating Riemann's parallels. Both transformations can be achieved using the *add* rule. If we set that the basic harmonic function that we start with does not necessarily need to contain three tones, but possibly also a single tone or two tones (substituting the basic harmonic function), using *add* rule we can indeed get a Riemann's function parallel. For example, we can get a parallel *cea* as $ce + a$, where *ce* substitutes the tonic *ceg* in *C major*. The *alter* rule, therefore, has a different role – moving the tone outside the key, and as such, creating sharp dissonances and alterations in the way that tonal harmony describes them (Schönberg [1922]).

Definition. *The* alter(t,alt) *rule between two sentential forms h and g is defined as follows:*

$$h \xrightarrow[alter(t,alt)]{} g \Leftrightarrow (h = t_1, \ldots t_i, t, t_{i+1}, \ldots t_n) \wedge (g = t_1, \ldots t_i, t_{alt}, t_{i+1}, \ldots t_n) \tag{3.2}$$

where t belongs to the same key as $t_1 \ldots t_n$, t_{alt} does not *belong to that key, and t_{alt} is created from t by augmentation ($alt = \sharp$) or diminution ($alt = \flat$) by a semitone. Moreover, t can not be one of the pitch classes of the original basic harmonic function h from the beginning of the derivation.*

The constraints we cast on the *add* and *alter* rules yield the rules of tonal harmony. First of all, we can not modify the original tones, because we do not want to lose the sense of the basic harmonic function. If we wish to weaken the basic harmonic functions, we always have the possibility to start with only a subset if its tones. Secondly, we can add the tones, but only from the same key. And thirdly, we can alter the tones to a different key, but only after they have been added. Such hierarchy agrees with Lerdahl's definition of tonal pitch space (Lerdahl [2001]). The reader can also take a short exercise to find out how many *add* or *alter* rules are needed to form the basic function parallels shown in Attachment A.5.

Note that, using *add* and *alter* rules, we are able to create arbitrary harmonies. The only thing we need to keep in mind is that, given a sentential form h, we first have to find out, which key h belongs to. There are 24 possibilities, comprising 12 major and 12 minor keys. Once we find the key, we can apply the *add* and *alter* rules to derive the chord in question.

Example. In a simple example we derive the tonic parallel (T_P) $cef\sharp g\sharp$ from the original tonic (T) ce, in the key *C major*:

$$ce \xrightarrow[add(f)]{} cef \xrightarrow[alter(f,\sharp)]{} cef\sharp \xrightarrow[add(g)]{} cef\sharp g \xrightarrow[alter(g,\sharp)]{} cef\sharp g\sharp. \tag{3.3}$$

We first define a chord complexity for a simple chord. Then we proceed to define the complexity of the transition between two chords (chord complexity distance).

Definition. *A* chord complexity $c(h)$ *in the given key k of a chord h is the minimal length of derivation of the chord's sentential form in the context of the key k.*

Example. For our example tonic parallel: $cef\sharp g\sharp$, the chord complexity in the key *C major* is 4. However, the chord complexity for this chord can be even lower. If we choose the key *G major* and treat the form ce as subdominant in *G major*, we will have 3 as the resulting chord complexity (due to the fact that the tone $f\sharp$ can be added in one step – it is a part of the key *G major*.

Understanding that the chord h can be treated in multiple keys, that it can have multiple derivations and multiple chord complexities, we always use the shortest derivation (lowest complexity) if the key is not specified. We label such chord complexity $c(h)$. We use a similar approach in our final definition for chords – chord complexity distance labeled ChordCD.

Definition. *Let us have the sentential form of a chord h_1 and the sentential form of a chord h_2. We now define a* chord complexity distance *$ChordCD(h_1, h_2)$. We differentiate between these possibilities:*

1. *Let us assume, that there is a common ancestor in both derivations of h_1 and h_2 – a sentential form, that h_1 and h_2 can be both derived from, h_1 in k_1 steps, and h_2 in k_2 steps. Then if $k_1+k_2 \leq c(h_1)+c(h_2)$, the chord complexity distance $ChordCD(h_1, h_2)$ equals to $k_1 + k_2$. Otherwise, $ChordCD(h_1, h_2) = c(h_1) + c(h_2)$.*

2. *Let us assume, that there is no such common ancestor. $\exists r_1, r_2, k_{h1}, k_{h2}$: h_1 can be derived from an original basic harmonic function r_1 in k_{h1} steps, h_2 can be derived from an original basic harmonic function r_2 in k_{h2} steps.*

   (a) *Let us assume that r_1, r_2 are such basic harmonic functions, that are from the same key and the sum $k_{h1}+k_{h2}$ is minimal. Then the chord complexity distance $ChordCD(h_1, h_2)$ equals to $k_{h1} + k_{h2}$.*

   (b) *The common key for r_1 and r_2 does not exist. Then the chord complexity distance $ChordCD(h_1, h_2)$ equals to u_1+u_2, where u_1 is the number of steps required to derive h_1 from the empty sentential form, and u_2 is the number of steps required to derive h_2 from the empty sentential form.*

In other words, we define the ChordCD as the complexity of the transition between two chords. The transition complexity is evaluated as the number of steps needed to disassemble the first chord into its basic harmonic function (rolling back the derivation), then we switch the function (which requires no complexity), and finally, we count the number of steps to assemble the new chord. The total number of *add*, *alter*, or reverse *add* and *alter* steps is the transition complexity. If there is a common ancestor in derivations, closer than the basic harmonic function, we disassemble and assemble only to and from this common ancestor. Finally, if no common key can be found for the two harmonies, we choose to disassemble the first chord all the way to an empty sentential form, and construct the second chord from an empty sentential form.

Notably, in each of the above scenarios, the main idea is the constructing and deconstructing of the chords, using our derivation rules. This is when the analogy with the computational complexity steps in. The chord complexity distance can be understood simply as the computational complexity of reconstructing the chord h_1 to h_2. Following this analogy, the whole harmonic complexity of a musical piece can be understood as the number of computational steps needed to traverse through all of the chords in the piece, one after another.

The reader may inquire about the use of a word *distance*, to see if the symmetricity and triangle inequality holds for the ChordCD, as is typical for the geometric distances. First of all, the ChordCD can be conceived as

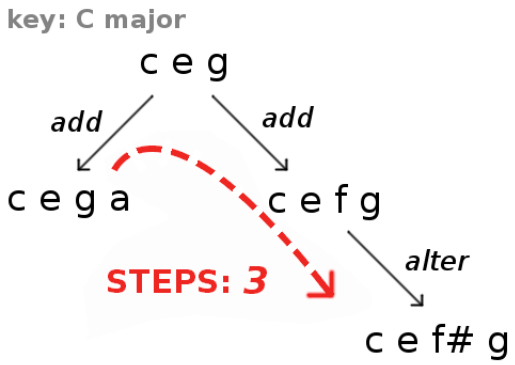


Figure 3.2: Chord complexity distance model based on adding/removing tones from the same key, and altering the tones outside the key.

a variation of *Levenshtein edit distance* for strings (Levenshtein [1966]), for which we know that it satisfies both symmetricity and triangle inequality. Indeed, the *add* and *alter* rules merely represent the insertions, removals or substitutions of characters, only with some more constraints. The changes between *T*–*S*–*D* can be conceived as a simultaneous substitution of characters, with a 0 distance. None of the changes break the symmetricity or triangle inequality.

Secondly, we show later in Section 3.2.3, that each derivation can be represented in a graph, where chords represent nodes. This analogy informally proves both symmetricity and triangle inequality of the ChordCD distance. For the formal proof of the symmetricity we could simply follow the definition directly: $ChordCD(h_1, h_2) = c(h_1) + c(h_2) = ChordCD(h_2, h_1)$, only we need to show it for all cases from the ChordCD definition. For the formal proof of the triangular inequality, the best way is to prove that the graph representation is indeed equivalent to our formal definitions of the model. That can be done by showing that the definitions do not allow negative distances, that every application of the *add* or *alter* rule creates a new chord with a distance of 1 from the source, and that for every pair of chords we can find a valid derivation.

An illustration of a chord complexity distance within one basic harmonic function (tonic) is shown in Figure 3.2. In this figure:

- Arrows represent the *add* and *alter* rules, starting from the basic harmonic function *ceg*.
- Dotted arrow represents the transition from the chord *cega* to $cef\#g$. One must remove (reverse add) the tone a, add the tone f and alter the tone f to $f\#$.
- The final complexity distance is 3, analogous to the edit distance of strings.

A more complex transition from a subdominant parallel to a dominant parallel is shown in Figure 3.3. This figure shows the harmonic functions in a directed graph:

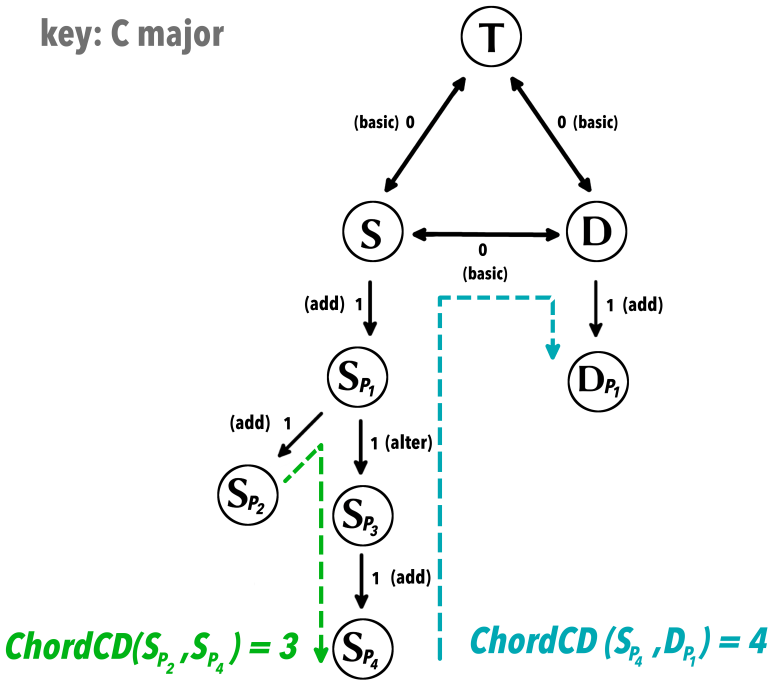


Figure 3.3: A more complex example of a chord complexity distance shown on a directed graph.

- Nodes represent the harmonic functions; edges represent the rules. In between the basic harmonic functions, there are edges with a 0 complexity, while the *add* and *alter* rules are assigned a complexity of 1. The type of each edge is shown in brackets.

- The dotted lines represent the final transitions. Note that the direction of the edges has meaning only for adding and altering the tones. When evaluating the final transition, it is possible to follow the edges in the opposite direction.

- If we spotlight on the functions S_{P_2} and S_{P_4}, we see a common ancestor in the derivation from S (S_{P_1}). The distance to and from this ancestor yields the ChordCD value of 3. The functions S_{P_4} and D_{P_1} do not have a common ancestor in derivation, but they share a common key. Therefore, their ChordCD is equal to the sum of their chord complexities ($3 + 1 = 4$).

Finally, we show some more difficult examples: A transition in between chords that do not share a common key is shown in Figure 3.4 *a)*:

- This time, we show the actual chords as nodes, by listing their pitch classes.

- The chords $dfga$ and $f\#a\#c\#$ cannot be treated as either of tonic, subdominant, or dominant of a common key. The chord complexity distance is therefore equal to the number of steps needed for deconstructing the chord $dfga$ to an empty sentential form plus the number

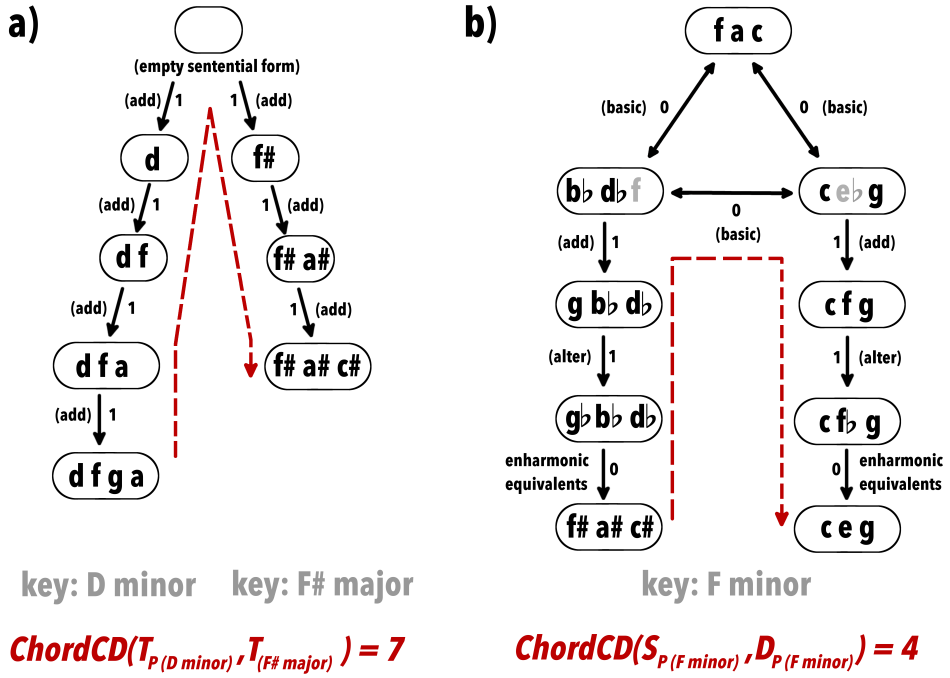


Figure 3.4: *a)*: An example of deriving a chord complexity distance for two chords that do not share a common key.
b): An example of seemingly impossible transition between chords $f\#a\#c\#$ and ceg via *F minor* key.

of steps needed to construct the chord $f\#a\#c\#$ $(4 + 3 = 7)$.

Next to it in Figure 3.4 *b)* we show an example of a difficult transition, where our algorithmic approach proved more effective than analyzing the transition by hand:

- The chords $f\#a\#c\#$ and ceg seemingly do not share a common key. However, our system searches for all the possible chord roots – including the less stable chord roots formed by only two tones (the missing tone is shown in gray) to find the best transition.
- It finds the transition within the *F minor* key, with the final ChordCD calculated using two *add* and *alter* rules on each side $(2 + 2 = 4)$.
- We also need to use the enharmonic equivalents of the tones, as they have the same pitch, only a different name (see Attachment A.1). We show this renaming by an edge with complexity 0, although we note that it is also possible to simply consider the equivalent chords as one node.

Moreover, this is only one of the possible transitions between the two chords, in this case, the one that yields the lowest ChordCD. When analyzing by hand, it may be impractical to find such transitions, as it takes additional time and is far from trivial. The power of the TSD model is that all such transitions can be found and evaluated momentarily, as we show in

Section 3.2.3 on implementation methods, or in our *harmony-analyser.org* application reviewed in Chapter 6.

3.2.2 Harmonic complexity of a musical piece

We can now proceed to the evaluation of harmonic complexity for a whole musical piece. Let us have a musical piece M. We can use different algorithms to extract the sequence of chords $\{C_i\}_{i \leq l}$ of the length l. Keeping in mind the analogy with the computational complexity, we are interested in obtaining an average number of steps needed to construct the following chord from its predecessor. We average the values because we want to abstract from the length of the progression, in the same fashion as the computational complexity theory abstracts from the length of the input.

Definition. *An average transition complexity (ATC) for a musical piece M, and a sequence of chords $\{C_i\}_{i \leq l}$ of the length l is defined as follows:*

$$ATC(M) = \frac{\sum_{i=0}^{l-1} ChordCD(C_i, C_{i+1})}{l-1} \tag{3.4}$$

We have experimentally studied also other aggregates of ChordCD for a musical piece, e.g., the *average chord complexity* (omitting the transitions, and looking only at the complexity of chords), or the *relative ATC* (ATC divided by the number of co-sounding tones present) (Maršík [2013]). These variations of a harmonic complexity may perform better in different situations. For example, if we used the average chord complexity, we would have gotten a high complexity result for a song that never uses a basic harmony function, while an analysis of the same song using ATC would not detect this anomaly. The relative ATC would assign more complexity to a solo instrument piece using a complex selection of tones, than to a polyphonic piece using a more narrow selection of tones, while the standard ATC would not detect that. However, for our initial studies on the datasets of popular songs, the ATC showed the most discriminative power (Section 3.5.3), and also has the closest analogy to the computational complexity, so we propose this definition as the main definition of harmonic complexity for this thesis.

3.2.3 Implementation of the model

For any given chord, multiple basic harmonic functions can be found, along with multiple different keys. This ambiguity is what makes the model a bit more challenging to implement. We propose two methods of implementation, based on the internal representation of musical data:

1. A query method – The fundamental rules of tonal harmony (keys, basic harmonic functions) are saved in a database, and several queries precedes the calculation of ChordCD for chords.

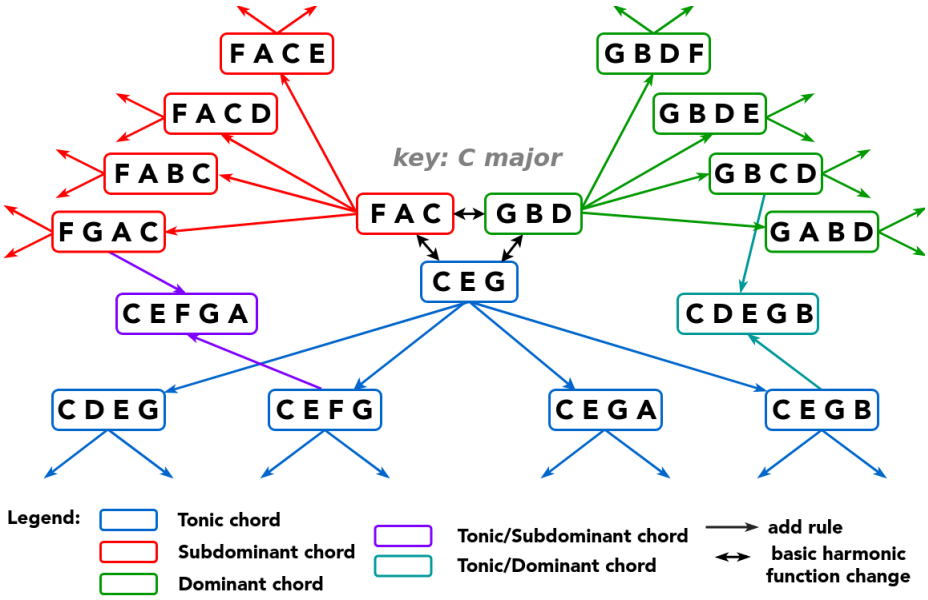


Figure 3.5: A graph representation of the TSD distance model

2. A graph method – We can abstract from the ambiguity of keys and functions and avoid complicated queries. Working within the limited space of all possible sets of tones in the octave, we have designed our derivation rules in such fashion, that for a given chord it is easy to calculate all chords with ChordCD equal to 1 from a given chord. This approach can result in constructing a graph representation of the problem, with edges representing a ChordCD equal to 1.

Graph method of implementation

The graph representation of the TSD distance model is depicted in Figure 3.5:

- Each existing chord is assigned a node.
- Only a fraction of the full graph is shown, as the full graph consists of 24 sub-graphs for 24 tonalities, each sub-graph centered around its basic harmonic functions.
- The edge exists between the two chords if their ChordCD is equal to 1. Depicted are only the *add* rules, but similarly, the *alter* rules can be added.
- The 0-complexity edge exists between the basic harmonic functions
- In the figure, colors represent the basic harmonic function (blue for tonic, red for subdominant, green for dominant). Some chords can represent multiple harmonic functions, e.g., the chord $cefga$ represents both tonic ceg and subdominant fac in the C *major* key, so we highlighted it in violet. Similarly, the sub-graphs from various keys share common chords extensively.

One interesting observation from the graph in Figure 3.5 is that the chords are shared in between the keys: e.g., the $fgac$ node, or the fac basic harmonic function would similarly exist in the *F major* subgraph. There is a different connection between the keys, as they are connected through the circle of fifths. The *C major* key is connected to *G major* key for instance via the *gbd* chord, the *G major* key is further connected to the *D major*, etc. The TSD distance model does not specify any complexity for these shifts in between the keys and the ChordCD distance is indeed equal to 0 for these transitions. For the TSD distance model, the problem of modulation was never meant to be the subject of the study. True to its name, the TSD distance model merely measures the distance from the basic tonic, subdominant and dominant. There is only a single rule that for a transition in between two chords. only one modulation can occur.

However, the modulations should be studied more, and there is a need for a separate model or an extension of the TSD distance model to handle modulations and their complexity in a song. A simple extension would be to assign a *modulation complexity* (distinct from the harmonic complexity) equal to the number of steps on the circle of fifths, each time a modulation occurs in between the chords. We can then represent each song by a combination of the two complexities (harmonic and modulation), rather than just a single complexity, which is analogous to Lerdahl's diatonic and fifths level (Lerdahl [2001]).

Remark. *The graph representation from Figure 3.5 offers a few analogies that we wanted to point out. One is that the harmonic complexity represents the energy needed to move within the music space where the tonic-subdominant-dominant triangle forms a „music gravitational field". In this analogy, a modulation would not be a movement, rather an immediate change of the music space.*

Another interesting analogy is that music being played can be represented as a path on the graph representation of the TSD distance model. Looking back on Figure 3.5, a chord change within the same key can be observed as a movement along one or more of the edges. A chord change that modulates to another key triggers a sudden change of the „routes", e.g., from the edges specific to C major *to those of* G major*.*

These imaginary changes of the key space or the routes are quite analogous to the perception of the tonality change by the listeners.

If one chooses to implement (pre-compute) the TSD distance model using the graph representation, the space complexity is limited to $2^{12} = 4,096$ nodes (possible combinations of 12 tones in one octave). To construct the edges between the nodes, a simple loop over the nodes can be utilized, trying to add or remove tones. This is a promising start for the implementation – although the modulations make the graph implementation a bit more challenging.

However, the difficulties arise when we also want to know which nodes belong to which key or function. To extend the graph to include all of this information inside the nodes would require us to pre-compute all answers

of the whole model in advance, which is computationally demanding. If we want to get a detailed output, the query method of implementation is the best solution. But, one lesson learned from this section is, that if one wants to achieve the best performance along with detailed output, a hybrid method can be used, that also exploits the graph analogy. We could, for example, output the distance quickly by the breadth-first search, while the additional queries may be run for the context information.

Query method of implementation

As we mentioned earlier, the query method of implementation brings computational overhead, but it brings the advantage of having all answers just one query away. The query method of implementation starts with all the labels for tones, chords, and keys of the tonal harmony being saved in a database. From there, model classes such as *Tone, Chord, Key* may be created to encapsulate the tonal harmony entities and basic operations within them. The next step is to create functions representing the basic tonal harmony rules. Finally, the TSD distance model definitions are to be programmed based on the definitions from this chapter.

The source code of our solution to the tonal harmony and TSD distance model is freely available at the domain *harmony-analyser.org*[4] and we discuss this project more in-depth in Chapter 6. The pseudocode of the ChordCD function shown in Algorithm 1 can be helpful as an overview of the used routines and how to put them together in the final calculation. Our implementation uses routines such as: *roots*, *chordComplexity*, *commonRoots*, or *nearestCommonAncestor*, which we will not elaborate to the details, but we provide a short description below:

- The *roots* function is arguably the fuel behind the whole system, outputting an array of possible root tones, keys and harmonic functions for an arbitrary chord. For example, for a chord $cc\#ef\#g$ it would output a possible *root ceg*, in the *C major* key as a tonic, but also *ceg* in the *F major* key as dominant and *ceg* in the *G major* key as a subdominant. This function needs to traverse all key-function combination and also include other matches, e.g., $f\#c\#$ as a less stable tonic of *F# major*.

- The *chordComplexity* function is another commonly used routine of the TSD model. Given the chord and the *root* (tuple: [chord, key and function]), it uses the *add* and *alter* rules to find the sequence needed for derivation, and outputs its length.

- The *commonRoots* function finds a common key context for two arrays of roots. For example, for an array 1: *[[C major, tonic], [G major, subdominant]]* and array 2: *[[D major, tonic], [G major, dominant]]*, the common roots would be: *[[G major, subdominant, dominant]]*. The output is essential for the next steps, as it specifies in which keys is it possible to proceed, and which functions the two chords represent.

[4] `http://harmony-analyser.org`

- Finally, the *nearestCommonAncestor* function finds the common ancestor in the derivation of two chords, as was shown in Figure 3.3. It reuses the *chordComplexity* code – the sequence of *add* and *alter* rules – to find the common subsequence.

Importantly, the output of the *nearestCommonAncestor* routine needs to be indexed by the used common roots. For example, the nearest common ancestor *ce* for chords *cega* and *cdef* is found within the context of a common root: [*C major*, tonic, tonic], but it could also be found in the context of a common root: [*C major*, tonic, subdominant]. We need to remember both of these contexts along with the *ce* common ancestor. For this purpose we use a dictionary construct, labeled as *map* in Algorithm 1. Our application makes use of multiple indexing mechanisms (e.g., a hash map) and remembers all possibilities so that the whole music space is searched. Knowing the context, we can then, on the lines 34 and 38 of Algorithm 1, find the minimum complexity. We traverse all the common roots, sum up the chord complexities, and output the lowest complexity found.

3.3 Chroma complexity difference

We now apply the concept of the TSD distance model to chroma vectors. The following definition is to be understood as an extension of the TSD distance model since we defined the model to work only with a string representation of chords. We start by converting the chords to a 12-dimensional vector: $< c_A, c_{A\#}, c_B, c_C, c_{C\#}, c_D, c_{D\#}, c_E, c_F, c_{F\#}, c_G, c_{G\#} >, c_t \in \{0, 1\}$, to achieve a representation similar to chroma vectors. For chroma vectors, the definition is equivalent, with $c_t \in \mathbb{R}$.

Our goal is to achieve a similar behavior as ChordCD(C_1, C_2) (C_1, C_2 are chords) for a new similarity measure ChromaCD(c_1, c_2) (c_1, c_2 are chroma vectors). The discrete adding and removing of the tones cannot be applied to real numbers, so we need to find another way of unifying the two distances.

We propose to preserve the main rules of the TSD model for chroma vectors. In particular:

1. The two chroma vectors should be aware of the context of the common key.

2. The two chroma vectors should be both aware of its chord root within the key (e.g., a chroma vector with prevalent tones *cdeg* has a chord root of *ceg* within *C major*). As with chords, the changes in between the chord root tones are not considered complex.

3. Adding (removing) the tone from the common key is assigned one unit of complexity (for chords: *add* rule).

4. Adding (removing) the tone from outside the common key is assigned twice as high complexity than for the tones from the common key (for chords: *add* rule followed by *alter* rule).

```
1: // input: chords c1 and c2
2: // output: ChordCD distance between the chords c1 and c2
3: function ChordCD(string c1, string c2)
4:     // search for the roots of each chord:
5:     // roots1 and roots2 are arrays of tuples
6:     // of the form: chord × key × function
7:     // for example: [C major, C major, tonic]
8:     array[array] roots1 ← roots(c1)
9:     array[array] roots2 ← roots(c2)
10:
11:    // search for common roots (multiple roots can be found):
12:    // the result is an array of tuples, this time in the form:
13:    // key × function × function
14:    // for example: [C major, tonic, dominant]
15:    array[array] commonRoots ← commonRoots(roots1, roots2)
16:
17:    // search for the nearest common ancestor (as shown in Figure 3.3)
18:    // commonRoots may be empty, in that case nca is also empty
19:    // The result is a map, indexed by the used commonroots
20:    map[array→string] nca ← nearestCommonAncestor(c1, c2, commonRoots)
21:
22:    // assigning a chord complexity to each chord, as shown in Section 3.2.1
23:    // commonRoots may be empty, in that case chordComplexity
24:    // goes up until the empty sentential form
25:    // The result is a map (dictionary), indexed by the used commonroots
26:    map[array→int] comC1 ← chordComplexity(c1, commonRoots)
27:    map[array→int] comC2 ← chordComplexity(c2, commonRoots)
28:
29:    // returning the ChordCD as defined in Section 3.2.1
30:    if (commonRoots.length > 0) then
31:        // common roots were found, so we need to sum the chord complexities
32:        // up until the nearest common ancestor
33:        map[array→int] comNCA ← chordComplexity(nca, commonRoots)
34:        return min_(commonRoots)(comC1 + comC2 - (2 * comNCA))
35:    else
36:        // no common roots were found, so we sum the chord complexities
37:        // up to the empty sentential form
38:        return min_(commonRoots)(comC1 + comC2)
39:    end if
40: end function
```

Algorithm 1: Computation of ChordCD

Following these rules, we propose a similarity measure for two chroma vectors based on a simple vector difference, improved by weighting each bin according to the tone it represents.

Definition. *A chroma complexity difference (ChromaCD) is defined as follows:*

$$ChromaCD(x,y) = \sum_{i=1}^{12} |w(x)_i x_i - w(y)_i y_i| \tag{3.5}$$

where, $w(x)$ and $w(y)$ are 12-dimensional weight vectors, which have their values dependent on the context for the transition from x to y, by the following rules:

$$\begin{aligned} w(x)_i = 0 &\Leftrightarrow c(x)_i = 1 \\ w(x)_i = 1 &\Leftrightarrow k(c(x), c(y))_i = 1 \wedge c(x)_i = 0 \\ w(x)_i = 2 &\Leftrightarrow k(c(x), c(y))_i = 0 \wedge c(x)_i = 0 \end{aligned}$$

where function $c(x)$ returns the chord estimate for a given chroma vector x, and function $k(c,d)$ returns the key estimation for chords c and d.

Both functions k and c have their return values in the form of binary vectors of size 12 (for a chord estimate $c(x)$ of a chroma vector x, $c(x)_i = 1$ if the pitch class i belongs to $c(x)$, $c(x)_i = 0$ otherwise). For a vector returned by the function c, exactly three values are 1, and for a vector returned by function k, exactly seven values are 1, which are the standard numbers of tones for chords and keys.

Example. As an example transition, we can imagine x and y both being estimated to *C major* chord ($c(x) = c(y) =$ *C major* chord) and *C major* being the common key ($k(c(x), c(y)) =$ *C major* key). The following 12-dimensional vectors start from the tone A:

$$c(x) = c(y) =< 0,0,0,1,0,0,0,1,0,0,1,0 >$$

$$k(c(x), c(y)) =< 1,0,1,1,0,1,0,1,1,0,1,0 >$$

and

$$w(x) = w(y) =< 1,2,1,0,2,1,2,0,1,2,0,2 >$$

The weight value of 0 occurs for the chord tones (c, e, g). The weight value of 1 occurs for the tones (d, f, a, b). The weight value of 2 occurs for the tones $(c\#, d\#, f\#, g\#, a\#)$.

In other words, using the weight vector $w(x)$, we suppress the changes in between chord tones ($w(x) = 0$), we take into account changes in between the non-chord tones from the key ($w(x) = 1$), and we assign highest weights to the non-chord and non-key tones ($w(x) = 2$), to emphasize the non-diatonic movements in music.

Functions for chord and key estimation can be chosen independently from the definition. We have employed a simple chord estimation, where $c(x)$, is the closest chord (using Euclidean distance) to the chroma vector x. The key estimation can be accomplished by a ranked list of chord usage for a given key (more used chords have higher ranks in the list). Function k then simply chooses a key for which the sum of ranks of the two chords is minimal.

What really the ChromaCD measure does, is that it observes the complexity of movements in tonal music in real time. The idea stays the same as for the TSD distance model: if the music stays within the basic tonic, subdominant, and dominant, the chroma vector distances stay very low. However, if interesting tones outside the basic functions are added (and especially, dissonances), the chroma vector distances detect this and have much higher values. Both ChordCD and ChromaCD distances are also robust to the key transposition of the song.

3.4 Comparison of TSD distance model with other models

One of the essential things when proposing a new model is a comparison to the other state-of-the-art models. This can be done experimentally, or theoretically. We devote three experiments in Chapter 4 to the comparison of TSD distance model to other models and features, as well as one case study at the end of this chapter.

For the theoretical comparison, we choose the most related models from the Chapter 2: Lerdahl's TPS, and Chew's Spiral model (Lerdahl [2001], Chew [2014]). First of all, this comparison needs to start by stating that although the models are related, they are all still very unique and their end goal is not always the same. Notably:

- The models are based on different studies. Lerdahl's model is based on music cognition. Chew's model is based on a geometric representation of music. TSD distance model is based on the tonal harmony analysis and formal grammars / computational distance of strings.

- The models are solving different problems. The TPS model tries to map the listeners' intuitions of the relative distances of pitches, chords, and regions to a set of rules and mathematical constructs, where the task is mainly the analysis. Chew's model tries to organize music elements in space, and by doing such solve MIR tasks such as key finding. TSD distance model tries to solve the definition of harmonic complexity, distance from the basic T–S–D flow, and extract descriptors from a musical piece.

On the other hand, the three models do not contradict each other. They can all coexist as each model treats the music from a different perspective. They also share the characteristics of:

- Robustness, as an arbitrary chord can be formed in each of the models.

- Working with pitch classes within one octave.
- Extracting a distance measure between the chords. It can be achieved in a non-contradicting way that two chords can have a different TPS distance (from the perspective of cognition), spiral distance (from the perspective of geometry) and TSD distance (from the perspective of complexity).

As the three models should be treated as theories in a music domain, there is hardly any way to dispute one or another. Only case studies can bring us closer to understanding the usefulness of each of the models. We look at the case studies starting in the following section.

3.5 Case studies on audio files

To experiment with the model presented in the previous sections, we start with standard music processing tasks – obtaining chroma vectors and segmentation, as described in Section 1.2. We omit the details of these steps, as they are described in Chapter 4. The aim of this section is mainly to complete the understanding of the proposed model.

However, there is one particular part of the music processing that is worth mentioning here. The part where we label the chords, commonly performed by using a chord dictionary (Section 1.2.4) is quite specific when using the TSD distance model. Our model has the capability of parsing an arbitrary chord, and in fact, it does a better job when dealing with complicated, dissonant chords. That is why, for each case study presented below, we allow more than three tones to form a harmony. A common way is to set up an *audible threshold*. If a chroma value at a particular bin is higher than a threshold, we consider that tone to be present.

As a result, even chords that are difficult to describe by music theory, containing multiple dissonances, can appear in our analysis. There are advantages and disadvantages to this approach. Dissonant tones can be treated as invalid information towards retrieval, such as noise. However, non-chord tones can also play an important role in the musical piece. An example can be a singer singing the voice above the chord accompaniment, causing a temporary dissonance. We find this approach quite unique in MIR, and it can have many applications in the future.

3.5.1 Time series descriptor for a musical piece

One of the final contributions of this chapter is to form a new descriptor for musical pieces. If we plot all *ChordCD* distances on a timeline, simply for each pair of successive chords, we get a characteristic fingerprint of the song. This time series descriptor was first extracted only from the curiosity of how complex the transitions in the song really are, but as we are about to show, it can also be used for retrieval.

The first case studies are performed on the songs Hallelujah by Bastian Baker and Wonderwall by Oasis. The characteristic curvatures of the songs can be seen in Figure 3.6. We also extracted TPS distances for comparison.

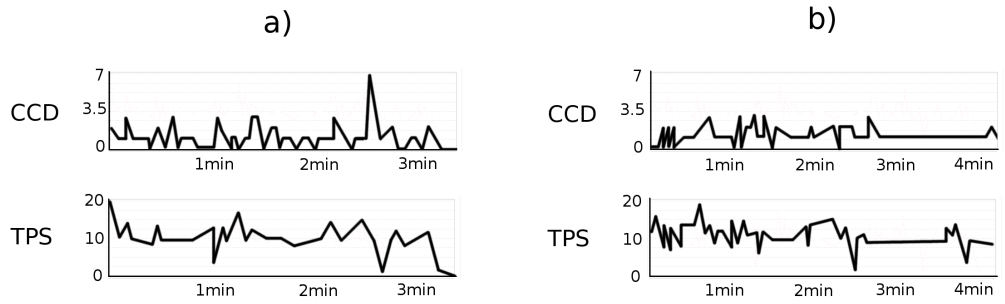


Figure 3.6: Analysis of songs: *a)* Hallelujah by Bastian Baker and *b)* Wonderwall by Oasis, for both Tonal Pitch Space distance (TPS) and Chord Complexity Distance (ChordCD). Peaks in analysis *a)* represent a complex chord progression before the chorus in 0:35, 1:15, or 2:30. Peaks in analysis b) around 0:40, 1:30 and 2:30 correspond to the A5 chord followed by B7sus4 chord.

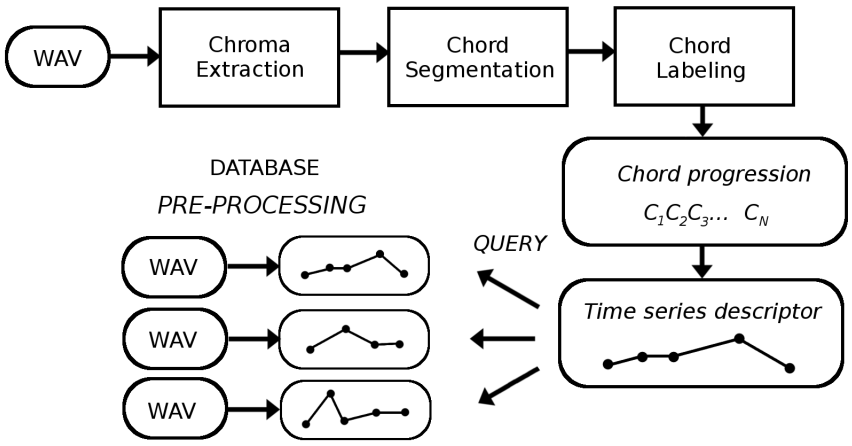


Figure 3.7: System outline for descriptor creation and retrieval based on fingerprints

In the figure, we can see that TPS has a bigger range than ChordCD. However, the peaks in Chord complexity distance are showing musically meaningful parts a bit more clearly. Peaks in Figure 3.6 *b)* around 0:40, 1:30 and 2:30 correspond to the A5 chord followed by B7sus4 chord. In Figure 3.6 *a)* there is a similar more complex chord progression before the chorus in 0:35, 1:15, or 2:30. These can also be observed by TPS, but not as clearly due to the bigger scale. An interesting attribute of the ChordCD is the sensitivity on the dissonances, which translated to the maximum peak in Figure 3.6 a) 2:30, and can be attributed to the ornamented singer's performance.

Overall, the results of the case study are in line with our expectation that the two distances are quite different. It also shows that forming a time series of chord distances indeed shows a curvature which is quite characteristic for a song. We want to highlight this by setting up a system for automated music retrieval shown in Figure 3.7, which we test in Chapter 4.

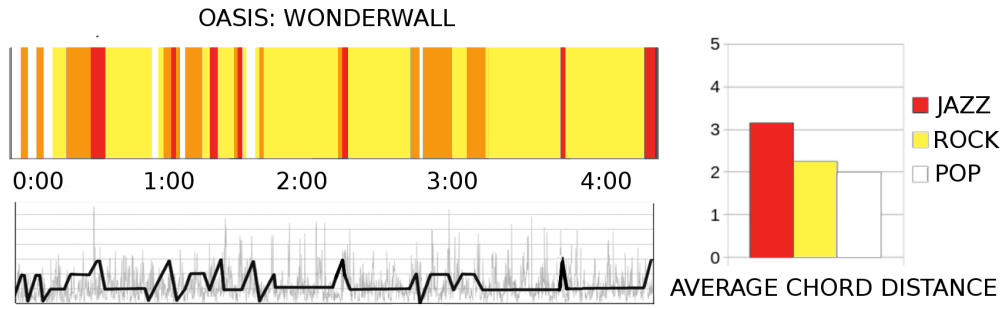


Figure 3.8: Visualization of the song Wonderwall by Oasis. On the top left, the chord progression is shown as a color temperature graph, on the bottom left as a line graph. Peaks around 0:40, 1:30, and 2:30 correspond to the A5 chord followed by B7sus4 chord. On the right there are average chord complexity distances for three genres: jazz, rock and pop on a corpus of 120 songs.

3.5.2 Visualization of interesting music parts

Another advantage of the time series descriptor introduced in the previous section is the ease of visualization of a musical piece. What follows is one way how we envision the visualization of harmonic complexity. As seen in Figure 3.8, both color temperature or line graph can be employed to show the progression. This visualization can be easily used while playing the music in a music player, or when performing live, the musician may expect difficulties and challenging chords before they happen. When the user retrieves a similar musical piece based on this descriptor, the visualization provides an understanding of why the piece was selected. The user sees all the important chord sequences in the visualization easily as the contrasting areas in the color temperature graph, or as the peaks in the line graph. These parts usually relate to unusual harmony movements, as are often in the bridge or before the start of the chorus. As we see in Figure 3.8, all the occurrences of the, arguably most interesting chord change from the song Wonderwall, A5 $\rightarrow$ B7sus4, are shown in red. There are also other chord changes highlighted in red, but some of them resulted from the singer's voice sounding together with the musical accompaniment. Although this may not be a good lead to find something interesting in the musical accompaniment, we may discover other things, e.g., the ornamentation of the voice.

On the right we also show the ATC (average transition complexity from Section 3.2.2, calculated as an averaging of ChordCD for the whole song). The experiments on a dataset of 120 songs highlight that the harmonic complexity can distinguish in between the musical genres. In this case, we clearly see that Jazz songs contain much more complex chord changes than Rock or Pop songs. Based on these experiments, we could say that, from the perspective of harmony, Jazz songs are more complex, or that they contain more interesting harmony progressions.

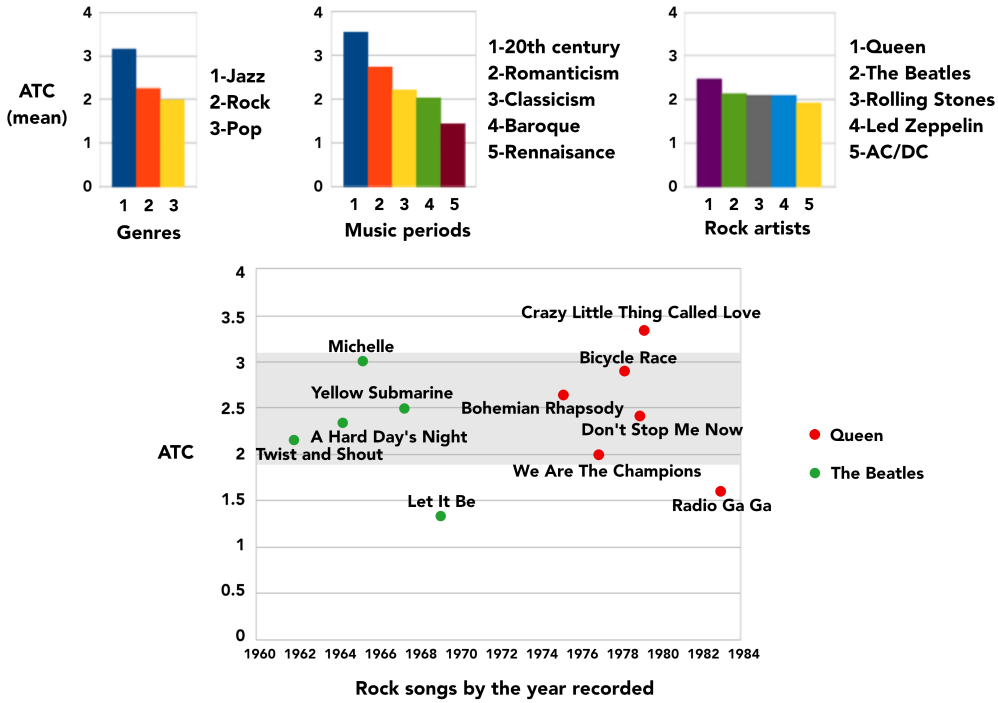


Figure 3.9: ATC harmonic complexity for genres, music periods, rock artists and individual rock songs

3.5.3 Case studies for music genres, music periods and artists

For our last case studies, we focus on determining whether the newly defined concept of harmonic complexity can indeed be a good descriptor for the music classification task. The aim is to show that *ATC* values can distinguish the different genres, artists in one genre, or even the songs from the same artist. The dataset was selected from the top five best-selling artists in 2013, according to renowned worldwide music charts and separate for each genre: rock and pop[5], jazz[6] and classical music[7]. In each genre, 25 different pieces were analyzed.

We split the experiments into three parts.

1. First, we measure the mean ATC for different music periods. Since classical music has developed through history and particular music periods are known to obey certain composition styles, we are interested to see the resulting complexity.

2. Second, we measure the mean ATC for different artists from the Rock genre.

3. Third we measure the ATC found for different songs selected from the 2 artists of Rock music: *The Beatles* and *Queen*.

[5]Source: `https://www.billboard.com`
[6]Source: `https://www.artistsdirect.com`
[7]Source: `https://www.classical-music.com`

Figure 3.9 shows the result of the analysis. The previous jazz/rock/pop classification from Figure 3.8 is included for reference. ATC values around 1 means that the transitions were not complex and were within the basic tonal harmony rules. On the other hand, values above 3 meant that the transitions needed on average more than 3 steps. As we mentioned in the previous section, our model gives the highest complexity ranking to jazz music. Rock music averaged around 2, and the lowest rankings were assigned to pop songs. This aligns with the expectations. There is an ambiguity between rock and pop genres, which is understandable, because the genres may overlap.

In classical music periods, we can observe how the music has developed through the centuries. This also corresponds with the theoretical knowledge, knowing that after Romanticism, the composers began to break the established harmony rules.

We have also discovered some interesting results concerning particular artists and songs. Queen songs were analyzed to be considerably more complex than the rest of the rock artists, which can be attributed to their famous ensembles sounding together in an unusual way. We have then plotted the ATC of rock songs by The Beatles and Queen on a chart by the year when they were recorded. This way we can visualize how artists change their harmonic complexity over time – naturally, a much higher number of songs would be needed for a proper analysis.

Certain songs have the ATC value that is below the area where most of the songs of the same genre belong to (highlighted in grey). In the particular case of *Radio Ga Ga* by *Queen*, we can conclude that the song is known to have a more pop song feeling than the rest of *Queen*'s production. There are indeed no choir ensembles or difficult chord changes, and this is why our algorithm calculated a lower complexity. The song *Let It Be* by *The Beatles* is known for its repetitive chorus containing 4 basic chords. On the other hand, the songs such as *Bohemian Rhapsody* and *Bicycle Race* are famous for their choir ensembles, and the song *Crazy Little Thing Called Love* has a slightly different style than the rest of the songs. This song is close to rock and roll genre, and we suspect that the fast pace of chord changes and the selection of chords may have contributed to the higher harmonic complexity. The song *Michelle* by *The Beatles* is known for its initial chord progression: *Fm/C–Caug–Fm7–Dm7b5/F–Bbm/F–C* (the reader can reference one of the available chord dictionaries online[8]). It is these types of progressions that trigger high *ChordCD* values.

3.6 Conclusion of the TSD distance model

In this chapter, we have proposed a new term that can be evaluated for a musical piece – the harmonic complexity. We have presented a computational model based on tonal harmony for evaluating harmonic complexity and compared it to the state-of-the-art musicology models. Lastly, we have successfully conducted a series of case studies to prove that the new feature is relevant to the music classification task. The results correspond with the

[8] `http://keychord.com`

expectations. Knowing that our model can evaluate the harmonic movements precisely we propose using this technique to obtain music descriptors for future music classification attempts, although that is out of the scope of this thesis focused on the CSI task.

In the discussion, we would also like to remind the idea of content-based music recommendation based on harmonic complexity. As Zanette [2008a] pointed out, the complexity and change in music together with the repetition of pleasant stimuli are two fundamental, but contradictory, principles that the listeners are looking for in music. One listener may prefer simple harmonies, while others might prefer more complex or more specific music (e.g., jazz or modern classical music). The model of harmonic complexity, therefore, presents a new way of understanding the need for listeners and can be used to recommend him/her music according to his/her needs. Harmonic complexity can also be used as a fitness function for evolutionary algorithms when we need to evaluate the population of generated songs.

We also remind the reader that harmonic complexity as described is only one of the possible types of complexities in music and other similar measures can be specified (space complexity, rhythm complexity, the complexity of modulations, etc.). We hope that by creating this model, we have induced some new ideas for future research.

4. Music harmony features comparison and CSI experiments

In this mostly experimental chapter, we use the standard MIR features and techniques (Chapters 1 and 2) along with our new features (Chapter 3) and newly proposed techniques to conduct multiple experimental studies. We have three main goals:

1. Experimentally study the performance of the new features proposed in Chapter 3 (ChordCD, ChromaCD, ATC).

2. Perform an extensive comparison of music harmony features, standard and new.

3. Perform a CSI experiment using the best features and scores coming from the previous analysis, with some of our own improvements.

4.1 Motivation and contributions

For the first goal (usage of ChordCD, ChromaCD, ATC) we have two motivations. The first is that no feature similar to harmonic complexity has been yet used for a MIR task to the best of our knowledge – perhaps with an exception of De Haas et al. [2012] and the errors in the derivation of the harmony tree (see Chapter 2). The second motivation is stemming from the fact that there are many harmony features, but there are not many examples of a successful harmony *fingerprint* of a song. Fingerprinting was made popular by applications such as Shazam[1] and algorithms such as the one presented and patented by Wang [2003]. Audio fingerprinting is one of the best examples where the MIR research found its way into the industry. However, the best attribute of audio fingerprinting is also one of its biggest drawbacks for the use in CSI: It finds the exact same music (albeit in a noisy environment), but when certain attributes of the music changes it may not find the match, as the fingerprints rely on the spectrogram peaks too heavily. Since harmony is considered to be one of the best characteristics to solve the CSI task, our motivation here is to form a useful harmony fingerprint of a song.

Although it may sound inferior to the other goals, the reader should make no mistake that the second goal – the feature comparison – was one of the most important for us during our whole research. With the vast amount of features available, the young researchers may consider it troublesome to choose the best features for their experiments. One of the contributions of this chapter is, therefore, to compare the features statistically and by the basic machine learning techniques (Section 4.4). We then continue to

[1] `https://shazam.com`

compare the features by probably the most popular method of MIR in the past: *dynamic time warping*. There are many studies benchmarking a few features, e.g., to compare a newly proposed MIR feature to the baseline features. However, it is very rare to find an actual study on a large dataset, that benchmarks 5 or more features. We try to fill this gap by providing a comparison of 6 and more relevant features in our experiments.

The third goal is wrapping it all together in a search for the best combination of the feature, the technique, and the similarity measure for a well-balanced CSI system. As we saw in Chapter 2, there is a great number of algorithms for the CSI task, and for almost 10 straight years, the algorithm of Serrà et al. [2009] retains the best performance on the MIREX benchmarking. The motivation for our CSI experiments is therefore made clear: If we do not improve the state-of-the-art techniques in terms of performance, we should try to offer new techniques and comparisons that could eventually change the thinking of the community, and result in further improvements.

In summary, the contributions of this chapter are:

1. Establishing the harmonic complexity in an experimental study.
2. Establishing a new harmony fingerprinting technique using a time series of chord distances.
3. A comparison of music harmony features useful for MIR researchers.
4. New results for the CSI by the DTW method: Selecting the best DTW scores, and showing which features have the biggest impact.
5. Performing an innovative experiment on our own karaoke dataset of $2,000$ songs with the best performance of 89.9% retrieval accuracy.

4.2 Machine learning experiments with harmonic complexity

Motivated by the promising case studies from Chapter 3, especially the discriminative power of the harmonic complexity for the genres, we have performed an experimental *music classification* study on a small dataset of popular songs. This experiment is supposed to be a supporting experiment to establish the harmonic complexity as a useful feature, as music classification is not the central theme of our work. But, intuitively, if a feature proves well for music classification of a genre, we can deem it a right candidate for the CSI task, too, as CSI is a classification of a song version, which is only a finer form of music classification than genre classification.

4.2.1 Music classification

In order to show that our new feature is efficient for music classification, we must first observe what are the common techniques used. There have been

a vast amount of proposals since music classification is one of the most popular MIR tasks. The most common are using hidden Markov models (Shao et al. [2004]), self-organizing maps (Rauber et al. [2002]), k-nearest neighbor (Tzanetakis and Cook [2002]), support vector machines (Xu et al. [2003]), or neural networks of different kinds (Soltau et al. [1998]). The state-of-the-art techniques include compressive sampling (Chang et al. [2010]), or low-rank semantic mapping (Panagakis and Kotropoulos [2013]). Generally, we may conclude that there are two major approaches: classifying music based on short excerpts or based on aggregating the values extracted from the whole piece. The popularity of music classification can also be documented by multiple surveys summarizing the state-of-the-art techniques, such as the one from Bertin-Mahieux et al. [2010], which we propose for further reference.

4.2.2 Neural network experiment setup

With the recent success of the machine learning and deep learning techniques (Krizhevsky et al. [2012]), we have taken an opportunity to conduct an experiment using neural networks, although neural networks are not the main topic of the thesis. Another motivation for this method was that we could exchange the set of features on the same dataset of songs and re-train the network multiple times. That is an effective way of seeing if a particular feature brings results for a given task. We were interested in comparing the performance of a neural network in two scenarios:

1. Each musical piece is represented by a basic set of features.
2. Each musical piece is represented by the same set of features with an added ATC feature (see Section 3.2.2).

4.2.3 Basic set of features

We have picked features commonly used for music classification tasks. The basic set of features needs to be strong enough for a satisfying classification, so that we may observe the added value of the harmonic complexity.

1. **Mel Frequency Cepstral Coefficients (MFCC) mean value and covariation matrix.** MFCC values are commonly used for music classification and speech recognition (Logan [2000]).
2. **Root Mean Square (RMS) Amplitude.** RMS Amplitude represents the volume of the piece.
3. **Tempo in Beats per Minute (BPM).** The values were extracted by GPL-Licensed Queen Mary Vamp plugins set[2].
4. **Chord transition probabilities.** The matrix of probabilities of transitions between the pairs of chords. The chords in the matrix were simplified and represented only by the root tone of the chord.

[2] `https://www.vamp-plugins.org`

5. **Number of similarity segments.** The number of segments that are similar in the piece, derived from the self-similarity matrix, extracted by Queen Mary Vamp plugins set.

6. **Other harmonic measures.** We completed the feature vector by meta-features describing the harmonic content: number of distinct keys, number of modulations, and number of distinct chord roots, acquired by Queen Mary Vamp plugins set.

4.2.4 ATC feature extraction

The whole process of obtaining the ATC values starts with the low-level feature extraction. For that purpose, another Vamp plugins, NNLS Chroma and Chordino[3] in version 0.2.1, were used. Developed by Mauch and Levy [2011], using the method of non-negative least squares, the NNLS Chroma plugin provided us with chroma vectors, i.e., the presence of frequencies mapped into semitone-spaced 12-dimensional vectors. Chordino plugin was used to locate the points of time when the harmony changes significantly, segmenting the piece into small parts. We then chose the four tones with the highest presence in the segment to represent the chord. Finally, a model of harmonic complexity described in Chapter 3 was used to obtain the ATC values.

4.2.5 Data set and choosing the genres

We selected five genres from modern music: *electronic, jazz, metal, rock*, and *pop*. It is evident that these genres are difficult to recognize and that they may overlap. As a dataset, we have used 100 songs (20 for each genre), featured in the 2013 and 2014 music charts[4].

4.2.6 Network parameters and validation

We used 70 songs represented as normalized feature vectors for the training of the neural network. 15 songs were used for validation of the training, and 10 songs for testing. The network contained 110 input neurons for the basic feature set and 111 input neurons for the feature set with ATC. In the hidden layer, there were 40 hidden neurons to achieve multi-layered learning. Finally, five output neurons were used to classify the music – each representing one genre. The neuron with the highest activation determined the genre.

To validate our results, we have used cross-validation with 10 rounds, in each round new sets of songs were used for training, training validation, and testing.

4.2.7 Results

The resulting overall confusion matrices for all tests are shown in Figure 4.1. If we consider the classification as a retrieval system, we can observe the

[3] `http://isophonics.net/nnls-chroma/`

[4] `https:/www.billboard.com`

1: Basic features

CLASSIFIED GENRE \ REAL GENRE	E	J	M	R	P	PRECISION
Electronic	**15**	1	1	4	2	0.65
Jazz	0	**15**	0	2	2	0.79
Metal	1	0	**13**	0	0	0.93
Rock	2	3	0	**12**	1	0.67
Pop	2	1	6	2	**15**	0.58
RECALL	0.75	0.75	0.65	0.60	0.75	OVERALL: 0.70

2: with ATC

CLASSIFIED GENRE \ REAL GENRE	E	J	M	R	P	PRECISION
Electronic	**13**	1	0	1	4	0.68
Jazz	2	**17**	0	0	0	0.89
Metal	1	0	**17**	0	1	0.89
Rock	2	2	0	**16**	4	0.67
Pop	2	0	3	3	**11**	0.59
RECALL	0.65	0.85	0.85	0.80	0.55	OVERALL: 0.74

Figure 4.1: Results of music classification task using harmonic complexity:
1: Scenario with basic set of features.
2: Scenario with added harmonic complexity (ATC value).

precision and *recall* values. On the example of jazz music, the precision is the number of real jazz pieces classified as jazz, divided by all pieces classified as jazz. The recall is the number of real jazz pieces classified as jazz, divided by all jazz pieces.

We can see that the use of harmonic complexity yields to up to 4% improvement in the overall precision (74% against 70%). While observing the successive rounds of cross-validation, we noticed that, when ATC was used, the overall precision fell below 70% only in one round, as opposed to 4 rounds when the basic features were used.

After a deeper exploration of the results, we found an interesting paradox concerning the use of ATC. Even though the overall precision was higher with ATC, its classification was less able to distinguish between electronic and jazz music. Knowing that jazz music usually achieves the highest complexity values, we attribute this to the fact, that electronic music contains segments in which the harmony is not clear (noises, raising or lowering pitches), and therefore obtains higher values too.

We were mostly interested in finding out which genres are better classified when ATC is used. Except for the metal genre, all four other genres were better classified with ATC. The best result, up to 10% improvement, we observed in jazz music, which yields to the expectations, since jazz music usually contains complex harmonic movements.

4.2.8 Conclusion of the neural network experiment

We have proposed a new feature, harmonic complexity, as a useful descriptor for the music classification task. First, we have provided a short survey on the related works, none of which uses harmonic complexity as one of the features. In the supporting experiments, we have then shown, that adding the ATC values indeed enhances the music classification using the neural network method. For future work, experiments on the larger data sets need to be performed. A comparative study of all possible harmonic features would be also be needed to help the researchers choose the right ones for the given task. That is the focus of the following section.

4.3 Comparison of harmony fingerprints using dynamic time warping

This section puts the research from the previous chapters finally to the context of the CSI task. As we mentioned earlier, we propose to use a harmonic complexity for retrieval in the form of a fingerprint (a time series of ChordCD, see Section 3.5). A fingerprint should carry over the distinct harmony signature of the song. It should be small enough so that the fingerprints can be found quickly in the database. Both of these requirements are achieved by a ChordCD time series.

However, what we are looking for is not a standard fingerprinting scenario (detection of a song by its fingerprint). With CSI, we are always working with two or more different songs sharing the same title or underlying composition. We wish for each such cover „cluster“(versions of the same song) to have a very similar fingerprint, but at the same time, we wish for this fingerprint to be different from fingerprints of the other cover clusters. That is a demanding requirement, which we put to the test by the industry standard dynamic time warping method. As a bonus, we also test up to five different dynamic time warping scores in the same set of experiments.

4.3.1 Dynamic Time Warping

Dynamic Time Warping (Müller [2007]) (DTW) is a method used for determining an optimal alignment between two time series. Based on this, an alignment score (distance) is calculated. The main advantages of DTW are the ability to compare sequences which differ in length and its generality. When introduced, DTW was used mainly for speech recognition, but thanks to the benefits mentioned, it quickly spread into other areas such as robotics, medicine, video games, music processing, and many more. Importantly for our work, DTW is a straightforward technique to identify a cover song played in a different tempo.

Principles of DTW

Let us consider two time series S and T. The basic idea behind DTW is to compare every data point from time series S with every data point from time

series T. For this purpose of point-to-point comparison, usually Euclidean or Manhattan distances are used, but any other distance can be employed. Consequently, the distance matrix is built in which a minimal warping path is found.

Definition. *A* warping path *in the distance matrix M is a sequence $W = (w_1, ..., w_l)$, where $w_l = M(i, j)$ and $l \in [1, |W|]$ that satisfies the following conditions:*

1. *Condition of boundaries: $w_1 = (1, 1)$ and $w_l = (|S|, |T|)$*
2. *Condition of a step size: $w_{l+1} - w_l \in \{(1, 0), (0, 1), (1, 1)\}$ for $l \in [1, |W| - 1]$*

In the end, DTW distance is calculated using Equation (4.1) or (4.2) (as described by Mueen and Keogh [2016])

$$DTW(S, T) = \min \left\{ \sqrt{\sum_{l=1}^{L} w_l} \right. \tag{4.1}$$

$$DTW(S, T) = \min \left\{ \frac{1}{L} \sqrt{\sum_{l=1}^{L} w_l} \right. \tag{4.2}$$

where L is the length of warping path W. Equation (4.2) for calculating the score is divided by the length of the warping path to account for the situations when compared sequences have different lengths. *Identity* occurs when $DTW = 0$.

Similarity in DTW

The standard equations (4.1) and (4.2) unfortunately share one main problem when we use them for calculating DTW score. The score is not normalized in the interval $[0, 1]$, which means that it is difficult to compare DTW results across different datasets. For that reason, we are testing three equations for calculating similarity by using the information from building the distance matrix and the warping path.

$$DTW_{sim}(S, T) = \frac{|S| + |T|}{|S| + l + |T| + u} \tag{4.3}$$

The first similarity score (4.3) (Vlachos et al. [2006]) which we use is simply obtained by dividing the original length of the sequences by the length of the sequences containing the edit information. Compared to other methods that deal with the alignment of sequences, for example, the Needleman-Wunsch algorithm (used for alignment of DNA sequences), DTW edits do not prolong the sequences in the resulting alignment. Edits in the DTW method occur in the situation when a continuous sequence of elements from one time series is collapsing into one point of another time series and vice versa (see Figure 4.2). In Equation (4.3) l and u are the values of horizontal and vertical movements in the warping path, where every l or u movement means the insertion of edit information into the sequence.

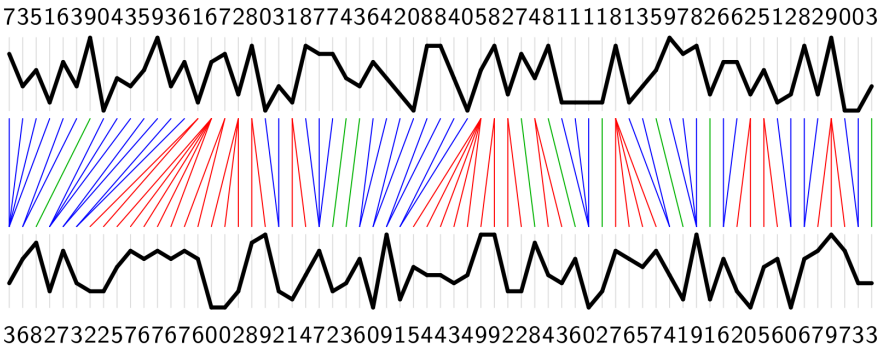


Figure 4.2: Example of DTW alignment between two time series. Directions of collapsing points are depicted by blue and red colors. The green lines connect elements mapped in one to one relationship.

The score from equation (4.3) is within the interval $[0, 1]$, and *identity* occurs when $DTW_{sim} = 1$. One disadvantage of this score is that when $DTW_{sim} = 1$ it does not necessarily mean that the compared sequences are identical. It only means that the compared time series are aligned to each other by one point to one point relationship. In the classic DTW, this score would never be 1 if the lengths of the compared sequences are different.

The second similarity score that we test is the following equation (Kocyan [2015]):

$$DTW_{sim}(S,T) = 1 - \frac{\sqrt{DTW(S,T)}}{\sqrt{DTW_{max}(S,T)}} \tag{4.4}$$

We calculate the similarity by dividing the score from $DTW(S,T)$ by maximal value DTW_{max}, which can theoretically occur. Let us consider time series $S \in O^{|S|}$ and $T \in O^{|T|}$, where O is the domain of the analyzed sequences. Maximal DTW score can be obtained by calculating DTW for sequences S', T', where $S' \in \{s'_1, ..., s'_i\}, \forall n \in \{1, ..., i\}(s'_n = min(O))$ and $T' \in \{t'_1, ..., t'_j\}, \forall n \in \{1, ..., j\}(t'_n = max(O))$. The result is subtracted from 1 so that *identity* occurs when $DTW_{max} = 1$ and difference when $DTW_{max} = 0$.

Our third similarity score for the comparison (4.5) is a slightly adjusted version of Equation (4.4) (Kocyan [2015]).

$$DTW_{sim}(S,T) = \left(1 - \frac{\sqrt{DTW(S,T)}}{\sqrt{DTW_{max}(S,T)}}\right) \cdot \frac{min(|S|,|T|)}{max(|S|,|T|)} \tag{4.5}$$

Equation (4.4) is multiplied by a coefficient calculated from differences in lengths of the compared sequences. By this, we penalize the comparison of time series with different lengths.

Obviously, it would be extremely inefficient to calculate DTW two times, one for standard DTW and one for DTW_{max}. Fortunately, the calculation

of DTW_{max} can be simplified as seen in Equation (4.6).

$$DTW_{max}(S,T) = \sqrt{(max(O) - min(O))^2 \cdot max(|S|,|T|)} \tag{4.6}$$

4.3.2 Experiments

The three features that we are testing are:

- ChordCD time series fingerprint based on the definition from Section 3.2.1 put in a timestamp–value format
- ChromaCD time series fingerprint based on the definition from Section 3.3 put in a timestamp–value format (although the timestamps are set by the frame rate of the chroma vectors).
- As a baseline feature we choose 12-dimensional chroma features by NNLS Chroma Vamp Plugin (Mauch and Dixon [2010]). Note that we choose a baseline *feature* and not a baseline *fingerprint*, simply because we do not have knowledge of a similar harmony fingerprinting, and we wanted to see how good or bad our fingerprints perform compared to a full-sized feature.

Remark. *To highlight the fact that the chroma features were not processed further (as opposed to post-processing required by ChromaCD), we denote the vectors as „raw chroma vectors“ in our experiments. We also denote ChordCD time series fingerprint and ChromaCD time series fingerprint simply as ChordCD and ChromaCD to save space in the tables.*

For feature extraction, including the chord segmentation, we have used a combination of Vamp Plugins[5] (NNLS Chroma, Chordino, and Key Detector). We have further processed the resulting features with our *harmony-analyser* software[6] to obtain the chord and chroma vector distances.

We have performed a series of experiments on two commonly used music datasets: covers80 (Ellis and Cotton [2007]) and SecondHandSongs (Bertin-Mahieux et al. [2011]). All experiments were run on the Anselm[7] supercomputer in IT4Innovations National Supercomputing Center, where we used one computational node (Two Intel Sandy Bridge E5-2665 processors, each having 8 cores, 2.4GHz and 32GB RAM).

The covers80 dataset consists of 160 songs organized as 80 musical works, each in two versions. The first versions of the songs are used as queries for the search in the whole set of the second versions of the songs.

SecondHandSongs is a set of 18,196 tracks with 5,854 cover song clusters (average cluster size is 3.11). For our comparison, we have taken a chunk of 999 songs (295 clusters) from SecondHandSongs train set - the first 999 songs listed in the official dataset information file.

[5] `https://www.vamp-plugins.org`
[6] `http://harmony-analyser.org`
[7] `https://docs.it4i.cz/anselm/hardware-overview`

4.3.3 Comparison of features and DTW scores

We aim to output results for each feature and DTW score combination, on both datasets. A standard evaluation metric for MIREX is the use of the Mean arithmetic of Average Precision (MAP). MAP score takes into account the ranking of each cover song and assigns the weights according to the rank. The results are within the interval $[0, 1]$; higher values mean that the cover songs ranked high and close together. For datasets of thousands of songs and small cover song clusters, MAP values are likely to be ~ 0.1 (as seen, e.g., in the work of Bertin-Mahieux and Ellis [2012]).

We can see the comparison in Table 4.1 and Table 4.2. Raw chroma vectors have outperformed chord and chroma vector distances. We attribute this to the fact that 12-dimensional chroma vectors contain much more information than the one-dimensional fingerprints. The information is reduced even more from frame-specific ChromaCD to ChordCD evaluated only for every chord transition. The execution time to obtain similarity matrices was: $\simeq 56s$ for raw chroma vectors, $\simeq 51s$ for ChromaCD and $25ms$ for ChordCD time series for covers80 dataset. We can see marginally better performance for ChromaCD. However, the performance difference is reduced on the larger dataset: on SecondHandSongs dataset the execution time is $\simeq 550s$ for raw chroma vectors and $\simeq 100s$ for ChordCD time series.

Comparison of all results over all data formats shows, that the best results are provided by DTW scores (4.1), (4.3) and (4.4) in this order. Scores (4.2) and (4.5) provide very poor results and we deem them as not suitable for the CSI task.

Score	Raw chroma vectors	ChromaCD	ChordCD
score (4.1)	**0.482**	0.094	0.142
score (4.2)	0.103	0.070	0.071
score (4.3)	0.417	0.174	0.156
score (4.4)	0.454	0.061	0.114
score (4.5)	0.082	0.041	0.034

Table 4.1: MAP results for each DTW score and feature for covers80 dataset

Score	Raw chroma vectors	ChromaCD	ChordCD
score (4.1)	**0.107**	0.031	0.019
score (4.2)	0.021	0.014	0.014
score (4.3)	0.029	0.035	0.021
score (4.4)	0.043	0.015	0.012
score (4.5)	0.008	0.008	0.009

Table 4.2: MAP results for each DTW score and feature for SecondHandSongs dataset

4.3.4 Comparison to the state-of-the-art

Besides MAP results we have examined Mean Average Rank (MAR) of the covers and the number of correctly identified covers (for covers80 dataset), to be able to compare our results with the state-of-the-art methods. MAR values are between $[1, N]$ where N is the number of songs, and we aim to achieve the lowest value possible.

An overview of MAR results can be seen in Table 4.3 and Table 4.4. For the SecondHandSongs dataset, the MAR metrics were evaluated before us by Bertin-Mahieux and Ellis [2012]. They achieved average rank of $2,939$ for over $12,960$ songs (Bertin-Mahieux and Ellis [2012]), and $308,369$ for 1 million songs (Bertin-Mahieux and Ellis [2011]). In our experiments, Raw chroma vectors have the average rank of 321 out of 999 songs, closely followed by ChromaCD (average rank 341). That is a worse result than the state-of-the-art, but a promising result for ChromaCD. For the covers80 dataset, the results were similar, raw chroma vectors outperforming the simple distances (best average rank 14 out of 80 songs). The number of correctly identified covers was 33, which was in fact achieved by the score (4.1) with a slightly worse average rank. This is similar to the first LabROSA system results (Ellis and Poliner [2007]). On the other hand, ChromaCD achieved only 6 identified covers, despite the promising average rank. This shows that ChromaCD feature keeps the relevant tonal information, but is not self-contained (too much data is reduced by the extraction, and the differences between the songs are getting lost).

We have also gathered the first experimental results with TPS chord distance (see Chapter 3) for CSI task. For this experiment, we used the symmetrical TPSD (Tonal Pitch Step Distance) proposed by De Haas et al. [2008]. We have extracted a time series fingerprint, exactly as the one with ChordCD, but we have used TPSD instead. TPSD has outperformed the more simple ChordCD, with the best result 0.198 MAP and 27.575 MAR for covers80 dataset. Note that these results are better in MAP than ChromaCD. These first experiments are promising for a further study of TPS for the CSI task.

Score	Raw chroma vectors	ChromaCD	ChordCD
score (4.1)	15.688	30.450	32.825
score (4.2)	29.350	35.500	37.663
score (4.3)	15.025	21.538	26.688
score (4.4)	**13.963**	36.587	35.688
score (4.5)	36.013	46.038	45.362

Table 4.3: Comparison of mean average rank score on covers80 dataset

Score	Raw chroma vectors	ChromaCD	ChordCD
score (4.1)	**321.033**	382.686	402.829
score (4.2)	403.961	438.722	444.141
score (4.3)	362.050	341.092	407.792
score (4.4)	374.675	414.561	426.181
score (4.5)	547.546	578.601	561.569

Table 4.4: Comparison of mean average rank score on SecondHandSongs dataset

4.3.5 Conclusion of the experiments with fingerprints

The baseline chroma features are 12-dimensional vectors with much more music information, and thus, there is no wonder that they perform better than our fingerprints. To showcase the possible use of our fingerprints, we again highlight the computation time: 56s for chroma vectors vs. 25ms for fingerprints (cover80) and 9 minutes vs. 1.5 minute (SecondHandSongs). The fingerprints achieve an accuracy where we can expect 20 – 40% of the cover songs retrieved for a medium-sized dataset (as opposed to a random algorithm where the % of songs retrieved would be close to 0. Moreover, as we saw in tables 4.3 and 4.4, the fingerprints tend to rank the covers in the upper half of the result list. As a result, we propose such fingerprints to form the first layer of database pruning techniques as described by Osmalskyj et al. [2013].

4.4 Comparison of features using statistics and machine learning

We now change our toolbox and use a statistical approach for our next experiment. One of the questions that come naturally to the researchers before they employ a new dataset is, whether the dataset is a good fit for their task, and which features to use. While we cannot answer all of these questions in advance, we decided to give a comprehensive analysis of features using some common techniques (correlation, distribution and filter methods) with regards to the CSI task.

There are three types of features based on their complexity: Single, Timestamp, and Frame. This bears meaning for CSI, as the recent studies have been focusing on using the Single features for the early database pruning techniques and the Frame features for the final cover song identification (Osmalskyj et al. [2013]).

The Single and Frame features can be easily compared and visualized because they are represented as scalars and vectors, which is not the case for the Timestamp features. Indeed, a Timestamp feature contains a structured data where a timestamp is associated with an event (such as a note onset or a chord change). The comparison of Timestamp features is, there-

fore, more challenging. We can convert the Timestamp feature to a Frame feature by assigning the time-bounded value to every successive frame until a new timestamp is found. For example, all frames contained between two chords are given the tag of the first chord. This technique allows comparison of a Timestamp feature to a Frame feature, and is used in Section 4.5.2. Other ways to approach the problem are to consider removing the time information, treating each timestamp-value pair as a Single feature, or using aggregate functions to convert a Timestamp feature into a Single feature (e.g., creating a histogram). These techniques allow us to create more comparisons between the Timestamp features and also allow us to compare an aggregated Timestamp feature to a Single feature, as we show in the next section.

4.4.1 Analysis of Single features

To gain a basic insight into the performances of the individual Single features, we visualize their performances by a set of experiments. We start with visualizing the distribution of the cover pairs and the non-cover pairs, with respect to each feature.

The dataset that provides the features for the analysis is our *Kara1k* dataset (the reader can find more about this contribution in Chapter 6).

Distribution of cover pairs and non-cover pairs per feature

One way to look at the CSI task is to pair up all songs into 2-tuples and observe whether the pair is a cover pair or a non-cover pair. CSI can then be considered a binary classification problem with a ground-truth, where we output 1 if the pair is a cover pair, and -1 if the pair is a non-cover pair. In *Kara1k*, there is by design 1,000 cover pairs with ground-truth 1 and $\binom{2,000}{2} - 1,000 = 1,998,000$ non-cover pairs with ground-truth -1. The amount on non-cover pairs is very high, as for each origin song there is only 1 song to form a cover pair, but 999 songs to form a non-cover pair. For the purpose of our following comparison, we need a $1:1$ ratio between the number of cover pairs and non-cover pairs to achieve the best visualization. So we use $1,000$ cover pairs and compare them to $1,000$ non-cover pairs selected from all the combinations (each of the $1,000$ origin songs was assigned a unique, non-matching song from the $1,000$ cover songs, using a random selection).

Figure 4.3 shows the visualizations of the distributions for selected features. In the graphs, we show the number of cover pairs in blue and the number of non-cover pairs in red according to the similarity of the two songs using a given feature. The similarity can be viewed as a function $sim(f(S_1), f(S_2))$ where S_1, S_2 is the pair of songs compared by the feature f. The choice of the similarity function *sim* often yields some basic understanding of the feature, so we show the choice underneath each of the graphs (e.g., for chord histogram, the cosine distance is appropriate).

Different levels of suitability can be observed for each of the features. If a feature is suitable for CSI, the distribution of cover song pairs in blue is distinct from the distribution of non-cover pairs in red. The perfect separation between the two distributions is not possible, but the goal is to select

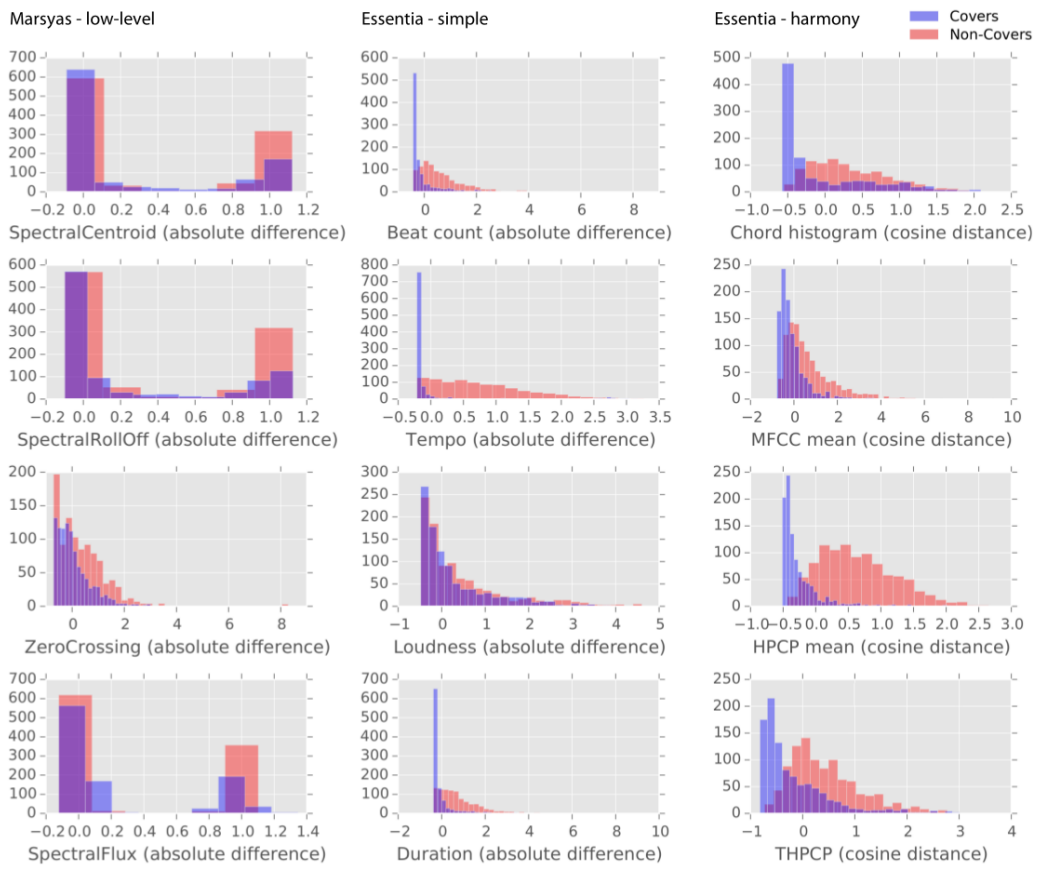


Figure 4.3: Distribution of cover pairs (blue) and non-cover pairs (red) with respect to individual features. A portion of the *Kara1k* dataset including 1,000 cover pairs and 1,000 non-cover pairs was used, and the representative Single features from each of the feature extractors are shown. Some features shown (Vamp histograms) are derived from the Timestamp-based features.

features that minimize the intersection. According to Figure 4.3, the pitch histogram is one of the best features to use for CSI, whereas the beat count is one of the worst. These visualizations can be a useful alternative to the principal component analysis, e.g., when we are about to perform a machine learning experiment.

Filter methods

To further help the researchers with their choice of individual features for their machine learning experiments, five filter methods for feature selection are compared: F-statistic of the ANOVA, Mutual Information (Yu and Liu [2003]), ReliefF (Kira and Rendell [1992], Kononenko [1994]), SURF Greene et al. [2010] and MultiSURF Granizo-Mackenzie and Moore [2013].

The filter methods are used as a preprocessing step before the classifier training phase and therefore are unaffected by the bias of the classifier. The process for feature selection by filter methods is divided into two steps.

1. Rank the features using a chosen scoring function.

2. Select k best features using a specified threshold.

The scores awarded to each feature by the five filter methods are summarized in Figure 4.4. The features with the highest positive score are considered the best for the CSI task. It should be noted that while the other methods have a minimum at 0, the ReliefF, SURF, and MultiSURF generally can be negative. We experienced the highest scores for CSI with the harmony-based features. The pitch histogram was reaffirmed as a suitable feature for CSI, but Vamp Plugins resulted in a very similar score with the chord histogram. The figure also shows the differences between the particular filter methods, which sometimes disagree quite significantly in their evaluation of the feature, but this can be attributed to the different definitions (Kononenko [1994]).

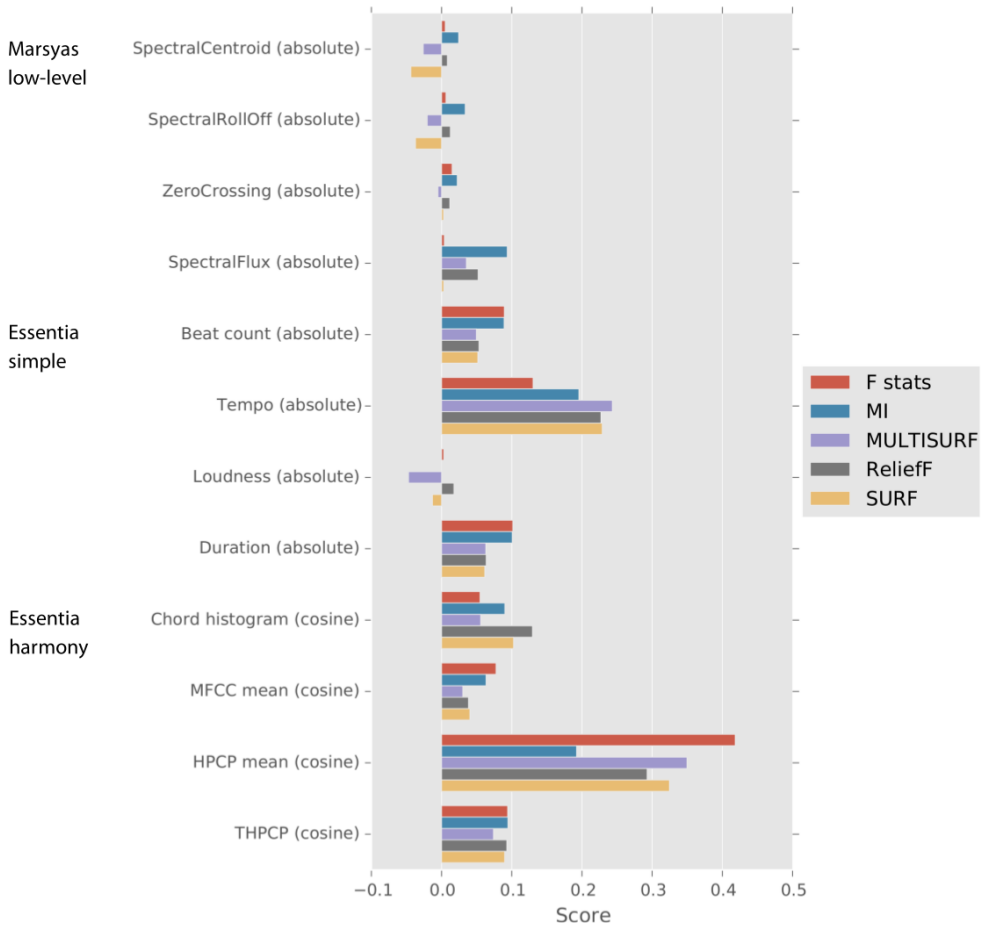


Figure 4.4: The scores for the selected *Kara1k* Single features, obtained by the filter methods: F-statistic, Mutual information, ReliefF, SURF and Multi-SURF. A portion of the *Kara1k* dataset including $1,000$ cover pairs and $1,000$ non-cover pairs was used. Some features shown (Vamp histograms) are derived from the Timestamp-based features.

Correlation of features

While results obtained in the previous analysis are useful as a preliminary analysis of the features, we need to remember that the filter methods evaluate each feature separately. Thus, the interactions between the features are not taken into account. If the features are correlated and used together, the CSI results may be distorted for some machine learning algorithms. As we can see from the correlation matrix in Figure 4.5, there are indeed many implicit relations between the features.

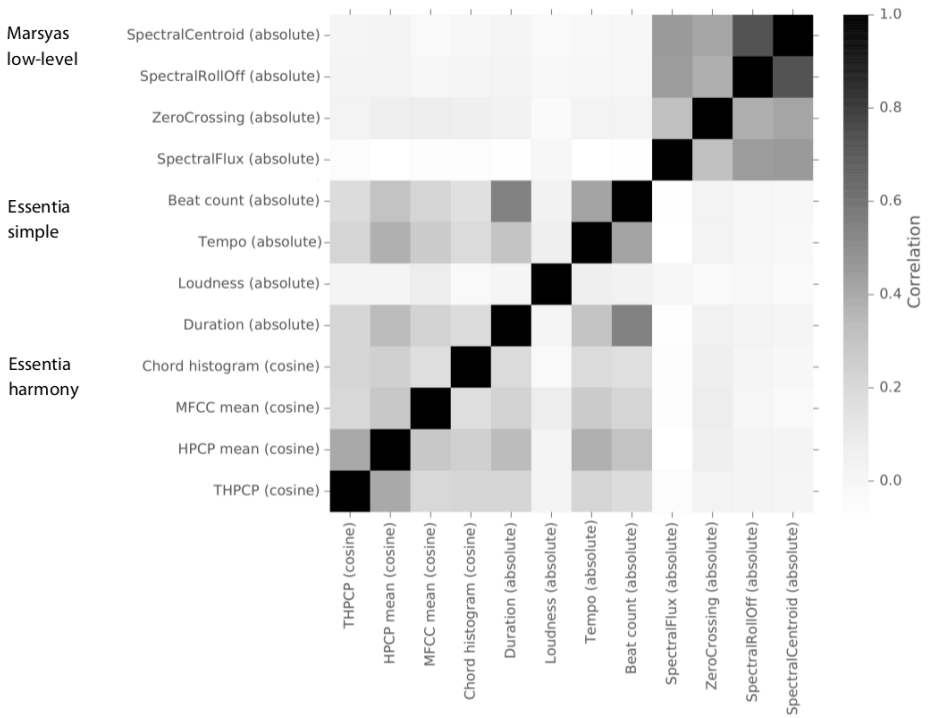


Figure 4.5: Correlation of the selection of the Single features from the *Kara1k* dataset. Some features shown (Vamp histograms) are derived from the Timestamp-based features.

Some of the expected correlations are shown. This is the case with the chord histogram extracted by Vamp and the one extracted by Essentia. There is also a high correlation between the beat count and the duration. However, the figure also reveals relations that may not have been apparent, such as the correlation between the chord and the pitch histogram. Overall, the harmony features based on the tones show some amount of mutual correlation.

4.5 Final dynamic time warping experiments

We can now perform a final set of experiments for the CSI task. With the knowledge of the results from previous sections, our goal is to learn from the lessons where the accuracy was not high enough, for both features and score

definitions. For our final experiment, our algorithm of choice is again DTW, as this method is highly justified when comparing original songs to cover songs since the cover version can differ in tempo. The Euclidean distance is used to compare the vectors of the two time series, and the corresponding warping path is created. Afterward, we chose the best-performing DTW scores described in Section 4.3.3 to obtain the similarity matrix. We detail the selection of features in Section 4.5.1, and the experiment details and results in Section 4.5.2.

4.5.1 Feature extraction for dynamic time warping experiment

Our previous experiments have shown that full-sized features are better than any fingerprints. That is why, for our final CSI experiments, we choose a selection of the following features from the *Kara1k* dataset. We mainly use full-sized features, but we still add a few fingerprints to reinforce their experimental results, although we do not expect them to perform well compared to the full-sized features. The selected features are:

- 12-dimensional chroma vectors extracted by the *NNLS Chroma Vamp plugin* (Mauch and Dixon [2010]). We prefer these sophisticated chroma features over the standard HPCP, as they share the same advantages as HPCP such as robustness against tuning differences, etc. (see Section 1.2.4), and are easily pluggable into our *harmony-analyser* software.
- 13 MFCC after the 0^{th} extracted by *YAAFE* (Mathieu et al. [2010]).
- 12-dimensional chord vectors extracted by the *Chordino Vamp plugin* (Mauch and Dixon [2010]) and post-processed by *harmony-analyser* (Maršík [2016]).
- 12-dimensional key vectors extracted by the *Key Detector* from the Vamp plugins (Noland and Sandler [2007]), further post-processed by *harmony-analyser* (Maršík [2016]).
- Tonal Pitch Step Distance (Lerdahl [2001], De Haas et al. [2008]) time series fingerprint from every pair of consecutive chords, extracted by *harmony-analyser* (Maršík [2016]).
- ChromaCD time series fingerprint, which is a series of differences of consecutive chroma vectors, extracted by *harmony-analyser* (Maršík [2016]). This feature was proposed as a simplification of chroma vectors for large-scale approaches in Chapter 3.

While chroma features are proven effective for CSI (Serrà et al. [2008, 2009]), MFCC features are considered useful for other tasks, and only a few studies have tested their impact on CSI. A good selection of an algorithm can result in a retrieval accuracy competitive to chroma-based algorithms, as was pointed out by Tralie and Bendich [2015].

As for this particular experiment we use karaoke songs that share the harmony of the instrumental background with the original songs (reference Chapter 6 for details), we propose to use the chord and the key features. To allow the comparison of chords and keys via DTW, we use chords and keys in a novel way. Not as string labels, as was done, e.g., by Khadkevich and Omologo [2013], but as binary vectors. The vectors contain the value *True* if the chord/key tone is present and the value *False* if it is absent.

We again choose to test the TPSD and ChromaCD time series fingerprints, first studied in Section 4.3. These fingerprints represent a characteristic curvature of a song as proposed in Chapter 3. We mainly choose these two fingerprints because of their quite similar score in the previous experiment from Section 4.3. It is also another challenge in between chroma-based and chord-based fingerprints, which of them have more discriminative power for cover versions.

We unify all features to be present at every frame, effectively spanning the vector (or distance value) over all of the frames until the next value occurs, as was mentioned in Section 4.4. As an example, a Timestamp feature *Chordino labels* from *Vamp Plugins* was converted to a Frame feature (post-processed by *harmony-analyser*). Thus, all features that we use in our experiments are in the form of a time series of scalars or vectors, which is vital for our DTW method described in the following section.

4.5.2 Retrieving karaoke version using a DTW method

As we now work with a Kara1k dataset (described in details in Chapter 6), we re-define our CSI task as: *For a given query song, the algorithm has to find the corresponding karaoke version out of* $1,000$ *songs.* To do so, we took the six frame-scale features discussed in Section 4.5.1, for each query song. We compared them with the features from the database of karaoke songs using the DTW method, and we built a similarity matrix containing each song combination. Each matrix is thus composed of $1,000$ query (original) songs and 1,000 karaoke songs. Karaoke songs are then sorted by their similarities to the query song. The first song in this list is then considered to be the detected cover (karaoke) version.

We use three metrics to compare the features. The MAP takes into account the whole sorted list of cover songs and assigns a weight to the ground-truth song according to its corresponding rank. As was described earlier in Section 4.3.3, MAP score can range between 0.0 and 1.0 and the higher the MAP, the better the algorithm. We again use also the Mean Average Rank (MAR) that averages the rank of the ground-truth song for all the queries. The MAR can range from 1.0 to the number of songs in the musical dataset and the lower the MAR, the better the algorithm. The third metric is the percentage of correctly detected karaoke songs out of the $1,000$ queries.

Table 4.5 compares the MAP, the MAR and the percentage of correctly detected cover songs for all features. The rows indicate the distinct features, and the columns show the three metrics for evaluation. The best MAP and accuracy is achieved by the chroma features. The MFCC and chord fea-

Table 4.5: The final results for the CSI task on the *Kara1k* dataset

	MAP	MAR	% detected
Chroma	**0.899**	48.112	**88.9%**
MFCC	0.878	**25.161**	86.3%
Chord	0.865	43.943	84.4%
ChromaCD	0.442	70.994	36.4%
TPS	0.257	141.941	19.3%
Key	0.203	109.954	11.0%

tures detect a similar amount of karaoke songs as the chroma features, while MFCC achieves better MAR than chroma features. This is an interesting result, caused by only a few songs being considered too distant by the chroma features, while MFCC treated them more close to each other. The result is captivating also for chord features, which, compared to chroma features, represent reduced information from float vectors to binary vectors restricted to a chord dictionary. We specify that since the karaoke songs are typically produced in the same key as the original songs, we did not need to shift the series of 12-dimensional vectors to all possible rotations before processing the comparison. Indeed, we did not achieve better results when we considered these rotations.

The results obtained with the use of the ChromaCD, TPS distance, and the key features show lower accuracy compared to the chroma and chord features. This became apparent already in Section 4.3.3 and we verified that it stays true also for the karaoke cover song identification.

4.5.3 Discussion and conclusions from our final CSI experiment

The main difference between the original recording and the karaoke version of one given song comes from the singing style and the played instruments. These differences are the reason why the chroma and the chord features display good results in our experiments. In fact, all of the 11.1% pairs of songs misclassified by the chroma features were a subject of our further testing to understand the differences in their features. It seems that a few of the karaoke songs are indeed very different from the original, in terms of tuning, key or the overall composition.

More importantly, our first result on the karaoke dataset helps us to form ideas toward a more difficult task, a cover song identification in an arbitrary musical dataset. For example, if the DTW comparison of two songs using a quite simple feature such as the chord vectors results in a good score, we can declare the two songs a cover pair, even without extracting other features. If not, we can proceed with more sophisticated features and methods. Our results show that chords are a well-suited feature for this initial check, while, e.g., keys are not. With the help of database pruning techniques (Osmalskyj et al. [2013]), this approach can be tested for the

large-scale cover song identification.

The reader can also note that the fingerprint ChromaCD displays an accuracy of 36.4%, which is not a bad result for a 12-times smaller feature. This goes in line with the conclusion on our previous experiment in Section 4.3.5 where we propose this feature for the first layer of a multi-layer retrieval system. On the other hand, the TPS feature has performed worse than ChromaCD, which reinforces our effort to try the complexity analysis for the chroma vectors. The harmonic complexity occurring on the frame level brings twice as much accuracy than if only evaluated for the chord change.

For the discussion, we propose to examine whether some particular audio fingerprinting algorithms may have good results for matching karaoke and original songs. The audio fingerprinting task is generally considered less challenging than CSI (Tralie and Bendich [2015]) and may introduce problems if the song is played in other tempo or ornaments. We thus suggest more experiments to explore the audio fingerprinting results on similar, but not the same, recordings, for example on karaoke songs from *Kara1k* dataset.

5. harmony-analyser application

We finally take time to present the application that we used in each of our experiments and harmony analysis. The *harmony-analyser.org* project has been launched in 2016, is now published to the open-source community and is regularly drawing attention and feedback. In the following sections, we discuss both the technical aspects of the project as well as the features from the user perspective.

5.1 Project introduction

We present a new Java library and tools for analysis of music harmony focused on chords, chord progressions, and chroma vectors. The underlying model is capable of creating, naming and analyzing chords, as well as evaluating chord distances including Tonal Pitch Space, geometric distances on the Tonnetz grid, or chord complexities. Special attention is given to the experimental distances between chroma vectors. Our system can take input in the form of text, MIDI input device, or the WAV file format. We provide tools for chordal analysis and for creating visual representations - chord segmentation or line graph visualizations of the chord or chroma distances. The system is extensible by creating additional plugins and provides an easy way of incorporating C++ based Vamp plugins, thus forming a suitable framework for music analysis in Java. Our *JHarmonyAnalyser* library and tools under harmony-analyser.org can be used for music analysis or feature extraction for retrieval tasks.

5.2 Motivation for the application

Automatic chord estimation has been a major task in MIR in the 21st century, ever since Fujishima [1999] has first compared the chroma vectors to a chord dictionary and obtained chord progressions in 1999. Since then, the effort has been put not only towards a great variety of techniques for chord estimation (McVicar et al. [2014]) but also towards improving recent MIR tasks by analyzing chroma or chord sequences. As we pointed out in Chapter 2, notable examples in the CSI task were the use of the reduced chroma information, such as averaging chroma for every beat, or chroma landmarks (Bertin-Mahieux et al. [2010]). Similarly for chord sequences, extracting and comparing fingerprints is a proven way to search for a cover song (Khadkevich and Omologo [2013]). While comparing chroma or chord sequences looks to be a standard way for today's harmony analysis, new ideas have arisen in the past years, pioneering similarity and distances between chords themselves (De Haas et al. [2008], Rocher et al. [2010b]) and calling for further research on this topic. The paradigm shift is to look at the chord as a point in the space and treat progression in the musical piece

as a path in this space.

Our work on the *harmony-analyser* application is following up on these efforts. We provide library and tools capable of creating, naming, and analyzing chords and visualize chord progressions. Furthermore, we gather all relevant chord distances and provide tools for their comparison. We also pioneer new distances between chroma vectors, offering multiple options to approach this novel concept.

In the following sections, we show the contents of our *JHarmonyAnalyser* library and explain more technically how we developed the definitions from Chapter 3 (chord complexity distance, chroma vector distances) or how we performed the experiments from Chapter 4. In Section 5.4, we present how new plugins can be created and added to the system in order to achieve custom analysis. We wrap up by showing snapshots of GUI tools in Section 5.5, and our course for future work in Section 5.6.

5.3 Technical specifications

The website *harmony-analyser.org* is the home for *JHarmonyAnalyser* library as well as simple GUI tools to present its main functions, together forming a cohesive system for harmony analysis. It contains two main packages:

- *chord_analyser* package, containing Java classes for evaluating chord distances
- *chroma_analyser* package, containing Java classes for evaluating experimental chroma distances

Both packages have a similar structure and fulfill these common requirements:

1. The packages contain base classes that supply them with the representations of tones, intervals, and scales (typically one Java class for each entity).
2. The packages contain the main classes, Chord and Chroma respectively for *chord_analyser* and *chroma_analyser* package. Multiple constructor inputs are supported, including text input.
3. The packages provide methods to derive distances between its main entities. For this purpose, multiple model classes are provided, representing some model of computation (e.g., TPS model).

To ease the work with a real-life input, the services package is included, containing tools for reading WAV files, or interpreting MIDI input from the external MIDI instrument (e.g., MIDI keyboard). We show how all the packages work together in an example scenario.

Example. Let us consider MIDI input keyboard plugged in to our machine. Classes available in services packages are able to get the keyboard and decode its input. We may use the *chord_analyser* package to construct chord,

or *chroma_analyser* package to construct chroma from the input from the keyboard. We choose to proceed with playing two chords (e.g., C major and G7) and creating two Chord objects in the memory. Then, using the TonalPitchSpace model located in *chord_analyser*, we calculate the distance between C major and G7, outputting the result to a text area in GUI. We explain the main contents of each package in the subsections below.

5.3.1 Chord analyser package

With *JHarmonyAnalyser* library being updated on a regular basis, new models are being included in the packages, to achieve an up-to-date selection. Brief summary of the current model classes contained in *chord_analyser* package in version 1.2 of *JHarmonyAnalyser* are:

- *TonalPitchSpace* – as defined by Lerdahl [2001], with a symmetricity fix as described by De Haas et al. [2008].
- *ChordComplexityDistance* (ChordCD) – as defined in Chapter 3, a complexity of the transition between two subsequent chords.

As we described in the previous chapters, ChordCD treats the added dissonant tones on the top of major or minor chord as additional layers of complexity. Rules are defined to add/remove/modify the dissonant tones. The resulting distance is defined as the complexity of constructing the next chord from the previous chord and is similar to the edit distance of two strings. The reader can refer back to Figure 3.2 for the illustration of the transition.

5.3.2 Chroma analyser package

In this package, we provide classes to calculate experimental distances between chroma vectors. Example models in version 1.2 of the library are: *SimpleDifference* and *ComplexityDifference*. As described in Section 3.3, Euclidean distance or Manhattan distance can be used for chroma vectors, but we wish to test out a more musicological approach, and so in this package, we can reuse an existing interface to add new distances. *chroma_analyser* package further allows us to use these distances directly for analysis of a musical piece.

5.4 Extensible plugins

The *harmony-analyser* system can also act as an easy-to-use framework for creating plugins for music analysis in Java (not necessarily related to chords or chroma vectors). There are ready-made plugins available in the plugins package of the library. Custom plugins can be created simply by following the proposed structure. Plugins form one of the core parts of the system since they are responsible for opening and analyzing WAV files using the models from the rest of the library. While distances between chords and chroma vectors can be calculated simply by calling a method from the model class, analysis of WAV file format usually takes more subsequent steps in

order to achieve the desired result. For example, we first need to derive chroma vectors from the WAV file, and then we need to get distances between subsequent chroma vectors. Optionally, we may want to plot a line graph of the resulting distances. The resulting 3-step process would be unfit for one wrapper method, especially if we want to observe the partial results. Custom and ready-made plugins serve this purpose of layered analysis. Inspired by GNU GPL licensed C++ *Vamp plugins*[1], a plugin has its own unique key and analyses the whole WAV input in its specific manner, producing results that are well documented. In fact, thanks to Java wrappers provided by Queen Mary University, we include a selection of Java-wrapped Vamp plugins in the *plugins/vamp_plugins* package. Differently from Vamp plugins, results from one plugin are very often taken by another plugin to continue with higher-level analysis (e.g., chroma vectors picked up by chord estimator, or chord progression picked up by chord distance plugin).

Each plugin needs to meet the following requirements:

1. It needs to have a *pluginKey*, a unique key for the plugin. An example key would be: *chord_analyser:tps_distance*.
2. It needs to provide a list of input files and their extensions.
3. It needs to provide a list of output files and their extensions.
4. It needs to implement the *analyse* method, which takes all the input files and creates correct output files.

Example plugins are (for the full list, please refer to the documentation on *harmony-analyser.org*):

- *chord_analyser:tps_distance* implementing TPS model analysis for a WAV file.
- *chord_analyser:chord_complexity_distance* implementing the same analysis as we showed in Section 3.5 (ChordCD analysis for a WAV file).
- *chroma_analyser:complexity_difference* implementing the chroma vector distance analysis (ChromaCD analysis for a WAV file).

5.5 GUI tools

Along with the library and plugins, *harmony-analyser.org* also provides simple GUI tools (jointly named *harmony-analyser* tools), which were developed using *java.awt* package, to show the main library functions. Screenshots of some of the main tools are in the figures 5.1, 5.3 and 5.2. Example tools are:

- Chord Transition Tool - using MIDI or text input to analyze chords or chroma vectors.
- Audio Analysis Tool - batch analysis using selected plugins for an input folder with WAV files.

[1] `https://www.vamp-plugins.org`

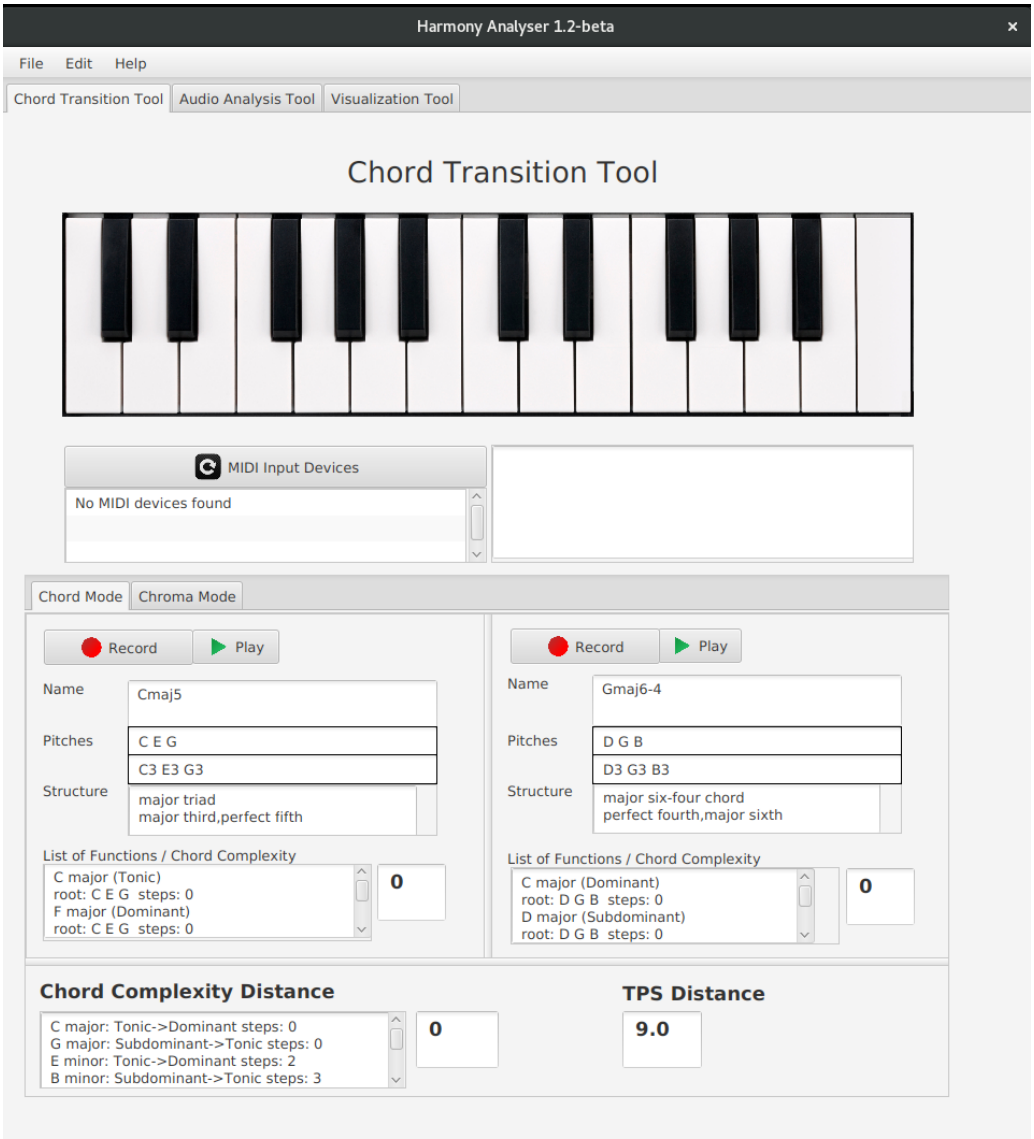


Figure 5.1: Chord Transition Tool: capturing the MIDI input and outputting the chord labels, functions and the chord distances. *C major* and *G major* chords are analysed.

- Visualization Tool - plotting graphs and diagrams with chord or chroma vector distances for a WAV file.

5.5.1 Chord Transition Tool

When the application starts, the default tool selected is the *Chord Transition Tool* (see Figure 5.1). The user can either use the MIDI keyboard plugged in via the USB port, or use a text input field, to specify two chords. The added value compared to other common MIDI software is a list of functions and chord distances, based on the tonal analysis (described in more details in Maršík et al. [2014b]). The fact that the chord can have multiple functions in music is commonly accounted for in the works on musicology, but less frequently in the MIR works. This is one of the many examples of a gap between MIR and musicology, which we discussed in the previous chapters.

Chord Transition Tool shows the chord and all of its tonal functions, and the user can observe various chord distances: ChordCD (Maršík et al. [2014b]), or TPS Distance (Lerdahl [2001]) as seen in Figure 5.1, which gives him/her a good overview of the chord transition.

5.5.2 Visualization Tool

After the user is familiar with the chordal analysis described in the previous section, the next step is to observe the chords, chord distances, or chroma vector distances extracted from the real audio. We offer the Visualization Tool (see Figure 5.2) to visually understand how the labels and distances can help to analyze a musical piece. In the musical piece analysis shown in Figure 5.2 (Hallelujah by Bastian Baker) we have time in seconds on the x axis, and chord distance values of *each pair of the subsequent chords* on the y axis. Visualized is the ChordCD time series fingerprint, one of the song fingerprints that we experimentally studied in previous chapters. The local peaks around 30^{th} or 80^{th} second represent the transition between *Ami* and *F* chords. The peaks after the 150^{th} second represent the same transition with the singer performing vocal ornaments in the last verse, yielding a higher (more complex) chord distance value, since the voice is accounted for in the chord estimation.

The same chart visualization can be shown for each plugin that extracts the values in the form of a time series (e.g., chroma vector distances), or labels with a timestamp (chord or key detection). Some plugins will output column charts, such as Average Chord Complexity Distance (Maršík et al. [2014a], named ATC in the Chapter 3) in Figure 5.2. We have shown how these averaged features improve the music genre detection in Section 4.2.

5.5.3 Audio Analysis Tool

The last step of the analysis after understanding the harmony features thoroughly is to apply the chosen analysis on a folder with WAV files. This can be achieved by the Audio Analysis Tool (Figure 5.3). The plugins are categorized in the plugin groups (*Vamp plugins, Chord analyser, Chroma analyser*) and the details and parameters of the selected plugin are shown. After hitting the *Analyse* button, the tool creates text files with the analysis results in the selected folder. These can be used as an input for another analysis plugin, or an input for a retrieval technique. There is also an additional *Post Processing* tab that serves various purposes, such as applying a smoothing filter to a time series. The additional tabs can also be helpful for importing or exporting other file types so that the application can be used for various projects. As an example, The Million Song Dataset from Bertin-Mahieux et al. [2011] uses *HDF5* files, and by providing conversion to text files, the dataset can be used easily with Audio Analysis Tool.

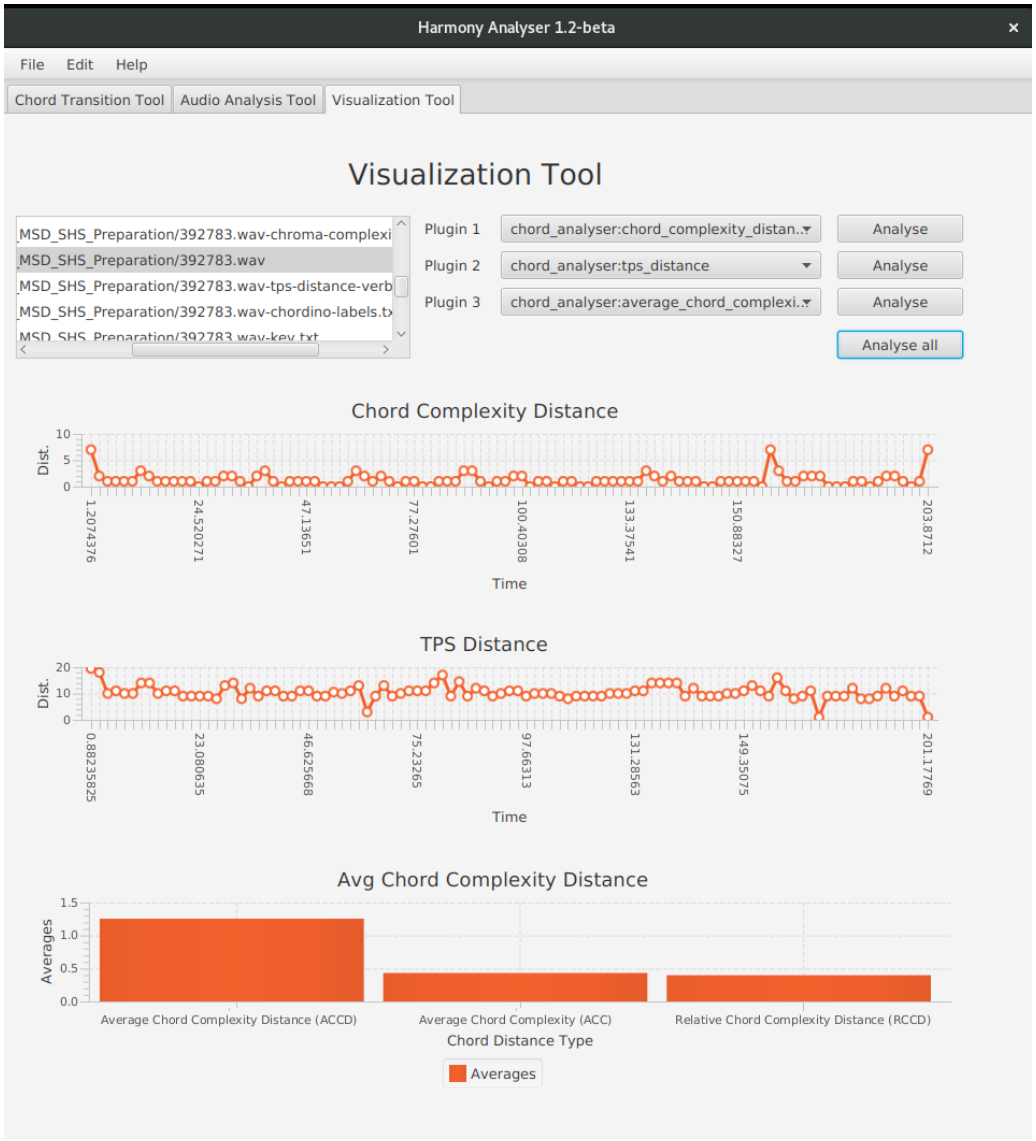


Figure 5.2: Visualization Tool: Analysis of Hallelujah by Bastian Baker is shown, containing results for Chord Complexity Distance, TPS Distance, and three types of averages for Chord Complexity Distance (Maršík et al. [2014a]).

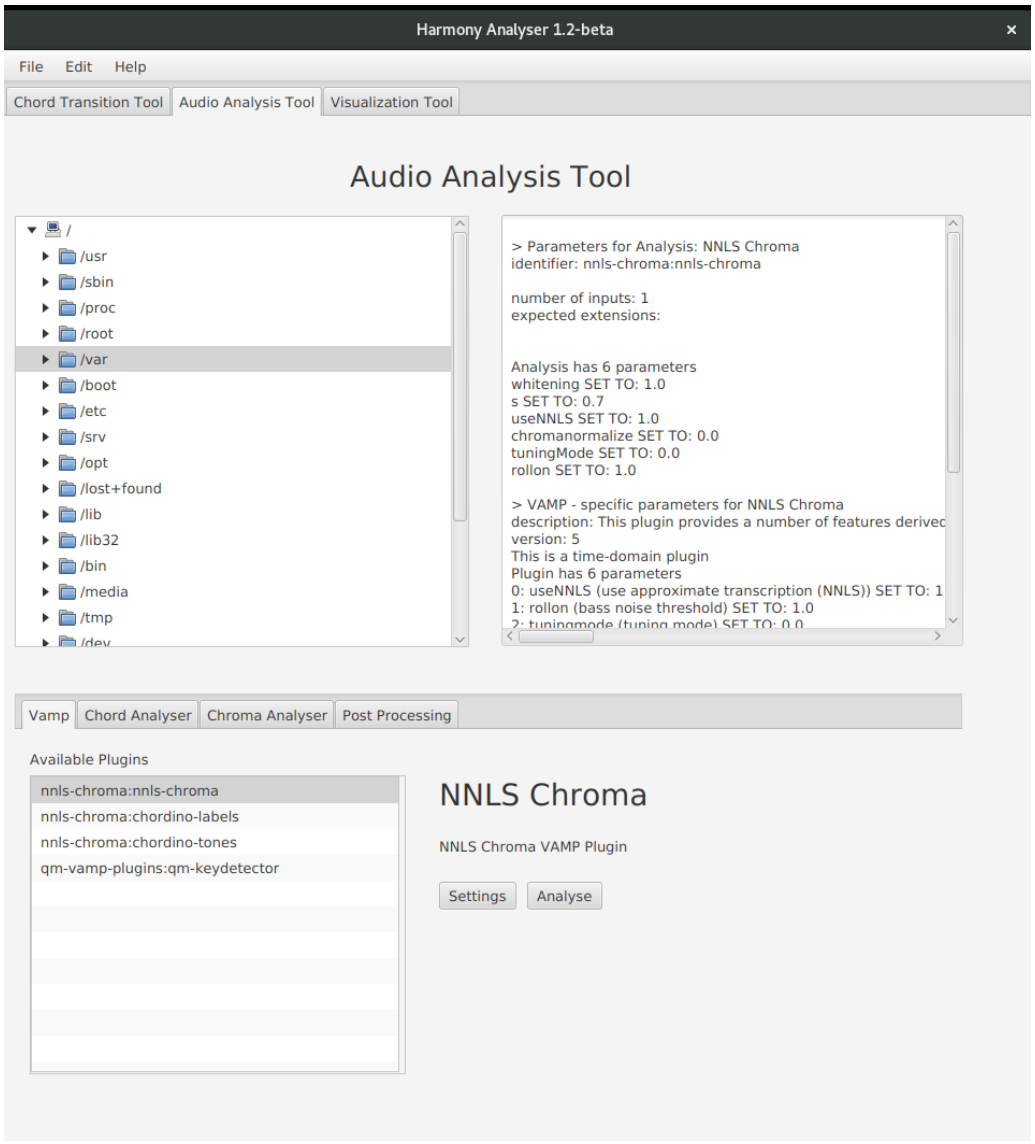


Figure 5.3: Audio Analysis Tool: selecting a folder with WAV files and choosing a desired plugin for analysis.

5.6 Experiments with harmony-analyser and future work

The *harmony-analyser* tools have already been tested on various MIR tasks, as we have shown in the previous chapters. The time series produced by *harmony-analyser* were tested on both covers80 dataset[2] and a subset of SecondHandSongs dataset[3] (999 songs). Additionally, *harmony-analyser* project was one of the key ingredients to form the *Kara1k* dataset, to which we devote Chapter 6. The analysis was successfully used in a project of three laboratories (Charles University, IT4Innovations, and Université Bordeaux) forming the *KaraMIR* initiative. The results using *harmony-analyser* were presented on several international conferences and were given prizes by the juries in the ACM SRC competition in 2017[4] and Fourier Prize Competition in 2019[5].

We provide a maintained, tested, and publicly available Java library and tools for chordal analysis, under *harmony-analyser.org* domain. *JHarmony-Analyser* library can be used for feature extraction, visual analysis of musical pieces, or comparison of chord and chroma distances, as well as a simple framework for the development of music analysis software in Java. As future work we plan to add new models to the *chord_analyser* package, starting with the model proposed by Chew [2014]. We also plan to broaden the plugin base with plugins and smoothing filters. Our other goal is to host the automatic analysis as an online service, rewriting its code to a programming language suitable for web applications.

By providing these free tools, we hope to encourage future research in harmony analysis, and we are open for contribution and feedback on our project page or a GitHub repository page[6].

[2] `https://labrosa.ee.columbia.edu/projects/coversongs/covers80`
[3] `https://labrosa.ee.columbia.edu/millionsong/secondhand`
[4] `https://src.acm.org`
[5] `https://atos.net/cs/ceska-republika/cena-josepha-fouriera`
[6] `https://github.com/lacimarsik/harmony-analyser`

6. KaraMIR project and Kara1k dataset

6.1 KaraMIR project introduction

A major challenge in the MIR field is the creation of large, clearly labeled musical datasets (Casey et al. [2008]), which are notably usable for cross-dataset comparison (Bogdanov et al. [2011]). The low number of studies dedicated to a given MIR task can be attributed to the low number of available musical datasets and their corresponding ground truths for the task. One of the tasks that require additional well-annotated datasets is CSI. The difficulty of the CSI task is often attributed to many ways in which the cover version can differ from the original recording (Ellis and Cotton [2007], Serrà et al. [2008, 2009]), such as melody, rhythm, or instrumentation. Datasets specializing in a single aspect of a cover song would allow the researchers to tailor their algorithms to the specific needs. Another recent MIR challenge is the analysis of the singing voice. To the best of our knowledge, only two musical datasets currently propose the annotations for the singing voice presence at a frame scale (Ramona et al. [2008], Kroher et al. [2015]), which is the requirement for further voice analysis (Bayle et al. [2016], Ghosal et al. [2013], Lehner et al. [2014, 2015]). In the light of these challenges, karaoke songs form an interesting subset of cover songs, which differ from the original recordings in the singer's voice, melody, and instrumentation, but resemble the original recordings in rhythm, tempo, harmony, and structure. Additionally, karaoke recordings have the benefit of a separate voice track from the instrumental track. The voice track can subsequently be used for the voice analysis, while the combined track can provide sample-precise annotations of the singing voice.

In this article, we fulfill these needs by creating a *KaraMIR* project[1] dedicated to forming the new tasks, datasets, and tools. As our base dataset we create *Kara1k*, a dataset of $1,000$ cover songs from a karaoke company Recisio[2], along with $1,000$ original recordings. Our goal is to propel new CSI studies with this dataset, as the current state-of-the-art CSI algorithm has not been outperformed for over eight years, and yields 73.5% accuracy (Serrà et al. [2009]). We also wish to contribute to the singing voice analysis tasks by offering a separate voice track with multiple ground truths. We tackle the need for a variety of features and, more importantly, their proper definitions, by using and referencing five feature extractors, both traditional (Tzanetakis and Cook [2002], Mauch and Dixon [2010], Mathieu [2002]) and emerging in the last years (Bogdanov et al. [2011], Maršík et al. [2017]), featured in recent MIR reviews (Moffat et al. [2015]). But, the main highlight of the *Kara1k* dataset is its originality, since, in Recisio karaoke application, the cover songs are performed by professional musicians in a similar setup as the original recordings, allowing for interesting comparisons. We elaborate and

[1] `http://yannbayle.fr/karamir`

[2] `https://www.recisio.com`

demonstrate these benefits in our supporting experiments.

In Section 6.2, we look at the current state of musical datasets, focusing on CSI and singing voice analysis tasks. In Section 6.3.3, we detail the characteristics, features and metadata of the *Kara1k* dataset. Although we already presented experiments using *Kara1k* dataset in Section 4.4 featuring CSI as the core task of this thesis, we also want to show the usefulness of the dataset for other MIR tasks. Section 6.4 thus adds a supporting experiment of the Singer Gender Classification task. Lastly, in Section 6.5, we summarize the project contributions to the community and discuss future work.

6.2 Related work

6.2.1 Datasets for cover song identification

The first academic attention to the CSI task emerged in 2006 when this task was added to the MIREX benchmarking. Ellis and Cotton [2007] provided *covers80*, one of the first publicly available datasets for CSI. Ellis and his team also won the first MIREX CSI benchmark with their LabROSA system. The best result on the MIREX benchmark was achieved by Serrà in 2009 (Serrà et al. [2009]). Serrà et al. used the HPCP and the cross recurrence quantification method to correctly identify $2,426$ out of $3,300$ (73.5%) cover songs, a result which was not outperformed since. This implies, that to further improve the results for CSI, the researchers need to focus on the partial tasks or use diverse datasets. Another milestone in CSI was the proposition of the *SecondHandSongs* dataset as a subset of the *Million Song Dataset* provided by Bertin-Mahieux et al. [2010]. *SecondHandSongs* dataset is currently the largest CSI dataset with 18,196 tracks, and as such it serves the large-scale approaches well. Features are extracted by The Echo Nest service for each track. However, the dataset does not provide all definitions of the features, and new feature extraction is not planned at the time of writing this article. The CSI studies of the last six years have been switching the focus from the accuracy and exact retrieval to the large-scale approaches and database pruning (Bertin-Mahieux et al. [2010], Osmalskyj et al. [2013]). It is now that more and more researchers call for new datasets and comparisons, in order to take new challenges (Bogdanov et al. [2011]).

6.2.2 Datasets for singing voice analysis

Many studies have been dedicated to the identification of the gender of the speaker, but only a few works have investigated a *Singer Gender Classification* (SGC) task (Schuller et al. [2010], Weninger et al. [2011b,a]). To the best of our knowledge, only two datasets currently propose the annotations for the singing voice presence at a frame scale. Ramona et al. [2008] annotated 93 songs of the *Jamendo*[3] dataset and Kroher et al. [2015] annotated 100 flamenco songs in *cante100*. However, the needed human labeling of the singing voice presence at a frame scale is prone to mistakes (Sturm [2015], Sturm [2015]) and is time-consuming (Skowronek et al. [2006]).

[3] `https://www.mathieuramona.com/wp/data/jamendo`

The human labeling was also used in *AudioSet*, a recent large-scale dataset of manually annotated audio events (Gemmeke et al. [2017]). *AudioSet* contains 5,800 hours of audio, and the annotations are made for 10-second sound clips drawn from YouTube videos. For each snippet, human labelers had to confirm the suggested metadata for the video, such as the "female singing" tag[4]. However, as a result, some snippets in the balanced train segments are labeled with the "female singing" tag but also contain less audible male background singing voice, as can be observed in some of the videos[5]. This kind of shortage weakens the performances of a SGC task, although *AudioSet* is a great example of a multi-purpose dataset.

Consequently, a dataset providing the singing voice track annotated at the time of recording, and the corresponding instrumental track for each song could highly benefit the SGC task, as well as the *Singing Voice Detection* (SVD) task (Bayle et al. [2016], Ghosal et al. [2013], Lehner et al. [2014]) and the *Singing Voice Separation* (SVS) task (Weninger et al. [2011a], Liutkus et al. [2014]). The identification of the singing voice presence needs to be straightforward, faultless, and sample-precise. These requirements are automatically reached in a dataset that contains the singing voice track and the instrumental track for each song. The zeros in the singing voice track signal can be used as objective annotations of the singing voice absence. One of the first datasets reaching the previously mentioned requirements is *MedleyDB* (Bittner et al. [2014]). Indeed, this musical dataset of 122 multi-track audio contains high-quality ground truths. *MedleyDB* also contains the mix, the raw audio, and the metadata for each of those songs. It includes 70 songs containing vocal tracks that make this dataset suitable for singing-voice related tasks, although, this number is limited for a scale-up analysis.

6.2.3 Karaoke datasets

The datasets MIR-1K (Hsu and Jang [2010]) and iKala (Chan et al. [2015]) were the first to provide karaoke songs to the MIR community. They contain up to 30-second excerpts for up to $1,000$ karaoke songs, and the corresponding ground truths. A recent addition was the work of Smith (Smith [2013]). The *Sing! Karaoke* part of the DAMP archive (Stanford's Digital Archive of Mobile Performances)[6] is the largest archive of karaoke songs, containing over 34,000 songs captured by the mobile devices around the world, using the *Sing!* application by Smule[7]. Audio files are available for download for both the singing voice track and the background track. The features provided are MFCC features and *pitch tracks*, which are the pitch estimations of the singing voice in time, which is useful for voice analysis, but the features that could help CSI are not provided per se. Overall, *Sing! Karaoke* dataset is an original contribution that facilitates the research of amateur singers' voices recorded all around the world, highlighting the detection of singer's age, nationality, or gender, and the possible use of deep learning methods.

[4] `https://research.google.com/audioset/dataset`
[5] `https://www.youtube.com/watch?v=3w-AEo6rgI8`
[6] `https://ccrma.stanford.edu/damp`
[7] `https://www.smule.com/apps`

To further improve these choices, we propose *Kara1k*, a multi-purpose database included in the *KaraMIR* project, to foster existing and new MIR tasks, and more specifically the CSI and singing voice analysis tasks. We do so by the inclusion of the original recording to the dataset and by providing an unmatched variety of features as well as new annotations – in addition to the DAMP archive we annotate backing vocals and their gender, duets, and we tag multiple languages and genres per song. Notably, the karaoke songs are recorded differently from the datasets mentioned above. In the next section, we describe the *Kara1k* dataset in details.

6.3 Description of *Kara1k*

6.3.1 Description of songs

We set up a partnership with Recisio[8], a company developing online music products and services, to propose the *Kara1k*[9] database that references a list of 1,000 cover songs provided along with a list of 1,000 original songs corresponding to the cover songs. The cover songs contain the singing voice and the instrumental tracks and were recorded in a studio with professional musicians following the original song scores. We refer to them as *karaoke* songs, but it is important to note the quality of the performance and the fact that they are used to guide the amateur singers in the karaoke mobile application.

The original songs are represented by a mixed track only and were assembled from sources distinct from Recisio, as Recisio can only distribute the cover versions. We undertook the challenge of manually checking the original songs for quality, and unifying their file type, to avoid some problems stemming from multiple sources (we refer the reader to *KaraMIR* website for more information). As a result, the original song and the karaoke song from *Kara1k* resemble each other in sound, but the listener is able to distinguish the difference in voices, timbre, slight variations in the tempo and structure, and the different recording conditions.

The 1,000 titles in *Kara1k* is a selection of the most sung songs for the month of January 2016 in the Recisio Karafun application[10]. The titles can be explored on our *KaraMIR* project website, which facilitates the analysis of *Kara1k* data and metadata. All cover songs from *Kara1k* are also available for listening on the Karafun application, although Recisio reserves the right to modify the list of available songs according to users' needs. In June 2018, Recisio provided 29,791 songs in their karaoke application and constantly improve their services, so we consequently plan to maintain the *KaraMIR* project, providing more features or creating bigger datasets.

The audio files in *Kara1k* were unified to mono tracks and were encoded in the 16-bit unsigned little-endian WAV format with a sampling rate of 44,100 Hz before the feature extraction. However, both karaoke and original songs are protected by the copyright of their owners, which is the reason

[8] `https://www.recisio.com`

[9] `http://yannbayle.fr/karamir/kara1k`

[10] `https://www.karafun.com`

why the audio files are not available in *Kara1k*. We decided to provide all the necessary musical features extracted from the audio files and are happy to provide additional features upon a simple request through the *KaraMIR* website.

In the following sections, we detail the provided metadata, features, and the ground truths for the CSI and SGC tasks.

6.3.2 Metadata and ground truths

Kara1k contains the artist name, the song name, and the custom identifier used by Recisio for each song. Moreover, we provide the International Standard Recording Code[11] (ISRC) given by the International Federation of the Phonographic Industry[12], as well as the YouTube[13] link, to reference each cover song to the corresponding original song easily. For each song, *Kara1k* also indicates the sung language, which includes English, French, German, Spanish, Italian, Dutch, Portuguese, and Korean. Hence, it is possible to classify the sung language. The distribution of language tags can be seen in Figure 6.1. Likewise, Recisio indicates the genre tag for the cover songs in the Karafun application. The songs can have as many genre tags as applicable. The 10 most prevalent genres are shown on Figure 6.2. Since *Kara1k* is dedicated to the singing voice analysis tasks, we provide the related ground truths. For each song, we indicate the presence of backing vocals, duets, and the gender of the singer, or the vocalists sounding simultaneously. We thus annotate each song with an annotation from {*female, females, male, males, mixed*}. This level of detail was achieved thanks to the Recisio original karaoke experience. The Karafun application contains multiple vocal tracks and allows to switch lead singers and backing vocals on and off. The relationship between the Karafun application and our ground truths is depicted on Figure 6.3 and the distribution of the singers' genders can be seen in Figure 6.4.

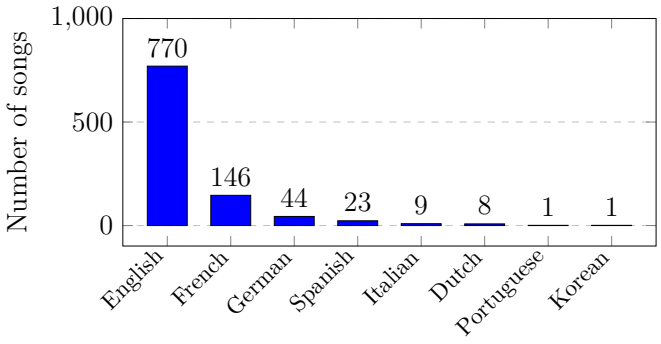


Figure 6.1: Distribution of the language tags in the *Kara1k* dataset. Songs can have multiple language tags, the sum of all occurrences of the tag is displayed.

[11] `https://isrc.ifpi.org/en`
[12] `https://www.ifpi.org`
[13] `https://www.youtube.com`

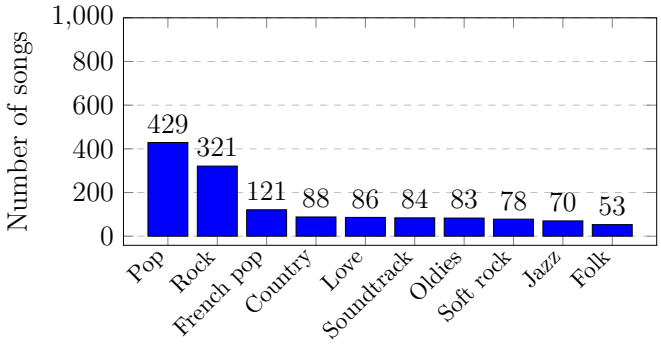
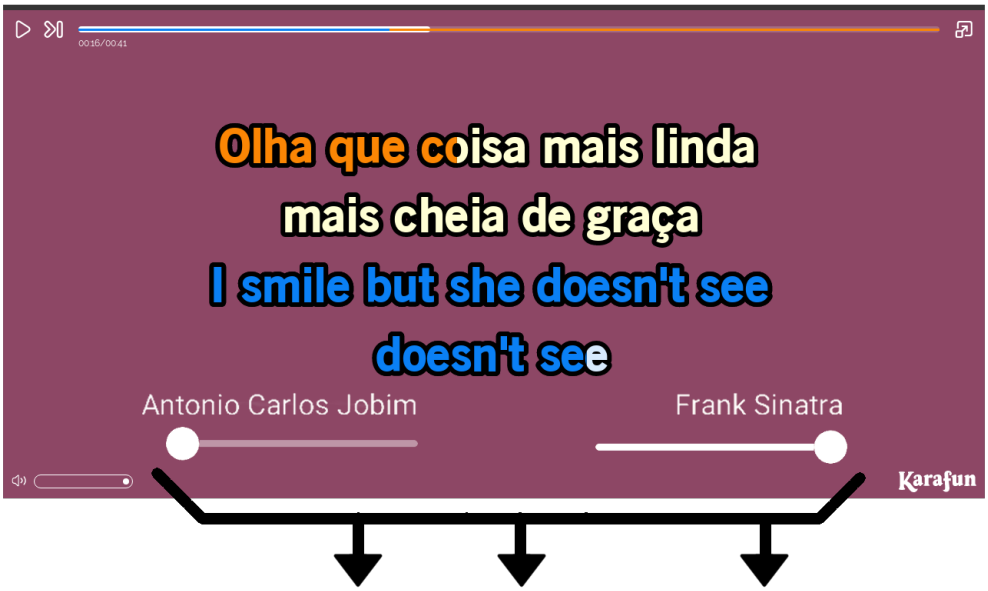


Figure 6.2: The 10 most prevalent genre tags in the *Kara1k* dataset. Songs can have multiple genre tags.



languages	genders	back_vocals	duet
English,Portuguese	male	0	1

Figure 6.3: The relationship between the Karafun application and the *Kara1k* dataset ground truths. The runtime of the Karafun application on the top shows two professional singers singing a duet, imitating the original duet by Antonio Carlos Jobim and Frank Sinatra. The user is able to switch the singers' tracks on and off. As a consequence of this feature, Recisio was able to provide details such as multiple sung languages and genders of the singers. Since the two male singers sing the duet one after another, the corresponding gender tag is *male*, meaning one male voice sounding at a time. For the male duets that are sung simultaneously, the gender tag would be *males*.

In addition to these ground truths, we also provide other non-standard ground truths such as year of the original recording, or, explicit language annotation, that Recisio uses to help its users know whether the song is suitable for the target audience.

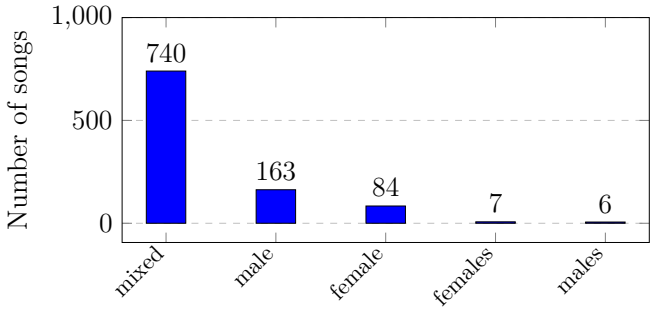


Figure 6.4: Distribution of the singers' gender annotation in the *Kara1k* dataset. The gender annotation includes both lead and back vocalists, and specifies mixed gender if at least for a few moments the male and female vocals are sounding in the song. On the contrary, females/males annotation strictly means only one gender in the whole song.

We freely provide all metadata and the ground truths for *Kara1k* on our *KaraMIR* website and are open for further annotation requests[14].

6.3.3 Audio features

Essentia in version 2.1 (Bogdanov et al. [2011])[15], *harmony-analyser* in version 1.2 (Maršík [2016])[16], *Marsyas* in version 0.5 (Tzanetakis and Cook [2002])[17], *Vamp plugins* (NNLS Chroma version 1.1 by Mauch and Dixon [2010], Key Detector version 1.7 by Noland and Sandler [2007][18]) and, finally, *YAAFE* 0.64 Mathieu [2002][19] have been used to extract audio features for each track in *Kara1k*.

All available features in *Essentia* have been extracted. *Vamp plugins NNLS Chroma* and *Chordino* Mauch and Dixon [2010] were used to extract the chroma features and the chord labels. *Key Detector* from *QM Vamp Plugin* set[20] was used for the key finding. The *harmony-analyser* tool was then used for further high-level processing, such as the extraction of *Tonal Pitch Space* (TPS) distances Lerdahl [2001] in between every pair of the subsequent chords, and also to convert chord and key features to vector format, resulting in eight additional harmony features. We used *Marsyas* to extract 68 features for each track as was done by Gouyon *et al.* Gouyon et al. [2014]. These features are the mean and standard deviation of the zero crossing rate, the spectral centroid, the roll-off and flux, and the first nts13 MFCC. Regarding *YAAFE*, 13 MFCC after the 0^{th}, the delta and the double delta have been extracted for each frame of each song with an analysis frame of $93ms$ and 50% of overlapping.

Table 6.1 details the features extracted and the software used. There

[14] `http://yannbayle.fr/karamir/contact`
[15] `https://essentia.upf.edu`
[16] `http://harmony-analyser.org`
[17] `http://marsyas.info`
[18] `https://www.vamp-plugins.org`
[19] `https://yaafe.sourceforge.net`
[20] `https://www.vamp-plugins.org/plugin-doc/qm-vamp-plugins.html`

are three types of features provided. The first type, named Single, refers to one feature extracted, usually in the form of a scalar, tuple, or a low-dimensional vector (e.g., a global chroma vector for a track). The second type, named Timestamp, refers to structured features composed of a series of timestamp-value pairs (e.g., moments when a chord change occurs). The third type, named Frame, refers to a feature composed of values extracted for each frame of the signal. More details about the features and the feature extraction software are available on our *KaraMIR* website. We encourage other researchers to contact us in the case of a specific need of features that we do not offer yet.

Table 6.1: The main features available in the *Kara1k* dataset. The complete list can be found on the project website.

Software	Features	Type	Description
Essentia (Bogdanov et al. [2011])	47 low-level features	Single	Statistical measures for: barkbands, dissonance, erbbands hfc, melbands, pitch salience, silence rates spectral, GFCC and MFCC features
	13 rhythm features	Single Timestamp	Beats (count, BPM, loudness, 6 histograms), danceability, onset rate, beat position
	24 harmony features	Single	Statistical measures for: chords, keys, tuning, HPCP, THPCP
	9 metadata	Single	loudness, length, lossless, sample and bit rate replay gain, codec, downmix
harmony-analyser (Maršík [2016])	10 harmony features	Frame Timestamp	high-level harmony features: chord and key vectors TPS, ChordCD, ChromaCD distances
Marsyas (Tzanetakis and Cook [2002])	68 scalar features	Single	Statistical measures for: Zero Crossing Rate, Spectral Centroid, Roll-off, Flux, 13 MFCCs
Vamp plugins (Mauch and Dixon [2010], Noland and Sandler [2007])	4 harmony features	Frame Timestamp	NNLS Chroma, Key Detector Chordino labels and tones
YAAFE (Mathieu [2002])	1 MFCC feature	Frame	13 MFCCs

6.4 Supporting experiment of singer gender classification

We add one final experiment as a part of this dissertation – a supporting experiment for the singer gender classification (SGC) task. The *Kara1k* dataset may be used for a variety of purposes, but the choice for SGC is obvious because of the provided metadata and ground truths.

6.4.1 Description of the task

The SGC task aims at identifying whether the singing voice comes from a female or male adult singer. In this supporting experiment on a subset of the *Kara1k* songs we want to showcase how the *Kara1k* dataset can aid SGC and other singing voice analysis tasks. In our scenario, we wish to quantify the loss in the classification performance when considering the singing voice track alone (Vocal) and when the voice track is mixed with its corresponding instrumental track (Mix). This quantification thus measures the degree of

suitability of the chosen features for SGC under a "signal and noise" model. The signal is constituted here by the singing voice and the noise by the instrumental background. We assume that if perfect features are used, the performance would be the same in both the Vocal and the Mix conditions. We believe that this kind of experiment is needed for further comparison of novel features and to reassess the importance of common features for the SVD, such as the MFCC.

6.4.2 Methods

In this experiment, we chose those songs from *Kara1k* that have no mixed gender in the leading singing voice. We thus discarded the songs that contain backing vocals, choirs, and duets. We also needed pure singing voice tracks for our experiments. *Kara1k* comes with an *instrument bleeding* annotation, which tells us whether the microphone recordings of the leading singing was made spatially too close to the instrumental background, and a minor instrumental background is present in the voice track. Using these filters, we selected 25 songs that were only sung by an adult female and 50 sung by an adult male. The observed female/male song ratio is consistent with the one depicted by Schuller et al. [2010].

The next step removed the silences in the singing voice tracks and the corresponding samples in the instrumental tracks. The silence removal is necessary to avoid the extraction of the features during the silent part of the song. Figure 6.5 illustrates the silence removal process.

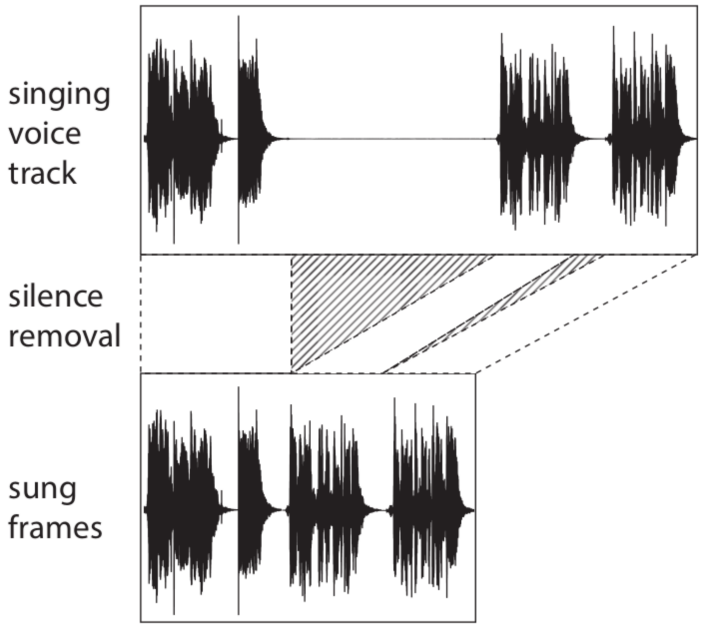


Figure 6.5: Schema illustrating the silence removal process on a singing voice track. Stripes indicate the removed frames while the blanks show the copying of the singing voice frames.

Afterwards, *YAAFE* extracted 13 MFCC, the delta and double delta for each frame. The MFCC features were chosen because they are well known

to capture the singing voice variations at a frame scale (Bayle et al. [2017], Lehner et al. [2014]). In total, there were 410, 130 male frames and 215, 460 female frames. We then classified the features of the female and male frames for the Vocal condition and gathered the classification results. We did the same under the Mix condition. A 5-fold cross validation method with artist filtering (Flexer [2015]) was used to select the train and test sets. The features in these sets were classified by a Random Forest algorithm provided by the python package scikit-learn (Pedregosa et al. [2011]). Tree-based algorithms like Random Forest were used in this experiment because they are known to perform well with singing voices (Bayle et al. [2016], Lehner et al. [2014]).

In the following section, we compare the weighted average precision, recall, f-score, and accuracy for the Vocal and Mix conditions.

6.4.3 Results

Table 6.2 depicts the weighted average precision, recall, f-score and accuracy for the SGC task. Our results indicate that the performances achieved for the Vocal condition are higher than for the Mix condition. Indeed, the precision for the Vocal and Mix conditions had a statistically significant difference (one-way ANOVA, $F = 8.12$, $p = 0.021$, significance threshold $= 0.05$). The same tendency is observed for the recall, the f-score and the accuracy (one-way ANOVA, respectively $F = 8.74$, $p = 0.018$; $F = 8.02$, $p = 0.022$; $F = 8.74$, $p = 0.018$; $F = 0$, $p = 0$; significance threshold $= 0.05$).

Table 6.2: Results for the SGC task on the *Kara1k* dataset. The rows indicate distinct metrics, and the columns show the Vocal and the Mix conditions. The metrics displayed are the weighted average precision, recall, f-score and accuracy $\pm$ the standard deviation.

Metric	Vocal	Mix
Precision	0.721 ± 0.034	0.654 ± 0.040
Recall	0.725 ± 0.029	0.665 ± 0.035
F-score	0.722 ± 0.032	0.656 ± 0.041
Accuracy	0.725 ± 0.029	0.665 ± 0.035

6.4.4 Discussion

The results presented for this supporting experiment are not meant to perform better than the state-of-the-art algorithms but serve as a baseline reference for further experiments on *Kara1k*. Indeed, a more in-depth benchmark of existing SGC algorithms (Schuller et al. [2010], Weninger et al. [2011b], Weninger et al. [2011a]) is needed for the *Kara1k* dataset. However, it can be nicely seen how the algorithm has the highest precision for the Vocal tracks, and how musical accompaniment may confuse the algorithm and lessen the accuracy.

We also propose to study new features such as the pitch contour (Bittner et al. [2014]) and the vocal variance (Lehner et al. [2014]), since they have not been tested yet and they both seem promising for the SGC task.

6.5 Conclusion for the KaraMIR project

This chapter described the *KaraMIR* project that includes *Kara1k*, a dataset dedicated for particular MIR tasks, notably CSI and singing voice analysis. We described the benefits of *Kara1k* for both of the tasks, we described the differences from the other karaoke datasets, and we discussed the dataset content. The main advantages are the quality of the cover songs recording, pure singing voice track, a large variety of features and ground truths, including non-standard features from Recisio such as duet, backing vocals or explicit language.

We provided two sets of use-case experiments on the *Kara1k* dataset. The first was presented earlier in Section 4.5 and uses the cover songs from *Kara1k* to benchmark the existing and novel features dedicated to the CSI task. Our experiments showed that the more complex features such as chroma, MFCC, and chord features are well suited toward CSI using a direct application of the DTW algorithm, while e.g., key features are not. From the fingerprints, ChromaCD has the best accuracy on *Kara1k* dataset. An interesting result was the similar accuracy for both the complex chroma feature time series and the simple chord feature time series. We also provided a statistical analysis of the features earlier in Section 4.4, and we assessed the harmony features being the best for the CSI task.

The second set of experiments exploits the singing voice track of *Kara1k* for a SGC task. We found that the MFCC features and corresponding delta and double delta are not well suited to discriminate female and male singing voices because the performances drop when an instrumental background is introduced.

We provide multiple ideas to challenge the state-of-the-art in CSI and SGC with the help of our *Kara1k* dataset. Finally, we freely distribute the dataset, ground truths, and our source code under the *KaraMIR* project website, where we are available for the MIR community feedback. As *KaraMIR* is a community project, researchers are more than welcome to join the project.

For future work, we plan to continue maintaining the project, the dataset, and to work with Recisio in including new songs and metadata. The dataset containing tens of thousands of tracks would be more suitable for the emerging deep learning techniques. We are also motivated by the idea of combining audio fingerprinting approaches with CSI approaches in order to develop fast and precise retrieval algorithms for cover songs that are resembling the originals. The new approaches should later be tested not only for karaoke cover songs, but also for other specific cover versions, e.g., live performances from the original artist, or, multiple recordings of the same classical music piece. We will, therefore, guide the *KaraMIR* project toward breaking down the difficult CSI task to meaningful subtasks, with the aim of developing new-generation music discovery applications.

Conclusion and future work

In this thesis, our topic was to analyze harmonic progressions in musical pieces, using a custom model, extract relevant features, and use them for a cover song identification task.

This was a very broad task, given that only a custom model based on music theory proved itself to be challenging enough so that the whole books exist on a topic (Lerdahl [2001], Chew [2014]). We took a straightforward way of creating a model based on tonal harmony, computational complexity, and formal grammars. Even though these concepts are hardly ever mentioned together, they all fit well in our final TSD distance model. Along the way, we also defined and extracted a new feature, that was spoken of in MIR works but never formally defined: a *harmonic complexity*. As our comparison to related works as well as our case studies, the model was chosen well and the proposed feature has the potential to be discriminative for MIR tasks, including music classification.

The second part of the thesis was dedicated to the review of harmonic features and our MIR task of choice: cover song identification. Knowing that the state-of-the-art has not been challenged for almost 10 years, we have chosen to contribute in multiple ways, in which the topic of CSI seems to need some fresh ideas.

- We have compared the features to get more insight into which features are suitable for CSI.
- We have compared DTW scores to show which one performs the best.
- We have tested harmony fingerprints to see when they are meaningful to use to save space or time for the CSI experiments.
- We have deployed a new dataset for CSI and singing voice analysis tasks.
- We have launched an open-source *harmony-analyser* application along with *KaraMIR* project for improvement of the current MIR tasks.

By doing so, we hope that this thesis has filled the gap for the researchers, when they are sometimes left with no other choice than comparing, testing, or creating the datasets themselves.

Our results can be interpreted and used in multiple ways, but we consider the following as the main achievements:

1. Developing a music harmony model useful for direct comparison of the songs, and putting this model in the context of other harmony models
2. Defining a harmonic complexity, a concept not clearly defined before although referenced frequently, to be used for various musicology or music information retrieval tasks

3. The contribution of surveying all common harmony features, and comparing them in a way that was not yet done before (direct application of dynamic time warping, as well as machine learning and statistical approaches), including the newly formed features.

4. High-accuracy results for cover song identification task specializing in karaoke songs, which also displays arguable size and performance complexity savings

5. Publishing the application *harmony-analyser* for the music analysis and the *Kara1k* dataset for cover song identification.

The thesis also had a goal to introduce MIR and harmony analysis research to any reader who had not yet experienced these interesting interdisciplinary challenges. As we provided everything from the basic definitions to complex models, attachments describing the background as well as the guide to experiments, we deem this task fulfilled.

Future work

While working on the thesis, we have also left several doors open for future work.

First of all, we are interested in trying the musicological distances in some other way than the one that was described. Instead of creating a time series of one song, we could look at the comparison of chords or chroma vectors from two songs. A DTW-based algorithm would possibly benefit if we swapped the Euclidean or Manhattan distance with one of our proposed musicological distances. We only had a chance to try this on a small portion of a dataset, for chroma features. We swapped Euclidean distance for ChromaCD in the DTW computation, and our early studies show that the final projected accuracies were close, but Euclidean distance still had a small edge. We are therefore motivated to review this idea with more experiments in the future work.

We also are aware of the massive popularity that deep learning methods are encountering. We believe that deep learning methods are the future lead also for the CSI task, although a breakthrough did not happen yet for this task compared to some other tasks, e.g., the visual recognition challenge. This is why we have commenced to include deep learning methods into our *KaraMIR* project and we attribute a large part of the future work to go along this direction.

As the whole thesis was combining multiple research fields, we hope that in the end, these were adequately referenced and the common ground was explained without lacking the depth. Consequently, we hope that some of our work (model, features, or dataset) is picked by future research, whether it is in artificial intelligence, musicology, or any other discipline. The interdisciplinary fields are notoriously more difficult to understand, but we also think that their popularity will only increase in the future, as some really exciting applications remain to be seen. That is especially true for music and computing.

Bibliography

M. A. Bartsch and G. H. Wakefield. To Catch a Chorus: Using Chroma-Based Representations for Audio Thumbnailing. In *Workshop on Applications of Signal Processing to Audio and Acoustics*, WASPAA 2001. IEEE, 2001. URL https://ieeexplore.ieee.org/document/969531.

Y. Bayle, L. Maršík, M. Rusek, P. Hanna, K. Slaninová, J. Martinovič, and J. Pokorný. Kara1k: a karaoke dataset for cover song identification and singing voice analysis. In *2017 IEEE International Symposium on Multimedia*, ISM 2017. IEEE, 2017. URL http://ieeexplore.ieee.org/document/8241597/.

Yann Bayle, Pierre Hanna, and Matthias Robine. Classification à grande échelle de morceaux de musique en fonction de la présence de chant. In *Journées d'Informatique Musicale*, 2016. URL https://hal.archives-ouvertes.fr/hal-01484201.

Juan Pablo Bello. Audio-Based Cover Song Retrieval Using Approximate Chord Sequences: Testing Shifts, Gaps, Swaps and Beats. In *Proceedings of the 8th International Society for Music Information Retrieval Conference*, ISMIR 2007, 2007. URL https://pdfs.semanticscholar.org/75ae/d0cb8103f4e2e5127df324ec5b6fdd6b9924.pdf.

Tony Bergstrom, Karrie Karahalios, and John C. Hart. Isochords: Visualizing Structure in Music. In *Proceedings of Graphics Interface*, GI '07. ACM, 2007. URL http://www.graphicsinterface.org/proceedings/2007.

Thierry Bertin-Mahieux and Daniel P. W. Ellis. Large-Scale Cover Song Recognition Using Hashed Chroma Landmarks. In *IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics*, WASPAA 2011. IEEE, 2011. URL https://ieeexplore.ieee.org/document/6082307.

Thierry Bertin-Mahieux and Daniel P. W. Ellis. Large-Scale Cover Song Recognition Using the 2D Fourier Transform Magnitude. In *Proceedings of the 13th International Society for Music Information Retrieval Conference*, ISMIR 2012, 2012. URL https://academiccommons.columbia.edu/doi/10.7916/D8Z60ZBV.

Thierry Bertin-Mahieux, Douglas Eck, and Michael I. Mandel. Automatic Tagging of Audio: The State-of-the-Art. In *Machine Audition: Principles, Algorithms and Systems*. IGI Publishing, 2010. URL http://tbertinmahieux.com/Papers/machineaudition10.pdf.

Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The Million Song Dataset. In *Proceedings of the 12th International Society for Music Information Retrieval Conference*, ISMIR 2011, 2011. URL http://ismir2011.ismir.net/papers/OS6-1.pdf.

R. M. Bittner, J. Salamon, M. Tierney, M. Mauch, C. Cannam, and J. Bello. MedleyDB: A Multitrack Dataset for Annotation-Intensive MIR Research. In *Proceedings of the 15th International Society for Music Information Retrieval Conference*, ISMIR 2014, 2014. URL https://www.semanticscholar.org/paper/MedleyDB%3A-A-Multitrack-Dataset-for-MIR-Research-Bittner-Salamon/e0ea1bb742b4958f5c84ece964ac9e3247d44015.

D Bogdanov, J Serrà, and N Wack. Unifying low-level and high-level music similarity measures. *IEEE Transactions on Multimedia*, 13(4):687–701, 2011. URL https://ieeexplore.ieee.org/document/5728926.

William E. Caplin. *Analyzing Classical Form: An Approach for the Classroom*. Oxford University Press, 2013. URL http://www.music.mcgill.ca/acf/.

M. A. Casey, R. Veltkamp, M. Goto, M. Leman, C. Rhodes, and M. Slaney. Content-based music information retrieval: Current directions and future challenges. *Proceedings of the IEEE*, 96(4):668–696, 2008. URL https://ieeexplore.ieee.org/document/4472077.

Tak-Shing Chan, Tzu-Chun Yeh, Zhe-Cheng Fan, Hung-Wei Chen, Li Su, Yi-Hsuan Yang, and Roger Jang. Vocal activity informed singing voice separation with the iKala dataset. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, ICASSP 2015. IEEE, 2015. URL https://ieeexplore.ieee.org/document/7178063.

Kaichun K. Chang, Jyh-Shing Roger Jang, and Costas S. Iliopoulos. Music Genre Classification via Compressive Sampling. In *Proceedings of the 11th International Society for Music Information Retrieval Conference*, ISMIR 2010, 2010. URL http://ismir2010.ismir.net/proceedings/ismir2010-66.pdf.

Sungkyun Chang, Juheon Lee, Sang Keun Choe, and Kyogu Lee. Audio Cover Song Identification using Convolutional Neural Network. In *Workshop on ML4Audio: Machine Learning for Audio Signal Processing at NIPS 2017*, 2017. URL https://arxiv.org/abs/1712.00166.

Ning Chen. Cover Song Identification Based on Similarity Fusion. In *Music Information Retrieval Evaluation eXchange*, MIREX 2016, 2016. URL https://www.music-ir.org/mirex/abstracts/2016/CL1.pdf.

Ning Chen, Mingyu Li, and Haidong Xiao. Two-layer similarity fusion model for cover song identification. *EURASIP Journal on Audio, Speech and Music Processing*, 2017(12):1–15, 2017. URL https://link.springer.com/article/10.1186/s13636-017-0108-2.

Elaine Chew. *Towards a Mathematical Model of Tonality*. PhD thesis, Massachusetts Institute of Technology, 2000. URL https://dspace.mit.edu/handle/1721.1/9139.

Elaine Chew. *Mathematical and Computational Modeling of Tonality*. Springer US, 2014. URL https://www.springer.com/la/book/9781461494744.

Albin Andrew Correya, Romain Hennequin, and Mickaël Arcos. Large-Scale Cover Song Detection in Digital Music Libraries Using Metadata, Lyrics and Audio Features. In *arXiv.org*, volume 1808.10351. Cornell University Library, 2018. URL https://arxiv.org/abs/1808.10351.

S. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(4):357–366, 1980. URL https://ieeexplore.ieee.org/document/1163420.

W. Bas De Haas. *Music information retrieval based on tonal harmony*. PhD thesis, Utrecht University, 2012. URL https://dspace.library.uu.nl/handle/1874/226713.

W. Bas De Haas, Remco C. Veltkamp, and Frans Wiering. Tonal Pitch Step Distance: A Similarity Measure for Chord Progressions. In *Proceedings of the 9th International Society for Music Information Retrieval Conference*, ISMIR 2008, 2008. URL https://dspace.library.uu.nl/bitstream/handle/1874/32376/TPSD-ISMIR-2008.pdf.

W. Bas De Haas, José Pedro Magalhães, and Frans Wiering. Improving Audio Chord Transcription by Exploiting Harmonic and Metric Knowledge. In *Proceedings of the 13th International Society for Music Information Retrieval Conference*, ISMIR 2012, 2012. URL http://ismir2012.ismir.net/event/papers/295_ISMIR_2012.pdf.

W. Bas De Haas, Anja Volk, and Frans Wiering. Structural segmentation of music based on repeated harmonies. In *2013 IEEE International Symposium on Multimedia*, ISM 2013. IEEE, 2013a. URL https://ieeexplore.ieee.org/document/6746800/.

W. Bas De Haas, Frans Wiering, and Remco C. Veltkamp. A geometrical distance measure for determining the similarity of musical harmony. *International Journal of Multimedia Information Retrieval*, 2(3):189–202, 2013b. URL https://dspace.library.uu.nl/handle/1874/303073.

J. Stephen Downie, Kris West, Andreas F. Ehmann, and Emmanuel Vincent. The 2005 Music Information retrieval Evaluation Exchange (MIREX 2005): Preliminary Overview. In *Proceedings of the 6th International Society for Music Information Retrieval Conference*, ISMIR 2005, 2005. URL http://recherche.ircam.fr/anasyn/vincent/downie_ISMIR05.pdf.

J. Stephen Downie, Mert Bay, Andreas F. Ehmann, and M. Cameron Jones. Audio Cover Song Identification: MIREX 2006-2007 Results and Analyses. In *Proceedings of the 7th International Society for Music Information Retrieval Conference*, ISMIR 2008, 2008. URL https://ismir2008.ismir.net/papers/ISMIR2008_265.pdf.

Alexey Egorov and Gene Linetsky. Cover Song Identification with IF-F0 Pitch Class Profiles. In *Music Information Retrieval Evaluation eXchange*, MIREX, 2008. URL https://www.music-ir.org/mirex/abstracts/2008/cbms_cover_song_id.pdf.

Daniel P. W. Ellis. Identifying 'Cover Songs' with Beat-Synchronous Chroma Features. In *Music Information Retrieval Evaluation eXchange*, MIREX 2006, 2006. URL https://www.ee.columbia.edu/~dpwe/pubs/Ellis06-coversongs.pdf.

Daniel P. W. Ellis and Courtenay V. Cotton. The 2007 LabROSA Cover Song Detection System. In *Music Information Retrieval Evaluation eXchange*, MIREX 2007, 2007. URL https://www.ee.columbia.edu/~dpwe/pubs/EllisC07-covers.pdf.

Daniel P. W. Ellis and Graham E. Poliner. Identifying 'Cover Songs' with Chroma Features and Dynamic Programming Beat Tracking. In *Proceedings of the IEEE International Conference on Acoustics*, ICASSP 2007. IEEE, 2007. URL https://ieeexplore.ieee.org/document/4218379/.

Leonhard Euler. *Tentamen Novae Theoriae Musicae*. Saint Petersburg Academy, 1739. URL http://eulerarchive.maa.org/pages/E033.html.

Arthur Flexer. Improving visualization of high-dimensional music similarity spaces. In *Proceedings of the 16th International Society for Music Information Retrieval Conference*, ISMIR 2015, 2015. URL http://ismir2015.uma.es/proceedings.html.

Jonathan Foote. Content-based retrieval of music and audio. In *Multimedia Storage and Archiving Systems II, Proceedings Volume 3229*, 1997. URL https://www.spiedigitallibrary.org/conference-proceedings-of-spie/3229/1/Content-based-retrieval-of-music-and-audio/10.1117/12.290336.short.

Peter Foster, Simon Dixon, and Anssi Klapuri. Identifying Cover Songs Using Information-theoretic Measures of Similarity. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(6):993–1005, 2015. URL https://dl.acm.org/citation.cfm?id=2876420.

Takuya Fujishima. Realtime Chord Recognition of Musical Sound: A System Using Common Lisp Music. In *Proceedings of the International Computer Music Conference*, ICMC 1999, 1999. URL http://quod.lib.umich.edu/i/icmc/bbp2372.1999.446.

J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, C. Moore, M. Plakal, and M. Ritter. Audio Set: An ontology and human-labeled dataset for audio events. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, ICASSP 2017. IEEE, 2017. URL https://ieeexplore.ieee.org/document/7952261.

Arijit Ghosal, Rudrasis Chakraborty, Bibhas Chandra Dhara, and Sanjoy Kumar Saha. A hierarchical approach for speech-instrumental-song classification. *SpringerPlus*, 2(526):1–11, 2013. URL https://www.ncbi.nlm.nih.gov/pubmed/25694856.

E. Gómez. *Tonal Description of Music Audio Signals*. PhD thesis, Universitat Pompeu Fabra, 2006. URL http://mtg.upf.edu/node/472.

Fabien Gouyon, Bob L. Sturm, Joao Lobato Oliveira, Nuno Hespanhol, and Thibault Langlois. On Evaluation Validity in Music Autotagging. In *arXiv.org*, volume 1410.0001. Cornell University Library, 2014. URL https://arxiv.org/abs/1410.0001.

Delaney Granizo-Mackenzie and Jason H. Moore. Multiple Threshold Spatially Uniform ReliefF for the Genetic Analysis of Complex Human Diseases. In *Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics*, pages 1–10, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-37189-9. URL https://link.springer.com/chapter/10.1007/978-3-642-37189-9_1.

Casey S. Greene, Daniel S. Himmelstein, Jeff Kiralis, and Jason H. Moore. The Informative Extremes: Using Both Nearest and Farthest Individuals Can Improve Relief Algorithms in the Domain of Human Genetics. In *Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics*, pages 182–193, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN 978-3-642-12211-8. URL https://link.springer.com/chapter/10.1007/978-3-642-12211-8_16.

John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Boston, second edition, 2001. URL https://dl.acm.org/citation.cfm?id=1177300.

David A. Hounshell. Elisha Gray and the Telephone: On the Disadvantages of Being an Expert. *Technology and Culture*, 16(2):133–161, 1975. ISSN 0040165X, 10973729. URL http://www.jstor.org/stable/3103488.

Chao-Ling Hsu and Jyh-Shing Roger Jang. On the improvement of singing voice separation for monaural recordings using the MIR-1K dataset. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 18(2):310–319, 2010. URL https://ieeexplore.ieee.org/document/5153305.

Eric J. Humphrey, Oriol Nieto, and Juan Pablo Bello. Data Driven and Discriminative Projections for Large-Scale Cover Song Identification. In *Proceedings of the 14th International Society for Music Information Retrieval Conference*, ISMIR 2013, 2013. URL https://www.semanticscholar.org/paper/98e0b6562ce77918b323e3ed061abc394f423307.

Nanzhu Jiang, Peter Grosche, Verena Konz, and Meinard Müller. Analyzing Chroma Feature Types for Automated Chord Recognition. In *Audio Engineering Society 42nd International Conference: Semantic Audio*, 2011. URL http://www.aes.org/e-lib/browse.cfm?elib=15943.

Maksim Khadkevich and Maurizio Omologo. Large-Scale Cover Song Identification Using Chord Profiles. In *Proceedings of the 14th International Society for Music Information Retrieval Conference*, ISMIR 2013, 2013. URL http://www.ppgia.pucpr.br/ismir2013.

Kenji Kira and Larry A. Rendell. A Practical Approach to Feature Selection. In *Proceedings of the Ninth International Workshop on Machine Learning*, ML 1992, pages 249–256, San Francisco, CA, USA, 1992. Morgan Kaufmann Publishers Inc. ISBN 1-5586-247-X. URL http://dl.acm.org/citation.cfm?id=141975.142034.

Donald E. Knuth. The Complexity of Songs. *SIGACT News*, 9(2):344–346, 1977. URL http://doi.acm.org/10.1145/1008354.1008355.

Tomáš Kocyan. *Adapting Case-Based Reasoning for Processing Natural Phenomena Data*. PhD thesis, VŠB – Technical University of Ostrava, 2015. URL https://dspace.vsb.cz/handle/10084/111489.

Igor Kononenko. Estimating Attributes: Analysis and Extensions of RELIEF. In *European Conference on Machine Learning*, ECML 1994, pages 171–182. Springer Verlag, 1994. URL https://link.springer.com/chapter/10.1007/3-540-57868-4_57.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS 12, 2012. URL https://dl.acm.org/citation.cfm?id=2999257.

Nadine Kroher, José-Miguel Díaz-Báñez, Joaquin Mora, and Emilia Gómez. Corpus COFLA: A research corpus for the Computational study of Flamenco Music. *J. Comp. Cultural Heritage*, 9(2):1–24, 2015. URL https://dl.acm.org/citation.cfm?id=2875428.

Carol L. Krumhansl. *Cognitive Foundations of Musical Pitch*. Oxford University Press, New York, NY, US, 1990. URL https://www.oxfordscholarship.com/view/10.1093/acprof:oso/9780195148367.001.0001/acprof-9780195148367.

Carol L. Krumhansl. The Cognition of Tonality – As We Know It Today. *Journal of New Music Research*, 33(3):253–268, 2004. URL http://dx.doi.org/10.1080/0929821042000317831.

Carol L. Krumhansl. The Geometry of Musical Structure: A Brief Introduction and History. *Computers in Entertainment*, 3(4):1–14, 2005. URL http://doi.acm.org/10.1145/1095534.1095542.

Carol L. Krumhansl and Edward J. Kessler. Tracing the Dynamic Changes in Perceived Tonal Organization in a Spatial Representation of Musical Keys. *Psychological Review*, 89(4):334–368, 1982. URL https://www.ncbi.nlm.nih.gov/pubmed/7134332.

Jozef Laborecký. *Hudobný terminologický slovník*. Slovenské pedagogické nakladateľstvo, 1997. URL https://www.cbdb.cz/kniha-171771-hudobny-terminologicky-slovnik.

Juheon Lee, Sungkyun Chang, Sang Keun Choe, and Kyogu Lee. Cover Song Identification Using Song-to-Song Cross-Similarity Matrix with Convolutional Neural Network. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, ICASSP 2015. IEEE, 2018a. URL https://ieeexplore.ieee.org/document/8461395.

Juheon Lee, Sungkyun Chang, Donmoon Lee, and Kyogu Lee. CoverNet: Cover Song Identification Using Cross-Similarity Matrix with Convolutional Neural Network. In *Music Information Retrieval Evaluation eXchange*, MIREX 2018, 2018b. URL https://www.music-ir.org/mirex/abstracts/2018/LCLL1.pdf.

Kyogu Lee. Identifying Cover Songs from Audio Using Harmonic Representation. In *Music Information Retrieval Evaluation eXchange*, MIREX 2006, 2006. URL https://pdfs.semanticscholar.org/3558/802c4973d2a4f7132a8318482e6e38dcf298.pdf.

Bernhard Lehner, Gerhard Widmer, and Reinhard Sonnleitner. On the reduction of false positives in singing voice detection. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, ICASSP 2014. IEEE, 2014. ISBN 9781479928927. URL https://ieeexplore.ieee.org/abstract/document/6855054.

Bernhard Lehner, Gerhard Widmer, and Sebastian Böck. A low-latency, real-time-capable singing voice detection method with LSTM recurrent neural networks. In *23rd European Signal Processing Conference*, EUSIPCO 2015. EURASIP, 2015. URL https://www.eurasip.org/Proceedings/Eusipco/Eusipco2015/papers/1570097385.pdf.

Fred Lerdahl. *Tonal Pitch Space*. Oxford University Press, Oxford, UK, 2001. URL https://www.oxfordscholarship.com/view/10.1093/acprof:oso/9780195178296.001.0001/acprof-9780195178296.

Fred Lerdahl and Carol L. Krumhansl. Modeling Tonal Tension. *Music Perception: An Interdisciplinary Journal*, 24(4):329–366, 2007. URL http://www.jstor.org/stable/10.1525/mp.2007.24.issue-4.

Vladimir I. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions, and Reversals. *Soviet Physics-Doklady*, 10(8):707–710, 1966. URL https://nymity.ch/sybilhunting/pdf/Levenshtein1966a.pdf.

David Lewin. A Formal Theory of Generalized Tonal Functions. *Journal of Music Theory*, 26(1):23–60, 1982. URL http://www.jstor.org/stable/843354.

David Lewin. *Generalized Musical Intervals and Transformations*. Yale University Press, New Haven, CT, USA, 1987. URL https://www.oxfordscholarship.com/view/10.1093/acprof:oso/9780195317138.001.0001/acprof-9780195317138.

R. J. Lewis, B. Fields, and T. Crawford. Addressing the Music Information Needs of Musicologists. In *Proceedings of the 16th International Society for Music Information Retrieval Conference*, ISMIR 2015, 2015. URL https://tm.web.ox.ac.uk/blog/addressing-the-music-information-needs-of-musicologists.

A Liutkus, D Fitzgerald, Z Rafii, B Pardo, and L Daudet. Kernel Additive Models for Source Separation. *IEEE Transactions on Signal Processing*, 62 (16):4298–4310, 2014. URL https://ieeexplore.ieee.org/abstract/document/6842708.

Beth Logan. Mel Frequency Cepstral Coefficients for Music Modeling. In *Proceedings of the 1st International Society for Music Information Retrieval Conference*, ISMIR 2000, 2000. URL http://musicweb.ucsd.edu/~sdubnov/CATbox/Reader/logan00mel.pdf.

Christopher Longuet-Higgins. Letter to a musical friend. *The Music Review*, 23(3):244–248, 1962. URL https://openmusiclibrary.org/article/221869/.

L. Maršík, M. Rusek, K. Slaninová, J. Martinovič, and J. Pokorný. Evaluation of Chord and Chroma Features and Dynamic Time Warping Scores on Cover Song Identification Task. In *Proceedings of the 16th IFIP International Conference on Computer Information Systems and Industrial Management*. Springer, 2017. URL https://link.springer.com/chapter/10.1007/978-3-319-59105-6_18.

L. Maršík, P. Martišek, J. Pokorný, M. Rusek, K. Slaninová, J. Martinovič, M. Robine, P. Hanna, and Y. Bayle. KaraMIR: A project for cover song identification and singing voice analysis using a karaoke songs dataset. *International Journal of Semantic Computing*, 12(4):501–522, 2018. URL https://www.worldscientific.com/doi/abs/10.1142/S1793351X18400202.

Ladislav Maršík. Music Harmony Analysis: Towards a Harmonic Complexity of Musical Pieces. Master's thesis, Comenius University in Bratislava, 2013. URL http://www.dcs.fmph.uniba.sk/diplomovky/obhajene/getfile.php/Music_Harmony_Analysis-Towards_a_Harmonic_Complexity_of_Musical_Pieces.pdf?id=366&fid=649&type=application/pdf.

Ladislav Maršík. harmony-analyser.org - Java Library and Tools for Chordal Analysis. In *Proceedings of 2016 Joint WOCMAT-IRCAM Forum Conference*, WOCMAT 2016. Kainan University, 2016. URL http://www.harmony-analyser.org/download/paper-wocmat2016-harmony-analyser.pdf.

Ladislav Maršík. Student Research Abstract: Using Chord Distance Descriptors to Enhance Music Information Retrieval. In *Proceedings of the 32nd Annual ACM Symposium on Applied Computing*, SAC 2017. ACM, 2017. URL https://dl.acm.org/citation.cfm?id=3019939.

Ladislav Maršík, Jaroslav Pokorný, and Martin Ilčík. Towards a Harmonic Complexity of Musical Pieces. In *Proceedings of the Dateso 2014 Annual International Workshop on Databases, Texts, Specifications and Objects*, volume 1139 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2014a. URL http://ceur-ws.org/Vol-1139.

Ladislav Maršík, Jaroslav Pokorný, and Martin Ilčík. Improving Music Classification Using Harmonic Complexity. In *Procedings of the 14th Conference Information Technologies - Applications and Theory - Workshops and Posters*, ITAT 2014. Ústav informatiky AV ČR, 2014b. URL http://artax.karlin.mff.cuni.cz/~bajel3am/itat2014/local/13_Marsik.pdf.

B. Mathieu, S. Essid, T. Fillon, J. Prado, and R. Gaël. YAAFE, an Easy to Use and Efficient Audio Feature Extraction Software. In *Proceedings of the 11th International Society for Music Information Retrieval Conference*, ISMIR 2010, 2010. URL http://www.quaero.org/media/files/bibliographie/yaafe_ismir2010_mathieu.pdf.

Benoit Mathieu. Outils informatiques d'analyse musicale. Master's thesis, ENST Bretagne, 2002. URL http://recherche.ircam.fr/equipes/repmus/Rapports/mathieu2002/outils-analyse-BM-2002.pdf.

Mathias Mauch and Simon Dixon. Approximate Note Transcription for the Improved Identification of Difficult Chords. In *Proceedings of the 11th International Society for Music Information Retrieval Conference*, ISMIR 2010, 2010. URL ismir2010.ismir.net/program/program-ismir-2010.

Matthias Mauch and Mark Levy. Structural Change on Multiple Time Scales as a Correlate of Musical Complexity. In *Proceedings of the 12th International Society for Music Information Retrieval Conference*, ISMIR 2011, 2011. URL https://pdfs.semanticscholar.org/c590/17230da17dba92f5b804e6a4a890cf91ebd9.pdf.

Matt McVicar, Raúl Santos-Rodriguez, Yizhao Ni, and Tijl De Bie. Automatic Chord Estimation from Audio: A Review of the State of the Art. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(2):556–575, 2014. URL https://ieeexplore.ieee.org/document/6705583.

David Moffat, David Ronan, and Joshua D Reiss. An evaluation of audio feature extraction toolboxes. In *Proceedings of the 18th International Conference on Digital Audio Effects*, DAFx 2015, 2015. URL https://qmro.qmul.ac.uk/xmlui/bitstream/handle/123456789/13075/Moffatt%20AN%20EVALUATION%20OF%20AUDIO%20FEATURE%202015%20Published.pdf?sequence=2.

Abdullah Mueen and Eamonn J. Keogh. Extracting Optimal Performance from Dynamic Time Warping. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD 2016. ACM, 2016. URL https://dl.acm.org/citation.cfm?id=2945383.

Meinard Müller. *Information Retrieval for Music and Motion*. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-74047-6. URL https://www.springer.com/gp/book/9783540740476.

Meinard Müller and Sebastian Ewert. Towards Timbre-Invariant Audio Features for Harmony-Based Music. *IEEE Transactions on Audio, Speech and Language Processing*, 18(3):649–662, 2010. ISSN 1558-7916. URL https://ieeexplore.ieee.org/document/5410051.

Alexander Móži. *Študijné texty k dejinám hudby 1*. Academy of Performing Arts, Bratislava, Slovakia, 1994. URL https://sclib.svkk.sk/sck01/Record/000047648.

Katy Noland and Mark Sandler. Signal Processing Parameters for Tonality Estimation. In *Audio Engineering Society Convention 122*, 2007. URL http://www.aes.org/e-lib/browse.cfm?elib=14140.

J. Osmalskyj, S. Piérard, M. Van Droogenbroeck, and J.-J. Embrechts. Efficient Database Pruning for Large-Scale Cover Song Recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, ICASSP 2013. IEEE, 2013. URL https://ieeexplore.ieee.org/document/6637741.

Julien Osmalskyj. *A Combining Approach To Cover Song Identification*. PhD thesis, University of Liege, 2017. URL https://orbi.uliege.be/bitstream/2268/214942/1/phd-thesis-osmalskyj.pdf.

Jean-François Paiement, Douglas Eck, and Samy Bengio. A Probabilistic Model for Chord Progressions. In *Proceedings of the 6th International Society for Music Information Retrieval Conference*, ISMIR 2005, 2005. URL http://publications.idiap.ch/index.php/publications/show/344.

Yannis Panagakis and Constantine Kotropoulos. Music Classification by Low-Rank Semantic Mappings. *EURASIP Journal on Audio, Speech and Music Processing*, 2013(13):1–15, 2013. URL http://dx.doi.org/10.1186/1687-4722-2013-13.

Steffen Pauws. Musical Key Extraction from Audio. In *Proceedings of the 5th International Society for Music Information Retrieval Conference*, ISMIR 2004, 2004. URL http://www.ee.columbia.edu/~dpwe/ismir2004/CRFILES/paper142.pdf.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12(1):2825–2830, 2011. URL http://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf.

Juraj Pospíšil. *Hudobná teória pre konzervatória: Učebnica pre 1. ročník konzervatórií*, volume 1. Slovenské pedagogické nakladateľstvo, Bratislava, Slovakia, 1985. URL https://sclib.svkk.sk/sck01/Record/000220498/Description.

Hendrik Purwins, Benjamin Blankertz, and Klaus Obermayer. A New Method for Tracking Modulations in Tonal Music in Audio Data Format. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks*, IJCCN 2000. IEEE, 2000. URL https://ieeexplore.ieee.org/document/859408.

Mathieu Ramona, Gaël Richard, and Bertrand David. Vocal detection in music with support vector machines. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, ICASSP 2008. IEEE, 2008. URL https://ieeexplore.ieee.org/document/4518002.

Andreas Rauber, Elias Pampalk, and Dieter Merkl. Using Psycho-Acoustic Models and Self-Organizing Maps to Create a Hierarchical Structuring of Music by Musical Styles. In *Proceedings of the 3rd International Society for Music Information Retrieval Conference*, ISMIR 2002, 2002. URL http://ismir2002.ismir.net/proceedings/02-FP02-4.pdf.

Suman Ravuri and Daniel P. W. Ellis. The Hydra System of Unstructured Cover Song Detection. In *Music Information Retrieval Evaluation eXchange*, MIREX 2009, 2009. URL https://www.music-ir.org/mirex/abstracts/2009/RE.pdf.

Ana Rebelo, Ichiro Fujinaga, Filipe Paszkiewicz, Andre R. S. Marcal, Carlos Guedes, and Jaime S. Cardoso. Optical music recognition: state-of-the-art and open issues. *International Journal of Multimedia Information Retrieval*, 1(3):173–190, 2012. URL https://link.springer.com/article/10.1007/s13735-012-0004-6.

Hugo Riemann. *Harmony Simplified*. Augener Ltd., London, UK, 1896a. URL https://archive.org/details/cu31924022305357.

Hugo Riemann. *Dictionary of Music*. Augener Ltd., London, UK, 1896b. URL https://archive.org/details/dictionaryofmusi00riem/page/n6.

Matthias Robine, Pierre Hanna, Pascal Ferraro, and Julien Allali. Adaptation of String Matching Algorithms for Identification of Near-Duplicate Music Documents. In *Proceedings of the International SIGIR Workshop on Plagiarism Analysis, Authorship Identification, and Near-Duplicate Detection*, SIGIR-PAN 2007, 2007. URL https://pan.webis.de/downloads/publications/papers/robine_2007.pdf.

Matthias Robine, Pierre Hanna, Thomas Rocher, and Pascal Ferraro. Structured Representation of Harmony for Music Retrieval. In *Proceedings of the International Computer Music Conference*, ICMC 2009, 2009. URL https://www.semanticscholar.org/paper/Structured-Representation-of-Harmony-for-Music-Robine-Hanna/c08f6052a67f59128273d859aa27bdc609c7b699.

Thomas Rocher. *Analyse et modélisation des informations tonales dans la musique occidentale*. PhD thesis, Université Bordeaux 1, 2011. URL http://www.theses.fr/2011BOR14406.

Thomas Rocher, Matthias Robine, Pierre Hanna, and Myriam Desainte-Catherine. A Survey of Chord Distances With Comparison For Chord Analysis. In *Proceedings of the International Computer Music Conference*, ICMC 2010, 2010a. URL quod.lib.umich.edu/i/icmc/bbp2372.2010.036/.

Thomas Rocher, Matthias Robine, Pierre Hanna, and Laurent Oudre. Concurrent Estimation of Chords and Keys from Audio. In *Proceedings of the 11th International Society for Music Information Retrieval Conference*, ISMIR 2010, 2010b. URL https://pdfs.semanticscholar.org/d2c1/5308f6c1926471159449cd5b5c0aa49825da.pdf.

Craig S. Sapp. Visual Hierarchical Key Analysis. *Computers in Entertainment*, 3(4):1–19, 2005. URL http://doi.acm.org/10.1145/1095534.1095544.

B Schuller, C Kozielski, F Weninger, F Eyben, and G Rigoll. Vocalist Gender Recognition in Recorded Popular Music. In *Proceedings of the 11th International Society for Music Information Retrieval Conference*, ISMIR 2010, 2010. URL https://www.semanticscholar.org/paper/Vocalist-Gender-Recognition-in-Recorded-Popular-Schuller-.Kozielski/8f26c9eefdb9967b3a43987cd83698f3c1099f20.

Arnold Schönberg. *Theory of Harmony*. University of California Press, Los Angeles, CA, USA, 1922. URL https://www.ucpress.edu/book/9780520266087/theory-of-harmony.

Dirk Schönfuss. Content-Based Music Discovery. In *Exploring Music Contents*, volume 6684 of *Lecture Notes in Computer Science*. Springer, 2011. URL http://dx.doi.org/10.1007/978-3-642-23126-1_22.

Joan Serrà. *Identification of Versions of the Same Musical Composition by Processing Audio Descriptions*. PhD thesis, Universitat Pompeu Fabra, 2011. URL http://mtg.upf.edu/node/1951.

Joan Serrà, E. Gómez, Perfecto Herrera, and Xavier Serra. Chroma Binary Similarity and Local Alignment Applied to Cover Song Identification. *IEEE Transactions on Audio, Speech and Language Processing*, 16 (6):1138–1151, 2008. URL https://ieeexplore.ieee.org/document/4523006.

Joan Serrà, Xavier Serra, and Ralph G. Andrzejak. Cross Recurrence Quantification for Cover Song Identification. *New Journal of Physics*, 11 (9):1–20, 2009. URL https://iopscience.iop.org/article/10.1088/1367-2630/11/9/093017/meta.

Xi Shao, Changsheng Xu, and Mohan S. Kankanhalli. Unsupervised Classification of Music Genre Using Hidden Markov Model. In *Proceedings of 2004 IEEE International Conference on Multimedia and Expo*, ICME 2004, 2004. URL https://ieeexplore.ieee.org/document/1394661.

Roger Newland Shepard. Geometrical approximations to the structure of musical pitch. *Psychological review*, 89(4):305–333, 1982. URL https://psycnet.apa.org/record/1982-27250-001.

Sigurdur Sigurdsson, Kaare Petersen, and Tue Lehn-Schiøler. Mel Frequency Cepstral Coefficients: An Evaluation of Robustness of MP3 Encoded Music. In *Proceedings of the 7th International Society for Music Information Retrieval Conference*, ISMIR 2006, 2006. URL https://www.semanticscholar.org/paper/Mel-Frequency-Cepstral-Coefficients%3A-An-Evaluation-Sigur%C3%B0sson-Petersen/22f4747308bfdbf6ddd2e9f5d76e18232b7d8362.

Janto Skowronek, Martin F. McKinney, and Steven V. De Par. Ground truth for automatic music mood classification. In *Proceedings of the 7th International Society for Music Information Retrieval Conference*, ISMIR 2006, 2006. URL http://ismir2006.ismir.net/PAPERS/ISMIR06105_Paper.pdf.

Jeffrey Christopher Smith. *Correlation Analyses of Encoded Music Performance*. PhD thesis, Stanford University, 2013. URL https://ccrma.stanford.edu/damp/magicpiano/jeffsmiththesis.pdf.

Hagen Soltau, Tanja Schultz, Martin Westphal, and Alex Waibel. Recognition of Music Types. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, ICASSP 1998. IEEE, 1998. URL https://ieeexplore.ieee.org/abstract/document/675470.

Stanley Smith Stevens, John E. Volkmann, and E. B. Newman. A Scale for the Measurement of the Psychological Magnitude Pitch. *The Journal of the Acoustical Society of America*, 8(3):185–190, 1937. URL https://asa.scitation.org/doi/10.1121/1.1915893.

Bob L. Sturm. Faults in the Latin Music Database and with its Use. In *Proceedings of the 16th International Society for Music Information Retrieval Conference*, ISMIR 2015, 2015. URL http://ismir2015.uma.es/LBD/LBD7.pdf.

David Temperley. *The cognition of basic musical structures*. MIT Press, Cambridge, MA, USA, 2001. URL https://psycnet.apa.org/record/2001-10077-000.

Christopher J. Tralie. Cover Song Identification Using Similarity Fusion of HPCPs, MFCCs, and MFCC SSMs. In *Music Information Retrieval Evaluation eXchange*, MIREX 2017, 2017a. URL https://www.music-ir.org/mirex/abstracts/2017/CT2.pdf.

Christopher J. Tralie. Early MFCC and HPCP Fusion for Robust Cover Song Identification. In *Proceedings of the 18th International Society for Music Information Retrieval Conference*, ISMIR 2017, 2017b. URL https://ismir2017.smcnus.org/proceedings/.

Christopher J. Tralie and Paul Bendich. Cover Song Identification with Timbral Shape Sequences. In *Music Information Retrieval Evaluation eXchange*, MIREX 2015, 2015. URL http://ismir2015.uma.es/articles/277_Paper.pdf.

George Tzanetakis and Perry Cook. Musical Genre Classification of Audio Signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, 2002. URL https://ieeexplore.ieee.org/document/1021072.

Michail Vlachos, Marios Hadjieleftheriou, Dimitrios Gunopulos, and Eamonn Keogh. Indexing Multidimensional Time-Series. *The International Journal on Very Large Data Bases*, 15(1):1–20, 2006. URL https://link.springer.com/article/10.1007/s00778-004-0144-2.

Jaroslav Volek. *Struktura a osobnosti hudby*. Panton, Prague, Czech Republic, 1988. URL https://tritius.amu.cz/detail/1254713.

Anja Volk, Elaine Chew, Elizabeth Hellmuth Margulis, and Christina Anagnostopoulou. Music Similarity: Concepts, Cognition and Computation. *Journal of New Music Research*, 45(3):207–209, 2016. URL https://www.tandfonline.com/doi/full/10.1080/09298215.2016.1232412.

Piet G. Vos. Tonality Induction: Theoretical Problems and Dilemmas. *Music Perception: An Interdisciplinary Journal*, 17(4):403–416, 2000. URL http://mp.ucpress.edu/content/17/4/43.

Avery L. Wang. An Industrial-Strength Audio Search Algorithm. In *Proceedings of the 4th International Society for Music Information Retrieval Conference*, ISMIR 2003, 2003. URL https://www.ee.columbia.edu/~dpwe/papers/Wang03-shazam.pdf.

Bo Wang, Aziz M. Mezlini, Feyyaz Demir, Marc Fiume, Zhuowen Tu, Michael Brudno, Benjamin Haibe-Kains, and Anna Goldenberg. Similarity network fusion for aggregating data types on a genomic scale. *Nature Methods*, 11:333–337, 2014a. URL https://www.nature.com/articles/nmeth.2810.

Ju-Chiang Wang, Ming-Chi Yen, Yi-Hsuan Yang, and Hsin-Min Wang. Automatic Set List Identification and Song Segmentation for Full-Length Concert Videos. In *Proceedings of the 15th International Society for Music Information Retrieval Conference*, ISMIR 2014, 2014b. URL `http://mac.citi.sinica.edu.tw/~yang/pub/wang14ismir.pdf`.

Felix Weninger, Jean-Louis Durrieu, Florian Eyben, Gaël Richard, and Björn Schuller. Combining monaural source separation with long short-term memory for increased robustness in vocalist gender recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, ICASSP 2011. IEEE, 2011a. URL `https://ieeexplore.ieee.org/document/5946764`.

Felix Weninger, Martin Wöllmer, and Björn Schuller. Automatic assessment of singer traits in popular music: Gender, age, height and race. In *Proceedings of the 12th International Society for Music Information Retrieval Conference*, ISMIR 2011, 2011b. URL `https://pdfs.semanticscholar.org/910f/227c85bdc3fe78870a7391f6f8d78bd3d363.pdf`.

Changsheng Xu, Namunu C. Maddage, Xi Shao, Fang Cao, and Qi Tian. Musical Genre Classification Using Support Vector Machines. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, ICASSP 2003. IEEE, 2003. URL `https://ieeexplore.ieee.org/document/1199998`.

Cheng Yang. MACS: Music Audio Characteristic Sequence Indexing for Similarity Retrieval. In *IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics*, WASPAA 2001. IEEE, 2001a. URL `https://ieeexplore.ieee.org/document/969558`.

Cheng Yang. Music Database Retrieval Based on Spectral Similarity. In *Proceedings of the 2nd International Society for Music Information Retrieval Conference*, ISMIR 2001, 2001b. URL `http://ismir2001.ismir.net/posters/yang.pdf`.

Fan Yang and Ning Chen. Cover Song Identification Based on Cross Recurrence Plot and Local Alignment. *Journal of East China University of Science and Technology*, 42(2):247–253, 2016. URL `http://en.cnki.com.cn/Article_en/CJFDTotal-HLDX201602015.htm`.

Lei Yu and Huan Liu. Feature Selection for High-dimensional Data: A Fast Correlation-based Filter Solution. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML 2003, pages 856–863. AAAI Press, 2003. ISBN 1-57735-189-4. URL `http://dl.acm.org/citation.cfm?id=3041838.3041946`.

Damián Horacio Zanette. Music, Complexity, Information. In *arXiv.org*, volume 0807.0565. Cornell University Library, 2008a. URL `https://arxiv.org/abs/0807.0565`.

Damián Horacio Zanette. Playing by numbers. *Nature*, 453(19):988–989, 2008b. URL `https://www.nature.com/collections/tngvwkhycp`.

Pavel Zika and Miloslav Kořínek. *Tonálna harmónia pre 1.-3. ročník inštrumentálnych a speváckych oddelení konzervatória*. Slovenské pedagogické nakladateľstvo, Bratislava, Slovakia, 1990. URL `https://sclib.svkk.sk/sck01/Record/000256646`.

List of Figures

List of Tables

List of Abbreviations

ATC Average Transition Complexity

ChordCD Chord Complexity Distance

ChromaCD Chroma Complexity Difference

CNN Convolutional Neural Network

CSI Cover Song Identification

DCT Discrete Cosine Transform

DFT Discrete Fourier Transform

DTW Dynamic Time Warping

HPCP Harmonic Pitch Class Profiles

iDFT Inverse Discrete Fourier Transform

IRCAM Institut de Recherche et Coordination Acoustique/Musique

ISMIR International Society for Music Information Retrieval

LaBRI Laboratoire Bordelais de Recherche en Informatique

MAP Mean arithmetic of Average Precision

MAR Mean Average Rank

MFC Mel-Frequency Cepstrum

MFCC Mel-Frequency Cepstral Coefficients

MIDI Musical Instrument Digital Interface

MIR Music Information Retrieval

MIREX Music Information Retrieval Evaluation eXchange

NNLS Non-negative least squares

PCP Pitch Class Profiles

SGC Singer Gender Classification

SMC Sound and Music Computing

SNF Similarity Network Fusion

STFT Short-time Fourier Transform

SVD Singing Voice Detection

SVS Singing Voice Separation

THPCP Transposed Harmonic Pitch Class Profiles

TPS Tonal Pitch Space

TPSD Tonal Pitch Step Distance

TSD Tonic Subdominant Dominant

WAV Waveform Audio File Format

List of Publications

L. Maršík, P. Martišek, J. Pokorný, M. Rusek, K. Slaninová, J. Martinovič, M. Robine, P. Hanna, and Y. Bayle. KaraMIR: A project for cover song identification and singing voice analysis using a karaoke songs dataset. *International Journal of Semantic Computing*, 12(4):501–522, 2018. URL `https://www.worldscientific.com/doi/abs/10.1142/S1793351X18400202`.

Ladislav Maršík. Student Research Abstract: Using Chord Distance Descriptors to Enhance Music Information Retrieval. In *Proceedings of the 32nd Annual ACM Symposium on Applied Computing*, SAC 2017. ACM, 2017a. URL `https://dl.acm.org/citation.cfm?id=3019939`.

L. Maršík, M. Rusek, K. Slaninová, J. Martinovič, and J. Pokorný. Evaluation of Chord and Chroma Features and Dynamic Time Warping Scores on Cover Song Identification Task. In *Proceedings of the 16th IFIP International Conference on Computer Information Systems and Industrial Management*. Springer, 2017. URL `https://link.springer.com/chapter/10.1007/978-3-319-59105-6_18`.

Y. Bayle, L. Maršík, M. Rusek, P. Hanna, K. Slaninová, J. Martinovič, and J. Pokorný. Kara1k: a karaoke dataset for cover song identification and singing voice analysis. In *The 19th IEEE International Symposium on Multimedia*, ISM 2017. IEEE, 2017. URL `http://ieeexplore.ieee.org/document/8241597/`.

Ladislav Maršík, Jaroslav Pokorný, and Martin Ilčík. Towards a Harmonic Complexity of Musical Pieces. In *Proceedings of the Dateso 2014 Annual International Workshop on Databases, Texts, Specifications and Objects*, volume 1139 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2014a. URL `http://ceur-ws.org/Vol-1139`.

Ladislav Maršík. harmony-analyser.org - Java Library and Tools for Chordal Analysis. In *Proceedings of 2016 Joint WOCMAT-IRCAM Forum Conference*, WOCMAT 2016. Kainan University, 2016. URL `http://www.harmony-analyser.org/download/paper-wocmat2016-harmony-analyser.pdf`.

Ladislav Maršík, Jaroslav Pokorný, and Martin Ilčík. Improving Music Classification Using Harmonic Complexity. In *Procedings of the 14th Conference Information Technologies - Applications and Theory - Workshops and Posters*, ITAT 2014. Ústav informatiky AV ČR, 2014b. URL `http://artax.karlin.mff.cuni.cz/~bajel3am/itat2014/local/13_Marsik.pdf`.

Ladislav Maršík, Jaroslav Pokorný, and Martin Ilčík. A Survey on Music Retrieval Systems Using Microphone Input. In *Proceedings of the Dateso 2015 Annual International Workshop on Databases, Texts, Specifications*

and Objects, volume 1343 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2015. URL http://ceur-ws.org/Vol-1343.

Ladislav Maršík. Analysing Musical Pieces Using harmony-analyser.org Tools. In *Proceedings of the Dateso 2017 Annual International Workshop on Databases, Texts, Specifications and Objects*, 2017b. URL http://www.cs.vsb.cz/dateso/sbornik/dateso17-prews.pdf.

A. Attachments

A.1 Tones dictionary

In this attachment, the reader can find more information about music tones:

- Background on the tone definitions and naming.
- Dictionary that can help with the ambiguous naming of the tones.

This and the following attachments provide an extended basis of music theory and tonal harmony, recommended if the reader is interested in the topic of Chapter 3 (harmony model) and seeks to understand the chapter in more details.

From the spectrum of all audible pitches, the western music only uses a narrow set with frequencies in such distribution, that their differences may be clearly recognized by an ear (88 tones of today's piano keyboard). In this set, the two pitches, one with a double frequency of the other, blend in the sound while played simultaneously, so they resemble one sound, although they have different pitches. To these pitches, a distance of one *octave* is assigned. Within an octave, we differentiate a scale of seven tones that is periodically repeated. These tones were assigned the alphabet letters, forming the basis of a *musical alphabet*:

$$a, b, c, d, e, f, g$$

However, with stabilizing the tone c as the beginning of what became a *major scale*, we more often refer to the tone order: $c,\ d,\ e,\ f,\ g,\ a,\ b$.

To distinguish the different octaves, a labeling was established. In the *Helmholtz notation* commonly used by musicians, we label the octaves from the middle and up: „one-line" (c') , „two-line" (c''), „three-line" (c''') and from the middle down: „small" (c), „great" (C) and „contra" ($C,$). Some

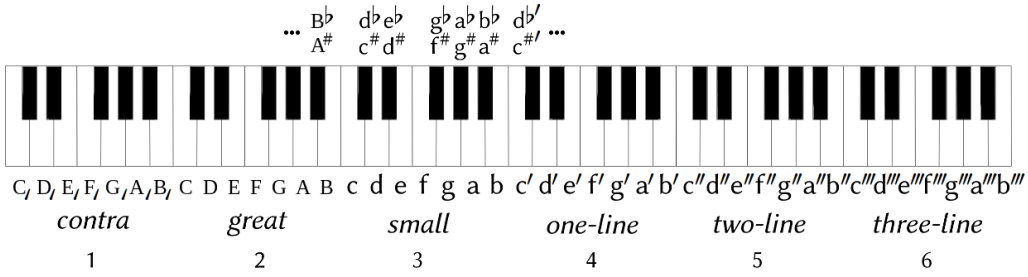


Figure A.1: Tones arranged on the piano keyboard in commonly used six octaves, from *contra* octave to *three-line* octave. A tone with one pitch can hold multiple names: e.g. a tone $c\#$, alternatively $d\flat$, stands in between the c and d tones and has one semitone distance from c and d.

authors prefer the *scientific notation*, simply labeling the octaves chronologically: 1, 2, 3, 4, 5, 6. On the standard piano, tones are ranging from the *sub contra a* ($A0$) to the *five-line c* ($C8$), MIDI tones range even from *double sub contra c* ($C\text{-}1$) to *six-line g* ($G9$). In Figure A.1 we show the most commonly used tones.

Due to the practical reasons of including a *semitone* interval (described in the details in Attachment A.2 *Intervals dictionary*), multiple tones out of the basic major scale were added (black tones on the piano keyboard), and we differentiate two ways how to describe their presence – by two types of *accidentals*:

- If the tone can be described as created by augmenting the original tone by a semitone, we mark it with the accidental ♯ next to the original tone, and call it „**sharp**“ (*c sharp*: $c\sharp$, *d sharp*: $d\sharp$, *f sharp*: $f\sharp$, *g sharp*: $g\sharp$, and *a sharp*: $a\sharp$).
- If the tone can be described as created by diminishing the original tone by a semitone, we mark it with the accidental ♭ next to the original tone, and call it „**flat**“ (*d flat*: $d\flat$, *e flat*: $e\flat$, *g flat*: $g\flat$, *a flat*: $a\flat$ an *b flat*: $b\flat$).

With the inclusion of these *accented* tones, new possibilities have arisen to form a musical scale. The basic scale c, d, e, f, g, a, b (called *C major*) can be shifted up or down, by the operation called *transposition*. The resulting scale of a transposition up by a semitone is a *C# major* scale: $c\#$, $d\#$, $e\#$, $f\#$, $g\#$, $a\#$, $b\#$. Importantly for the tone naming, the *C# major* scale can also be treated as a basis for augmenting or diminishing the tones. In fact, some musical instruments are constructed in the way that they prefer other scales than *C major* and having the possibility for augmenting/diminishing the tones for these scales is essential. That has two notable consequences:

- A doubled accidental. Examples: a double-flat b: $b\flat\flat$, or a double-sharp c: $c\#\#$ (usually denoted as $c\times$ in the sheet music)
- The fact that an otherwise unaccented tone can have the same pitch as a tone with double accidental. Example: The tone $b\flat\flat$ has the same pitch as the tone a. Note that we avoid specifying that the tones equal, or are the same. They only share the same pitch. This phenomenon is referred to as the tone $b\flat\flat$ being an *enharmonic equivalent* to the tone a.

Following this naming, Table A.1 shows the dictionary of common the tone pitches and the possible tones.

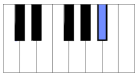
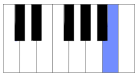
semitones from c	**frequency (small octave)**	**enharmonic equivalents**	**on the keyboard**
0	130.81	c, $b\#$, $d\flat\flat$	
1	138.59	$c\#$, $d\flat$	
2	146.83	d, $c\#\#$, $e\flat\flat$	
3	155.56	$d\#$, $e\flat$	
4	164.81	e, $d\#\#$, $f\flat$	
5	174.61	f, $e\#$, $g\flat\flat$	
6	185.00	$f\#$, $g\flat$	
7	196.00	g, $f\#\#$, $a\flat\flat$	
8	207.65	$g\#$, $a\flat$	
9	220.00	a, $g\#\#$ $b\flat\flat$	
10	233.08	$a\#$, $b\flat$	
11	246.94	b, $a\#\#$, $c\flat$	

Table A.1: Dictionary of tones: Tone pitches are shown as a distance from c in semitones, frequencies within the small octave, all the possible enharmonic equivalents as the tone names and a visualization on the piano keyboard.

A.2 Intervals dictionary

According to Schönberg [1922], we can explain the basic pitches and intervals as having been found through imitation of nature. A musical sound can be considered as a composite made up of a series of tones sounding together. The most prominent fundamental tone is accompanied by a series of *overtones*, produced by the partial oscillations of the material, forming the **harmonic series**. The frequency of the original wave is called the *fundamental frequency* or the *first harmonics* and represents the *fundamental tone* in the composite, whereas the higher frequencies are referred to as the overtones or the *higher harmonics* (2^{nd}, 3^{rd}, ...). From a fundamental C, the higher harmonics are:

$$c,\ g,\ c',\ e',\ g',\ b\flat'\ c'',\ d'',\ e'',\ f'',\ g'',\ etc.$$

The tones that occur first in the series also have a stronger presence in the composite. In fact, the presence of some of the harmonics may be omitted, as it depends on the musical instrument being played, and as such it translates to the timbre of the tone.

We now step away from the music acoustics and summarize how the harmonic series helps to define intervals. Schönberg specifies that the mathematical ratios found in harmonic series play a significant role in music, and, in fact, helped to form the piano keyboard as we know it today. While that is true, the mapping between the overtone series and the piano keyboard is not straightforward, as can be seen in Figure A.2. We can observe the definitions of the first few intervals (frequency ratio: $2:1$ *perfect octave*, frequency ratio: $3:2$ *perfect fifth*, and so on), but we can also see a few discrepancies. Starting with the sixth overtone, the intervals seem to be repeating themselves, if we look at the number of tones in between. The truth is that for western music, the mathematically precise intervals have become impractical for some musical instruments. Therefore, a common interval ratio of a *semitone* was established, with the value of $\sqrt[12]{2} : 1$. The definition came from diving the perfect octave interval into 12 semitones, and a logarithmic unit of *cents* was introduced, where 100 cents represent one semitone. This tuning became known as the *tempered tuning*, as opposed to the *just tuning* coming from the exact ratios of the harmonic series.

Looking back at Figure A.2 we may now point out, that the harmonic series is not perfectly aligned with the tones on the piano keyboard, starting already with the 2^{nd} overtone being two cents above the tone from the tempered tuning. Following the harmonic series, the discrepancies would be more frequent. Notwithstanding the mathematical inaccuracies, according to Laborecký [1997] it is the frequency ratio $9:8$ that is most commonly denoted as a *major second* interval consisting of two semitones (thus often named a *whole tone* interval, too). The *minor second* interval (consisting of one semitone) is even harder to match to the mathematical ratio, but the difference between the perfect fourth and major third leads us to the ratio $16:15$ which is commonly denoted as the *just tuning semitone*. On

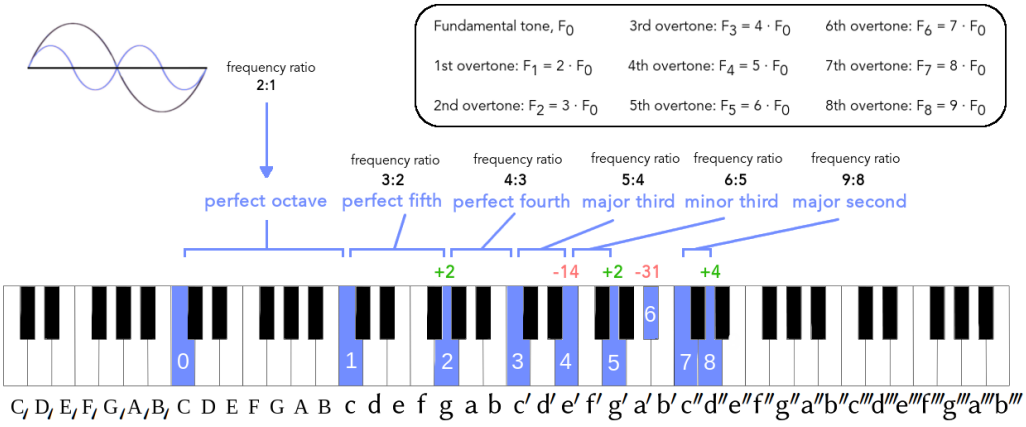


Figure A.2: First eight overtones of a harmonic series mapped onto the piano keyboard. Frequencies are shown as multiples of the fundamental frequency F_0. Frequency ratios of the just tuning are shown on the piano keyboard. Above the keyboard, the difference of the keyboard's tempered tuning and the just tuning is shown in cents. As an example of how the piano keyboard is not mathematically aligned with the harmonic series, the sixth overtone is diminished by 31 cents (out of 100 cents of a semitone) from the $b\flat$ of a piano keyboard, and is also the first one not to fit the basic scale of tones.

the other hand, the difference between the major third and the minor third intervals leads us to the frequency ratio of 25 : 24. These ratios – albeit very different – are almost indistinguishable by an ear, which only reinforces the motivation of the western music to implement the tempered tuning of 12 equal semitones within one octave.

The naming itself consists of two parts:

1. *unisone, second, third,* etc., are inherited from the scales, where *second* represents the second tone of a scale, *third* the third tone of a scale, etc.

2. *perfect*, *major* or *minor* are qualities inherited from the mode of the scale (the reader may refer to Attachment A.3 *Scales dictionary* for more information on scales). *Major* intervals appear in the major scales, while *minor* intervals appear in the minor scales. For example, the 3$^{\text{rd}}$ tone in a major scale and the 3$^{\text{rd}}$ tone in a minor scale have a different distance from the 1$^{\text{st}}$ tone, and they were named *major third* and *minor third* respectively. *Perfect* intervals hold such naming, because they appear in both major and minor scales.

Table A.2 summarizes all the definitions and gives the reader a quick reference for the interval naming. Same as with the tones, there are possible enharmonic equivalents for the interval names. As an example, the tones c and b represent an interval of major seventh in the *C major* scale. If the composer decides to augment the tone b to $b\#$, the resulting interval

distance in semitones	name	frequency ratio (just tuning)	type	enharmonic equivalents	on the keyboard
0	perfect unisone	1 : 1	consonant	diminished second	
1	minor second	16 : 15	dissonant	augmented unisone	
2	major second	9 : 8	dissonant	diminished third	
3	minor third	6 : 5	consonant	augmented second	
4	major third	5 : 4	consonant	diminished fourth	
5	perfect fourth	4 : 3	consonant	augmented third	
6	tritone	N/A	dissonant	diminished fifth, augmented fourth	
7	perfect fifth	3 : 2	consonant	diminished sixth	
8	minor sixth	N/A	consonant	augmented fifth	
9	major sixth	N/A	consonant	diminished seventh	
10	minor seventh	N/A	dissonant	augmented sixth	
11	major seventh	N/A	dissonant	diminished octave	
12	perfect octave	2 : 1	consonan	atugmented seventh	

Table A.2: Dictionary of intervals: The most common intervals are shown as the distance measure in semitones, a common name, an equivalent frequency ratio from the just tuning, the possible enharmonic equivalents, and a visualization on the piano keyboard.

between c and $b\#$ has to hold the name of the original interval (*seventh*) and is therefore named *augmented seventh*. This helps the musicians to describe any tone precisely by using a reference tone and an interval name (e.g., augmented seventh up from c is $b\#$, and not c') and makes the musical speech less confusing.

A.3 Scales dictionary

Historically, the scales as we know them today were preceded by *modes* (from Latin word „modus“: „measure, standard, manner“) introduced in Ancient Greece, and later the *church modes* used in medieval church music (Schönberg [1922]). Both eras introduced multiple modes (e.g., Mixolydian, Lydian, Phrygian to name a few Greek modes), but only two of them took the lead role in the tonal harmony period in the 19^{th} century: Ionian and Aeolian, today forming the *major* and *minor* scale. The word *mode* had also taken on an additional meaning at that time, and we now speak of a *major mode* or *minor mode* in reference to the scales, keys, or tonalities.

It is also possible to look for the foundations of a scale in the music acoustics. According to the harmonic series explained in Attachement A.2, for the fundamental tone c it is the tone g as the second most important component, and as such, our ear represents it as a harmony when these two tones sound together. A similar assumption can also be made about the next tone appearing in the series, tone e. Consequently, for the tone g, the closest higher harmonics would be g', d'', g'' and b'' and therefore we may conclude that d and b as another essential harmony. Taking the tone c as the midpoint, we should also consider the other direction (following the inversion principle, very common in music theory). We have c as the first overtone in the harmonic series of the tone f and the tone a found in the following overtones. Following these guidelines, the seven tones of the major scale are found, and it is only a matter of choosing the I. degree to differentiate the major and minor scale (*C major*: c, d, e, f, g, a, b, c vs. *a minor*: a, b, c, d, e, f, g, a).

The scales are named based on the first tone of the scale, and based on their mode, e.g., *C major*, or *a minor*. The convention says that the major scales should be labeled by a capital letter, whereas the minor scales by a non-capital letter.

We provide a comparison of the major and minor scales from the tone c in Table A.3. The reader can observe that the major and minor scales differ in the $III., VI.$ and $VII.$ degree, and that both are true to their definitions of containing major and minor thirds respectively (major thirds in *C major* in between c and e, f and a, and g and b; minor thirds in *c minor* in between c and $e\flat$, f and $a\flat$, and g and $b\flat$). As we focus on the tonal harmony, we omit the other scales commonly used by music genres, the scales of modal harmony, chromatic scale, etc. We also consider only *natural* scales for both modes, while there also exist variants (*harmonic*, *melodic*, ...).

The final selection of 24 scales commonly used by tonal harmony is shown in Table A.4. Musicologists have been trying to visualize the relationships of the scales (keys, tonalities) for over a century, with many notable results such as the *Tonnetz* grid or a spiral array discussed in Chapter 2. However, the standard visualization of the 24 scales remains until this day the *circle of fifths* (Figure A.3). Scales are presented on a circle, and the neighboring scales have the fifth interval between their I. degree. Starting from *C major*

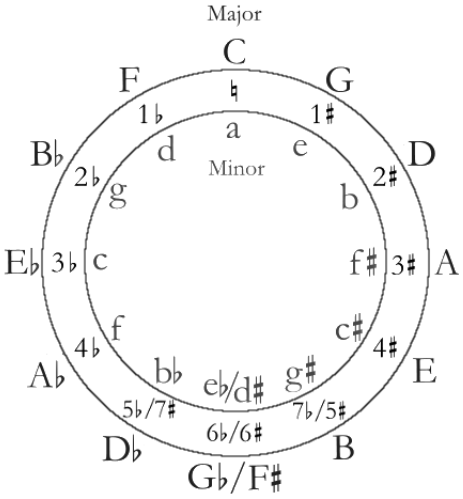


Figure A.3: Circle of fifths: 12 major and 12 minor scales (keys) presented on a circle, with the fifth interval in between the I. degree of the neighboring scales.

and *a minor*, the keys following by fifth intervals in one direction, have an increasing number of tones with sharp accidentals, and starting from the same (*C major*, *a minor*) going the other direction, have an increasing number of tones with flat accidentals. This is also a common way to determine how many augmented or diminished tones are there in the particular key – finding out the number of steps in the circle of fifths. The set of accidentals for a particular scale or key is referred to as its *signature*.

tones	**scale**	**on the keyboard**
c d e f g a b c	*C major*	
c d e♭ f g a♭ b♭ c	*c minor*	

Table A.3: Diatonic scales: natural *C major* and *c minor* scales and their visualization on the piano keyboard.

mode	**scales**
major	*C major*, *C# major/D♭ major*, *D major*, *D# major/E♭ major*, *E major*, *F major*, *F# major/G♭ major*, *G major*, *G# major/A♭ major*, *A major*, *A# major/B♭ major*, *B major*
minor	*c minor*, *c# minor/d♭ minor*, *d minor*, *d# minor/e♭ minor*, *e minor*, *f minor*, *f# minor/g♭ minor*, *g minor*, *g# minor/a♭ minor*, *a minor*, *a# minor/b♭ minor*, *b minor*

Table A.4: 24 diatonic scales used by tonal harmony, 12 for each mode, denoting the enharmonic equivalent scales.

A.4 Chords dictionary

According to Schönberg [1922]: *If the scale is an imitation of the tone on the horizontal plane, that is, note after note, then chords are imitation on the vertical, notes sounded together. If the scale is an analysis, then the chord is a synthesis of the tone.* Schönberg further describes that the obvious way to form a chord to imitate the euphony of a single tone is to omit the more distant overtones and to reinforce the more immediate. Such imitation forms the **major triad** as the fundamental tone, major third and perfect fifth (e.g. *C major*: c, e, g).

The **minor triad** seemingly does not have a direct resemblance in the harmonic series, but the analogy is only less apparent. The inversion principle, best described by the Riemann's *dualist system* (Lewin [1987]), reminds us to consider the inversional relationship between *major* and *minor* mode, with minor triads being considered „upside down" versions of major triads. So while a major triad is formed of major third and a minor third from bottom to top, a minor triad is formed of the major third and minor third from top to bottom (major third in between the III. and V. degree, example: $e\flat$ and g in $c, e\flat, g$). Alternatively, we can conclude that the tone $e\flat$ belongs to the triad, but not as a prominent overtone of the fundamental tone c, but rather as a g being a prominent overtone of $e\flat$.

We show both of the basic triad's structure in Table A.5 and we further apply the 1st inversion and 2nd inversion on the major triad in Table A.6.

structure	chord	picture
major third, perfect fifth	major triad	
minor third, perfect fifth	minor triad	

Table A.5: The basic triad's structure.

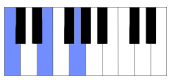
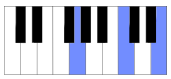
type	name	picture
root position	triad	
1st inversion	sixth chord	
2nd inversion	four-six chord	

Table A.6: The basic triad's inversions, formed by rearranging the bottom tone to the top of the chord.

Besides major and minor triads we also distinguish **diminished triad**

(minor third, diminished fifth) and **augmented triad** (major third, augmented fifth), shown in Table A.7.

structure	**chord**	**picture**
minor third, diminished fifth	diminished triad	
major third, augmented fifth	augmented triad	

Table A.7: Diminished and augmented triads.

A.5 Harmonic functions dictionary

It is possible to build a triad on every degree of a diatonic scale, using the tones of that scale. Every such triad we can then assign a *harmonic function*, also known as *diatonic function*.

Note that, a function here means a certain *role* that the triad plays towards the root of the key, the tonic. In the thesis, in Section 1.1.3, we have discussed that the three main roles (three main functions) are: tonic (T), subdominant (S) and dominant (D), built on I., IV., and V. degree accordingly. The triads on the other degrees we perceive as variants, or parallels, of the three main functions. They share characteristic tones with the main functions, and are therefore capable of substituting them in certain cases.

In Section 1.1.3, we have also discussed that forming a parallel is done by modifying one of its tones. In particular: *„. . . by extending the highest tone (i.e., V. degree in the tonic, I. degree in the subdominant and II. degree in the dominant) by a whole tone, or by diminishing the root tone by a semitone.“* This is a simplified definition, and music theory describes this process in much more details. But, at the same time, there are multiple theories to choose from, each using its own convention. We forward the reader who wishes to understand the details on function parallels to Lewin [1987], Riemann [1896a] or Zika and Kořínek [1990].

Consequently, the naming for the functions differs in the literature. Some theories assign each of the seven triads a function on its own – as is commonly taught in North America (Caplin [2013]) – whereas the others assign the name with respect to the main function – German approach (Riemann [1896a]). We can nevertheless call the triad with the name of the degree it is built on, simply I, II, III, . . . , VII (some theories would use lower-case Roman numerals if the triad is minor). In the thesis we implement a simplification of the German approach, where the two possible variants – parallel (German: *„Parallel“*, denoted by P subscript, e.g T_P) and leading tone exchange (German: *„Gegenparallelklang“*, denoted either by L subscript or by G subscript, e.g., T_G) are unified, both to be denoted as a parallel with a subscript P. So we refer to both tonic variations as T_P. All of these namings are summarized in Figure A.4.

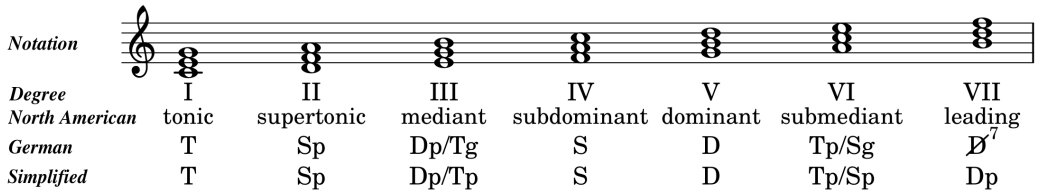


Figure A.4: Diatonic functions of *C major* key built as a triad on every degree of a diatonic scale, using the tones of the *C major* scale.