Freischütz Digital: Processing Audio Signals in Complex Music Scenarios

Freischütz Digital: Verarbeitung von Audiosignalen in komplexen Musikszenarien

Dissertation

Der Technischen Fakultät

der Friedrich-Alexander-Universität Erlangen-Nürnberg

zur

Erlangung des Doktorgrades
Doktor der Ingenieurwissenschaften (Dr.-Ing.)

vorgelegt von

Thomas Prätzlich

aus

Saarbrücken

Als Dissertation genehmigt
von der Technischen Fakultät
der Friedrich-Alexander-Universität Erlangen-Nürnberg

Tag der mündlichen Prüfung:	23.11.2016
Vorsitzender des Promotionsorgans:	Prof. Dr.-Ing. Reinhard Lerch
1. Gutachter:	Prof. Dr. Meinard Müller
2. Gutachter:	Prof. Dr. Gerhard Widmer

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus anderen Quellen oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet. Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder ähnlicher Form in einem Verfahren zur Erlangung eines akademischen Grades vorgelegt.

Erlangen, 30.09.2016

Thomas Prätzlich

Abstract

This thesis was conducted within the project "Freischütz Digital" (FreiDi),[1] an interdisciplinary cooperation between musicologists and computer scientists. A general goal of the FreiDi project was to develop and introduce computer-based methods that enhance human involvement with music. The opera "Der Freischütz" by Carl Maria von Weber provides a complex music scenario offering a large number of sources, including different versions of the musical score, the libretto, and audio recordings. Motivated by the FreiDi scenario, I applied methods from signal processing and music information retrieval (MIR) to process audio material with the goal of detecting and establishing semantic relationships across various music recordings and symbolic representations. This thesis presents novel, content-based methods for music synchronization, audio segmentation, and interference reduction. First, I present a novel memory-efficient algorithm for music synchronization based on a multiscale dynamic time warping (DTW) approach, which allows for specifying a constant upper bound on memory requirements. Secondly, I adapt automated segmentation procedures based on synchronization and matching techniques to segment different recordings of the opera consistently according to a given reference segmentation. Thirdly, I present a method to reduce interference in multitrack music recordings that iteratively estimates both the power spectral density of each instrument/voice and its corresponding strength in each microphone signal, based on Wiener-filtering and non-negative matrix factorization (NMF). Specific applications within the FreiDi scenario demonstrate the practicability of the proposed algorithms. Furthermore, systematic experiments on real-world music recordings beyond this scenario not only illustrate benefits and limitations of automated methods, but also deepen the understanding of inconsistencies and variations within the recordings.

[1]The project "Freischütz Digital – Paradigmatische Umsetzung eines genuin digitalen Editionskonzepts" was funded by the Federal Ministry of Education and Research in Germany (Funding Code 01UG1239A to C).

Zusammenfassung

Diese Arbeit wurde im Rahmen des Projekts „Freischütz Digital“ (FreiDi)[2] durchgeführt, einer interdisziplinären Kooperation zwischen Musikwissenschaftlern und Informatikern. Allgemeines Ziel des FreiDi-Projekts war die Entwicklung von computer-basierten Methoden, die die Benutzerinteraktion mit Musikdaten erleichtern und bereichern sollen. Die Oper „Der Freischütz“ von Carl Maria von Weber stellt mit seiner reichhaltigen Quellenlage mit verschiedenen Versionen des Notentexts, des Librettos und einer Vielzahl von Audioaufnahmen ein komplexes Musikszenario dar. Ausgehend vom FreiDi-Szenario habe ich Methoden der Signalverarbeitung und des Music Information Retrievals auf die Audioaufnahmen angewendet, um semantische Verknüpfungen zwischen den verschiedenen Musikaufnahmen und symbolischen Repräsentationen zu erkennen und herzustellen. In dieser Arbeit werden neue, inhaltsbasierte Methoden zur Musiksynchronisation, Audiosegmentierung und zur Reduktion von Übersprechen vorgestellt. Insbesondere wird ein speichereffizienter, auf einem multiskalen dynamic time warping-Ansatz basierender Musiksynchronisationsalgorithmus vorgestellt, dessen Speicherverbrauch sich durch eine obere Schranke begrenzen lässt. Weiterhin werden synchronisations- und matching-basierte automatisierte Segmentierungsverfahren entwickelt, um eine durch eine Referenzaufnahme festgelegte einheitliche Segmentierung der verschiedenen Aufnahmen der Oper herzustellen. Als weiteren Hauptbeitrag wird ein iteratives Verfahren zur Reduktion des Übersprechens in Multikanal Musikaufnahmen vorgestellt. Hierbei wird die spektrale Leistungsdichte der beteiligten Instrumente sowie deren Stärke in den Mikrofonen durch ein auf Wiener-Filter und Nicht-negativer Matrix-Faktorisierung (NMF) basierendes Verfahren geschätzt. Spezifische Anwendungsszenarien im FreiDi-Projekt veranschaulichen die Praktikabilität der vorgestellten Algorithmen. Über das FreiDi-Szenario hinaus werden in systematischen Experimenten mit realen Musikaufnahmen nicht nur die Vorteile und Einschränkungen von automatisierten Methoden aufgezeigt, sondern auch das Verständnis der in den Aufnahmen selbst liegenden Inkonsistenzen und Variabilitäten vertieft.

[2]Das Projekt „Freischütz Digital – Paradigmatische Umsetzung eines genuin digitalen Editionskonzepts“ wurde durch das Bundesministerium für Bildung und Forschung (BMBF) gefördert (Förderkennzeichen 01UG1239A bis C).

Contents

Chapter 1

Introduction

The increasing availability of computational resources has led to the field of *digital humanities*, where computer-based methods and digital resources are systematically used to approach research within the humanities. While early computer-aided music research relied primarily on symbolic representations of the musical score, the focus of recent research efforts has shifted towards the processing and analysis of various types of music representations including text, audio, and video [67, 79]. For example, in the case of an opera, one can typically find digitized versions of the libretto (literary text of the opera), different versions of the musical score, as well as a large number of performances available as audio and video recordings (see Figure 1.1). These different versions and representations constitute the body of sources of a musical work. Providing tools for accessing, analyzing, and comparing music sources is an important task in the field of *music information retrieval* (MIR).

One of the goals of the “Freischütz Digital”[1] project (FreiDi) was to develop and apply automated methods to support musicologists in editing, analyzing and comparing the various musical sources. Within the project, the opera “Der Freischütz” by Carl Maria von Weber—a work of central musical importance offering a rich body of sources—served as a complex music scenario. A central task was to establish linking structures across different musical sources, including sheet music and audio material. Having established linking structures between the musical score and available audio versions, one can listen to an audio recording and have the current position in the musical score highlighted.[2] Also, it is possible to use the score as an aid to navigate within an audio version and vice versa. Furthermore, one can use these linking structures to seamlessly switch between different recordings, thus making it easier to compare different performances.

[1] Freischütz Digital: Paradigmatische Umsetzung eines genuin digitalen Editionskonzepts, BMBF Funding Code 01UG1239A to C
`http://freischuetz-digital.de/`

[2] A demonstration of such an interface can be found at `http://freischuetz-digital.de/demos/syncPlayer/`

In this thesis, focused on the audio domain, we describe how these and other audio-related MIR tasks were approached within the FreiDi project. A central MIR task within this thesis is music synchronization, which aims to establish linking structures between two different versions of the same piece of music. As mentioned above, these can be used to simplify accessing and navigating music data. Large-scale musical works such as operas usually consist of different parts, and a full recording of an opera is usually very long. Different recordings may even deviate structurally, such that a straightforward comparison of two recordings might not be possible. As an important pre-processing step for further analysis, we approached the task of consistently segmenting different recordings of the opera "Der Freischütz" according to a given reference segmentation. In this context, we highlight various challenges that arise when (even well-established) techniques are applied to real-world scenarios. Lastly, we approach the task of interference reduction in multitrack music recordings, where the aim is to reduce unwanted interference (e.g. from other instruments or voices) in the individual tracks.

1.1 Structure

This thesis is structured as follows.

In Chapter 2, we introduce the FreiDi project and its general goals. The research described in this thesis was performed in the context of the FreiDi project. We provide an overview of the opera "Der Freischütz" and review the different data sources that were relevant within the project. We then describe the audio-related tasks that played a role in the project, and which are also in the focus of this thesis.

Next, in Chapter 3, we summarize fundamentals of music audio signal processing that are used throughout this thesis. We introduce basics on discrete-time signals, the short-time Fourier transform, and spectrogram representations derived from it. Building on spectrograms, we review features that are relevant for the tasks addressed in this thesis.

In Chapter 4, we deal with music synchronization techniques. We first introduce the basics of music synchronization and then introduce a novel memory-efficient variant of dynamic time warping (DTW). Experiments show that restricting the memory consumption of our proposed procedure to eight megabytes basically yields the same alignments as the standard DTW procedure. We extended an existing multiscale DTW (MsDTW) alignment approach that uses an alignment on a coarse resolution to constrain an alignment on a finer-grained resolution. In our modified approach, we proceed in a block-by-block fashion, where an additional block size parameter is introduced to explicitly control the memory requirements. Similar to previously introduced multiscale alignment strategies, our novel procedure drastically reduces the memory requirements and runtimes. In contrast to the previous approaches, having a linearly growing

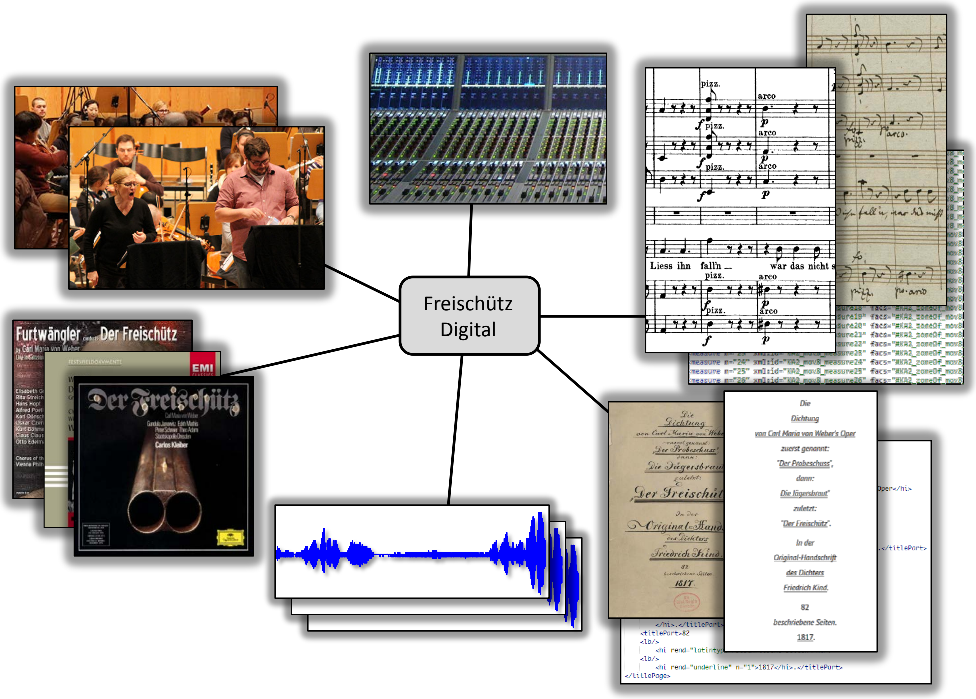


Figure 1.1. Music-related information in multiple modalities illustrated by means of the opera "Der Freischütz" by Carl Maria von Weber.

memory requirement, our block-by-block processing strategy allows for specifying a constant upper bound on the memory requirements.

Chapter 5 is dedicated to consistently segmenting different recordings of a musical work according to a reference specified by a domain expert. In the first part of the chapter, we approach the segmentation of mostly complete recordings of the opera. As it turns out, the task is more complex than one may think, due to significant acoustic and structural variations across the various recordings. We reveal and discuss these variations by systematically adapting segmentation procedures based on synchronization and matching techniques. Large-scale musical works such as operas may have a performance duration of several hours and typically involve a large number of musicians. For such compositions, one often finds different arrangements and abridged versions (often shorter than an hour), which can also be performed by smaller ensembles. Abridged versions still convey the flavor of the musical work and contain the most important excerpts and melodies. In the second part of the chapter, we show how a reference segmentation of an original, complete version of a musical work can be transferred onto strongly abridged versions. We adapt a frame-level matching approach for transferring the reference segment information to the unknown version. Considering the opera "Der Freischütz" as an example scenario, we discuss how to balance the flexibility and robustness of our proposed segmentation procedure.

In Chapter 6, we explore novel ways to analyze music alignments without the need for ground-truth annotations. Alignment methods are typically evaluated by comparing the computed alignments to given ground-truth annotations. Creating such annotations is usually very labor-intensive. For many musical pieces, especially in classical music, there exists a multitude of recordings. In this work, we investigate whether an evaluation of music alignment algorithms can be performed without ground-truth annotations when at least a triplet of recordings of the same piece of music is available. The main idea is to align the time points of a fixed reference version, in a circular way, back through a second and third version by using their pairwise alignments. A triple error is then computed by comparing these time points with their circularly aligned version. We formalize the idea of the triple error and discuss its potential and limitations. We present typical examples for the triple error and compare it to the pairwise alignment error based on ground-truth. We then present a case study to indicate the potential of the triple error to analyze alignments and to compare different alignment methods without the need of ground-truth annotations. Finally, we put the idea of the triple error in the general context of cross-version approaches which incorporate different versions of a piece of music into a music analysis or processing task.

In Chapter 7, we approach the task of reducing unwanted interference in multitrack music recordings. When recording a live musical performance, the different sound sources, such as the instrument groups or soloists of an orchestra, are typically recorded in the same room simultaneously, with at least one dedicated microphone assigned to each sound source. However, it is difficult to acoustically shield the microphones. In practice, each one contains interference from every source. To address this problem, we introduce an algorithm called MIRA (Multitrack Interference Reduction Algorithm) aiming to recover only the isolated source assigned to a given microphone by reducing the interference from other sources. For each source, MIRA iteratively estimates its power spectral density (PSD), its corresponding spatial image in the assigned microphone, and its amount of interference in each microphone signal. The PSDs and spatial images are estimated through Wiener filtering. The amount of interference of each source in each microphone is modeled through an interference matrix, which is learned via non-negative matrix factorization (NMF). The trade-off between distortion and separation can be controlled by the user through the number of iterations of the algorithm. We conducted an extensive evaluation of the algorithm on controlled mixtures using objective evaluation measures for source separation. Furthermore, we present application scenarios for the algorithm in which we use multitrack data from the FreiDi project.

In Chapter 8, we conclude the thesis and give an outlook to possible extensions and applications of the work presented in the previous chapters.

1.2 Contributions

The main contributions of this thesis can be summarized as follows.

- Memory restricted multiscale dynamic time warping (MrMsDTW), a novel multiscale dynamic time warping (MsDTW) variant that allows for specifying an upper bound on the memory usage (Section 4).
- Two approaches for reference-based segmentation of musical works (Section 5): one for complete recordings (Section 5.2) and one for abridged recordings (Section 5.3).
- A discussion of variations that typically occur in collections containing audio versions of the same musical work (Chapter 5).
- A novel approach for ground-truth-independent analysis of music alignments (Section 6).
- An iterative algorithm for interference reduction in multitrack recordings and estimation of an interference matrix via non-negative matrix factorization (NMF) (Chapter 7).
- The *Sync-Toolbox*, which provides implementations of our proposed MrMsDTW algorithm and utility scripts for music synchronization (Appendix C).
- The Freischütz Digital Multitrack Dataset (Appendix A).
- Web-based demos (Appendix E).

1.3 Main Publications

The main contributions of this thesis are based on the following publications, which were presented at conferences in the field of audio signal processing and music information retrieval.

[87] Meinard Müller, Thomas Prätzlich, Benjamin Bohl, and Joachim Veit. Freischütz Digital: a multimodal scenario for informed music processing. In *Proceedings of the International Workshop on Image and Audio Analysis for Multimedia Interactive Services (WIAMIS)*, pages 1–4, Paris, France, 2013.

[99] Thomas Prätzlich and Meinard Müller. Freischütz Digital: a case study for reference-based audio segmentation of operas. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 589–594, Curitiba, Brazil, 2013.

[100] Thomas Prätzlich and Meinard Müller. Frame-level audio segmentation for abridged musical works. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 307–312, Taipei, Taiwan, 2014.

[96] Thomas Prätzlich, Rachel Bittner, Antoine Liutkus, and Meinard Müller. Kernel additive modeling for interference reduction in multi-channel music recordings. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Brisbane, Australia, 2015.

[102] Thomas Prätzlich, Meinard Müller, Benjamin W. Bohl, and Joachim Veit. Freischütz Digital: Demos of audio-related contributions. In *Demos and Late Breaking News of the International Society for Music Information Retrieval Conference (ISMIR)*, Málaga, Spain, 2015.

[98] Thomas Prätzlich, Jonathan Driedger, and Meinard Müller. Memory-restricted multiscale dynamic time warping. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 569–573, Shanghai, China, 2016.

[101] Thomas Prätzlich and Meinard Müller. Triple-based analysis of music alignments without the need of ground-truth annotations. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 266–270, Shanghai, China, 2016.

[88] Meinard Müller, Thomas Prätzlich, and Christian Dittmar. Freischütz Digital: When computer science meets musicology. In Kristina Richts and Peter Stadler, editors, *„Ei, dem alten Herrn zoll' ich Achtung gern'“: Festschrift für Joachim Veit zum 60. Geburtstag*, pages 551–574. Allitera Verlag, München, 2016.

[97] Thomas Prätzlich, Rachel M. Bittner, Antoine Liutkus, Juan P. Bello, and Meinard Müller. Interference reduction for multitrack music recordings. *IEEE Transactions on Audio, Speech, and Language Processing*, 2016, in preparation.

1.4 Additional Publications

The following publications are also related to music signal processing, but are not considered in this thesis.

[89] Meinard Müller, Thomas Prätzlich, and Jonathan Driedger. A cross-version approach for stabilizing tempo-based novelty detection. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 427–432, Porto, Portugal, 2012

[26] Jonathan Driedger, Harald Grohganz, Thomas Prätzlich, Sebastian Ewert, and Meinard Müller. Score-informed audio decomposition and applications. In *Proceedings of the ACM International Conference on Multimedia (ACM-MM)*, pages 541–544, Barcelona, Spain, 2013.

[22] Christian Dittmar, Bernhard Lehner, Thomas Prätzlich, Meinard Müller, and Gerhard Widmer. Cross-version singing voice detection in classical opera recordings. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 618–624, Málaga, Spain, 2015.

[24] Christian Dittmar, Thomas Prätzlich, and Meinard Müller. Towards cross-version singing voice detection. In *Proceedings of the Jahrestagung für Akustik (DAGA)*, Nuremberg, Germany, 2015.

[110] Daniel Röwenstrunk, Thomas Prätzlich, Thomas Betzwieser, Meinard Müller, Gerd Szwillus, and Joachim Veit. Das Gesamtkunstwerk Oper aus Datensicht – Aspekte des Umgangs mit einer heterogenen Datenlage im BMBF-Projekt “Freischütz Digital”. *Datenbank-Spektrum*, 15(1):65–72, 2015.

[7] Stefan Balke, Lukas Lamprecht, Vlora Arifi-Müller, Thomas Prätzlich, and Meinard Müller. Automatisierte Identifikation von Audioaufnahmen anhand symbolisch codierter musikalischer Themen. In *Proceedings of the Deutsche Jahrestagung für Akustik (DAGA)*, Nuremberg, Germany, 2015.

[13] Benjamin W. Bohl, Thomas Prätzlich, Meinard Müller, and Joachim Veit. Der Gang durch die Domänen. In *Jahrestagung der Digital Humanities im deutschsprachigen Raum: „Von Daten zu Erkenntnissen: Digitale Geisteswissenschaften als Mittler zwischen Information und Interpretation“*, Graz, Austria, 2015.

[28] Jonathan Driedger, Thomas Prätzlich, and Meinard Müller. Let It Bee – Towards NMF-inspired audio mosaicing. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 350–356, Málaga, Spain, 2015.

[130] Christof Weiß, Vlora Arifi-Müller, Thomas Prätzlich, Rainer Kleinertz, and Meinard Müller. Analyzing measure annotations for Western classical music recordings. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 517–523, New York City, NY, USA, 2016.

[121] TJ Tsai, Thomas Prätzlich, and Meinard Müller. Known-artist live song id: A hashprint approach. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 427–433, New York City, NY, USA, 2016.

[120] Sebastian Stober, Thomas Prätzlich, and Meinard Müller. Brain Beats: Tempo extraction from EEG data. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 276–282, New York City, NY, USA, 2016.

1.5 Acknowledgments

Throughout my work on this thesis, I was lucky to have many people support me in many different ways. I would like to take the opportunity to express my gratitude to these people here.

First of all, I want to thank my supervisor Prof. Dr. Meinard Müller, without whom this thesis would not have been possible. He always made himself available for discussions and always re-focused me onto the most essential tasks at hand whenever re-focussing was needed. Not only did he provide valuable input whenever needed, but also a great working environment with an amazing set of colleagues.

I would like to thank Prof. Dr. Gerhard Widmer for reviewing my thesis and his forthcoming approach and flexibility.

During the work on my thesis, I had the opportunity to work in the interdisciplinary project “Freischütz Digital”, and I would like to offer my special thanks to the project team. I greatly enjoyed working with you all. Throughout the course of this project and during our regular meetings in Detmold, Paderborn, Frankfurt, and Erlangen, I learned so much about how to make interdisciplinary collaboration work out, and as a plus I also learned so much about musicology and the work of musicologists and the requirements of their research for scholarly

critical editions. In particular, I wish to thank Joachim Veit, without whom the "Freischütz Digital" project never would have come into being. It was his open-minded approach that brought together this wonderful team of people from so many different fields. Furthermore, he was always open for questions and never tired of explaining even simple musicological terms to a non-musicologist. I would like to thank Benjamin Bohl for his wonderful motivation and drive, which really enriched the entire project and helped building many bridges. I truly enjoyed working with you in our short but highly productive sessions. Johannes Kepper, thank you for your great determination while working with me on converting the early stage MEI music encodings into a piano-roll representation. Anna Maria Komprecht, thank you for your practical attitude and amazing organizational skills, especially when managing the project documentation. Thomas Betzwieser, Solveig Schreiter, and Janette Seuffert, thank you for introducing me to the concept of genetic text editions. Janette, I'd also like to thank you for the outstandingly constructive project meeting in Frankfurt. I really enjoyed our presentation at the Max Planck Institute for Empirical Aesthetics. That meeting was a real milestone for me, where all the bits and pieces finally came together to form a full picture, and I was able to grasp the individual project goals for each cooperation partner as well as the goal we were all pursuing together. This event really taught me the importance of establishing a shared terminology and of learning all you can about the topics of each others' research. Daniel Röwenstrunk, thank you for all the rides you gave me around Detmold and Paderborn, which always included inspiring discussions on the way. The insights you shared as a computer scientist in a group of musicologists proved invaluable to me and helped me to get a clear grasp of the tasks we tackled in the project.

Talking about Detmold, I would especially like to thank Lothar Gräwe and Eva Sturm-Gräwe, who provided me with a second home during project meetings and offered me support and encouragement throughout my thesis.

One of the components of the "Freischütz Digital" project was the recording of a selection of numbers from the Freischütz opera in cooperation with Tonmeisters from the Erich-Thienhaus-Institut in Detmold. This was an amazing opportunity to learn about recording techniques and production processes, for which I would like to thank Prof. Dipl-Tonm. Bernhard Güttler, Prof. Dipl-Tonm. Michael Sandner, and their students Stefan Antonin, Florian Bitzer, and Matthias Kieslich.

In this context I would also like to thank my parents Marietta and Olaf Prätzlich for sharing their passion and appreciation for music with me, and for nurturing my musical interests from the very beginning.

In the summer of 2014, Antoine Liutkus gave me the opportunity to visit INRIA in Nancy. Merci Antoine! We had a week of dense activity and I learned a lot. In this context, I also want to thank Elias Kokkinis, who kindly shared his code with us.

I would also like to express my gratitude to Juan P. Bello, who invited me to visit his lab at NYU in February 2016 and gave me the opportunity to collaborate with Rachel Bittner.

Over the first three years, I was funded by the project “Freischütz Digital” (Funding Code 01UG1239A to C) for which I want to thank the Federal Ministry of Education and Research in Germany. In the fourth year, I was financed both by the DFG-project SIAMUS[3] (DFG MU 2686/6-1), and the project “Computergestützte Analyse harmonischer Strukturen” (DFG MU 2686/7-1) for which I would like to thank the German Research Foundation

I count myself lucky to have worked in Meinard Müller’s research group, which provided a great environment and which was never short of smart and friendly people. My grateful thanks for great collaborations, discussions, and barbecues go to my current and former group colleagues (in no particular order): Stefan Balke, Christian Dittmar, Jonathan Driedger, Patricio López-Serrano, Vlora Arifi-Müller, Christof Weiß, Harald G. Grohganz, Nanzhu Jiang, Sebastian Ewert, Peter Grosche, Verena Konz, Andreas Baak, Thomas Helten, and Frank Zalkow.

Meinard’s group forms part of the International Audio Laboratories Erlangen,[4] which provided me with the best research environment for audio signal processing I could have wished for. I would like to thank all the people who helped to create such an inspiring work environment, especially the professors Tom Bäckström, Bernd Edler, Emanuël Habets, Jürgen Herre, and Meinard Müller. During my time at the AudioLabs, I was lucky to receive amazing support from the administrative staff with all kinds of matters from travel organisation and managing contracts to technical assistance with all kinds of appliances. Elke Weiland, Tracy Harris, Day-See Riechmann, Claudia Erbel, Jennifer Kohlhage, Carola Seibold, and Stefan Turowski—thank you all for making the AudioLabs such a great place to work at!

I also had the opportunity to work with a number of student assistants and students writing their Master’s theses with us, which often helped me to gain a new perspective on our work. Thank you, Logeshwaran Thamilselvan, Mageshwar Selvakumar, Lisa Machata, Alexander Valyalkin, Lena Krauß, Sanu Pulimootil Achankunju, Moritz Berendes, Helga Denninger, Marina Engelhardt, Lukas Lamprecht, and Simon Gebauer.

As the proverb says “all work and no play makes Jack a dull boy”—thankfully I have never had to count that one of my problems. This is mainly due to my colleagues and friends, whom I would like to thank not only for offering their support, knowledge, and input during my work on this thesis, but also for the great times we had while discussing things other than work: Alexander Adami, Rachel Bittner, Sebastian Braun, Soumitro Chakrabarty, Esther Fee Feichtner, Johannes Fischer, María Luis Valero, Christian Helmrich, Colin Raffel, Michael Schöffler, Hendrik Schreiber, Fabian-Robert Stöter, Maja Taseska, Armin Taghipour, and Nils Werner.

[3]SIAMUS = Score-Informed Audio Parameterization of Music Signals.

[4]The International Audio Laboratories Erlangen are a joint institution of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and Fraunhofer IIS.

Last but not least, I want to express my deepest gratitude to my partner Annika Gräwe for her unconditional love and support throughout the course of this thesis and the ups and downs of life.

Chapter 2

Musical Background: The Freischütz Digital Project

This chapter is mainly based on the publications [87] and [88].

FREISCHÜTZ
DIGITAL

In this chapter, we present the musical background for this thesis provided by the project Freischütz Digital (Section 2.1). We then introduce the opera "Der Freischütz" and the different data modalities involved (Section 2.2), and finally describe audio-related tasks that played a role within the project (Section 2.3).

2.1 The Freischütz Digital Project

The project *Freischütz Digital* (FreiDi)[1] offered an interdisciplinary platform for musicologists and computer scientists to jointly develop and introduce computer-based methods that enhance human involvement with music.[2] The opera "Der Freischütz" by Carl Maria von Weber served

[1]Freischütz Digital: Paradigmatische Umsetzung eines genuin digitalen Editionskonzepts, BMBF Funding Code 01UG1239A to C)

[2]Various research groups from different disciplines were involved in the project. Musicologists worked on a digital edition of the musical score and the libretto of the opera, while computer scientists developed techniques

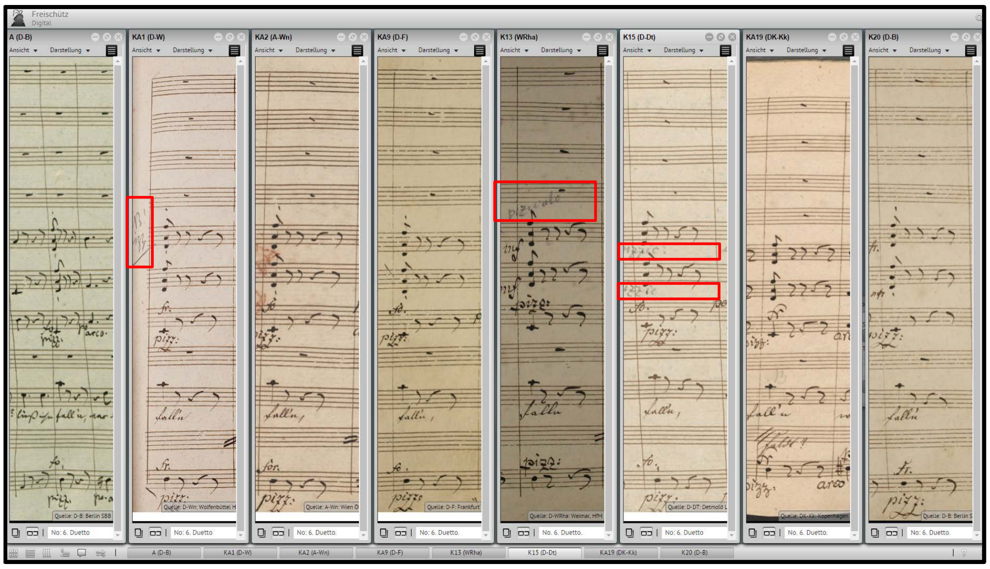


Figure 2.1. Screenshot of the *Edirom Online* software displaying multiple sheet music sources for measure 38 from No. 6 of Carl-Maria von Weber's "Der Freischütz". The left-most score shows the autograph (original version written by Weber), whereas the others are versions from copyists. In some of the copies, an additional "pizzicato" performance instruction, which is not present in the original autograph score, has been later added to the violins by pencil (see red rectangles). See [58] for a discussion of this example.

as an example scenario. This work plays a central role in the Western music literature and is of high relevance for musicological studies. Also, this opera was chosen because of its rich body of available sources—including different versions of the musical score, the libretto (literary text of the opera), and audio recordings.

A major goal of the project was the conception of a genuine digital edition of the opera "Der Freischütz" employing the new opportunities arising from the fields of digital humanities and music information retrieval. Traditionally, scholarly-critical editions in musicology aim at creating a coherent version of the musical work that is likely to be reflecting the intentions of the composer using (mostly) analog media. To achieve this, an editor takes into account historic sources of the work, e.g. the *autograph score* (the version of the score originally written by the composer) and copies of it, as well as letters and texts about the musical work itself. As a final result, a version of the musical score is published together with a critical commentary of the editor in which he justifies the decisions made during the editing process. The new applications possible through the use of digital technologies can improve the usability of such an edition.

For example, an editor can use and provide a side-by-side comparison of different sources on a computer screen as shown in Figure 2.1 (a screenshot of the Edirom Online[3] software showing

and tools for processing, comparing and presenting the data. The following institutions were involved in the project: Musikwissenschaftliches Seminar Detmold/Paderborn Institut für Musikwissenschaft, Goethe-Universität Frankfurt, Institut für Informatik, Universität Paderborn, International Audio Laboratories Erlangen.

[3]http://edirom.de/

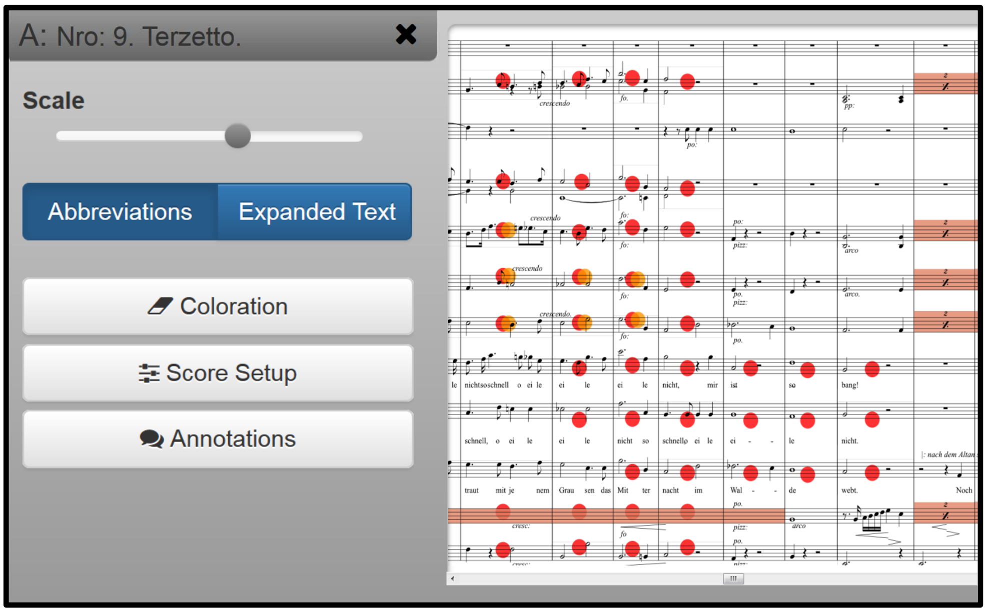


Figure 2.2. Dynamic score rendering as provided by Johannes Kepper. The screenshot shows the dynamic score rendering tool in which a score can be arbitrarily zoomed, voices can be shown/hidden, and annotations can be dynamically visualized in the score. The red dots mark editorial comments, whereas the other colorations mark tremolo instructions and abbreviations in the score.

an excerpt of "Der Freischütz"[4]). Furthermore, using a digital edition, editors can compare and point to relevant source material by means of visualizations and annotations which can be displayed dynamically in the musical score. Figure 2.2 shows an example of a dynamic score rendering system, where the user can arbitrarily zoom in and out of the score, as well as visualizing annotations or highlight specific score elements[5]. A user of such a digital edition studying a section of the musical score could then transparently follow the editor's comments and the relevant source material. On the side of the libretto, a central goal was the work on a genetic edition of the text which aims at highlighting the different text stages the main libretto text has gone through. Figure 2.3 shows an interface where a user can display the different text stages[6]. Furthermore, the inter-textually shared topics and relations have been annotated and incorporated into a Topic Map system which allows a user to search for topics and display their relationship in-between the different libretto text, see Figure 2.4[7]. Note that this encoding and annotation process usually involve a lot of manual work and is not as trivial as a simple text search, since the same concepts may be spelled or named differently in the various libretto texts.

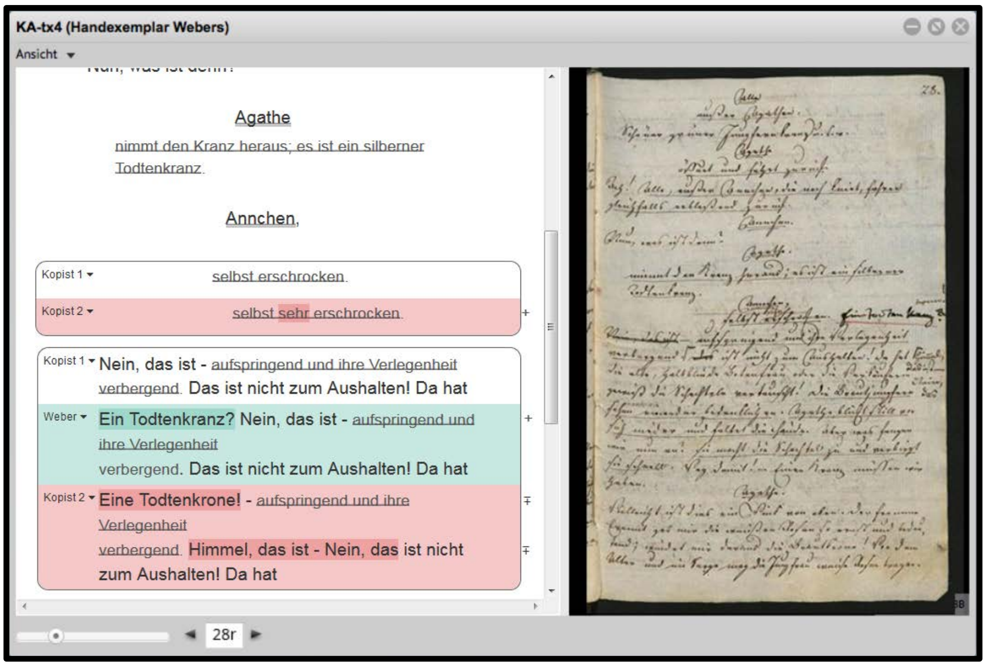
Another goal leading towards a digital edition was to explore techniques for establishing a virtual

[4]The digital edition of "Der Freischütz" can be accessed at `http://www.freischuetz-digital.de/edition/`

[5]This demo has been developed by Johannes Kepper and can be accessed at `http://www.freischuetz-digital.de/demos/dynamicScoreRendering/index.html`

[6]This demo has been developed by Solveig Schreiter and Daniel Röwenstrunk and can be accessed at `http://www.freischuetz-digital.de/demos/libretto/index.html`

[7]This demo has been developed by Benjamin Bohl and Janette Seuffert and can be accessed at `http://www.freischuetz-digital.de/exist/apps/topicMapViz/index.html`



Agathe

nimmt den Kranz heraus; es ist ein silberner Todtenkranz.

Annchen.

Kopist 1 selbst erschrocken.

Kopist 2 selbst sehr erschrocken.

Kopist 1 Nein, das ist - aufspringend und ihre Verlegenheit verbergend. Das ist nicht zum Aushalten! Da hat

Weber Ein Todtenkranz? Nein, das ist - aufspringend und ihre Verlegenheit verbergend. Das ist nicht zum Aushalten! Da hat

Kopist 2 Eine Todtenkrone! - aufspringend und ihre Verlegenheit verbergend. Himmel, das ist - Nein, das ist nicht zum Aushalten! Da hat

Figure 2.3. Genetic text edition as provided by Solveig Schreiter and Daniel Röwenstrunk. The above screenshot illustrates the genesis of the libretto text through the different text stages.

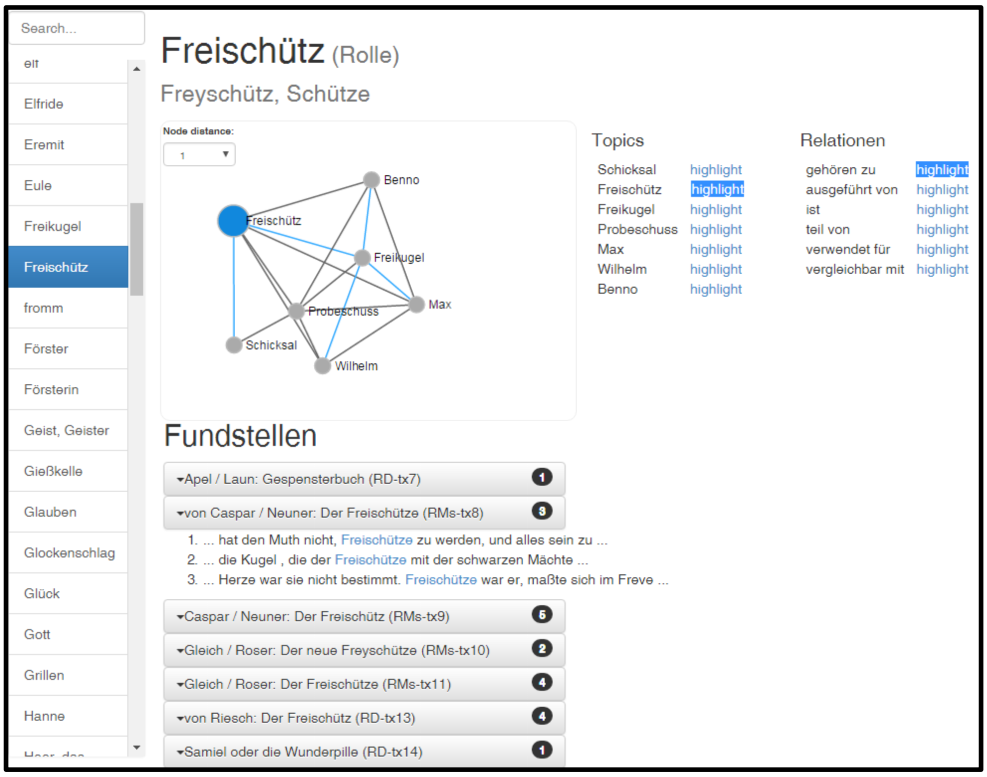
archive of relevant digitized objects, including symbolic representations of the autograph score and other musical sources (encoded in MEI[8]), transcriptions and facsimiles of libretti and other textual sources (encoded in TEI[9]) as well as (multitrack) audio recordings of the opera (see Figure 1.1). A more abstract goal within the Computational Humanities was to gain a better understanding of how automated methods may support the work of a musicologist beyond the development of tools for mere data digitization, restoration, management, and access. One particular challenge of the project was to investigate how automated methods and computer-based interfaces may help to coordinate the multiple information sources.

2.2 Musical Background and Data

Music is complex and manifested in many different formats and modalities [67, 79] (see Figure 1.1). In the opera "Der Freischütz", we encounter a wide variety of multimedia representations, including *textual* representations in form of the libretto (literary text of the opera), *symbolic* representations (musical score), *acoustic* representations (audio recordings), and *visual* representations (video recordings). In the following, we give some background information on "Der Freischütz" while discussing how different music representations naturally appear in

[8]MEI stands for the music encoding initiative, which is an open-source effort to define a system for encoding musical documents in a machine-readable structure [51, 103, 109]. See also `http://music-encoding.org/`

[9]TEI stands for the text encoding initiative [56]. See also `http://www.tei-c.org/`



Freischütz (Rolle)

Freyschütz, Schütze

Topics		Relationen	
Schicksal	highlight	gehören zu	highlight
Freischütz	highlight	ausgeführt von	highlight
Freikugel	highlight	ist	highlight
Probeschuss	highlight	teil von	highlight
Max	highlight	verwendet für	highlight
Wilhelm	highlight	vergleichbar mit	highlight
Benno	highlight		

Fundstellen

- Apel / Laun: Gespensterbuch (RD-tx7) 1
- von Caspar / Neuner: Der Freischütze (RMs-tx8) 3
  1. ... hat den Muth nicht, Freischütze zu werden, und alles sein zu ...
  2. ... die Kugel , die der Freischütze mit der schwarzen Mächte ...
  3. ... Herze war sie nicht bestimmt. Freischütze war er, maßte sich im Freve ...
- Caspar / Neuner: Der Freischütz (RMs-tx9) 5
- Gleich / Roser: Der neue Freyschütze (RMs-tx10) 2
- Gleich / Roser: Der Freischütze (RMs-tx11) 4
- von Riesch: Der Freischütz (RD-tx13) 4
- Samiel oder die Wunderpille (RD-tx14) 1

Figure 2.4. Topic Maps of Freischütz libretto sources as provided by Janette Seuffert and Benjamin Bohl. The topic Freischütz is highlighted as blue circle and the relation "gehören zu"—meaning "belongs to"—is highlighted as blue lines.

various formats and multiple versions in the context of this opera.

Composed by Carl Maria von Weber, "Der Freischütz" is a German romantic opera (premiere in 1821), which plays a key role in musicological and historical opera studies. The overture is followed by 16 numbers in the form of the German "Singspiel," where the music is interspersed with spoken dialogues [129]. This kind of modular structure allows an opera conductor for transposing, exchanging, and omitting individual numbers, which has led to many different versions and performances of "Der Freischütz".

As for text-based documents, there are detailed accounts on Friedrich Kind's libretto and its underlying plot, which is based on an old German folk legend (e.g., [114]). Since its premiere, the libretto has undergone many changes that were introduced by Kind, not to speak of changes made for individual performances. Furthermore, there are versions of the opera in other languages such as French, Russian, or Italian based on translated versions of the libretto. Finally, there exists a rich body of literature on the opera's reception.

With regards to the musical score, there exists a wide range of different sources for the opera. For example, variations have resulted from copying and editing the original autograph score. Changes were not only made by Weber himself, but also by copyists who added further perfor-

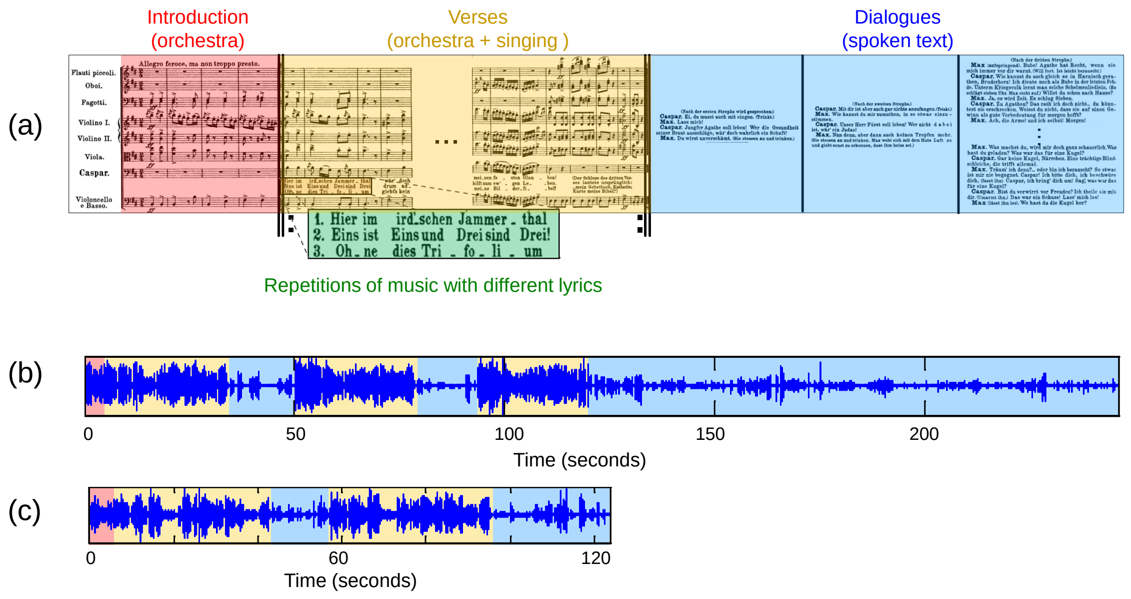

Introduction (orchestra)
Verses (orchestra + singing)
Dialogues (spoken text)
(a)
1. Hier im ird'schen Jammer - thal
2. Eins ist Eins und Drei sind Drei!
3. Oh - ne dies Tri - fo - li - um
Repetitions of music with different lyrics
(b)
0
50
100
150
200
Time (seconds)
(c)
0
60
120
Time (seconds)

Figure 2.5. Different representations of the song "Hier im ird'schen Jammerthal" (No. 4) of "Der Freischütz." **(a)** Score representation. In this song, after an intro (red), the repeated verses (yellow) are interleaved with spoken dialogues (blue). According to the score, there are three verses. **(b)** Waveform of a recorded performance conducted by Carlos Kleiber. The performance follows the structure specified by the above score. **(c)** Waveform of a recorded performance conducted by Otto Ackermann. In this performance, the structure deviates from the score by omitting the second dialogue and the third verse as well as by drastically shortening the final dialogue.

mance instructions and other details to clarify Weber's intention. A scholarly-critical edition of Weber's work[10] keeps track and discusses these variations. The recent Music Encoding Initiative (MEI) aims at developing representations and tools to make such enriched score material digitally accessible. Furthermore, there are various derivatives and arrangements of the opera such as piano transcriptions (e. g., by Liszt) or composed variants of the originally spoken dialogues (e. g., by Berlioz).

As mentioned above, our main focus in this thesis is the audio domain for which the opera "Der Freischütz" also offers a rich body of available sources, including a large number of recorded performances by various orchestras and soloists. For example, the catalogue of the German National Library[11] lists 1200 entries for sound carriers containing at least one musical number of the opera. More than 42 complete recordings have been published and, surely, there still exist many more versions which were produced for radio and TV broadcasts. Appendix B gives an overview of the recordings used in this project.

The opera covers a wide range of musical material including arias, duets, trios, and instrumental pieces. Some of the melodic and harmonic material of the numbers is already introduced in the

[10] `http://www.weber-gesamtausgabe.de/en/`
[11] `http://www.dnb.de/EN/`

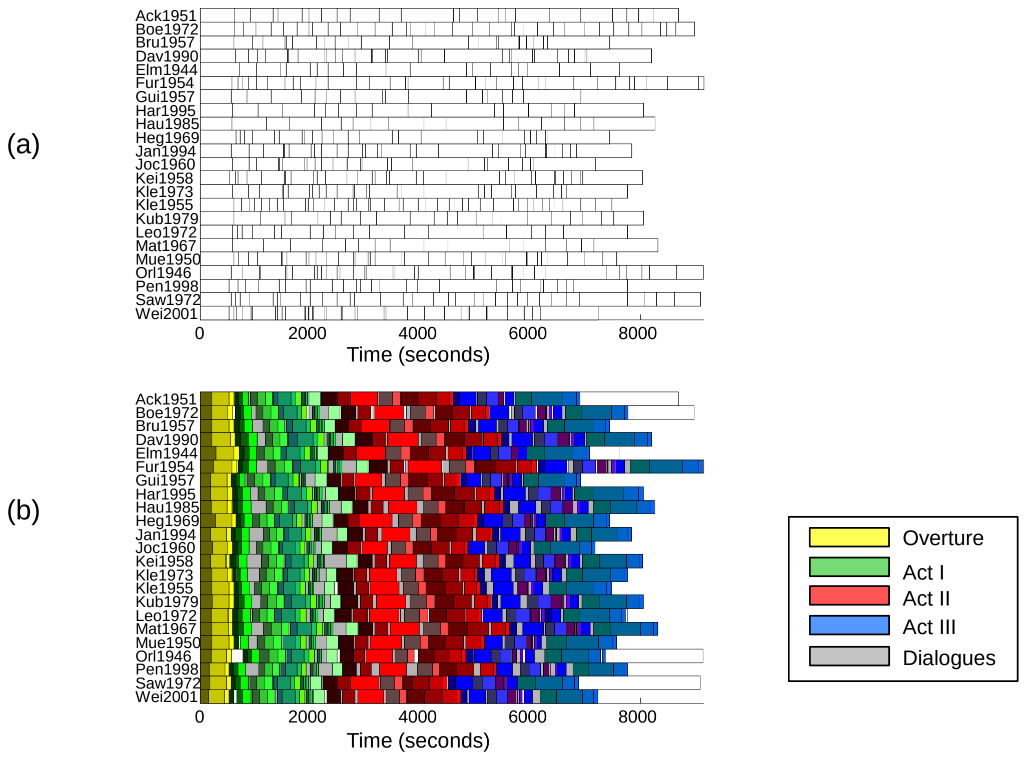


Figure 2.6. Segmentation of 23 different versions of "Der Freischütz" obtained from commercial CD recordings. **(a)** Segmentation according to the original CD tracks. **(b)** Segmentation according to a reference segmentation specified by a musicologist. The reference segmentation includes 38 musical sections as well as 16 spoken dialogue sections (gray).

overture. Furthermore, there are numbers containing repetitions of musical parts or verses of songs. The various performances may reveal substantial differences not only because of the variations mentioned above in the score and libretto, but also because a conductor or producer may take the artistic freedom to deviate substantially from what is specified in the musical score. This is illustrated in Figure 2.5 that indicates the structure of No. 4 of "Der Freischütz": an introduction and three verses that are interspersed with dialogues (Figure 2.5). Furthermore, two audio versions of No. 4 are shown with their structure (Figure 2.5b+c), where the second one deviates from the structure specified in the musical score by omitting the second dialogue and the last verse. Besides differences in the number of repetitions played, further deviations include omissions of entire numbers as well as significant variations in the spoken dialogues. For example, the recording of "Der Freischütz" conducted by Elmendorff in 1944 (Elm1944) omits the dialogues completely (see Elm1944 in Figure 2.6b). Apart from such structural deviations, audio recordings of the opera usually differ in their overall length, sound quality, language, and many other aspects. For example, the available recordings show a high variability in their duration, which can be explained by significant tempo differences and also by omissions of material (see Figure 2.6). Historic recordings in particular may be of poor acoustic quality

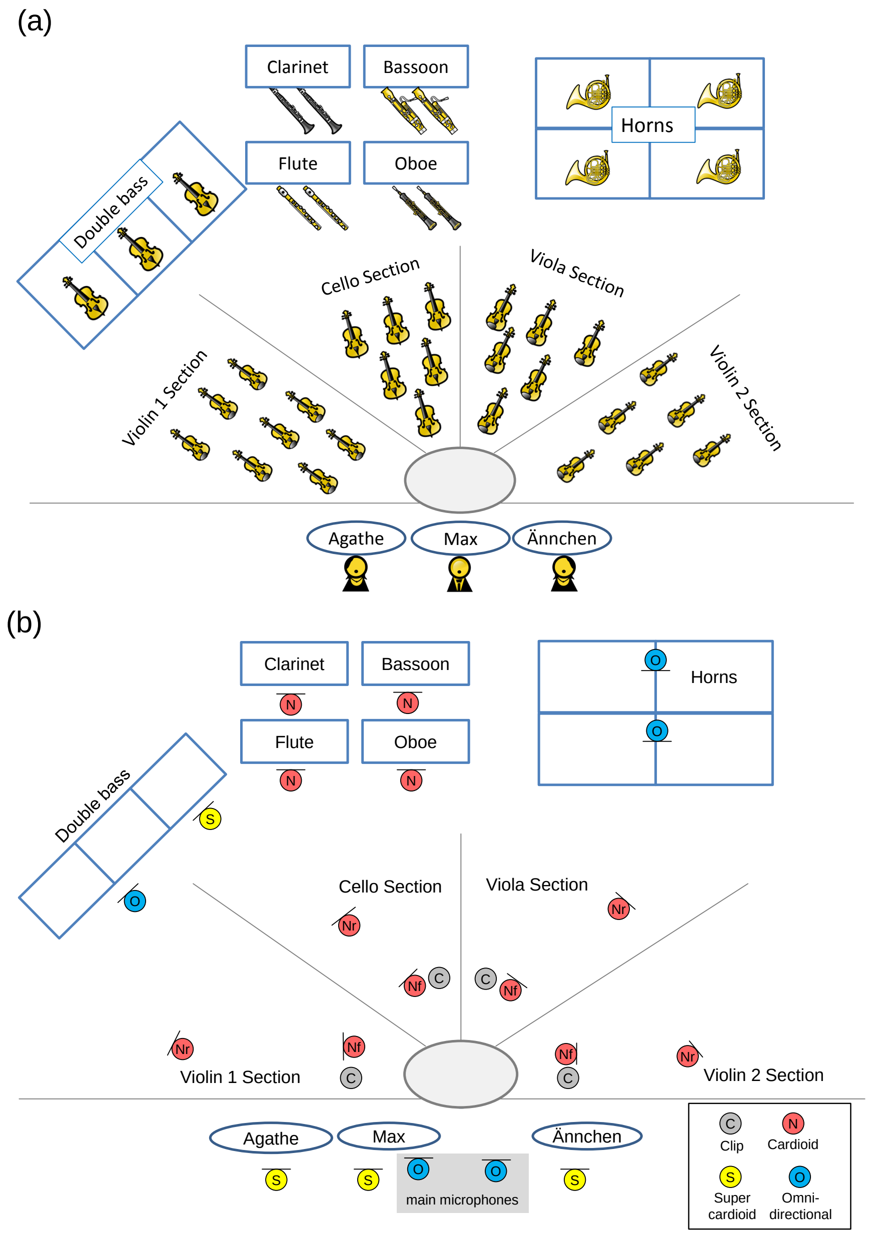


Figure 2.7. Recording setup used in the Freischütz project. **(a)** Seating plan (German/European style). **(b)** Setup of the 25 microphones used in the recordings, involving two main microphones for recording a stereo image and at least one spot microphone for each instrument section. For each string section, a spot microphone at the front (Nf) and at the rear (Nr) position was used. Additionally, clip microphones (C) were used for principal musicians of the string sections.

due to noise, recording artifacts, or tuning issues (also partly resulting from the digitization process). Working out and understanding the variations and inconsistencies within and across the different sources was a major task we addressed in this project.

Within FreiDi project, a professional recording of No. 6 (duet), No. 8 (aria), No. 9 (trio) of "Der

Freischütz" was produced in cooperation with Tonmeister students from the Erich-Thienhaus-Institute (ETI) in Detmold. The main purpose for the recording sessions was to produce royalty free audio material that can be used for demonstration and testing purposes. The generated audio material contains multitrack recordings of the raw microphone signals (one audio track for each microphone) as well as stereo mixes of specific instrument sections and a professionally produced stereo mix of the whole orchestra. Additionally, several variants of the musical score that are relevant for the scholarly edition were recorded to illustrate how these variants sound in an actual performance [58].[12]

Orchestra recordings typically involve a huge number of musicians and different instruments. Figure 2.7a shows the orchestra's seating plan that indicates where each voice (instrument section or singer) was positioned in the room. The seating plan also reflects the number of musicians that were playing in each instrument section. Overall, 44 musicians were involved in the recording session. For large-scale ensembles such as orchestras, interaction between the musicians is very important. For example, each instrument section has a principal musician who leads the other musicians of the section. To make this interaction possible, the different voices are usually recorded in the same room simultaneously. Figure 2.7b shows the microphones used for the different voices and their relative position in the room. Two main microphones were used for recording a stereo image of the sound in the room. For capturing the sound of individual voices, at least one additional spot microphone was positioned close to each voice.

For some of the instrument sections, additional spot microphones were used, see Figure 2.7b. The first violin section, for example, was recorded with three microphones: one at the front position, one at the rear position, and a clip microphone attached to the principal musician's instrument. The audio tracks recorded by the spot microphones allow a sound engineer to balance out the volume of the different voices in the mixing process. However, this balancing is limited as each microphone not only records sound from its dedicated voice, but also from all others in the room (see Figure 2.9a). This results in recordings that do not feature isolated signals, but rather mixtures of a predominant voice with all others being audible through what is referred to as *interference, bleeding, crosstalk*, or *leakage*. Further information on the FreiDi multitrack recordings can be found in Appendix A.

[12]The recordings are available for download at
`http://audiolabs-erlangen.de/resources/MIR/FreiDi/MultitrackDataset/`
For additional audio examples and a further discussion of the production, we refer to
`http://freischuetz-digital.de/audio-recording-2013.html` and
`http://freischuetz-digital.de/audio-production-2014.html`

2.3 Audio-Related Tasks

While our project partners focused on the encoding and processing of text- and score-based representations, our main objective was to research on ways that improve the access to audio-based material. To this end, we applied techniques from signal processing and music information retrieval to automatically process the music recordings.

A musical work is typically organized in a hierarchical fashion starting from high-level structures such as the movements of a symphony, over mid-level structures such as the parts associated with a musical form, down to low-level structures that correspond to note or even sub-note events. The automatic segmentation of the audio data into musically meaningful structural elements constitutes a central task in the FreiDi project. Within this scenario, we now highlight some high-, mid-, and low-level segmentation tasks that were relevant within the project.

On the high-level side, one first task is to automatically segment a given recorded performance into sections that correspond to the various numbers of the opera. Such sections often correspond to tracks as found on CD recordings of the opera. Using audio material obtained by ripping CD recordings, this problem seems to be already solved. However, in practice one often starts with a single audio file containing the entire opera (e. g., an audio track obtained from a video recording), which then needs to be segmented. Furthermore, the track segmentation found on different CDs is far from being unique (see Figure 2.6). In particular, in some recordings the dialogues appear as separate tracks and in others they are concatenated with musical tracks of the opera's numbers. Finally, different performances of the "Der Freischütz" can differ substantially: Dialogues are often shortened, and certain numbers may even be missing. In the case that both a reference version and a reference track segmentation of the opera are available, one can use cross-version retrieval techniques to consistently segment other unsegmented versions. Here, the reference tracks function as "queries" to identify corresponding sections in an unsegmented version, which can then be segmented accordingly, see also Chapter 5.

Once the high-level track segmentation has been determined, each track can be further subdivided into mid-level musical sections that reflect the musical form of the respective number. For example, these sections can be the individual verses of a given song (as in the example of Figure 2.5) or the subdivision of the overture into the slow introduction, the fast middle part, and the fortissimo concluding part. In general, the task of recovering a description of the musical form from a given music recording is referred to as *audio structure analysis*, see, e. g., [94, 95]. Again, as a major challenge, one needs to deal with musical and acoustic variabilities that may concern the tempo, the sound of instruments, room acoustics, just to name a few. This requires musically informed audio features that capture the respective desired musical properties (e. g. harmony, tempo, timbre), see [94, 95].

On the low-level side, a key issue for exploiting the rich body of different sources are alignment

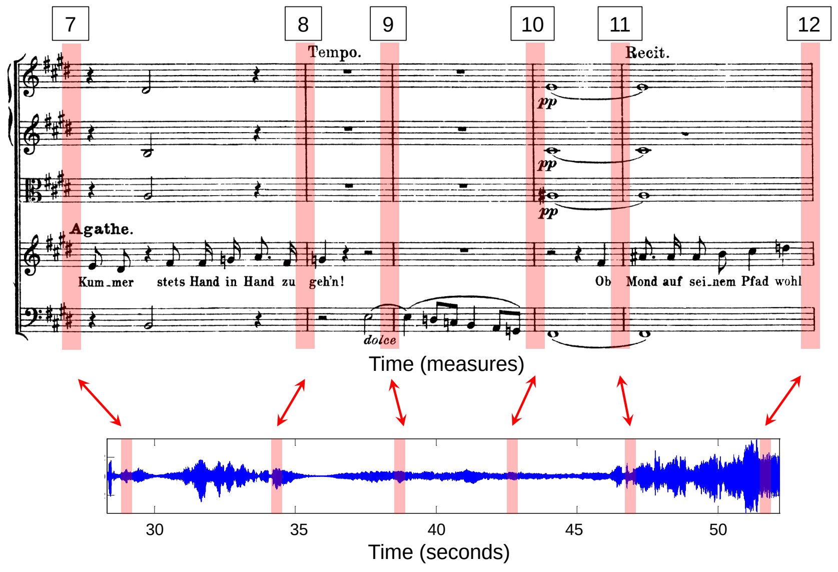


Figure 2.8. Measure-wise alignment between a sheet music representation and an audio recording. The links are indicated by the bidirectional red arrows.

or synchronization techniques which aim to identify and link semantically corresponding events present in different representations [54, 2, 36, 18, 43, 57]. Depending on the respective data types, one can distinguish between different synchronization scenarios. For example, in *SheetMusic-Audio* synchronization the objective is to link regions (given as pixel coordinates) within the scanned images of a given sheet music representation to semantically corresponding physical time positions within an audio recording. Such linking structures are useful as navigation and browsing aids in the context of digital music libraries, e. g., to highlight the current position in the scanned score during playback of the recording [18]. In *SymbolicScore-Audio* synchronization, the goal is to link time positions in a symbolic score representation (specified on a musical time axis given in measures) with corresponding time positions of an audio recording (see Figure 2.8). In *Audio-Audio* synchronization, the task is to time align two different audio recordings of a piece of music. These alignments can be used to jump freely between different interpretations, thus making efficient and convenient audio browsing possible. Such synchronization tasks are well-defined and tractable as long as the two versions which shall be aligned are structurally similar so that any event in one version has a counterpart in the other version. Synchronization becomes much more difficult if one only has sparse information or partial similarities. This is the case in *Lyrics-Audio* synchronization with the goal to align given lyrics to an audio recording of the underlying song, which is useful not only for retrieval but also for karaoke applications. Here, the localization of sung lyrics in complex polyphonic music turns out to be a very challenging problem, which can be alleviated when exploiting additional structural or harmonic information [43].

To approach the above mentioned segmentation problems within the FreiDi project, we adapt

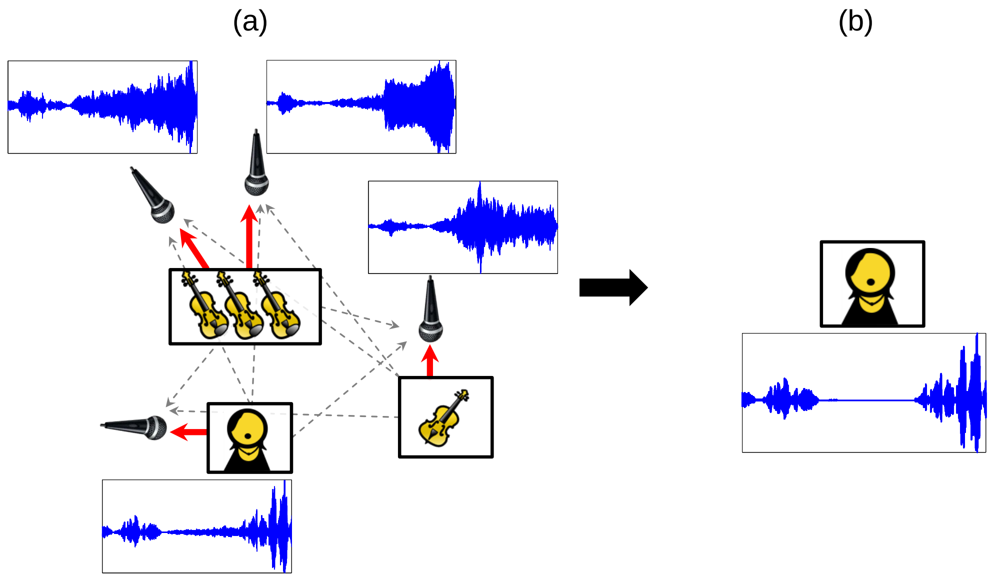


Figure 2.9. **(a)** Illustration of interference problem in a recording with three voices (violin section, bass, singing voice). A solid line (red) indicates that a voice is associated to a microphone, a dashed line (gray) indicates interference from another voice into a microphone. Each voice is associated with at least one of the microphone channels. **(b)** Interference reduced version of the singing voice signal.

existing synchronization and segmentation techniques for generating various linking structures across the different versions and representations of the opera, see Chapter 4 and Chapter 5.

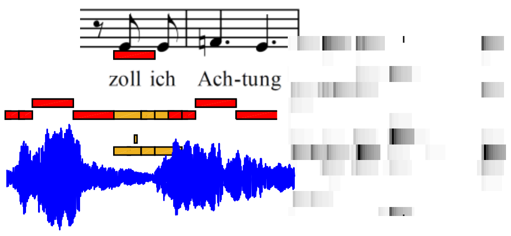
Another central task is the isolation of individual musical voices from a recording. As mentioned above, a recording of an opera involves a large number of different instruments. In such a recording, listening only to the melodic lines of the violins for example, is usually not possible. Having this option can be useful for pedagogical reasons[13], e.g. when a student wants to study a recording, or for "music minus one" applications (mixtures where a particular voice has been removed). In music source separation, the objective is to separate a sound mixture into its constituent components [126]. In our FreiDi scenario, the goal would be to separate a recording into the individual voices (instrument sections and singers). Audio source separation in general is a very difficult problem where performance is highly dependent on the signals considered. However, recent studies demonstrate that separation methods can be very effective if prior information about the signals is available (see e.g. [69] and references therein). A problem closely related to source separation is interference reduction where the objective is to reduce unwanted interferences from recordings, e.g. removing the orchestra instruments from the singer's microphone, see Figure 2.9. In Chapter 7, we present a method that aims to reduce interference in multitrack recordings to recover only the isolated voices. Here, we exploit the fact that each voice can be assumed to be predominant in its dedicated microphones. Figure 2.9b shows an

[13]`http://freischuetz-digital.de/audio-production-2014.html` provides examples of mixtures using the Freichütz Digital Multitrack Dataset that feature only specific instrument sections.

example of an interference reduced version of the singing voice signal from Figure 2.9a. Especially in the middle of the corresponding waveforms, it is easy to spot differences. In this region, there was no singing voice in the recording. Hence, the recorded signal in this region originated entirely from interference of other instrumental voices.

Chapter 3

Technical Background: Audio Signal Processing



In this chapter, closely following the notions in [77], we introduce fundamental concepts from audio signal processing that are used throughout this thesis. We first introduce music representations like *audio signals*, *sheet music*, and *MIDI* (Section 3.1). Then, we review the *Fourier transform* and the *short-time Fourier transform* (STFT), which are indispensable tools in audio signal processing (Section 3.2). Finally, building on the STFT, we introduce musically motivated audio features that correlate with the musical aspects of *pitch* and *harmony* (Section 3.3).

3.1 Music Representations

Music can be represented in many different ways. We first introduce audio signals, a representation that allows for the closest reproduction of recorded music (Section 3.1.1). In audio signals, most musical parameters such as tempo, pitch, and loudness are only implicitly given.

Another representation is sheet music, which specifies musical parameters such as tempo, pitch, or loudness more explicitly (Section 3.1.2). These parameters, however, are usually specified on a relative scale which leaves some room for interpretation to the musician. Finally, we introduce MIDI (Musical Instrument Digital Interface), which can be seen as a representation between audio and sheet music, as it has both a musical and a physical time axis, and furthermore specifies performance parameters such as tempo, pitch, or loudness in an explicit and reproducible way (Section 3.1.3).

3.1.1 Audio Signal

An audio signal encodes the variation of air pressure over time that we can perceive as sound. Following [77, Section 2.2.1], a *continuous-time* (CT) or *analog* audio signal is a function $f\colon \mathbb{R} \to \mathbb{R}$ that assigns an amplitude value $f(t) \in \mathbb{R}$ to each point in time $t \in \mathbb{R}$. This formulation allows us to model infinitesimally small changes both in time and amplitude and therefore to represent arbitrary sound signals. However, when using digital technology, only a finite number of parameters can be stored and processed [77, Section 2.1.3]. Therefore, analog audio signals need to be converted into finite representations. Two main steps are involved in this process. First, the signal's time axis needs to be *sampled* at a finite set of discrete time positions, and second, its amplitude needs to be *quantized* to a finite set of values. This process is referred to as *digitization.* We will exemplary explain the sampling process in the form of *equidistant sampling* [77, Section 2.2.2.1], and refer to [77, Section 2.2.2.2] for the quantization.

The main idea of *equidistant sampling* is to only retain values of a given CT-signal at time positions that are integer multiples of a sampling period T—this process is therefore often called *T-sampling.* Formally, given an analog signal $f\colon \mathbb{R} \to \mathbb{R}$ and a positive real number $T > 0$, one defines a function $x\colon \mathbb{Z} \to \mathbb{R}$ by

$$x(n) = f(n \cdot T) \tag{3.1}$$

for $n \in \mathbb{Z}$. Since x is only defined at discrete points in time, it is called a *discrete-time* (DT) signal. The value $x(n)$ is called a *sample.* The inverse of the sampling period T

$$F_s = 1/T \tag{3.2}$$

is called the *sampling rate* of the DT signal x. The sampling rate specifies the number of samples per second and is measured in Hertz (Hz). In practice, we can only store a finite number of samples in a computer, and therefore restrict the DT signal x to be finite. We will only consider a finite set of samples $x(0), x(2), \ldots, x(N-1)$ and assume that $x(n) = 0$ for $n \notin [0 : N-1]$, with $[0 : N-1] := \{0, 1, \ldots, N-1\}$.

Figure 3.1a shows an example of a DT signal with a sampling rate of $F_s = 22050$ Hz and a

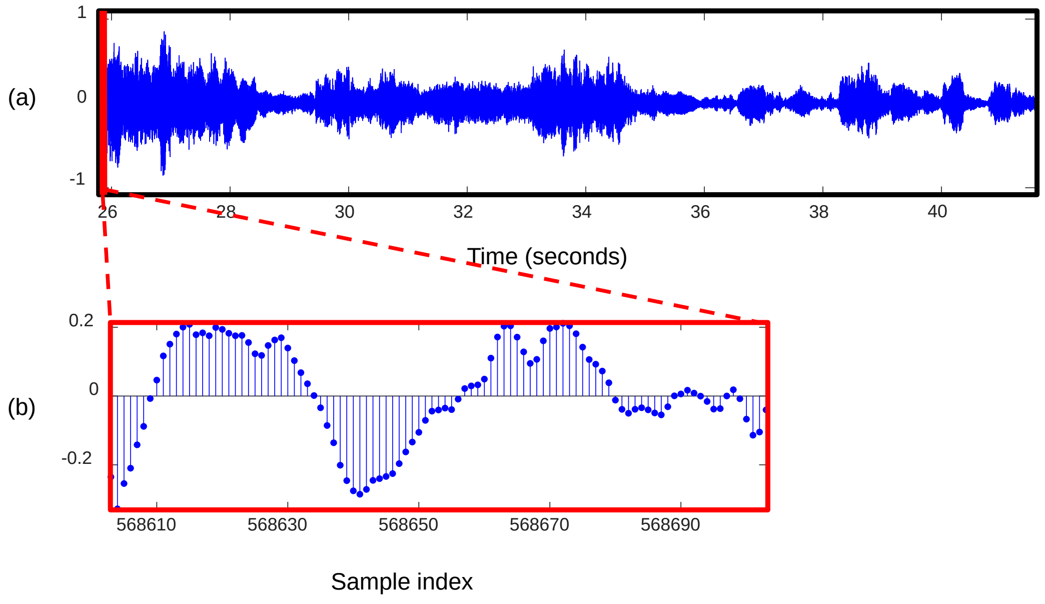


Figure 3.1. Audio signal corresponding to measures 15–23 from No. 6 of Carl Maria von Weber's "Der Freischütz" conducted by Bloemeke (2013). The sampling rate is $F_s = 22050$ Hz. **(a)** Full excerpt with time axis in seconds. **(b)** Short excerpt showing only 100 samples.

duration of 16 seconds. Figure 3.1b shows an excerpt of 100 samples of the signal where the individual samples can be seen.

3.1.2 Sheet Music

Sheet music is a performance instruction for musicians. It specifies musical parameter such as pitch, duration, tempo, or loudness explicitly. Figure 3.2 shows an excerpt of the orchestral score from Carl-Maria von Weber's opera "Der Freischütz". This special type of score is meant to be used by a conductor who needs to have an overview of the different instruments playing in the piece. The score shows 11 *staves*. A score usually defines a *time signature*, which indicates the duration of a measure and beat. This specifies how the musical time axis (measured in measures or beats) is divided. The score in Figure 3.2 has a time signature of 6/8, meaning that each measure contains notes with a total duration of six eighth notes and that the beat has a length of three eighth notes (this is a convention for this time signature). The measures are separated by vertical lines through the staffs, also referred to as *bar lines*.

Each staff usually contains the musical material for one instrument or instrument section. For example, the first staff contains the melodic material of the flute. A staff is made of five lines. The lines and the positions between and above the lines encode the pitch height according to a clef. For example, the G-clef (also called treble clef), see the first symbol in the flute's staff

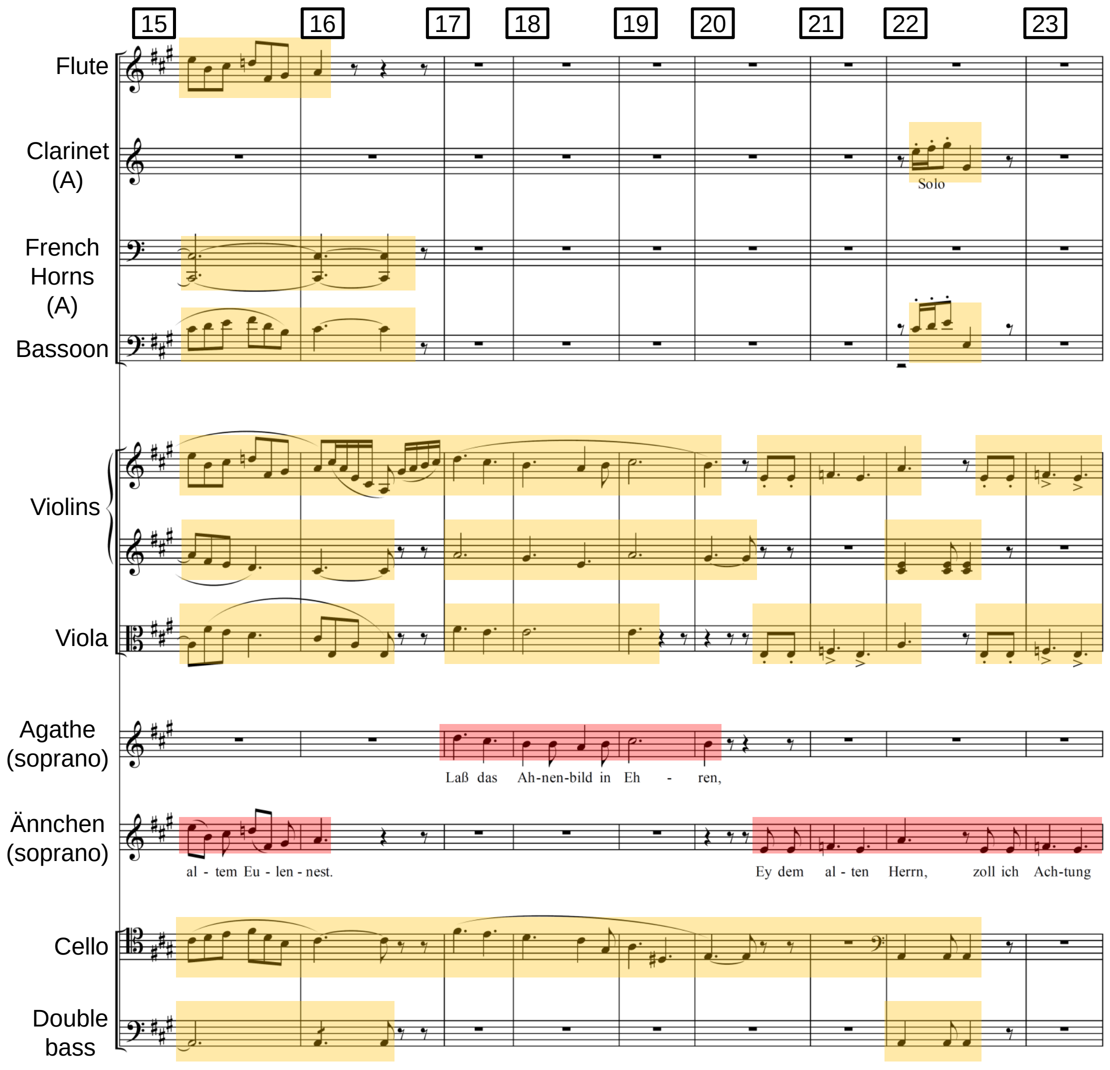


Figure 3.2. Score excerpt: measures 15–23 from No. 6 of "Der Freischütz".

in Figure 3.2, defines that the pitch G4 (in Scientific Pitch Notation [77, Section 1.1.1]) lies on the second lowest line of a staff. The F clef (also called bass clef), see first symbol in the double bass' staff, defines that the pitch F3 lies on the second highest line of a staff.

The *note symbols* 𝅝,𝅗𝅥,♩,♪,𝅘𝅥𝅯 encode the duration of a whole, half, quarter, eighth, and sixteenth note, respectively. Their horizontal position indicates their rhythmical position, whereas their vertical positioning in the staff lines encodes the pitch. In measure 15 in Figure 3.2, the flute and the soprano singer "Ännchen" play both the notes E5, B4, C♯5 D5 F♯4 G♯4 which have each a duration of an eighth note. Rest symbols 𝄻,𝄽,𝄾,𝄿 are used to indicate that an instrument does not play at a certain position. For example the clarinet in Figure 3.2 plays only in measure 22, and has rests in all other measures.

For an extended introduction of sheet music, music notation, and some musical background, we

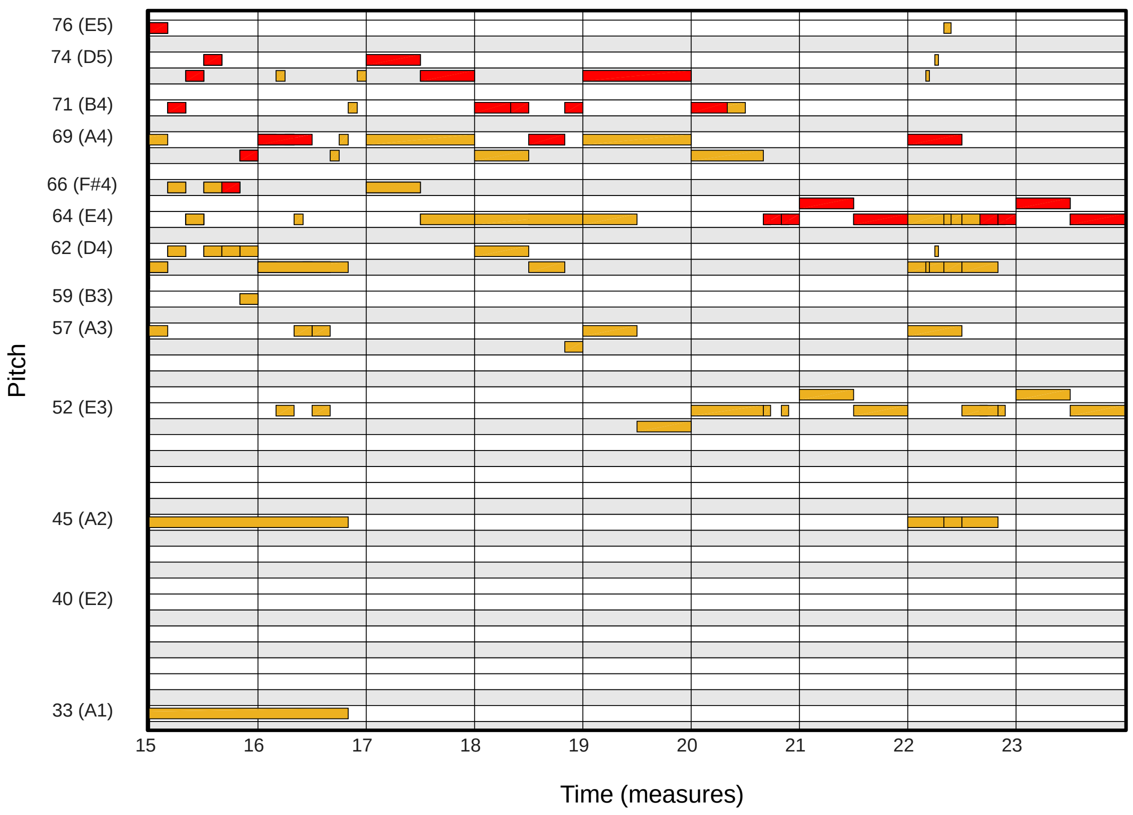


Figure 3.3. Piano roll visualization of a MIDI excerpt for No. 6 of Carl Maria von Weber's "Der Freischütz" (measures 15–23). The rectangles represent individual notes and their duration. The orchestra is colored in yellow and the singing voices in red. Note that not all notes from each instrument are visible separately as some instruments play the same notes. For example the flute and the soprano voice (Ännchen) have the same melody in measure 15. Individual notes are represented by rectangles. The singing voice's track is indicated in red.

refer to [77, Section 1.1]

3.1.3 MIDI

MIDI (Musical Instrument Digital Interface) is a protocol that was designed to control digital instruments and to enable communication between them [77, Section 1.1.2]. In the form of MIDI files, it is often used to encode symbolic music and performance data in music sequencing programs. It is important to note that MIDI is not a music notation format. However, it has been widely used as an exchange format for symbolic music data. The basic information stored in a MIDI file is called *message*. For our purpose, the most important messages are *note onset* events and *note offset* events. They both have a time stamp and carry pitch information given by a *MIDI note number*, and volume information given by the *note velocity*, both ranging from 0 to 127. Furthermore, they also carry a channel information. Channels are associated to *program numbers* that are mapped to a specific instrument or synthesizer. The MIDI note numbers

$p \in [0:127]$ correspond to the pitches C0 ($p = 0$) to G$^\sharp$9 ($p = 127$). The MIDI events are given on an abstract time axis measured in *ticks*. Through events that define the length of a tick in millseconds and in quarter notes, it can both be linked to the physical time axis and the musical time axis. The frequency measured in Hertz (Hz) associated to a MIDI pitch is given by

$$F_{\text{pitch}}(p) = 2^{(p-69)/12} \cdot 440 \,, \tag{3.3}$$

with $p \in [0:127]$. Furthermore, MIDI can encode different *tracks* which are used to group the notes that belong to a single instrument or instrument section.

Figure 3.3 shows a *piano roll* visualization of a MIDI file on the musical time axis (in measures) where a rectangle encodes the onset time, offset time, and the corresponding pitch of a note. The different orchestra instruments are colored in yellow, and the singing voices in red.

Note that a MIDI file has no sound. It can be sonified (leading to an audio signal) by using a digital instrument or synthesizer. With its more exactly specified performance parameters, and its musical and physical time axis, MIDI can be used as a bridge between audio signals and sheet music.

It is important to note the differences between sheet music and a recorded performance in form of a MIDI or audio signal. Sheet music leaves room for interpretation to the performer. A performer may play some notes a bit longer or shorter, or choose the onset of a note a bit earlier or later. This may even vary if the same performer plays the piece a second time.

3.2 Fourier Transform and Spectrograms

The *Fourier Transform* is one of the most important tools in audio signal processing. It allows for transforming a signal from the time domain into the frequency domain. Following [77, Section 2.4.2], the *discrete Fourier transform* of length $N \in \mathbb{N}$ for a DT-signal x is defined as

$$X(k) = \sum_{n=0}^{N-1} x(n) \exp(-2\pi i k n / N) \tag{3.4}$$

for $k \in [0 : N-1]$. The complex value $X(k)$ is called *Fourier coefficient*. Intuitively, the magnitude of $X(k)$ encodes how well the signal coincides with a sinusoidal of a given frequency

$$F_{\text{coef}}(k) = \frac{k \cdot F_s}{N} \,. \tag{3.5}$$

The Fourier transform is an invertible transform, which can be transformed back to a time

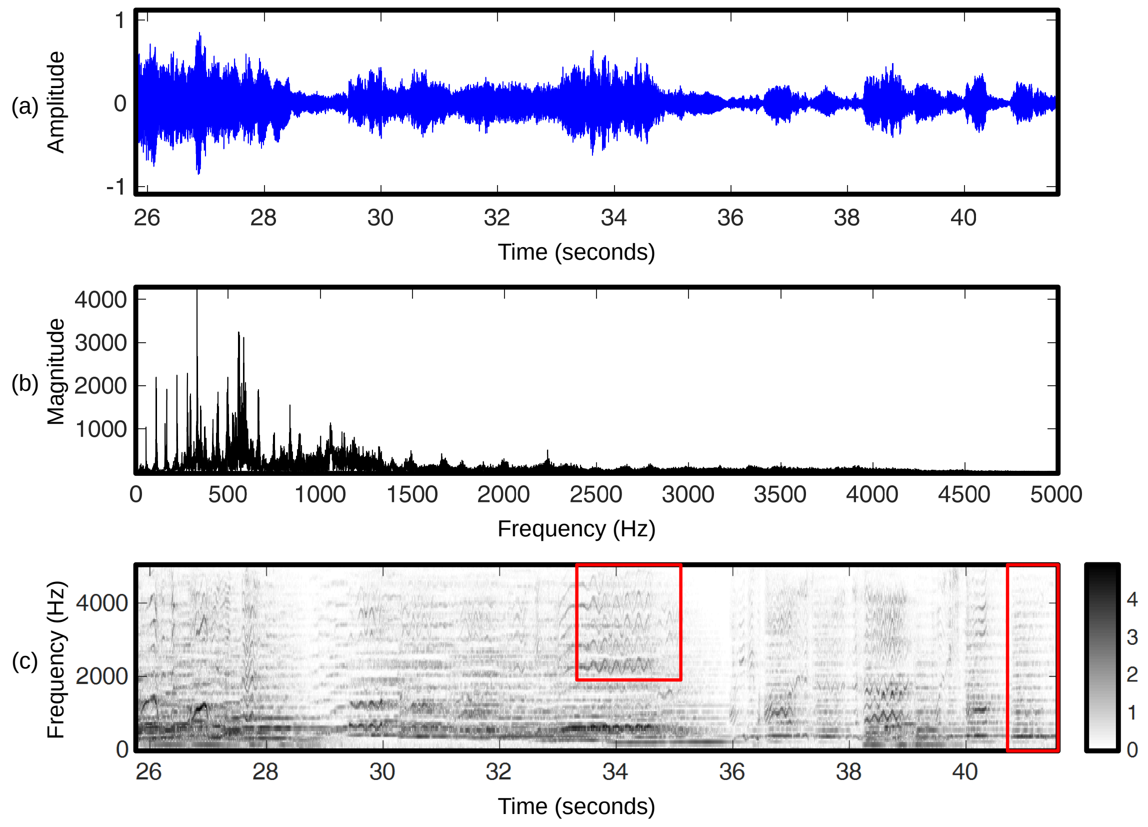


Figure 3.4. Signal representations of measures 15–23 from No. 6 of Carl Maria von Weber's "Der Freischütz" conducted by Bloemeke (2013). **(a)** Waveform. **(b)** Magnitude Fourier spectrum. **(c)** Magnitude spectrogram Y_{mag}. We applied a log compression to the values for visualization purpose.

domain signal by the *inverse Fourier transform*:

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) \cdot \exp(2\pi i k n / N) \,, \tag{3.6}$$

for $n \in [0 : N-1]$.

Figure 3.4a shows the waveform of a signal, and Figure 3.4b the magnitude Fourier spectrum. In the waveform, one can roughly see when a sound event is happening, but not its frequencies, whereas in the Fourier representation, one can see the frequency components of the signal but not at which time they occur. This illustrates a main drawback of the Fourier transform, namely its lack of time information. We know which frequencies are present in the signal, but not when. This issue is addressed by the *short-time Fourier transform* (STFT) [45]. The main idea is to apply the Fourier transform on small excerpts of the signal which we call *frames*. A frame has the length of L samples which are usually weighted by a window function $w\colon \mathbb{Z} \to \mathbb{R}$. We assume that $w(n) = 0$ for $n \notin [0 : L-1]$. The STFT of a signal is then defined by

$$X(m,k) = \sum_{n=0}^{L-1} w(n) \cdot x(n+mH) \cdot \exp(-2\pi ikn/L) \ , \tag{3.7}$$

where $m \in \mathbb{Z}$ is the *frame index*, $k \in [0 : L-1]$ is the *frequency index*, and H the *hop size*. In the following, we assume that the physical point in time corresponding to a frame is its center. To assign the first frame in Equation (3.7) to the physical time of zero seconds, we assume that the signal x has been shifted by half a window length to the right [77, Section 2.5.3]. In practice, this is often realized by *zero padding* the signal with $\lfloor L/2 \rfloor$ zeros at the beginning. Each frame index m can now be associated to a physical time position by

$$T_{\text{coef}}(m) := \frac{m \cdot H}{F_s} \ . \tag{3.8}$$

Similarly to Equation (3.5), we can compute the frequency in Hertz (Hz) associated to a frequency index k by

$$F_{\text{coef}}(k) := \frac{k \cdot F_s}{L} \ . \tag{3.9}$$

The magnitude of an STFT is referred to as *magnitude spectrogram*:

$$Y_{\text{mag}}(m,k) := |X(m,k)|. \tag{3.10}$$

In signal processing, often also the *power spectrogram* is considered, which is simply the square of the magnitude spectrogram:

$$Y_{\text{pow}}(m,k) := |X(m,k)|^2. \tag{3.11}$$

Figure 3.4c shows the STFT magnitude spectrogram for our example with a window size $L = 1024$ and a hop size of $H = 128$ samples. Opposed to the Fourier spectrum, we can now see when a certain frequency is active in the signal. Furthermore, we can even spot frequency components that are associated to specific voices. For example, around 34 seconds, there are wave-like structures stemming from the vibrato of the singer.

For a more in-depth discussion of the Fourier transform and discrete-time signal processing, we refer to [77, 91, 90]

3.3 Chroma-Based Feature Representations

The relevant information for most music processing tasks is only implicitly given in the audio signal. For processing and analyzing music signals, an important step is thus to extract *features* capturing aspects of music signals that are relevant for approaching a given task.

For example, the harmonic content of a music signal is an important aspect for tasks such as structure analysis, music segmentation, or music synchronization [123, 116, 95, 8, 77, 94].

Before we can compare an audio signal to a MIDI representation, we need to transform the given representations into suitable mid-level feature representations that facilitate a direct comparison. Features that are often used for modeling the harmonic content of a signal are *chroma features* [77]. Chroma features (often also called *pitch class profiles* [44, 49]) transform a given signal into a *time-chroma* representation. A time-chroma representation reduces all occuring piches into the twelve pitch classes C, C$^\sharp$, D, D$^\sharp$, E, F, F$^\sharp$, G, G$^\sharp$, A, A$^\sharp$, B by ignoring octave information.

In the following, we first introduce a *pitch-based log frequency spectrogram* which serves as an intermediate step for the chroma computation (Section 3.3.1). We then introduce chroma features (Section 3.3.2) and different chroma variants such as the chroma energy normalized statistics (CENS) features (Section 3.3.3) and chroma onset features (Section 3.3.4).

3.3.1 Log-Frequency Spectrogram and Pitch Features

As indicated by Equation 3.3, the relation between frequency and musical pitch is logarithmic. However, the spectrogram introduced in Equation (3.2) has a linearly spaced frequency axis. Following [77, 3.1.1], we explain how such a spectrogram can be transformed into a *time-pitch representation.*

First, we use Equation (3.3) to compute a set of associated frequency bins for a given pitch p

$$P(p) = \{k : F_{\mathrm{pitch}}(p - 0.5) \leq F_{\mathrm{coef}}(k) < F_{\mathrm{pitch}}(p + 0.5)\} \;, \tag{3.12}$$

with $p \in [0 : 127]$ [77, Section 3.1.1]. For a given time frame and pitch p, we compute a *log-frequency* spectrogram $Y_{\mathrm{LF}} : \mathbb{Z} \times [0 : 127] \to \mathbb{R}_{\geq 0}$ by simply adding up all the frequency bins associated to p:

$$Y_{\mathrm{LF}}(m, p) := \sum_{k \in P(p)} Y_{\mathrm{pow}}(m, k). \tag{3.13}$$

This spectrogram has now a frequency bin for each MIDI pitch. Figure 3.5a shows an example of a pitch-based log-frequency spectrogram for our running example.

Note that a single musical note played by an instrument is usually a mixture of frequencies [77, Section 1.3]. This makes the task of extracting the pitch corresponding to a musical note from a spectrogram much more complex than computing a log-frequency spectrogram, see e.g. [29].

Another approach to derive a time-pitch representation uses filter banks for the individual pitches [85] to compute *short-time mean square power pitch* (STMSP) features. Here, after using the filter banks for decomposing the signal into pitch bands, the short-time power is computed on subbands of the signals in a frame-wise fashion. Note that this is very similar to the approach described above, as the computation of the STFT power spectrogram results in a frame-wise computation of the signal's power in all frequency components of the STFT. The grouping of the STFT's frequencies into pitches results thus in a similar representation as the STMSP features.

3.3.2 Chroma Features

Two pitches sound similar when they are an octave apart from each other (12 tones in the equal tempered scale). We say that these pitches share the same chroma or pitch class. Chroma features exploit the above observation, by adding up all pitches that belong to the same chroma. Following [77], we derive chroma features from the pitch-based log frequency spectrogram by

$$\mathcal{C}(m,c) = \sum_{\{p\in[0:127]\,:\,p \bmod 12 = c\}} Y_{\mathrm{LF}}(m,p)\ , \tag{3.14}$$

for $c \in [0:11]$ and $m \in \mathbb{Z}$.

Figure 3.5 visualizes the computation of chroma features from a pitch-based spectrogram. In Figure 3.5a, the $\mathrm{C}^\sharp$ pitches are marked by red rectangles in the pitch-based log-frequency spectrogram. Figure 3.5b shows the resulting chromagram, where the resulting $\mathrm{C}^\sharp$ chroma bin is also marked by a red rectangle. In the marked region, the $\mathrm{C}^\sharp$ chroma is dominating. The corresponding pitches are played by the bassoon and the second violin. Other instruments play pitches whose overtones contribute to this chroma bin, e.g. the french horns playing an A1 and A2.

3.3.3 CENS Features

In this section, we describe *chroma energy normalized statistics* (CENS) features that were introduced in [83]. These features have been proven particularly useful in audio matching tasks [76, 63]. The basic idea of CENS features is to make chroma features invariant to local tempo variations and to changes in the dynamics by temporally smoothing and normalizing a chroma representation.

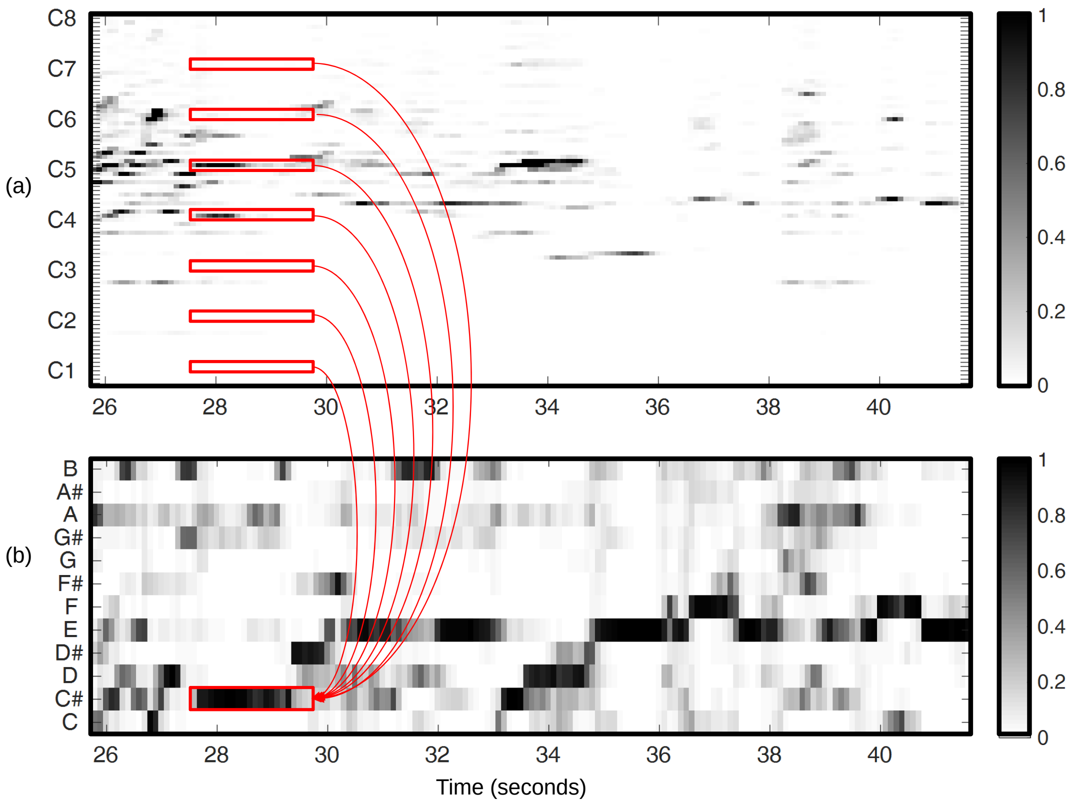


Figure 3.5. Feature representations of measures 15–23 from No. 6 of Carl Maria von Weber's "Der Freischütz" conducted by Bloemeke (2013) **(a)** Pitch-based log-frequency spectrogram Y_{LF}. **(b)** Chromagram $\mathcal{C}$.

First, each chroma vector in $\mathcal{C}$ is normalized with respect to the ℓ^1-norm. The resulting feature vectors are then quantized in a logarithmic fashion inspired by the logarithmic sensation of the sound intensities in the human ear, see [76] for details. Then, the chroma vectors are smoothed with a Hann window of length s, and downsampled by a factor of d. Finally, each chroma vector is ℓ^2 normalized. We refer to the resulting features by CENS_d^s

Figure 3.6a shows chroma features with a feature rate of 10 Hz. Figure 3.6b shows CENS_5^{21} with a feature rate of 2 Hz, derived from the 10 Hz chroma features. Through the smoothing over 21 of the original chroma vectors, they roughly cover 2.1 seconds of the audio material in each feature vector. Figure 3.6c shows CENS_{10}^{41} with a feature rate of 1 Hz, also derived from the 10 Hz chroma features. Here each feature vector covers roughly 4.1 seconds.

An advantage of CENS features is that different feature resolutions can be efficiently derived from the same chroma features. This saves the recomputation of the pitch-time representation on different resolution levels.

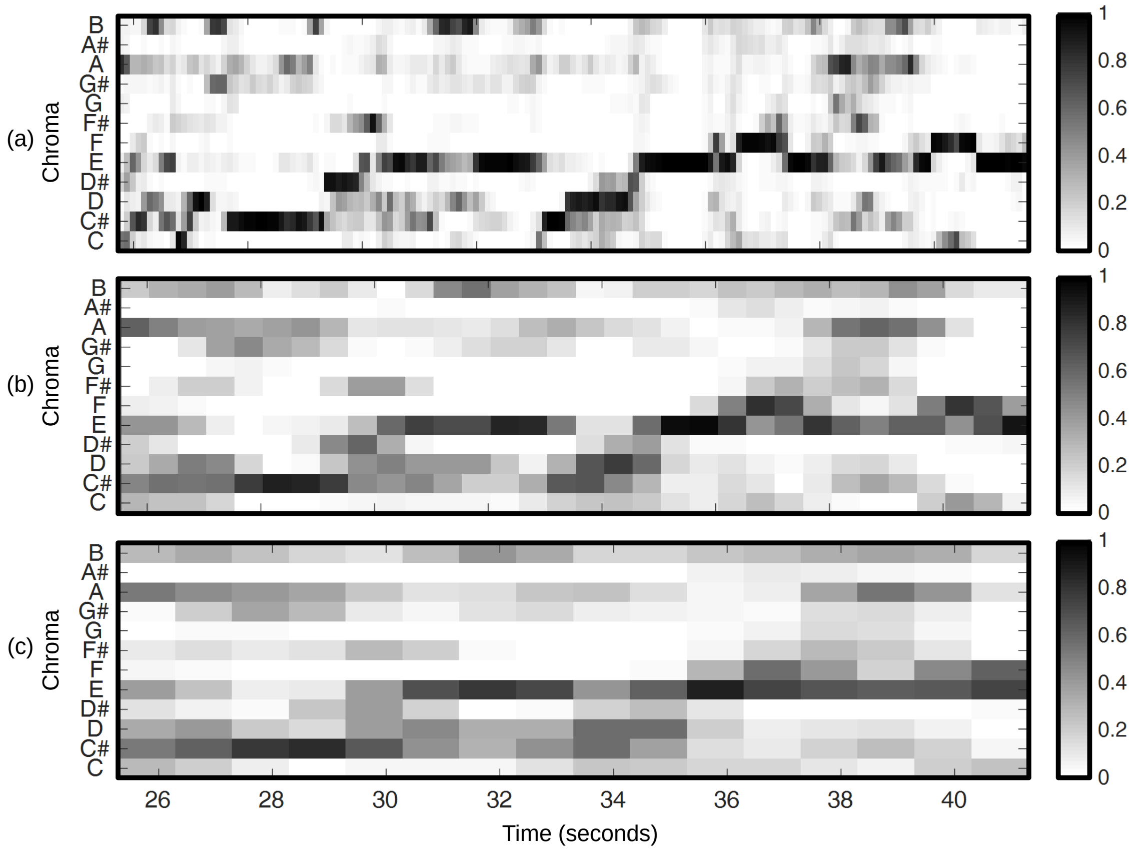


Figure 3.6. Feature representations on audio excerpt corresponding to measures 15–23 from No. 6 of Carl Maria von Weber's "Der Freischütz" conducted by Bloemeke (2013). **(a)** Chroma feature (10 Hz). **(b)** CENS_{5}^{21} (2 Hz) features derived from (a). **(c)** CENS_{10}^{41} (1 Hz) features derived from (a).

3.3.4 Chroma Onset Features

Another important aspect of musical notes is their onset time. Determining the onset of each note in a complex sound mixture is a very difficult task. However, combining a set of note onsets from different pitches into a chroma representation can stabilize this process. The main idea of chroma onset features is an onset representation for each chroma band. In the following, we describe how chroma onset features can be derived from a time-pitch representation like the pitch-based log-frequency spectrogram introduced in Section 3.3.1.

First, the temporal derivative of the time-pitch representation is computed. Then, the derivative is half wave rectified to account for the fact that a rise in energy is associated with note onsets. Before adding each pitch band into chroma classes, we apply a *logarithmic compression* to account for logarithmic sensation of sound intensity [36]. After this step, the chroma onset (CO) features are then similarly derived as chroma features by simply adding up the logarithmically scaled values corresponding to the same pitch class. The CO features are normalized by dividing each feature vector by the local maximum over the sequence of the ℓ^2-norms of the feature vectors

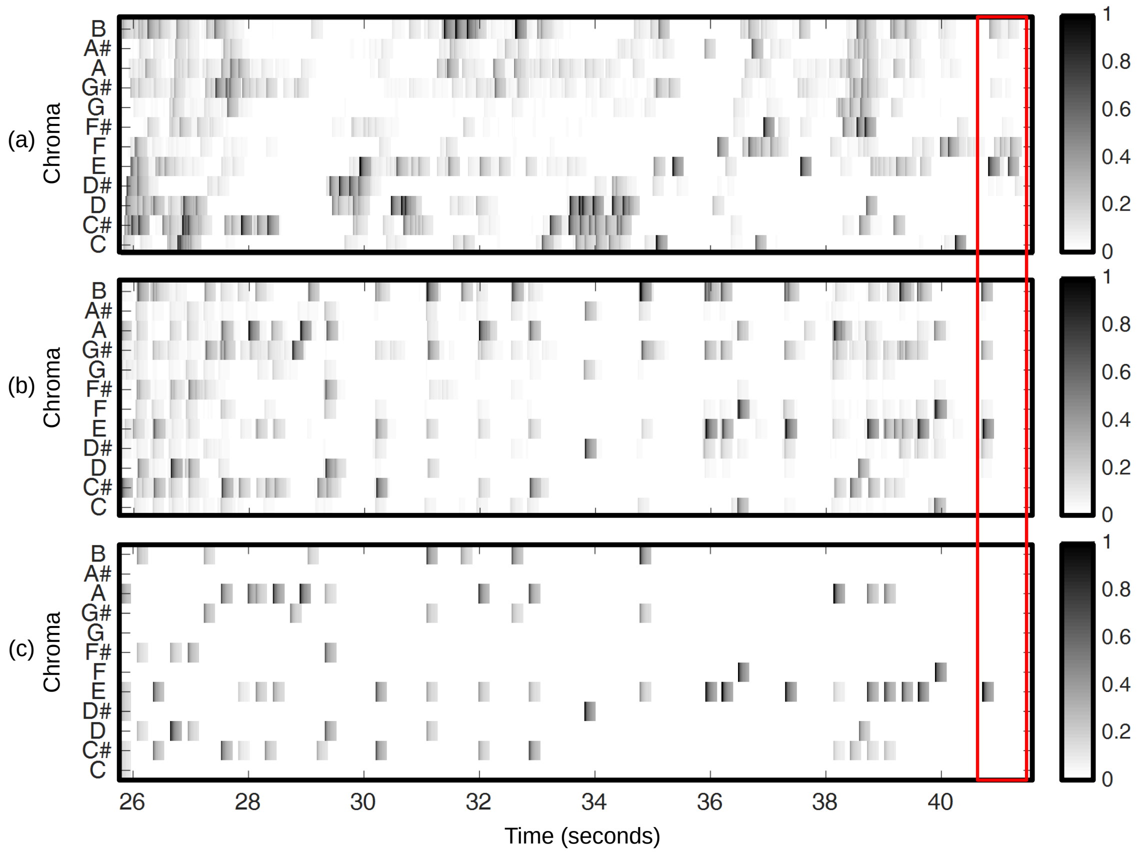


Figure 3.7. Decaying locally adaptive normalized chroma onset (DLNCO) features on excerpt corresponding to measures 15–23 from No. 6 of Carl Maria von Weber's "Der Freischütz" **(a)** DLNCO derived from audio recording conducted by Bloemeke (2013). **(b)** DLNCO derived from piano sonification of MIDI excerpt. **(c)** DLNCO derived from MIDI excerpt. The red rectangle marks the chroma stemming from the pitches E3 and E4 corresponding to measure 23.5 in the score.

within a two second window. Intuitively, this gives a lower weight to onsets with low energy in passages with high energy than in passages with low energy [36]. The resulting feature is called LNCO (locally adaptive normalized CO). In a final step, the decaying of a note is simulated by copying each LNCO feature vector n times and multiplying the resulting sequence with decreasing weight. The resulting feature is called DLNCO (decaying LNCO).

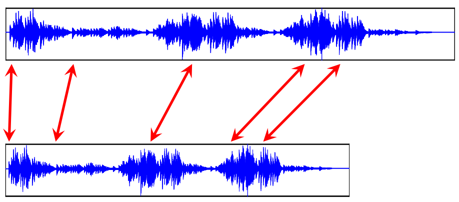
Figure 3.7 shows examples for DLNCO representations for an orchestra recording (Figure 3.7a), a piano recording (Figure 3.7b), and features directly derived from a MIDI representation (Figure 3.7c). The DLNCO features stemming from the MIDI representation are very clean as they were directly derived from the symbolic pitch information in the MIDI file. The piano recording that was synthesized from the MIDI shows onsets in chroma bands that are not present in the MIDI. These originate from overtones. For example, let us consider the position around 41 seconds in Figure 3.7c where only the E chroma band is active. At the same position in Figure 3.7b, the chroma band G$^\sharp$ and B are activated as well. These two chroma bands actually correspond to overtones of E.

DLNCO features have been successfully used to improve the temporal resolution of music alignments [36]. Similar features that are suitable for online music alignments have been proposed in [6].

Chapter 4

Efficient Music Synchronization

This chapter is mainly based on our contribution in [98].



A central task in the FreiDi project was to link the different information sources such as a given musical score and the many available audio recordings by developing and adapting synchronization techniques. In a typical music synchronization scenario, one has to deal with pieces of music having a duration of less then ten minutes, whereas in the case of a full opera, the recordings are much longer. For example, to synchronize two complete recordings of "Der Freischütz", one has to deal with roughly five hours of audio material (each recording being 2 and a half hours long), leading to computational challenges with regard to memory requirements and running time.

Dynamic time warping (DTW) is an established method for finding a global alignment between two feature sequences. However, having a computational complexity that is quadratic in the input length, memory consumption becomes a major issue when dealing with long feature sequences. Various strategies have been proposed to reduce the memory requirements of DTW. For example, online alignment approaches often have a constant memory consumption by applying forward path estimation or block-by-block processing strategies. However, this comes at the cost of robustness. Efficient offline DTW based on multiscale strategies constitutes another approach. While methods built on this principle are usually robust, their memory requirements are still dependent on the input length. By combining ideas from online alignment approaches and offline multiscale strategies, we introduce a novel alignment procedure that allows for specifying a constant upper bound τ on its memory requirements.

In this chapter, we introduce a memory-restricted alignment procedure that combines concepts from multiscale DTW (MsDTW) [86, 111] with the idea of using rectangular local constraint regions as already proposed for online alignment procedures [72]. We first introduce some basic notions (Section 4.2), review the basics of DTW (Section 4.3.1) and MsDTW (Section 4.3.2), and then describe our proposed memory-restricted MsDTW approach (MrMsDTW) in detail (Section 4.3.3). Finally, we discuss the influence of the parameter τ on the robustness of MrMs-DTW within a music synchronization application scenario (Section 4.4), and summarize our analysis (Section 4.5). In a final section, we describe additional application scenarios of music alignments (Section 4.6).

4.1 Related Work

The task of finding a global alignment between two feature sequences has received large research interest in the context of music information retrieval (MIR) and beyond. For example, the goal in music synchronization is to align musically corresponding positions in different representations of a piece of music. The different music representations may include sheet music (images), symbolic score data, or audio recordings. Many methods for *symbolic score-to-audio* alignment [19, 57, 107], *sheet music-to-audio* alignment [64], and *audio-to-audio* alignment [25, 86, 122] have been developed. Furthermore, alignment techniques are important in MIR tasks such as automatic accompaniment [20, 106] or score following [17, 92, 3]. Many different approaches have been used in alignment systems. For example, Hidden Markov Models or particle filters have been employed in score to audio alignment, where the current score position and tempo are modeled in a statistical sense [30, 75]. In this thesis, we focus on methods based on dynamic time warping (DTW) [25, 104, 77]. DTW is an effective technique to find an optimal alignment of two feature sequences. However, since the complexity of DTW is proportional to the product of the feature sequences' lengths [77], memory consumption becomes an issue when dealing with long feature sequences. But also aligning shorter sequences may become problematic with the standard DTW procedure when working for example on mobile devices, which usually only provide a limited amount of memory. To reduce the memory requirements of DTW, several strategies were proposed. On a global level, the Sakoe-Chiba band or the Itakura parallelogram impose a *constant global constraint region* on the set of possible alignments [104]. Using these constraints is problematic, as the optimal alignment may lie outside these regions. Another approach is to use *adaptive global constraints* such as in multiscale DTW (MsDTW) [86, 111]. Here, a projection of an alignment on a coarse feature resolution level (see Figure 4.1a) is used to constrain the

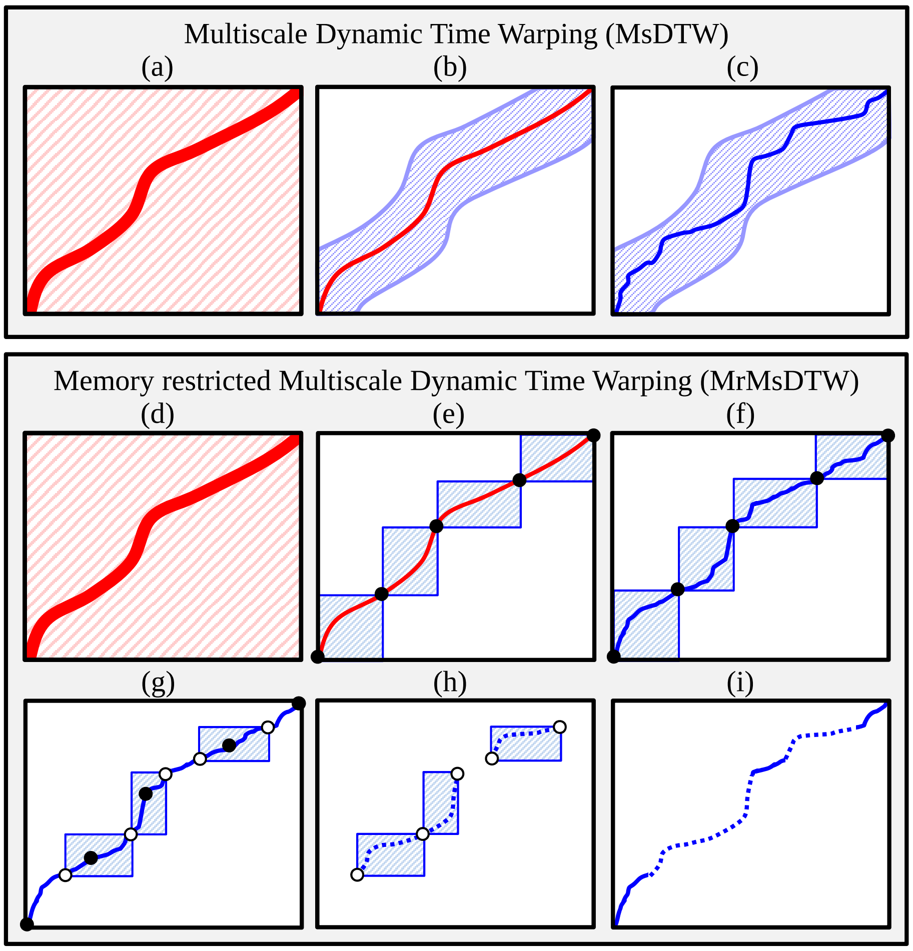

Multiscale Dynamic Time Warping (MsDTW)
(a)
(b)
(c)
Memory restricted Multiscale Dynamic Time Warping (MrMsDTW)
(d)
(e)
(f)
(g)
(h)
(i)

Figure 4.1. Illustration of MsDTW (a)-(c) with global tubular constraint regions and the proposed MrMsDTW (d)-(e) with rectangular local constraint regions. **(a)** Alignment on coarse feature resolution. **(b)** Projection of (a) onto finer level with resulting tubular constraint region. **(c)** Refinement of (a) on fine resolution within tubular constraint region. **(d)** Warping path P_C^* on coarse level (LEVEL C). **(e)** Projected warping path P_F on fine level (LEVEL F) and derived anchor sequence A. **(f)** First refinement of P_F by computing local paths within rectangular constraint regions defined by the anchor sequence A. **(g)** Derivation of the anchor sequence A' in a neighborhood of the anchor points in A. **(h)** Second refinement constrained by the rectangular constraint regions defined by A'. **(i)** Refined warping path P_F^*.

computation of a refined alignment on a finer feature resolution level. In the refinement, the alignment is restricted to lie within a tubular constraint region that is constructed from the projected alignment (see Figure 4.1b+c). Note that there are cases where alignments computed by MsDTW deviate from the globally optimal alignment, see [86, 111].

In online scenarios, *local constraints* are imposed on the alignment. Here, the alignment is usually computed using greedy forward path estimation [25] or by using block-by-block processing where a block defines a local rectangular constraint region [72].

In this chapter, we propose an MsDTW variant using the idea of building a global path from local alignments. Our memory-restricted MsDTW procedure, in the following called MrMsDTW, uses local rectangular constraint regions in the refinement step. The size, and therefore the required memory to store them, is bounded by a memory restriction parameter τ. The constraint regions are inferred from anchor points on the projected alignment (see Figure 4.1e). This yields a set of local alignments (see Figure 4.1f). In a second refinement step, these local alignments are corrected in a neighborhood around the initial set of anchor points. Again, this is done in rectangular constraint regions that are restricted in size by τ. This makes the memory requirement of our method basically constant in dependence of τ, whereas the memory requirement of MsDTW is linearly and DTW is quadratically growing with the lengths of the feature sequences (see Figure 4.4).

4.2 Basic Notions

Closely following the modeling in [77, 98], we introduce some basic notions that will be used to describe our proposed algorithm. Let $\mathcal{F}$ be a suitable feature space. Furthermore, let $X := (x_1, \ldots, x_N)$ and $Y := (y_1, \ldots, y_M)$ be feature sequences with $x_n, y_m \in \mathcal{F}$, where $n \in [1 : N] := \{1, \ldots, N\}$ and $m \in [1 : M]$. A *cell* is a tuple

$$p := (n, m) \in [1 : N] \times [1 : M]\,,$$

which encodes the correspondence between two feature vectors x_n and y_m. We define the operators

$$\pi_1(p) := n \text{ and } \pi_2(p) := m$$

to refer to the elements of a cell. A *path* is a sequence of cells

$$P = (p_1, \ldots, p_L)\,,$$

with $p_\ell \in [1 : N] \times [1 : M]$ for $\ell \in [1 : L]$ satisfying the *step size condition*

$$(p_{\ell+1} - p_\ell) \in \Sigma := \{(1, 0), (0, 1), (1, 1)\}\,.$$

We refer to the *length of a path* by

$$|P| := L\,.$$

A *warping path* is a path that additionally fulfills the *boundary conditions* $p_1 = (1, 1)$ and $p_L = (N, M)$ (see Figure 4.2a). Note that a warping path constitutes a global alignment between two feature sequences. We define the operator

$$h(P) := \lfloor |P|/2 \rfloor + 1$$

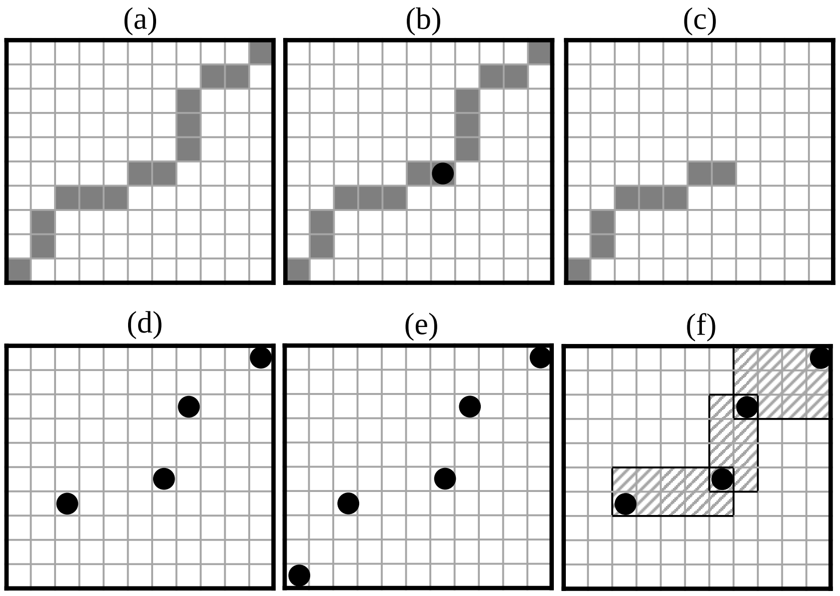


Figure 4.2. Illustration of basic concepts. **(a)** Warping path P of length $|P| = 14$. **(b)** Center $h(P) = 8$ of warping path P. **(c)** Subpath $P[1:8]$. **(d)** Anchor sequence $A \subseteq P$. **(e)** Anchor sequence fulfilling boundary conditions $A' \sqsubseteq P$. **(f)** Maximum rectangular extent $\mathcal{R}(A) = 12$ of anchor sequence A.

to return the center index of a sequence P (see Figure 4.2b). For retrieving the first, center, and last element of a path, we define the operators

$$B(P) := p_1,\ H(P) := p_{h(P)},\ E(P) := p_L\,,$$

respectively. Furthermore, we define a *subpath* as $P[i : j] := (p_i, \ldots, p_j)$ with $1 \le i < j \le |P|$ (see Figure 4.2c). An *anchor point* is a specified cell $a = (n, m)$ and an *anchor sequence* is a sequence of anchor points

$$A = (a_1, \ldots, a_K)$$

with $a_k \in [1 : N] \times [1 : M]$ that fulfills the condition of strict monotonicity $a_{k+1} - a_k \in \mathbb{N} \times \mathbb{N}$. We write

$$A \subseteq P$$

if all cells of A are also cells of P (Figure 4.2d). Furthermore, we write

$$A \sqsubseteq P$$

if $A \subseteq P$ and A additionally fulfills the boundary conditions $a_1 = p_1$ and $a_K = p_L$ (Figure 4.2e). Let

$$d_k := a_{k+1} - a_k + (1, 1).$$

We define the *maximum rectangular extent* of an anchor sequence A as

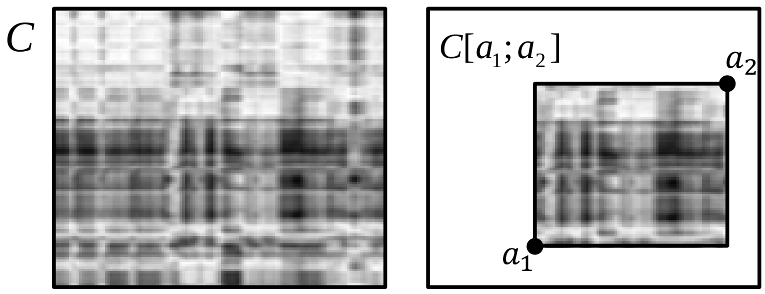


Figure 4.3. Full cost matrix C (left) and a local cost matrix $C[a_1; a_2]$ (right), constrained by the anchor points a_1 and a_2.

$$\mathcal{R}(A) := \max_{k\in[1:K-1]} \pi_1(d_k) \cdot \pi_2(d_k)\,,$$

which is the maximum rectangular area spanned between consecutive anchor points in A (Figure 4.2f). This concept will later be used to make the memory restriction of the algorithm explicit. Finally, to compare two features in $\mathcal{F}$, we use a local cost measure $c\colon \mathcal{F} \times \mathcal{F} \to [0, 1]$. By comparing each pair of elements in the sequences X and Y, we obtain the *cost matrix*

$$\mathbf{C}(n, m) := c(x_n, y_m).$$

Given two anchor points a_1 and a_2, we define the *local cost matrix*

$$\mathbf{C}[a_1; a_2] := \mathbf{C}(n, m)_{\substack{\pi_1(a_1)\le n\le\pi_1(a_2),\\ \pi_2(a_1)\le m\le\pi_2(a_2)}}\,,$$

which is a submatrix of $\mathbf{C}$ (Figure 4.3).

4.3 Efficient DTW Procedures

In this section, we review the classical DTW approach to alignment (Section 4.3.1), sketch the previously introduced multiscale DTW (Section 4.3.2), and introduce our proposed algorithm in (Section 4.3.3).

4.3.1 Classical Dynamic Time Warping (DTW)

The goal of DTW is to compute an optimal warping path between the two feature sequences X and Y with respect to the cost measure c. The total alignment cost of a warping path P

between two sequences X and Y is defined by

$$c_P(X,Y) := \sum_{\ell=1}^{L} \mathbf{C}(\pi_1(p_\ell), \pi_2(p_\ell))\,.$$

We say that a warping path P^* is optimal, if it has minimal alignment costs

$$P^* := \min\{c_P(X,Y) | P \text{ is warping path}\}\,.$$

To derive an optimal warping path, we first compute the *accumulated cost matrix* $\mathbf{D}$ given by

$$\mathbf{D}(n,m) = \min_{(i,j)\in\Sigma}\{\mathbf{D}\,(n-i, m-j) + w_{ij}\mathbf{C}(n,m)\}$$

with $\mathbf{D}(n,1) = \sum_{k=1}^{n} \mathbf{C}(k,1)$ for $n \in [1:N]$, $\mathbf{D}(1,m) = \sum_{k=1}^{m} \mathbf{C}(1,k)$ for $m \in [1:M]$, and $w_{ij} \in \mathbb{R}$ being a local weight parameter. The optimal warping path can be obtained by backtracking the steps through $\mathbf{D}$, see [104] for details. Note that if $w_{ij} = 1$ for all $(i,j) \in \Sigma$, the DTW is biased towards diagonal steps. To lower this bias slightly, we set the weights to $(w_{01}, w_{10}, w_{11}) := (1.5, 1.5, 2)$. Note that the time and space complexity of the algorithm to compute $\mathbf{D}$ and the optimal warping path is $O(NM)$.

4.3.2 Multiscale DTW (MsDTW)

In this section, we review the basic concepts of MsDTW originally introduced in [86, 111]. MsDTW aims to reduce the computational requirements of DTW by first computing an alignment on a coarse feature resolution level (Level C). The coarse alignment is then projected onto a finer feature resolution level (Level F) and refined using a tubular constraint region.

More precisely, let P_C^* be an optimal warping path computed on Level C (see Figure 4.1a). The optimal warping path P_C^* is projected onto Level F, resulting in a (potentially non-optimal) warping path P_F (see Figure 4.1b). To compute an optimal warping path P_F^* on Level F, a tubular constraint region is constructed by adding $\delta \in \mathbb{N}$ cells to the left, top, right and bottom of P_F. Within this constraint region, P_F^* is computed via DTW (see Figure 4.1c). The procedure can be recursively applied by introducing further coarse resolution levels. Note that, depending on the choice of δ, the warping path P_F^* might not coincide with the globally optimal warping path computed with DTW. Using the MsDTW strategy reduces the memory requirements compared to classical DTW. However, the required memory is still linearly dependent on the length of the feature sequences [111]. For a detailed description of the procedure, we refer to [86, 111].

4.3.3 Memory-restricted MsDTW (MrMsDTW)

We now describe our proposed MrMsDTW procedure. Given a memory restriction parameter τ that sets an upper bound on the number of cells that can be used for the alignment computation, our main idea is to use rectangular local constraint regions that have a size of at most τ instead of a single global tubular constraint region. On each rectangular constraint region, a local alignment is computed on a local cost matrix. Each local alignment is computed by applying standard DTW on the local cost matrix. Furthermore, the computations of the local alignments are independent of each other. They can therefore be computed in a sequential way such that at most τ cells are used at any time.

As in Section 4.3.2, we use two resolution levels to describe the algorithm. The main difference to the classical MsDTW approach is the refinement on Level F, where rectangular local constraint regions are derived from an alignment that is computed on Level C. As before, let P_C^* be an optimal warping path computed on Level C (Figure 4.1d). At this point, we assume that the computation of P_C^* did not require more than τ cells. Furthermore, let P_F be a suitable projection of P_C^* onto Level F (Figure 4.1e). In the next step, we use P_F to construct a set of rectangular constraint regions to refine the alignment on Level F. To this end, we derive an anchor sequence $A \sqsubseteq P_F$ (black dots in Figure 4.1e). Each consecutive pair of anchor points in A defines a rectangular constraint region. We further require A to fulfill the condition $\mathcal{R}(A) \leq \tau$. Such an anchor sequence can be obtained by initially setting

$$A := (B(P_F), E(P_F)).$$

If A does not fulfill $\mathcal{R}(A) \leq \tau$, the warping path P_F is recursively divided at its center into subpaths from which the first and last element are used as anchors. This is repeated until $\mathcal{R}(A) \leq \tau$ (see Figure 4.1e). In the next steps, we use A to compute a refined warping path P_F^* from P_F:

(Step 1) Refinement between anchor points. We compute a set of local paths $Q_1, \ldots, Q_{K-1}$ constrained by the anchor sequence $A = (a_1, \ldots, a_K)$. Each local path Q_k fulfills the boundary conditions $B(Q_k) = a_k$ and $E(Q_k) = a_{k+1}$ and is computed by using standard DTW on the local cost matrix $\mathbf{C}_k := \mathbf{C}[a_k; a_{k+1}]$ (see Figure 4.1f). If A contains only two anchor points ($K = 2$), we set $P_F^* := Q_1$. Otherwise, we proceed with the following steps.

(Step 2) Refinement in the neighborhoods of the anchor points. In the first refinement step, the warping path P_F was not refined at the anchor points in A. In this step, we therefore recompute the refinement in neighborhoods of these anchor points. To this end, we derive pairs of anchors A'_k with $k \in [1 : K-2]$, each fulfilling $\mathcal{R}(A'_k) \leq \tau$. We initialize each $A'_k := (H(Q_k), H(Q_{k+1}))$ and iteratively decrease the size of the respective rectangular constraint regions until they obey the memory restriction constraint (see white dots in Figure 4.1g).

Identifier	**Composer / Piece**
`ID-01`	Vivaldi / RV 269 (Spring), 1st movement
`ID-02`	Shostakovich / Jazz Suite No. 2, 6th movement (Waltz)
`ID-03`	Beethoven / Symphony 5, Op. 67, 1st movement
`ID-04`	Weber / "Der Freischütz", No. 8
`ID-05`	Wagner, "Meistersinger", Prelude
`ID-06`	Ravel / "Bolero"
`ID-07`	Beethoven / Symphony 9, Op. 125, 4th movement
`ID-08`	Schubert / Symphony 8, D759, 1st movement (Unfinished)
`ID-09`	Weber / "Der Freischütz", full opera
`ID-10`	Wagner / "Das Rheingold", WWV 86A

Table 4.1. Selection of pieces in our dataset.

Now, we compute the set of local paths $Q'_1, \ldots, Q'_{K-2}$ where each local path Q'_k is constrained by the anchor pair A'_k (see Figure 4.1g+h).

(Step 3) Concatenation. The local paths from the previous steps are combined to a global warping path P^*_F by concatenation (see Figure 4.1i).

When computing the local alignments sequentially, the maximum memory requirement of the algorithm in the two refinement steps can be directly inferred from the used anchor sequences A and A' by computing $\mathcal{R}(A)$ and $\max_k \mathcal{R}(A'_k)$ respectively.

As a final remark, note that the procedure described above can be applied in a recursive fashion by introducing further levels of decreasing feature resolution, similar as in the classic MsDTW procedure. In practice, we choose the coarsest feature resolution level such that we need at most τ cells for the computation of the full DTW procedure on this level. Note that this leads to a constant memory requirement for MrMsDTW.

4.4 Experiments

In this section, we first describe our experimental setting. Then, we investigate the robustness of our proposed MrMsDTW procedure in dependency of the memory restriction parameter τ in the context of an audio-to-audio music synchronization scenario.

In the experiment, we use four fixed feature resolution levels (50 Hz, 10 Hz, 2 Hz, 1 Hz), referred to as Level 1–4 where Level 1 corresponds to the finest feature resolution and Level 4 to the coarsest. In the case that the feature sequences are so long that the initial alignment would violate our memory constraint, we add a fifth level choosing a feature resolution that still fulfills the memory requirement. We use the same feature set as described in [36].

Our dataset contains 53 pairs of classical music recordings, each pair consisting of two different performances of the same piece of music. Table 4.1 shows the musical works used in our experi-

Identifier	Recording 1		Recording 2		Memory		
	Performer 1	Dur. [s]	Performer 2	Dur. [s]	DTW [GB]	MsDTW [MB]	MrMsDTW [MB]
ID-01	Abbado	203.2	Nishizaki	213.2	0.807	4.880	0.763
ID-02	Chailly	223.7	Yablonsky	193.8	0.808	5.120	0.763
ID-03	Karajan	444.0	Bernstein	519.2	4.294	11.884	0.763
ID-04	Furtwängler	611.5	Jochum	454.0	5.171	13.996	0.763
ID-05	Armstrong	595.1	Neumann	563.8	6.249	13.621	0.763
ID-06	Abbado	862.7	Ozawa	901.1	14.480	20.625	0.763
ID-07	Karajan	932.6	Bernstein	928.5	16.129	21.346	0.763
ID-08	Solti	950.9	Sacchi	817.5	14.479	21.764	0.763
ID-09	Carlos Kleiber	7763.3	Davis	8204.1	1186.379	187.784	0.763
ID-10	Karajan	8752.1	Haitink	8930.3	1455.823	204.398	0.763

Table 4.2. Memory requirements for DTW, MsDTW with $\delta = 30$ (expansion size for tubular constraint region), and MrMsDTW with $\tau = 10^5$ (rectangular constraint region size) for a selection of recording pairs from our dataset, see also Table 4.1. ID-09 and ID-10 were not used in our robustness experiments, as the memory requirement for DTW is more than a terabyte. The alignments are therefore infeasible to compute for these pieces.

ments. Table 4.2 shows a representative selection of the recordings used in our experiments. For each entry, a lower bound for the memory requirement of DTW, MsDTW, and MrMsDTW is given. The lower bound is derived from the number of memory cells required to store the accumulated cost matrices $\mathbf{D}$ that are used in DTW, MsDTW, and MrMsDTW. The memory usage in megabytes (MB) is obtained by multiplying the required number of memory cells to store $\mathbf{D}$ by $\frac{8}{1024^2}$ (assuming a 64 bit machine using double precision floating point numbers). Note that these values may need to be multiplied by a factor of 2 or 3, depending on the implementation (for additionally storing the cost matrices and a matrix that saves the step sizes and indices leading to a cost-minimizing path in $\mathbf{D}$). However, this factor is the same for all three DTW variants. For MsDTW, we computed the lower bound by assuming a diagonal path between the two feature sequences, leading to a requirement of $2\delta \cdot N$ memory cells, where N denotes the length of the longer feature sequence. In our dataset, the shortest piece is roughly one minute long, and the longest about 15 minutes. Note that the memory requirement of DTW for the piece ID-07 in Table 4.2 is already higher than 16 gigabytes. As the memory usage of DTW grows quadratically with the length of the input feature sequences, it is not feasible to use it for longer pieces on a normal desktop computer (assuming a maximum of 16 gigabytes of available memory). For example, full DTW would require more than one terabyte of memory on the opera recordings ID-09 and ID-10 in Table 4.2. However, our proposed MrMsDTW approach has basically constant memory requirements, and is therefore capable to compute alignments for longer pieces. The memory requirements for DTW (quadratic), MsDTW (linear), and MrMsDTW (constant) are illustrated in Figure 4.4.

In the following, we investigate the connection between the memory restriction parameter τ and the robustness of our proposed MrMsDTW procedure, see Table 4.3. To this end, we

τ \ ε	0	1	2	4	8	16	τ [MB]
10^7	99.90	99.90	99.90	99.91	99.93	99.94	76.29
10^6	99.81	99.83	99.85	99.87	99.88	99.92	7.629
10^5	95.82	96.57	96.90	97.49	98.31	99.13	0.763
10^4	82.60	87.94	90.18	93.36	96.37	98.44	0.076
10^3	37.58	59.53	69.81	82.00	91.98	97.29	0.008

Table 4.3. Overall percentage of warping path cells in P_{MrMsDTW} that have a deviation $\leq \varepsilon$ cells from the corresponding cells of the optimal global warping path P_{DTW}. The value has been computed for each warping path and was then averaged over the whole dataset (to avoid that longer pieces have a stronger influence). The error tolerance ε defines a tolerance region around the P_{DTW} of $\pm\varepsilon$ frames in horizontal and vertical direction. One frame corresponds to 20 ms.

compare the alignments computed by MrMsDTW with the optimal alignments computed by the full DTW procedure as described in Section 4.3.1. When aligning two music recordings, the warping path P_{MrMsDTW} may deviate from the optimal global warping path P^*_{DTW}. The smaller the constraint regions, the more likely it is that such deviations occur. Since small deviations from the optimal alignment might be acceptable, we introduce a tolerance parameter ε for the evaluation. We say that a cell in P_{MrMsDTW} was correct in case it lies within a region of ε-cells to the left, top, right, and bottom around a cell from P^*_{DTW}. This means that for ε=0, a cell in the P_{MrMsDTW} is only considered to be correct if it coincides perfectly with a corresponding cell in the optimal warping path P^*_{DTW}. In contrast, for ε=16, a deviation of up to 0.32 seconds from the optimal path is allowed (assuming a feature resolution of 50 Hz).

Table 4.3 shows the overall percentage of correct warping path cells in dependency of the memory restriction τ (given in cells) and the tolerance ε (given in frames) for our dataset. Furthermore, the memory requirement for a given τ is stated. For τ=10^7, having a memory requirement of ≈76 MB, nearly all alignment cells coincide with the full DTW approach. As a comparison, the full DTW needs already more memory for pieces that are longer than 63 seconds. For τ=10^6, reducing the memory requirement to ≈7.6 MB, even with $\varepsilon = 0$, more than 99% of the alignment cells are still correct. The memory requirement of MsDTW with $\delta = 30$ is only smaller for pieces shorter than 334 seconds (see Figure 4.4). With τ=10^5, requiring less than 1 MB of memory, the deviations from the optimal path start to increase. In this case, MsDTW with $\delta = 30$ is only more memory efficient for pieces shorter than 34 seconds (see Figure 4.4). For $\tau \leq 10^4$ (when providing less than 100 kilobytes of memory), the procedure becomes unstable, leading to stronger deviations from the optimal warping path. When reducing the memory requirement down to 8 kilobytes ($\tau = 10^3$), only 37.58% of the warping cells correspond with the full DTW results with a tolerance $\varepsilon = 0$. However, within a tolerance of $\varepsilon = 16$ (0.32 seconds), more than 97% of the cells are still considered to be correct. Depending on the application, this accuracy might still be acceptable.

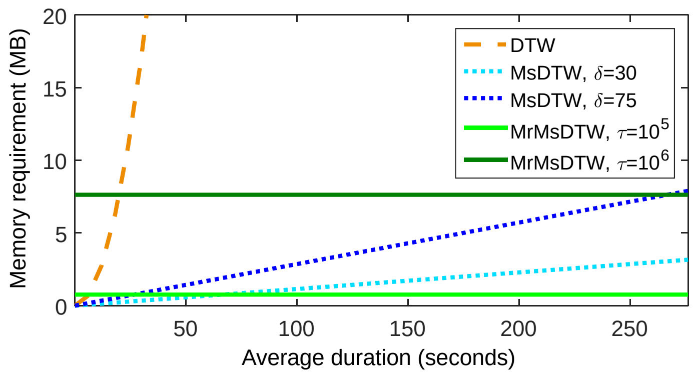


Figure 4.4. Memory requirement for storing the accumulated cost matrix $\mathbf{D}$ for DTW, MsDTW, and MrMsDTW in dependency of the average duration of the recordings. The curves for MsDTW are lower bounds assuming a diagonal path for the constraint region computation.

4.5 Conclusions

In this chapter, we proposed MrMsDTW, a new MsDTW variant using rectangular shaped local constraint regions. It combines the block-by-block processing from online alignment techniques with a multiscale strategy. The introduced procedure has a constant memory requirement, being explicitly controlled by a memory restriction parameter. In addition, MrMsDTW basically yields the same alignments as a full DTW approach, even when restricting it to require only eight megabytes of memory. As a final remark, opposed to classical MsDTW, our presented procedure has the potential of being implemented in a parallel fashion as the individual local paths can be computed independently of each other. This would trade the explicit memory restriction for a better runtime performance.

4.6 Further Notes

The MrMsDTW algorithm described in this chapter has been implemented in MATLAB as part of the Sync-Toolbox, which provides several utility functions and demo scripts for feature extraction, synchronization, sonification, and website generation. A more detailed documentation of the Sync-Toolbox can be found in Appendix C.

Within the FreiDi project, the Sync-Toolbox has been used to compute alignments between different audio recordings and symbolic encodings of a musical score for the opera. These alignments have been used in the *Synchronization & Interpretation Switcher*[1] realized as a web-based audio player. This player allows to playback different audio recordings while simultaneously highlighting the current playback position in the musical score (in the autograph or in a rendered version). Furthermore, through aligning all audio versions to a reference version, e.g. the musi-

[1] `http://freischuetz-digital.de/demos/syncPlayer/index.html`

cal score, a user can seamlessly switch from one audio recording to the corresponding position in another audio recording, see also Figure E.1 in Appendix E.

4.6.1 Applications of Music Alignments

Over the years, many musical works have seen a great number of reproductions, ranging from reprints of the sheet music to various audio recordings of performances. For many works this has led to a wealth of co-existing versions including arrangements, adaptations, cover versions, and so on. Establishing semantic correspondences between different versions and representations is an important step for many applications in Music Information Retrieval. For example, when comparing a musical score with an audio version, the goal is to compute an alignment between measures or notes in the score and points in time in the audio version. This task is motivated by applications such as score following [19], where the score can be used to navigate through a corresponding audio version and vice versa. The aligned score information can also be used to parameterize an audio processing algorithm such as in score-informed source separation [134, 39]. When working with two audio versions, alignments are useful for comparing different performances of the same piece of music [25, 37]. In cover song identification, alignments can be used to compute the similarity between two recordings [115]. Alignment techniques can also help to transfer meta data and segmentation information between recordings [25]. In [74], an unknown recording is queried against a database of music recordings to identify a corresponding version of the same musical work. After a successful identification, alignment techniques are used to transfer the segmentation given in the database to the unknown recording. In a live opera performance, online alignment techniques could be used to trigger opera subtitles [4].

Another application of alignments is *performance analysis* [131, 132, 133, 82, 81] which aims to analyze musical aspects such as dynamics or tempo. In the following section, we exemplarily describe the computation of tempo information from alignments [82].

4.6.2 Local Tempo Extraction from Music Alignments

An alignment between two music recordings of the same musical work assigns points in time from one recording to points in time of the other recording, and vice versa. From an alignment, one can easily derive the relative tempo between the two recordings. Let $p_\ell = (n_\ell, m_\ell)$ and $p_{\ell+1} = (n_{\ell+1}, m_{\ell+1})$ with $\ell \in [1 : L-1]$ be two consecutive cells in a strictly monotonic alignment. The value $n_\ell \in \mathbb{R}$ encodes time points (given in seconds) on the physical time axis of the first recording and $m_\ell \in \mathbb{R}$ time points (given in seconds) on the physical time axis of the second recording. Then we define a *tempo curve* $\mathcal{T} : [1 : L - 1] \to \mathbb{R}$ with

$$\mathcal{T}(\ell) = \frac{n_{\ell+1} - n_\ell}{m_{\ell+1} - m_\ell} \, ,$$

encoding the relative local tempo of the two recordings. If $\mathcal{T}(\ell) > 1$, then the first recording is slower, if $\mathcal{T}(\ell) < 1$, the second one is slower. Otherwise, they share the same local tempo. Let $b_\ell \in \mathbb{R}$ be the tempo given in beats per minute (BPM) associated to the time points $n_\ell \in \mathbb{R}$. To map the time points $m_\ell \in \mathbb{R}$ to an absolute tempo, we compute $b_\ell \cdot T(\ell)$.

In the remainder of this section, we present tempo curves for recordings of No. 6, 8, and 9 of "Der Freischütz" which are derived from *manual alignments*, *audio-audio alignments*, and *MIDI-audio alignments*. We start with the *manual alignments* and come back to the latter two later in this section. We manually created measure annotations for seven audio recordings of No. 6, 8, and 9 which we refer to as *manual alignments*. The measure annotations indicate the measure start positions (often also referred to as downbeat positions). By using the measure annotations, we can relate the local tempo of the performance to the musical time axis, see Figure 4.5, Figure 4.7, and Figure 4.9. This enables a user to relate the local tempo to musical measure positions in a score, and hence, to look for musical reasons of the tempo variations. For example, a user could be interested to compare the tempo variations to various performance instructions in the score. Furthermore, using a musical time axis as a reference time axis is also useful when comparing the local tempo of different performances, see Figure 4.6a, Figure 4.8a, and Figure 4.10a. For example, in Figure 4.6a, the recording `Fur1954` has the slowest tempo, but in measure 102 it is slightly faster than the `Kle1955` recording.

We now discuss the tempo curves of the Bloemeke 2013 recordings derived from manual alignments. Figure 4.5 shows the tempo curve of Weber's Freischütz No. 6 conducted by Bloemeke (2013) with a physical time axis (in seconds) and a musical time axis (in measures). This piece has a very steady tempo and there are only local tempo fluctuations within a small range. These small fluctuations can usually be related to musical phrases. In measure 102, the singer "Aennchen" ends a phrase, whereas the singer "Agathe" starts a new phrase. Here, we can see a slow down towards the end of the first phrase with an acceleration from the start of the second phrase. Figure 4.7 shows the tempo curve derived from manual annotations for the Bloemeke (2013) recording of No. 8. This piece has many tempo fluctuations, as some parts are performed as *recitative*. In such a part, the singer has quite some freedom in shaping the local tempo. By investigating the tempo curve, we can explain some tempo changes from the score. In measure 14, for example, there is a fermata, which results in a slower local tempo. Measures 35–40 are performed again as recitative. An interesting tempo fluctuation happens in measure 89, where the score has the instruction *stringendo*. This means that the tempo should be gradually increased. The stringendo ends in measure 91 which marks the highest tempo in the recording.
Figure 4.9 shows the tempo curve derived from manual annotations for the Bloemeke (2013)

recording of No. 9 of “Der Freischütz”. The tempo curve exhibits some interesting tempo changes. For example, in measure 9, the tempo drops roughly by a half. This is due to a fermata which extends the duration of a musical note. In measure 131, there is a sudden tempo increase. This is consistent with the instruction *vivace con fuoco* in the score, indicating a faster tempo than the *allegro* from the beginning of the piece. In measure 142, there is another tempo change which goes along with the instruction *andantino* which is a slower tempo than the *allegro*. The tempo changes again in measure 170 with the instruction *allegro vivace* in the score.

In the following, to illustrate that a tempo analysis can also be performed without having annotated each recording manually, we computed two alignments for each recording.

First, we computed an *audio-audio alignment* for each audio recording using the Bloemeke recordings from 2013 (Blo2013) as a reference, and used these computed alignments to transfer the measure annotations from Blo2013 to the other recordings. The corresponding tempo curves are shown in Figure 4.6b, Figure 4.8b, and Figure 4.10b for No. 6, 8, and 9, respectively. Furthermore, we computed a *MIDI-audio alignment*, aligning each audio recording with a MIDI version. From the MIDI version, we automatically extracted the measure positions which we again transferred onto all recordings by using the computed alignments. The resulting tempo curves for No. 6, 8, and 9 are shown in Figure 4.6c, Figure 4.8c, and Figure 4.10c, respectively.

When comparing the manual alignments with the audio-audio alignments and MIDI-audio alignments, we can see that the major tempo changes from the manual alignments are also reflected in the tempo curves derived from the automatically computed alignments. Locally, we can notice some outliers both for the audio-audio as well as the MIDI-audio alignments, see for example the red rectangles in Figure 4.10b and Figure 4.8c. Visually, it appears that the audio-audio alignments are closer to the manual alignments than the MIDI-audio alignments. Without going into a quantitative analysis, this is a plausible observation which coincides with the findings from [5], where the authors improved an online alignment algorithm by using another audio recording as a reference version instead of a MIDI version. The above observation can be explained in the following way. Remember that features derived directly from MIDI are very clean. In the example from Chapter 3 in Figure 3.7, we illustrated that the chroma features derived from MIDI were for example lacking contributions from overtones in the chroma bins. Hence, they do not match as closely as features derived from audio versions having the same instrumentation, which leads to errors in the alignments.

Note that the tempo analysis we performed in this section was focusing only on the local tempo extracted for each measure position. However, the tempo can be analyzed on different scales, for example on a motif, phrase, or section level. There are procedures that compute the tempo within a local window, as well as methods for deriving locally adaptive windows from the shape of the alignment for enhancing the tempo extraction, see [82] for details. Furthermore, instead of using alignments for the tempo analysis, one could also use beat tracking algorithms [12, 135, 53] that aim at directly extracting the musical beat from a music signal. However, extracting the measure positions by only relying on a beat tracker can be very error-prone, especially when dealing with complex orchestra music.

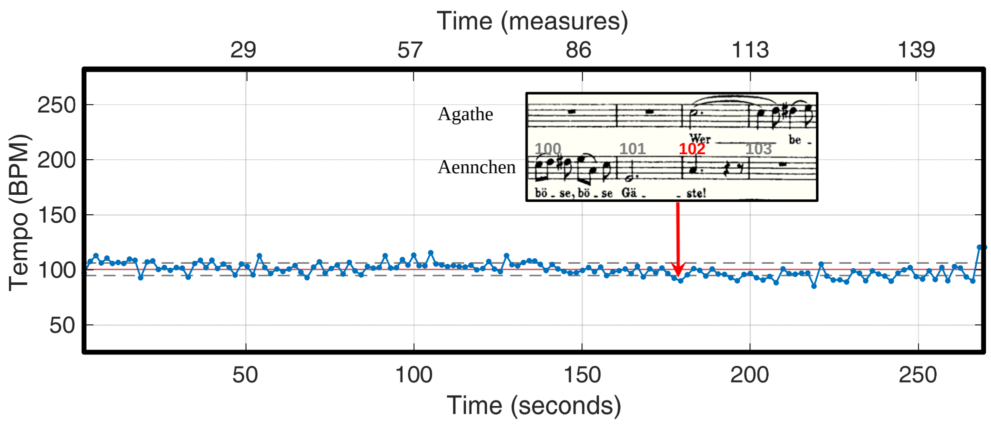


Figure 4.5. Tempo curve for Bloemeke 2013 recording of "Der Freischütz" No. 6, derived from manual annotations. The red horizontal line shows the mean tempo of the recording, and the gray dashed lines indicate the standard deviation of the tempo.

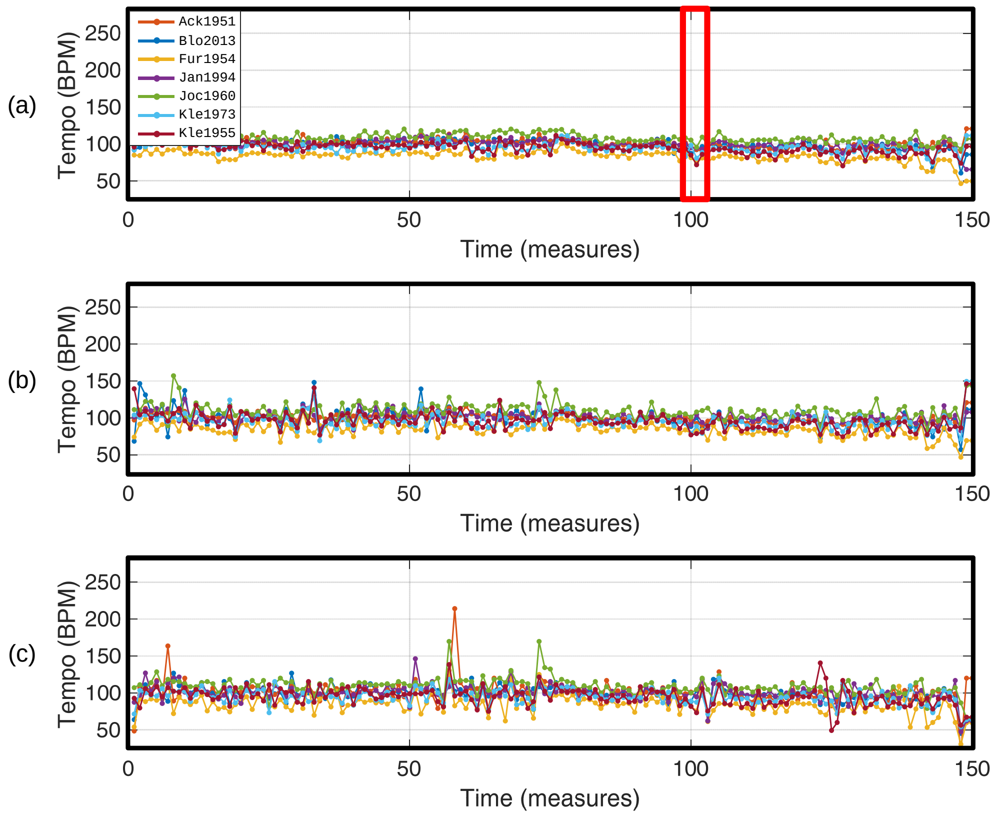


Figure 4.6. Tempo curves of "Der Freischütz" No. 6 derived from different data sources. **(a)** Manual annotations. **(b)** Alignment with Bloemeke 2013 (Blo2013). **(c)** Alignment with MIDI.

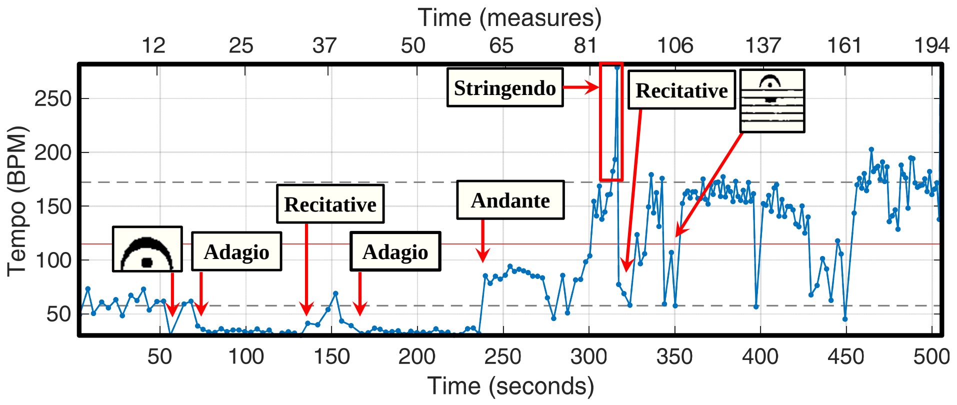


Figure 4.7. Tempo curve for Bloemeke 2013 recording of "Der Freischütz" No. 8, derived from manual annotations. The red horizontal line shows the mean tempo of the recording, and the gray dashed lines indicate the standard deviation of the tempo.

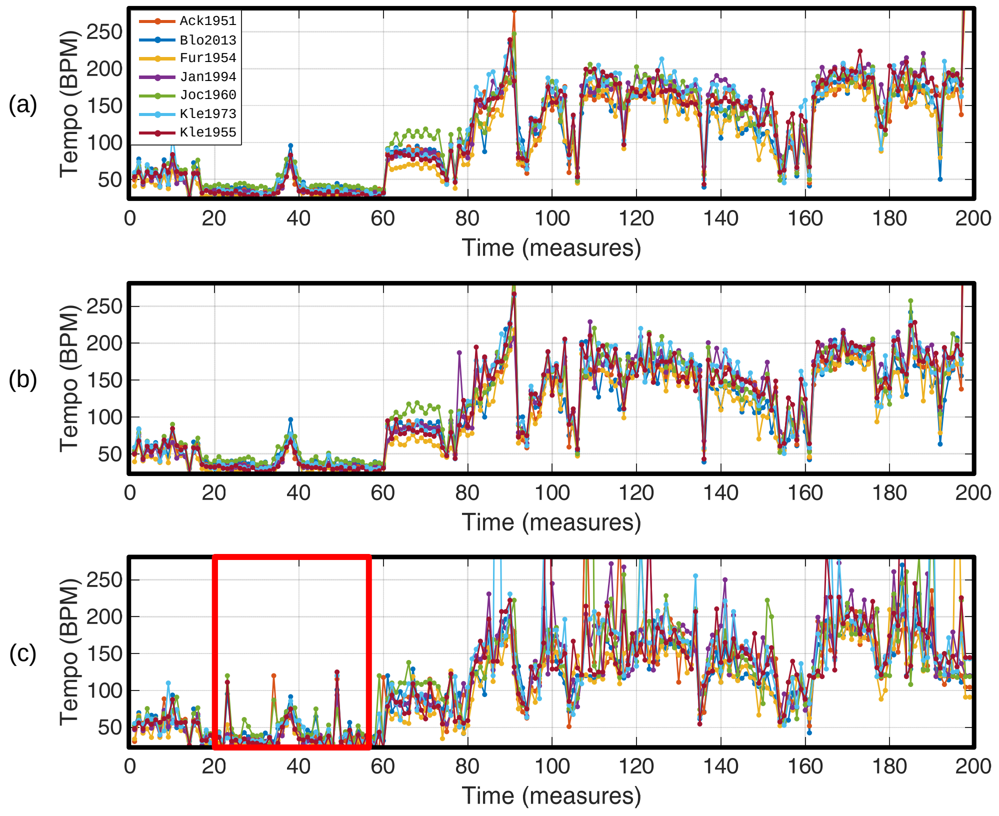


Figure 4.8. Tempo curves of "Der Freischütz" No. 8 derived from different data sources. **(a)** Manual annotations. **(b)** Alignment with Bloemeke 2013 (Blo2013). **(c)** Alignment with MIDI.

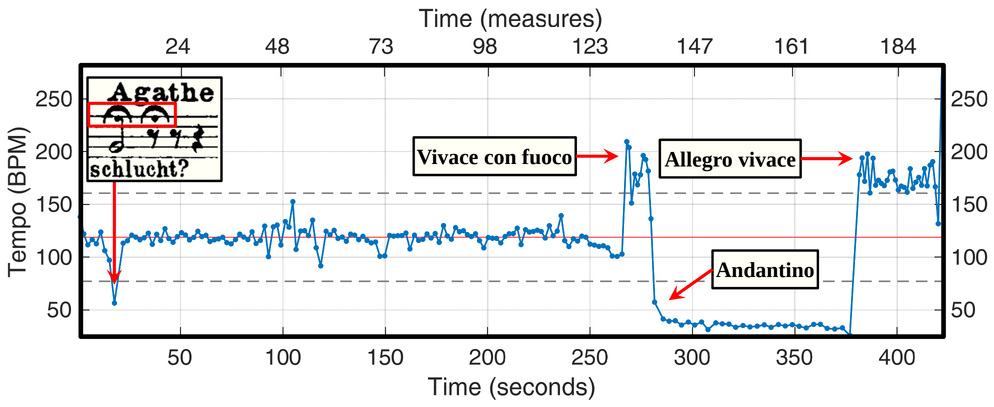


Figure 4.9. Tempo curve for Bloemeke 2013 recording of "Der Freischütz" No. 9, derived from manual annotations. The red horizontal line shows the mean tempo of the recording, and the gray dashed lines indicate the standard deviation of the tempo.

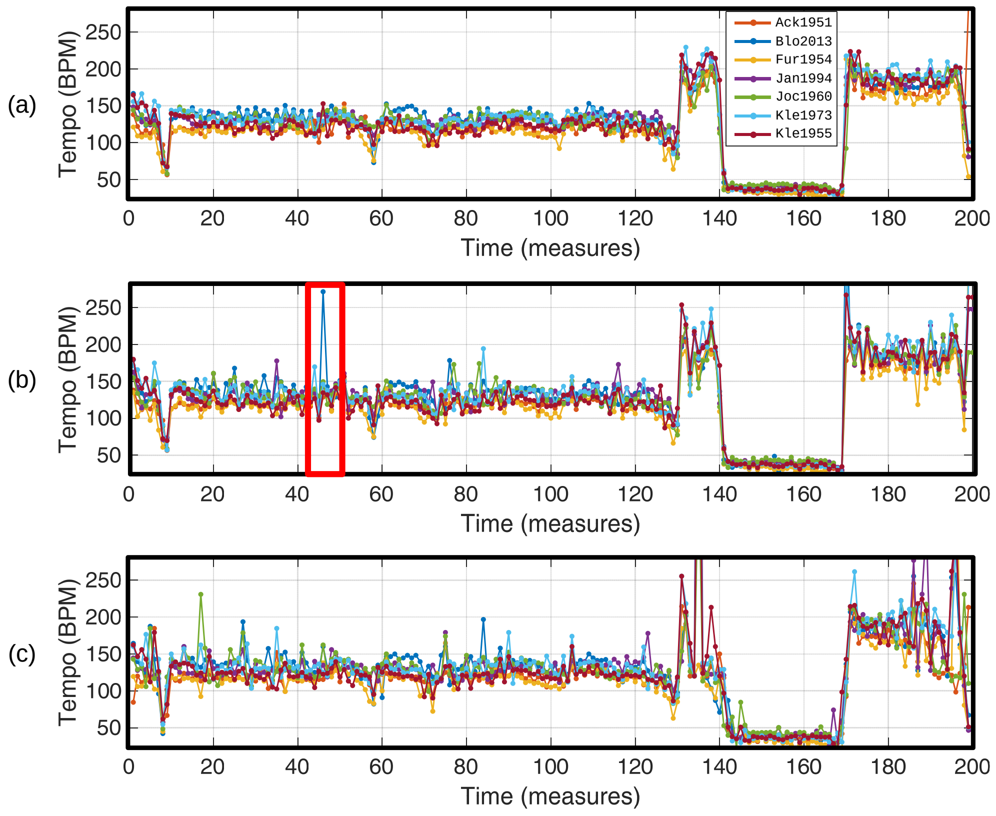


Figure 4.10. Tempo curves of "Der Freischütz" No. 9 derived from different data sources. **(a)** Manual annotations. **(b)** Alignment with Bloemeke 2013 (`Blo2013`). **(c)** Alignment with MIDI.

Chapter 5

Track Segmentation

This chapter is mainly based on our contributions in [99] and [100].

An essential audio processing task in the FreiDi project concerns the automated segmentation of all available audio recordings of the opera in a consistent way. In the context of the FreiDi project, such a segmentation was a fundamental requirement for further analysis and processing steps such as the computation of linking structures across different musical sources, including sheet music and audio material (see Chapter 4).

In our scenario, we assume that a musicologist may be interested in a specific segmentation of the opera. Therefore, as input of our algorithm, the user may specify a segmentation of the opera by manually annotating the desired segment boundaries within a musical score (or another music representation). This annotation is also referred to as *reference segmentation.* Our goal is to automatically transfer this reference segmentation onto all available recordings of the opera. The desired result of such a segmentation for 23 Freischütz versions is shown in Figure 2.6b.

In [99], we approached the segmentation task for complete recordings of the opera. Here, the task was approached by a segment-level matching procedure aiming at transferring a labeled reference segmentation onto an unknown version of the same musical work (see Figure 5.1). The contributions of [99] are twofold. First, we applied and adjusted existing synchronization and matching techniques to realize an automated reference-based segmentation procedure. The second and even more important contribution was to highlight the various challenges arising in the context of this seemingly easy segmentation scenario. In fact, the various audio recordings reveal significant acoustic and structural deviations. Considering digitized material from old sound carriers (shellac, LP, tape recordings etc.), one often has to deal with artifacts. Structurally, there are omissions or changes of numbers, repetitions, verses and dialogues. By systematically adjusting the segmentation procedure to reveal these variations, we not only successively improved the segmentation quality, but also gained a better understanding of the audio material.

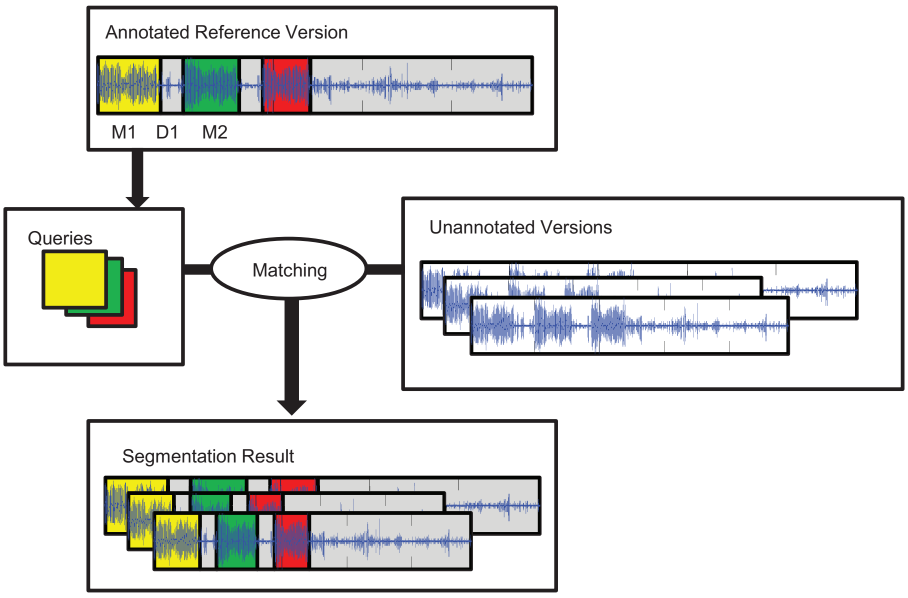


Figure 5.1. Illustration of the segment-level reference-based segmentation procedure.

One main assumption in [99] was that a given reference segment either appears more or less in the same form in the unknown version or is omitted completely. In abridged versions of an opera, however, this assumption is often invalid. Such versions strongly deviate from the original by omitting material on different scales, ranging from the omission of several musical measures up to entire parts (see Figure 5.8).

In [100], we addressed the problem of transferring a labeled reference segmentation onto an unknown version in the case of abridged versions. As our main contribution, instead of using a segment-based procedure as in [99], we applied a more flexible frame-level matching procedure. As illustrated by Figure 5.2, the idea is to establish correspondences between frames of a reference version and frames of an unknown version. The labeled segment information of the reference version is then transferred to the unknown version only for frames for which a correspondence has been established. Such a frame-level procedure is more flexible than a segment-level procedure. However, on the downside, it is less robust. As a main contribution of [100], we showed how to stabilize the robustness of the frame-level matching approach while preserving most of its flexibility.

In the remainder of this chapter, we give more detailed background information on the segmentation tasks and discuss related work (Section 5.1). We then describe our reference-based segmentation for complete recordings (Section 5.2) from [99]. Next, we describe our frame-level segmentation approach for abriged recordings (Section 5.3) from [100]. In Section 5.4, we conclude the chapter.

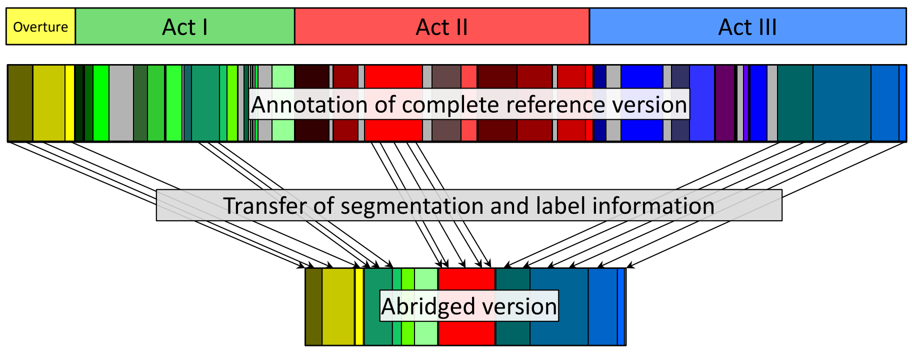

Overture
Act I
Act II
Act III
Annotation of complete reference version
Transfer of segmentation and label information
Abridged version

Figure 5.2. Illustration of the proposed frame-level segmentation procedure. Given the annotated segments on a complete reference version of a musical work, the task is to transfer the segment information to an abridged version.

5.1 Background

Remember that the Freischütz opera is structured in three acts which are further subdivided into an overture and 16 following numbers interspersed by spoken text passages (dialogues). The numbers cover a wide range of musical material (arias, duets, trios, instrumental pieces, etc.). Some of the melodic and harmonic material of the numbers is already introduced in the overture. Also, some of the numbers contain repetitions of musical parts or verses of songs. In the acoustic domain, these are not always part of the performance, as a conductor or producer may take the artistic freedom to deviate substantially from what is specified in the musical score. Besides differences in the number of played repetitions, further deviations include omissions of other parts or entire numbers as well as variations in the spoken text and the length of the dialogues. Apart from such structural deviations, audio recordings of the opera usually differ in overall length, sound quality, language and many other aspects. For example, our dataset includes historic recordings that are often prone to noise, artifacts, or tuning problems resulting from the digitization process. Furthermore, the recordings show a high variability in their duration, which can be explained by significant tempo differences and also by omissions of material, see Table 5.1 and Table 5.2 for details.

Large-scale musical works such as operas usually need a huge number of musicians to be performed. For these works, one often finds arrangements for smaller ensembles or piano reductions. Furthermore, performances of these works are usually very long. Weber's opera "Der Freischütz", for example, has an average duration of about two hours. Taking it to an extreme, Wagner's epos "Der Ring der Nibelungen", consists of four operas having an overall duration of about 15 hours. For such large-scale musical works, one often finds abridged versions. These versions usually present the most important material of a musical work in a strongly shortened and structurally modified form. Typically, these structural modifications include omissions of repetitions and other "non-essential" musical passages. Abridged versions were very common in the early recording days due to space constraints of the sound carriers. The opera "Der Freischütz" would have filled 18 discs on a shellac record. More recently, abridged versions or excerpts of a musical work can often be found as bonus tracks on CD records.

In a standard alignment scenario, abridged versions are particularly problematic as they omit material on different scales, ranging from the omission of several musical measures up to entire parts. For example, given a segment in a reference version, one may no longer find the start or ending sections of this segment in an unknown version, but only an intermediate section. Hence, alignment techniques that account for structural differences are needed. In [50], a music synchronization procedure accounting for structural differences in recordings of the same piece of music is realized with an adaptation of the Needleman-Wunsch algorithm. The algorithm penalizes the skipping of frames in the alignment by adding an additional cost value for each skipped frame. Thus, the cost for skipping a sequence of frames is dependent on the length of the sequence. In abridged versions, however, omission may occur on an arbitrary scale, ranging from several musical measures up to entire scenes of an opera. In such a scenario, a skipping of long sequences should not be more penalized as a skipping of short sequences. We will therefore use a different alignment strategy when approaching the segmentation of abridged recordings in Section 5.3.

A scenario related to our segment-level segmentation approach is described in [74], where the goal is to identify unknown audio recordings. By applying automated matching procedures, the unknown recordings are compared to well-annotated audio material in a database. Upon identification, the matching result also allows for segmenting the unknown recording. However, this segmentation is more a byproduct, which is not evaluated in detail. In our scenario, the focus lies on the segmentation and, in a certain sense, we follow a reversed approach as we start from known material that we match to a database which we assume to contain representatives of the same musical work.

Our raw audio data mostly originates from CD recordings, which were initially segmented in CD tracks. These track segmentations are not consistent, varying between 17 and 41 tracks per complete recording of the opera (see Table 5.1 for the number of tracks for each recording). In some recordings, each number of the opera was put into a separate track, whereas in others the numbers were divided into music and dialogue tracks, and sometimes the remaining music tracks were even subdivided.

In our scenario, a reference segmentation of the musical score into musically meaningful sections was specified by a domain expert (musicologist), who divided the opera into 38 musical segments and 16 dialogue segments. According to this reference segmentation, we manually created an annotation for each of the 28 audio recordings in our database (23 complete recordings, and 5 abridged recordings), resulting in over 1000 audio segments (see Figure 5.3 and Figure 5.8 for an overview). The objective in the following sections is to recover this annotation using automated

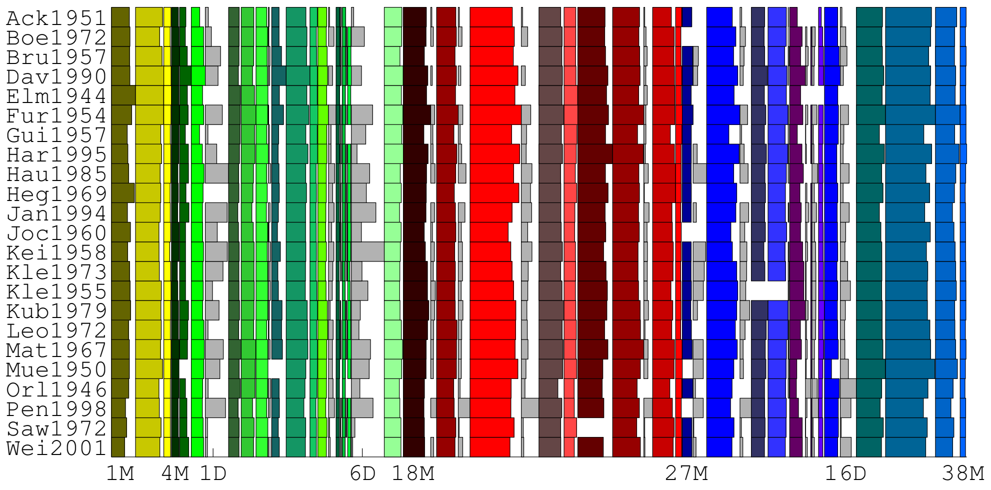


Figure 5.3. Segmentation result for 23 different audio recordings of "Der Freischütz" according to a reference segmentation specified by musicologists. The reference segmentation includes 38 musical sections (Overture: yellow, Act I: green, Act II: red, Act III: blue) as well as 16 spoken dialogue sections (gray).

methods and to get a better understanding of the variations and inconsistencies in the audio material.

5.2 Segmentation of Complete Recordings

As discussed before, the basic task is to segment an unknown audio recording (assuming no pre-segmentation) according to a given reference segmentation. In the following, we assume that this reference segmentation is specified on the basis of a reference audio recording. Then the objective of the segmentation task is to transfer the segmentation from the reference version to the unknown recording.

In this section, after introducing some basic notation (Section 5.2.1) and our evaluation measure to assess the accuracy of segmentation results (Section 5.2.2), we discuss various strategies to tackle the segmentation task based on global synchronization (Section 5.2.3) and local matching procedures (Section 5.2.4 – 5.2.7). Furthermore, we discuss the benefits and limitations of the respective procedures while revealing the musical and acoustic variations and inconsistencies in the audio material.

5.2.1 Basic Notations and Matching Techniques

In this section, we introduce some mathematical notions to model our segmentation problem and then review some standard audio synchronization and matching techniques that are applied in the subsequent section.

Similar as in Section 4.2, let $X := (x_1, x_2, \ldots, x_N)$ be a suitable feature representation of a given audio recording (the feature type is specified later). Then, a *segment* α is a subset $\alpha = [s:t] \subseteq [1:N]$ with $s \leq t$. Let $|\alpha| := t-s+1$ denote the length of α. Furthermore, we define a (partial) *segmentation* of X to be a sequence $(\alpha_1, \ldots, \alpha_I)$ of pairwise disjoint segments, i. e. $\alpha_i \cap \alpha_j = \emptyset$ for $i, j \in [1:I]$, $i \neq j$. Note that in this definition we do not assume that $[1:N]$ is completely covered by the segmentation.

In our scenario we assume that we have a reference sequence X with a reference segmentation $\mathcal{A} = (\alpha_1, \ldots, \alpha_I)$. Furthermore, let $Y := (y_1, y_2, \ldots, y_M)$ be a feature representation of an unknown audio recording. In the case that X and Y are structurally similar on a global scale, the transfer of the reference segmentation of X onto Y can be done by using standard synchronization or alignment techniques, see Chapter 4 and [25, 54, 76]. When synchronizing two audio recordings, the first step consists in transforming the recordings into feature representations, typically chroma-based audio features. In our experiments, we use CENS_5^{21} features of 2 Hz resolution as supplied by the chroma toolbox [78], see also Section 3.3.3 for details. Based on these feature representations and a suitable cost measure, one applies dynamic time warping (DTW) to compute a cost minimizing warping path which realizes the linking between X and Y, see [76, Chapter 4].

This synchronization-based transfer works as long as X and Y globally coincide. However, problems arise in the presence of significant structural differences. Furthermore, in case X and Y are long (as is the case for complete recordings of entire operas), running time and memory issues arise when performing DTW. Even though (multiscale, forward estimation) acceleration techniques exist [25, 86], such techniques are not suited when structural differences occur. As an alternative, one may apply more locally oriented *audio matching* techniques, where the individual segments α_i of the reference segmentation (used as "queries") are matched to subsegments of the unknown sequence Y (resulting in "matches" or "hits"), see [63]. In other words, the cost-intensive global DTW alignment is replaced by several smaller local alignments (realized by a subsequence variant of DTW), see also Figure 5.1 for illustration. Another positive effect is that using local matches allows for a better handling of missing segments and structural differences. On the downside, by querying the reference segments individually, one may loose temporal coherence, while the chance of obtaining local mismatches is increased (in particular for short segments).

In the subsequent section, we systematically apply, modify and combine both techniques—global synchronization and local matching—for performing our segmentation task. Here, besides the actual segmentation, our main goal is to obtain a better understanding of various kinds of variations and inconsistencies in the audio material.

5.2.2 Evaluation Measure

First of all, we need a measure that allows us to compare two given segments α and β. To this end, we define the *relative overlap measure* between α and β to be the value

$$\mu(\alpha, \beta) := \frac{|\alpha \cap \beta|}{|\alpha \cup \beta|} \in [0, 1],$$

which indicates the ratio of the absolute overlap and the length of the union segment. Note that $\mu(\alpha, \beta) = 1$ if and only if $\alpha = \beta$, and $\mu(\alpha, \beta) = 0$ if $\alpha \cap \beta = \emptyset$.

As before, let us assume that the reference version is represented by the sequence $X := (x_1, x_2, \ldots, x_N)$ and the *reference segmentation* by $\mathcal{A} := (\alpha_1, \ldots, \alpha_I)$. Furthermore, let $Y := (y_1, y_2, \ldots, y_M)$ be the unknown version to be segmented. For the purpose of evaluation, we assume that there is also a *ground truth segmentation* $\mathcal{B} := (\beta_1, \ldots, \beta_I)$ for Y, where each β_i musically corresponds to the α_i. The goal is to automatically derive the segmentation of Y. Let P denote such a segmentation procedure, which automatically transfers each reference segment α_i to a computed segment $\mathrm{P}(\alpha_i) \subseteq [1 : M]$. Then, the relative overlap measure $\mu(\beta_i, \mathrm{P}(\alpha_i))$ indicates the segmentation quality of the procedure P.

Because of the mentioned structural variations, the version Y does not necessarily contain a segment that musically corresponds to a reference segment α_i. In this case, the ground truth segment is set to $\beta_i = \emptyset$. Furthermore, the procedure P does not have to output a computed segment, which is modeled by setting $\mathrm{P}(\alpha_i) = \emptyset$. In the case that both the segment $\mathrm{P}(\alpha_i)$ and β_i are empty, we define $\mu(\beta_i, \mathrm{P}(\alpha_i)) = 1$ (a non-existing segment has been identified as such). Note that if only one of the segments is empty, $\mu(\beta_i, \mathrm{P}(\alpha_i)) = 0$.

5.2.3 Global Approach (S1, S2)

In the following matching procedures and evaluation, we only consider the musical sections (indicated by the non-gray segments in Figure 5.3) while leaving the dialogue sections (the gray segments in Figure 5.3) unconsidered. Exemplarily, we use a reference segmentation $\mathcal{A} = (\alpha_1, \alpha_2, \ldots, \alpha_{38})$ based on the recording conducted by Carlos Kleiber in 1973 (`Kle1973`), which is a performance that closely follows the musical score. Quantitative results for all procedures to be discussed are presented in Table 5.1 (relative overlap averaged over versions) and Table 5.2 (relative overlap averaged over segments).

In the two procedures S1 and S2, we apply a global synchronization approach. For S1, we employ DTW using the step size condition $\Sigma_1 = \{(1, 1), (1, 2), (2, 1)\}$, see Chapter 4.3.1 and [76, Chapter 4]. This strategy is usually very robust as long as there are no significant deviations in structure and tempo between the two versions compared. However, the procedure S1 is not able to compensate well for structural variations leading to an average relative overlap of 0.852, see Table 5.1 When using the step size condition $\Sigma_2 = \{(1,1),(1,0),(0,1)\}$ (calling this procedure S2), performance improves significantly, yielding the average relative overlap of 0.930, see Table 5.1. For example, in the version `Saw1972`, the dialogues are comparatively short, see also the gray rectangles in Figure 5.3. Such a situation causes S1 to fail, resulting in an overlap of 0.615 compared to 0.896 for S2, see Table 5.1. For both procedures, the alignment accuracy for α_{38} is very low with 0.714 (S1) and 0.724 (S2), see Table 5.2. This is due to audio material not belonging to the actual opera that is appended at the end (CD bonus tracks) in some versions. In this case, the global synchronization procedures do not allow to skip the final tracks. Despite the promising results of S2, this approach has several limitations. First, it is inefficient considering runtime and memory requirements, especially when increasing the feature resolution, see also Section 5.2.1. Secondly, it is not well suited to accommodate for structural changes in a controlled manner. And thirdly, the procedure does not give deeper insights into the musical and acoustic properties of the underlying audio material.

Our goal in the following sections is to develop a more flexible segmentation strategy that achieves a quality comparable to S2 while yielding better insight into the versions' properties.

5.2.4 Local Approach (M1)

The remaining approaches discussed below rely on a local matching procedure based on a subsequence variant of DTW using the step size condition Σ_1. Here, for each $\alpha_i \in \mathcal{A}$ (used as a query) applied to a given unknown version, we compute a ranked list of *matching candidates*. For the segmentation procedure M1, we only consider the top match in the list, see also Figure 5.1 for illustration of the general matching strategy.

In Figure 5.4a, the relative overlap values for M1 computed on all recordings in our dataset are presented in a gray-scale matrix visualization, where the rows indicate the audio versions and the columns indicate the segments. Black corresponds to $\mu = 0$ (no overlap) and white to $\mu = 1$ (perfect overlap). Row-wise, the segmentation accuracy of a specific version becomes obvious, whereas column-wise, segments which are problematic across versions can easily be spotted. An example for a problematic version is `Elm1944`, which generally seems to perform poorly, showing many black entries in Figure 5.4a and having a low average relative overlap of 0.705, see Table 5.1. A closer look at the audio material revealed that there are some issues concerning the tuning of this version, probably resulting from the digitization process. Furthermore, there are segments which show a poor segmentation accuracy across versions, see for example the black entries for α_{14} to α_{16} in Figure 5.4a. It turns out that these three segments correspond to the three verses of a song (No. 4) in the opera. The reason why this song has been divided into individual segments is that there are dialogues between the verses (recall that a requirement

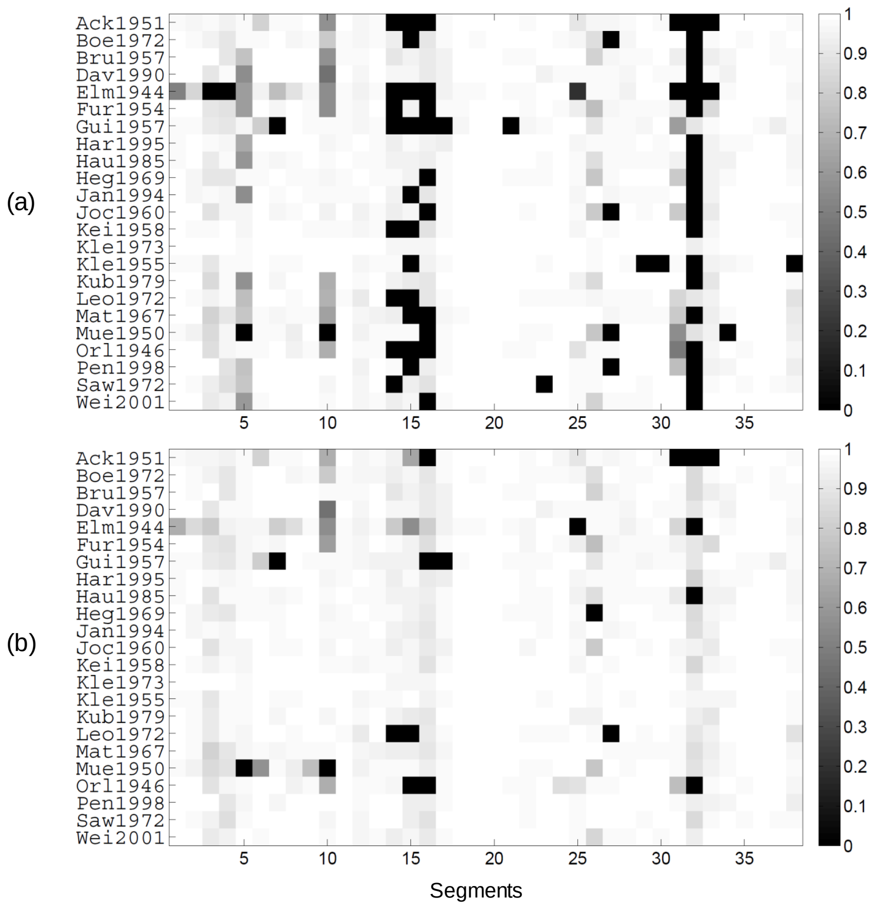


Figure 5.4. Matrix visualization of relative overlap values, where the versions correspond to rows and the segments to columns. **(a)**: P = M1. **(b)**: P = M4.

of the reference segmentation was to separate music and dialogue sections). The verses all share the same melodic and harmonic material and are thus easily confused with each other in the matching procedure. Another interesting problem appears for α_{32}, where M1 nearly fails for every version, resulting in an overall segmentation accuracy of 0.157, see Table 5.2 and Figure 5.4a. Actually, α_{32} (having a duration of only 12.4 seconds) is a short snippet of a chorus section for which many repetitions exist in the surrounding segments α_{31} (song with several verses and chorus sections) and α_{33} (chorus) which are interspersed by dialogues. Thus it is very likely that α_{32} is matched into the harmonically similar parts within α_{31} or α_{33}. For the version `Kle1955`, segment α_{38} seems to be problematic, see Figure 5.4a. Actually, α_{38} contains musical material which is already used in the overture of the opera (covered by α_3). A closer look into the matching results for `Kle1955` revealed that α_{38} matched indeed into the musically very similar section in the overture.

In conclusion, procedure M1 is more efficient[1], see also Section 5.2.1, while its main drawback

[1]On a 64bit machine, the average memory requirement for a global DTW run on one piece of our dataset is

Version	#O	*dur.*	S1	S2	M1	M2	M3	M4
`Ack1951`	19	6904.81	0.596	0.811	0.808	0.851	0.850	0.853
`Boe1972`	30	7771.77	0.784	0.931	0.889	0.865	0.962	0.962
`Bru1957`	24	7439.33	0.906	0.933	0.927	0.905	0.923	0.966
`Dav1990`	30	8197.88	0.972	0.984	0.926	0.926	0.950	0.961
`Elm1944`	19	7081.52	0.698	0.827	**0.705**	**0.777**	0.806	0.865
`Fur1954`	34	9121.69	0.923	0.936	0.866	0.861	0.938	0.949
`Gui1957`	18	6911.30	0.908	0.937	0.801	0.851	0.860	0.886
`Har1995`	17	8044.99	0.974	0.981	0.944	0.943	0.965	0.973
`Hau1985`	17	8245.23	0.955	0.957	0.935	0.933	0.932	0.943
`Heg1969`	25	7436.75	0.896	0.958	0.913	0.895	0.943	0.946
`Jan1994`	30	7843.21	0.881	0.987	0.916	0.917	0.964	0.976
`Joc1960`	26	7178.21	0.922	0.948	0.887	0.911	0.968	0.967
`Kei1958`	32	8043.00	0.886	0.965	0.904	0.902	0.976	0.975
`Kle1973`	29	7763.00	1.000	0.996	0.989	0.990	0.990	0.990
`Kle1955`	41	7459.35	0.776	0.873	0.849	0.876	0.980	0.980
`Kub1979`	23	8044.65	0.959	0.985	0.927	0.929	0.953	0.974
`Leo1972`	19	7726.17	0.861	0.926	0.905	0.900	0.875	0.896
`Mat1967`	17	8309.35	0.984	0.983	0.874	0.876	0.948	0.965
`Mue1950`	35	7559.97	0.814	0.881	0.825	0.824	0.885	0.895
`Orl1946`	32	7368.58	0.559	0.807	0.853	0.854	0.852	0.883
`Pen1998`	26	7768.00	0.866	0.904	0.890	0.891	0.968	0.977
`Saw1972`	29	6871.02	**0.615**	**0.896**	0.893	0.894	0.968	0.974
`Wei2001`	38	7220.13	0.859	0.974	0.915	0.916	0.965	0.975
∅	26	7665.65	**0.852**	**0.930**	0.884	0.891	0.931	0.945

Table 5.1. Relative overlap values averaged over segments for different versions and different procedures. The first column indicates the version, the second (#O) the number of segments on the original sound carrier, and the third column (*dur.*) the overall duration in seconds of the recording. S1, S2, M1, M2 M3, and M4 denote the respective segmentation procedures. The values shown in bold are discussed in the text.

is the loss of robustness due to confusion of local matches.

5.2.5 Tuning Issues (M2)

In real world scenarios, the tuning of a music performance often slightly deviates from the standard tuning, where a chamber tone of 440 Hz serves as reference frequency. This usually influences pitch related audio features such as chroma features. To compensate for different tunings, one typically integrates a tuning estimation procedure in the feature extraction process [49]. In the previous approaches, we already used tuned chroma features. But since an unkown version of the opera also contains a lot of non-music material (dialogues, applause, etc.), which is also considered in the tuning estimation, the resulting estimate may be incorrect.

With procedure M2, we evaluate the influence of the tuning estimation on the matching proce-

1.7 GB (2 Hz feature resolution) and 42.6 GB (10 Hz), computed from the length of the reference version and the average version length. Upper bounds for the local matching approaches (derived from the maximum query length and the average version length) are 114 MB (2 Hz) and 2.9 GB (10 Hz).

α_i	No.	*occ.*	*dur.*	S1	S2	M1	M2	M3	M4
1	0	23	216.5	0.995	0.994	0.968	0.975	0.975	0.975
2	0	23	283.3	0.996	0.995	0.977	0.976	0.976	0.976
3	0	23	081.4	0.962	0.972	0.881	0.918	0.918	0.918
4	1	23	069.9	0.888	0.927	0.900	0.937	0.937	0.937
5	1	22	070.9	0.808	0.938	0.753	0.747	**0.747**	**0.930**
6	1	23	138.4	0.826	0.986	0.969	0.970	0.952	0.952
7	2	23	122.9	0.854	0.983	0.930	0.932	0.932	0.932
8	2	23	152.4	0.959	0.992	0.977	0.977	0.977	0.977
9	2	23	139.8	0.987	0.987	0.986	0.988	0.970	0.977
10	3	22	073.1	0.930	0.945	0.775	0.772	**0.772**	**0.842**
11	3	23	230.3	0.989	0.992	0.985	0.985	0.985	0.985
12	3	23	074.6	0.990	0.993	0.964	0.967	0.967	0.967
13	3	23	092.1	0.939	0.982	0.979	0.976	0.976	0.976
14	4	23	034.6	0.749	0.876	0.617	**0.735**	**0.904**	0.904
15	4	23	029.3	0.635	0.798	0.496	**0.524**	**0.838**	0.838
16	4	20	026.4	0.550	0.692	0.519	**0.479**	**0.789**	0.789
17	5	23	186.0	0.979	0.985	0.930	0.930	0.930	0.930
18	6	23	287.8	0.984	0.994	0.987	0.989	0.989	0.989
19	7	23	223.9	0.963	0.972	0.992	0.992	0.992	0.992
20	8	23	499.4	0.989	0.997	0.995	0.994	0.994	0.994
21	9	23	258.6	0.979	0.992	0.945	0.988	0.988	0.988
22	9	23	137.6	0.971	0.978	0.985	0.980	0.980	0.980
23	10	22	337.3	0.944	0.951	0.944	0.943	0.987	0.987
24	10	23	301.9	0.977	0.986	0.989	0.988	0.981	0.981
25	10	23	243.8	0.910	0.986	0.933	0.932	0.924	0.924
26	10	23	059.7	0.740	0.889	0.908	0.847	0.883	0.883
27	11	19	104.5	0.631	0.725	0.807	0.807	**0.938**	0.938
28	12	23	356.9	0.882	0.988	0.982	0.982	0.982	0.982
29	13	22	161.5	0.794	0.940	0.943	0.943	0.986	0.986
30	13	22	208.8	0.814	0.951	0.945	0.944	0.984	0.987
31	14	23	168.4	0.729	0.923	0.796	0.790	**0.796**	**0.917**
32	14	19	**012.4**	0.439	0.643	**0.157**	**0.198**	**0.698**	0.735
33	14	22	057.7	0.714	0.846	0.864	0.869	0.827	0.913
34	15	23	147.2	0.745	0.946	0.938	0.937	0.980	0.980
35	16	23	303.6	0.827	0.996	0.990	0.989	0.989	0.989
36	16	23	503.2	0.812	0.965	0.994	0.994	0.994	0.994
37	16	23	241.2	0.781	0.894	0.987	0.987	0.987	0.987
38	16	23	068.2	**0.714**	**0.724**	**0.921**	**0.968**	0.968	0.968
∅		22.5	176.47	**0.852**	**0.930**	0.884	0.891	0.931	0.945

Table 5.2. Relative overlap values averaged over versions for different segments and different procedures. The first column (α_i) indicates the reference segment, the second column (No.) the musical number within the opera, the third column (*occ.*) the number of occurrences of α_i in the 23 versions of the dataset, and the fourth column (*dur.*) refers to the duration in seconds of α_i. S1, S2, M1, M2 M3, and M4 denote the respective segmentation procedures.

dure. This problem can either be addressed on the side of the unknown version or on the query side. In our approach, we use the same chroma sequence for the unknown version as in M1, and simulate the tuning deviations on the query side by computing the chroma sequence for the query with respect to six different reference frequencies (in the range of a semitone). Doing this for each query α_i, we then use the chroma sequence yielding the minimum cost in the matching.

For `Elm1944`, the local tuning adjustment indeed leads to a substantial improvement from 0.705 (M1) to 0.777 (M2), see Table 5.1. Also, there are improvements for certain segments, e.g., α_{38} with 0.921 (M1) compared to 0.968 (M2), see Table 5.2. In this example, the improvement mainly comes from the version `Kle1955`, where α_{38} is now matched onto the correct position.

5.2.6 Global Constraints (M3)

As mentioned in Section 5.2.4, the local matching procedure can easily confuse musically similar parts. Also, the computed segments obtained by individual matches may not be disjoint. In the procedure M3, we impose additional global constraints on the overall segmentation to cope with these two problems.

When using α_i as query, we now consider the entire ranked list of matches (instead of only using the top match as in M1 and M2). From each list we choose the best candidate so that the following global constraints are satisfied:

i) *Disjointness condition*: $\mathrm{P}(\alpha_i) \cap \mathrm{P}(\alpha_j) = \emptyset$

ii) *Temporal monotonicity*: $\alpha_i \prec \alpha_j \Rightarrow \mathrm{P}(\alpha_i) \prec \mathrm{P}(\alpha_j)$.

Here, we define the partial order $\prec$ on the set of segments by $\alpha_1 = [s_1 : t_1] \prec \alpha_2 = [s_2 : t_2] :\Leftrightarrow t_1 < s_2$. An optimal selection of matches from the ranked lists satisfying these global constraints can be computed using dynamic programming (similar to DTW). Howevever, note that in this case the dynamic programming is performed on the coarse segment level and not on the much finer frame level as in the case of global synchronization.

Applying this strategy does indeed improve the overall matching accuracy, on a version level as well as for individual segments, see Table 5.1 and Table 5.2. For example, for the segments $\alpha_{14}/\alpha_{15}/\alpha_{16}$, the results improve from 0.735/0.524/0.479 for M2 to 0.904/0.838/0.789 for M3. Also, the results for α_{32} improve from 0.198 (M2) to 0.698 (M3).

Another interesting example is the relative overlap of 0.938 for α_{27}. This segment is actually missing in four recordings of the opera. Using global constraints, the nonexistence of these segments was correctly identified by procedure $\mathrm{P} = \mathrm{M3}$ resulting in $\mathrm{P}(\alpha_{27}) = \emptyset$. However, the corresponding segment in `Leo1972` was misclassified as nonexistent by M3. A closer inspection revealed that the assumption modeled in the constraint that segments always appear in the same order as in the reference version was violated in this audio version. Here, the musical section covered by α_{27} was placed after α_{30} and used as an introduction before α_{31}. Thus, although strategy M3 stabilizes the overall matching, flexibility concerning the temporal order of segments is lost.

5.2.7 Structural Issues (M4)

Another problem occurs for the segments α_5, α_{10} and α_{31}, having the relative overlap values of 0.747, 0.772, and 0.796 for M3, respectively. According to the musical score, all these sections include repetitions of some music material. The segment α_5 for example should, according to the musical score, follow the structure $IA_1A_2B_1B_2O$, where I is an introductory and O a closing part. However, not all the repetitions are always performed. For example, the alternative structures IA_1B_1O, $IA_1A_2B_1O$, or $IA_1B_1B_2O$ for α_5 all appear in recordings of our dataset (similar variations occur for α_{10} and α_{31}). Such structural deviations can generally not be compensated well in the local matching procedure. Also, for further processing and analysis steps, such as the synchronization between corresponding segments in different recordings, it is important to know the exact structure of a given segment.

For M4, we investigate how structural correspondence of the query with an unknown version influences the segmentation quality. We manually annotated the musical structures occurring for α_5, α_{10} and α_{31} in the different audio versions of the opera. This information is then used in the matching to generate a query which structurally corresponds to the unknown version. The actual matching algorithm is the same as in M3. From the quantitative results in Table 5.2, we can conclude that the structural variations were indeed the cause of the poor performance for these segments: α_5 improves from 0.747 (M3) to 0.930 (M4), α_{10} from 0.772 (M3) to 0.842 (M4) and α_{31} from 0.796 (M3) to 0.917 (M4), see also Figure 5.4b.

5.2.8 Conclusions

In this section, we presented a case study on segmenting given audio versions of an opera into musically meaningful sections that have been specified by a domain expert. Adapting existing synchronization and matching techniques, we discussed various challenges that occur when dealing with real-world scenarios due to the variability of acoustic and musical aspects. Rather than presenting technical details, our main motivation was to show how automated methods may be useful for systematically revealing and understanding the inconsistencies and variations hidden in the audio material. Furthermore, we showed how a procedure based on a combination of local matching and global constraints yields a more flexible and efficient alternative to a global black-box synchronization approach. Besides yielding slightly better results, this alternative procedure also provides a more explicit control to handle the various musical aspects and yields deeper insights into the properties of the audio material.

5.3 Segmentation of Abridged Recordings

As pointed out in Section 5.1, for large-scale works, one often finds abridged versions that only contain fractions of the original work. In this section, we present another reference-based segmentation approach that, instead of trying to find entire segments, establishes correspondences on the frame-level.

Recall, that the approach presented in Section 5.2 applies segment-level matching techniques based on dynamic time warping (DTW) to transfer a labeled reference segmentation to an unknown version. Given a labeled segmentation $\mathcal{A}$ of X, each $\alpha_k \in \mathcal{A}$ is used as query to compute a ranked list of matching candidates in Y. The matching candidates are derived by applying a subsequence variant of the DTW algorithm using the step size conditions $\Sigma = \{(1,1),(1,2),(2,1)\}$, see [76, Chapter 5]. The result of the subsequence DTW procedure is a matching score and an alignment path $\mathcal{P} = (p_1, \ldots, p_L)$ with $p_\ell = (n_\ell, m_\ell)$. $\mathcal{P}$ encodes an alignment of the segment $\alpha_k := [n_1 : n_L] \subseteq [1 : N]$ and the corresponding segment $[m_1 : m_L] \subseteq [1 : M]$ in Y. To derive a final segmentation, one segment from each matching candidate list is chosen such that the sum of the alignment scores of all chosen segments is maximized by simultaneously fulfilling the following constraints. First, the chosen segments have to respect the temporal order of the reference segmentation and second, no overlapping segments are allowed in the final segmentation. Furthermore, the procedure was adapted to be robust to tuning differences of individual segments.

The basic procedure of our proposed frame-level segmentation is sketched in Figure 5.5. First, we use a *partial matching* algorithm (Section 5.3.2) to compute an alignment $\mathcal{L}$. Using $\mathcal{L}$ and the reference label function φ_r obtained from the reference annotation $\mathcal{A}$ of X, an *induced label function* φ_e to estimate the labels on Y is derived (Section 5.3.3). Finally, we apply a mode filter (Section 5.3.4) and a filling up strategy (Section 5.3.5) to derive the final segmentation result.

After introducing some basic notations (Section 5.3.1), we introduce our frame-level segmentation approach based on partial matching (Section 5.3.2). We then present results of a qualitative (Section 5.3.8) and a quantitative (Section 5.3.9) evaluation.

5.3.1 Basic Notations

Using the same notations as in Section 5.2.1, we now introduce some additional notions. Let $[1 : N]$ be an index set representing the *time line* of a discrete signal or feature sequence X. Similarly, let $[1 : M]$ be the time line of a second sequence Y. An *alignment* between two time

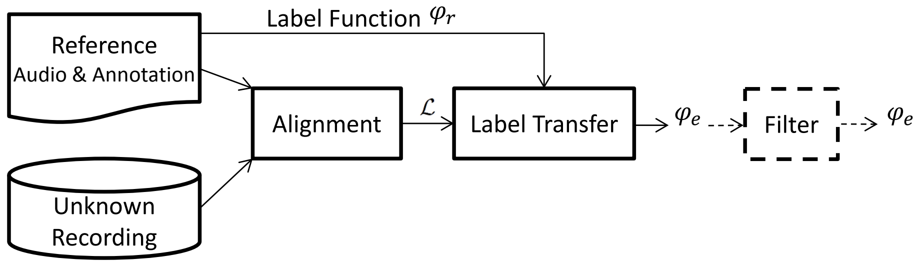


Figure 5.5. Illustration of the proposed frame-level segmentation pipeline. A reference recording with a reference label function φ_r is aligned with an unknown version. The alignment $\mathcal{L}$ is used to transfer φ_r to the unknown version yielding φ_e.

lines $[1:N]$ and $[1:M]$ is modeled as a set

$$\mathcal{L} = (p_1, \ldots, p_L) \subseteq [1:N] \times [1:M].$$

An element $p_\ell = (n_\ell, m_\ell) \in \mathcal{L}$ is called a *cell* and encodes a correspondence between index $n_\ell \in [1:N]$ of the first time line and index $m_\ell \in [1:M]$ of the second one. In the following, we assume $\mathcal{L}$ to be in lexicographic order. $\mathcal{L}$ is called a *match* if $(p_{\ell+1} - p_\ell) \in \mathbb{N} \times \mathbb{N}$ for $\ell \in [1:L-1]$. Note that this condition implies strict monotonicity and excludes the possibility to align an index of the first time line with many indices of the other and vice versa. An alignment can also be constrained by requiring $(p_{\ell+1} - p_\ell) \in \Sigma$ for a given set Σ of admissible step sizes. A typical choice for this set is $\Sigma = \{(1,1),(1,0),(0,1)\}$, which allows to align an index of one time line to many indices of another, and vice versa. Sometimes other sets such as $\Sigma = \{(1,1),(1,2),(2,1)\}$ are used to align sequences which are assumed to be structurally and temporally mostly consistent. If $\mathcal{L}$ fulfills a given step size condition, $\mathcal{P} = \mathcal{L}$ is called a *path*. Note that alignments that fulfill Σ_1 and Σ_2 are both paths, but only an alignment fulfilling Σ_2 is also a match.

Let $[0:K]$ be a set of labels. The label 0 plays a special role and is used to label everything that has not been labeled otherwise. A *label function* φ maps each index $n \in [1:N]$ to a label $k \in [0:K]$:

$$\varphi : [1:N] \to [0:K].$$

The pair $([1:N], \varphi)$ is called a *labeled time line*. Let $n \in [1:N]$ be an index, $\alpha = [s:t]$ be a segment, and $k \in [0:K]$ be a label. Then the pair (n,k) is called a *labeled index* and the pair (α, k) a labeled segment. A labeled segment (α, k) induces a labeling of all indices $n \in \alpha$. Let $\mathcal{A} := \{\alpha_1, \alpha_2, \ldots, \alpha_K\}$ be a segmentation of $[1:N]$ and $[0:K]$ be the label set. Then the set $\{(\alpha_k, k) \,|\, k \in [1:K]\}$ is called a *labeled segmentation* of $[1:N]$. From a labeled segmentation one obtains a label function on $[1:N]$ by setting $\varphi(n) := k$ for $n \in \alpha_k$ and $\varphi(n) := 0$ for $n \in [1:N] \setminus \bigcup_{k\in[1:K]} \alpha_k$. Vice versa, given a label function φ, one obtains a labeled segmentation in the following way. We call consecutive indices with the same label a

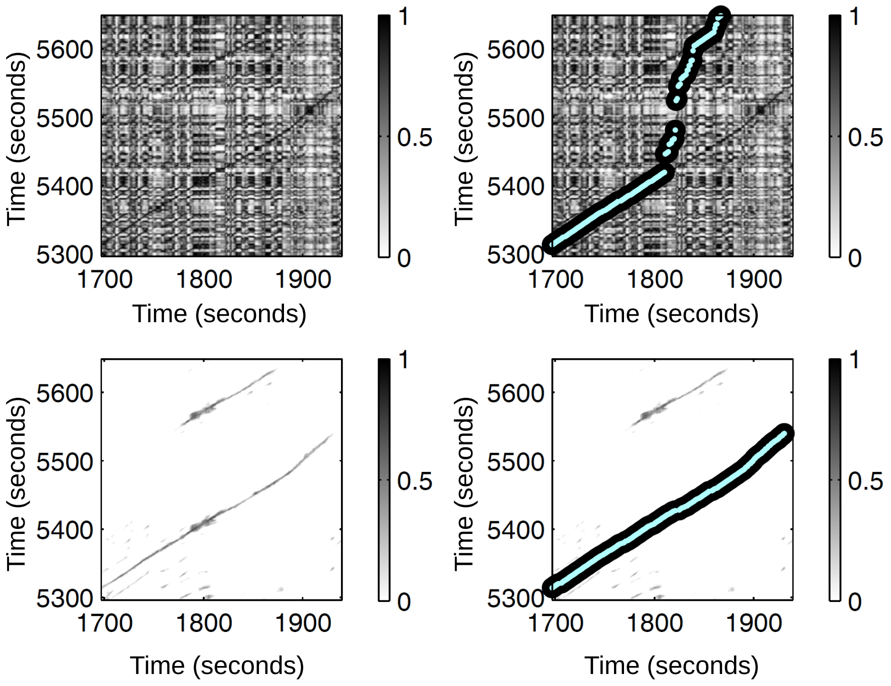


Figure 5.6. Excerpt of similarity matrices of the reference `Kle1973` and `Kna1939` before (top) and after enhancement (bottom), shown without match (left) and with match (right).

run. A segmentation of $[1 : N]$ is then derived by considering runs of maximal length. We call this segmentation the *segmentation induced by* φ.

5.3.2 Frame-Level Segmentation Approach

Now we describe a procedure for computing a partial matching between two sequences as introduced in [76]. To compare the two feature sequences X and Y, we compute a similarity matrix $S(n, m) := s(x_n, y_m)$, where s is a suitable similarity measure. The goal of the partial matching procedure is to find a score-maximizing match through the matrix S. To this end, we define the *accumulated score matrix* D by

$$D(n, m) := \max\{D(n-1, m), D(n, m-1), D(n-1, m-1) + S(n, m)\}$$

with $D(0, 0) := D(n, 0) := D(0, m) := 0$ for $1 \leq n \leq N$ and $1 \leq m \leq M$. The score maximizing match can then be derived by backtracking through D, see [76, Chapter 5]. Note that only diagonal steps contribute to the accumulated score in D. The partial matching algorithm is more flexible in aligning two sequences than the subsequence DTW approach, as it allows for skipping frames at any point in the alignment. However, this increased flexibility comes at the cost of loosing robustness. To improve the robustness, we apply path-enhancement (smoothing) on S, and suppress other noise-like structures by thresholding techniques [80, 115]. In this way, the algorithm is less likely to align small scattered fragments. Figure 5.6 shows an excerpt of a similarity matrix before and after path-enhancement together with the computed matches.

5.3.3 Induced Label Function

Given a labeled time line $([1:N], \varphi_r)$ and an alignment $\mathcal{L}$, we derive a label function φ_e on $[1:M]$ by setting:

$$\varphi_e(m) := \begin{cases} \varphi_r(n) & \text{if } (n,m) \in \mathcal{L} \\ 0 & \text{else,} \end{cases}$$

for $m \in [1:M]$. See Figure 5.7 for an illustration.

5.3.4 Local Mode Filtering

The framewise transfer of the labels may lead to very short and scattered runs. Therefore, to obtain longer runs and a more homogeneous labeling, especially at segment boundaries, we introduce a kind of smoothing step by applying a mode filter. The *mode* of a sequence $\mathcal{S} = (s_1, s_2, \ldots, s_N)$ is the most frequently appearing value and is formally defined by $\mathrm{mode}(\mathcal{S}) := \arg\max_{s \in \mathcal{S}} |\{n \in [1:N] \colon s_n = s\}|$. A *local mode filter* of length $L = 2q+1$ with $q \in \mathbb{N}$ replaces each element $s_n \in \mathcal{S}$, $n \in [1:N]$, in a sequence by the mode of its neighborhood $(s_{n-q}, \ldots, s_{n+q})$:

$$\mathrm{modefilt}_q\left(\mathcal{S}\right)(n) := \mathrm{mode}(s_{n-q}, \ldots, s_{n+q}).$$

Note that the mode may not be unique. In this case, we apply the following strategy in the mode filter. If the element s_n is one of the modes, s_n is left unmodified by the filter. Otherwise, one of the modes is chosen arbitrarily.

In our scenario, we apply the local mode filter on a labeled time line $([1:N], \varphi_e)$ by inputting the sequence $\varphi_e([1:N]) := (\varphi_e(1), \varphi_e(2), \ldots, \varphi_e(N))$ into the filter, see Figure 5.7 for an illustration. The reason to use the mode opposed to the median to filter segment labels, is that labels are nominal data and therefore have no ordering (integer labels were only chosen for the sake of simplicity).

5.3.5 From Frames to Segments (Filling Up)

In the last step, we derive a segmentation from the label function φ_e. As indicated in Section 5.3.1, we could simply detect maximal runs and consider them as segments. However, even after applying the mode filter, there may still be runs sharing the same label that are interrupted

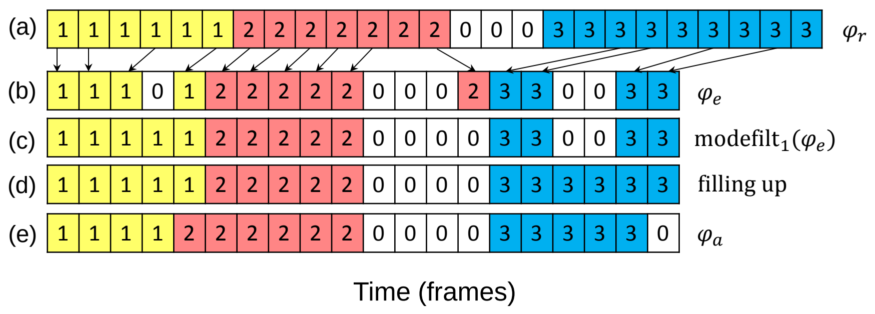

(a) 1 1 1 1 1 1 2 2 2 2 2 2 2 0 0 0 3 3 3 3 3 3 3 3 3 φ_r
(b) 1 1 1 0 1 2 2 2 2 2 0 0 0 2 3 3 0 0 3 3 φ_e
(c) 1 1 1 1 1 2 2 2 2 2 0 0 0 0 3 3 0 0 3 3 $\text{modefilt}_1(\varphi_e)$
(d) 1 1 1 1 1 2 2 2 2 2 0 0 0 0 3 3 3 3 3 3 filling up
(e) 1 1 1 1 2 2 2 2 2 2 0 0 0 0 3 3 3 3 3 0 φ_a
Time (frames)

Figure 5.7. Example of frame-level segmentation. The arrows indicate the match between the reference version and the unknown version. **(a)** Reference label function. **(b)** Induced label function. **(c)** Mode filtered version of (b) with length $L = 3$. **(d)** Filling up on (c). **(e)** Ground truth label function.

by non-labeled parts (labeled zero). In our scenario, we assume that all segments have a distinct label and occur in the same succession as in the reference. Therefore, in the case of a sequence of equally labeled runs that are interrupted by non-labeled parts, we can assume that the runs belong to the same segment. Formally, we assign an index in between two indices with the same label (excluding the zero label) to belong to the same segment as these indices. To construct the final segments, we iterate over each $k \in [1 : K]$ and construct the segments $\alpha_k = [s_k : e_k]$, such that $s_k = \min\{m \in [1 : M] \colon \varphi(m) = k\}$, and $e_k = \max\{m \in [1 : M] \colon \varphi(m) = k\}$, see Figure 5.7 for an example.

5.3.6 Evaluation

In this section, we compare the previous segment-level matching procedure with our novel frame-level segmentation approach based on experiments using abridged versions of the opera "Der Freischütz". First we give an overview of our test set and the evaluation metric. Subsequently, we discuss the results of the segment-level approach and the frame-level procedure on the abridged versions (Section 5.3.8). Finally, we present an experiment where we systematically derive synthetic abridged versions from a complete version of the opera (Section 5.3.9).

In the following experiments, we use the recording of Carlos Kleiber performed in 1973 with a duration of 7763 seconds as reference version. The labeled reference segmentation consists of 38 musical segments, see Figure 5.8. Furthermore, we consider five abridged versions that were recorded between 1933 and 1994. The segments of the opera that are performed in these versions are indicated by Figure 5.8. Note that the gray parts in the figure correspond to dialogue sections in the opera. In the following experiments, the dialogue sections are considered in the same way as non-labeled (non-musical) parts such as applause, noise or silence. In the partial matching algorithm, they are excluded from the reference version (by setting the similarity score in these regions to minus infinity), and in the segment-level matching procedure, the dialogue parts are not used as queries.

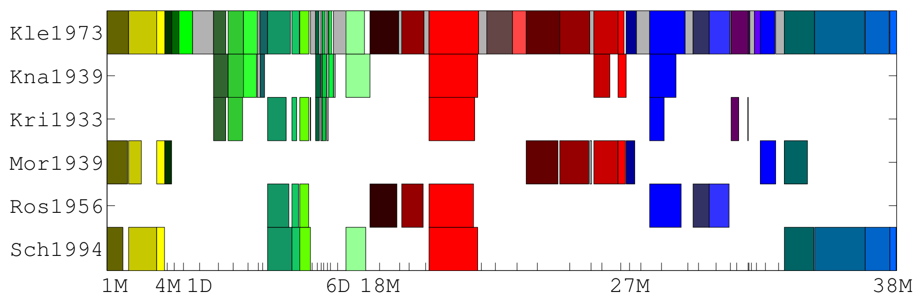


Figure 5.8. Visualization of relative lengths of the abridged versions compared to the reference version `Kle1973`. The gray segments indicate dialogues whereas the colored segments are musical parts.

Throughout all experiments, we use CENS_{10}^{41} features, see Section 3.3.3 and [76]. They are computed with a feature rate of 1 Hz (derived from 10 Hz pitch features with a smoothing length of 41 frames and a downsampling factor of 10). Each feature vector covers roughly 4.1 seconds of the original audio.

In our subsequent experiments, the following segment-level matching (M4) and frame-level segmentation (F1 – F4) approaches are evaluated:

(M4) Previously introduced segment-level matching, see Section 5.2.7 for details.

(F1) Frame-level segmentation using a similarity matrix computed with the cosine similarity s defined by $s(x, y) = \langle x, y\rangle$ for features x and y, see Section 5.3.2.

(F2) Frame-level segmentation using a similarity matrix with enhanced path structures using the SM Toolbox [80]. For the computation of the similarity matrix, we used forward/backward smoothing with a smoothing length of 20 frames (corresponding to 20 seconds) with relative tempi between $0.5 - 2$, sampled in 15 steps. Afterwards, a thresholding technique that retained only 5% of the highest values in the similarity matrix and a scaling of the remaining values to $[0, 1]$ is applied. For details, we refer to [80] and Section 5.3.2.

(F3) The same as in F2 with a subsequent mode filtering using a filter length $L = 21$ frames, see Section 5.3.4 for details.

(F4) The segmentation derived from F3 as described in Section 5.3.5.

5.3.7 Frame Accuracy

To evaluate the performance of the different segmentation approaches, we calculate the *frame accuracy*, which is defined as the ratio of correctly labeled frames and the total number of frames in a version. Given a ground truth label function φ_a and an induced label function φ_e, the frame

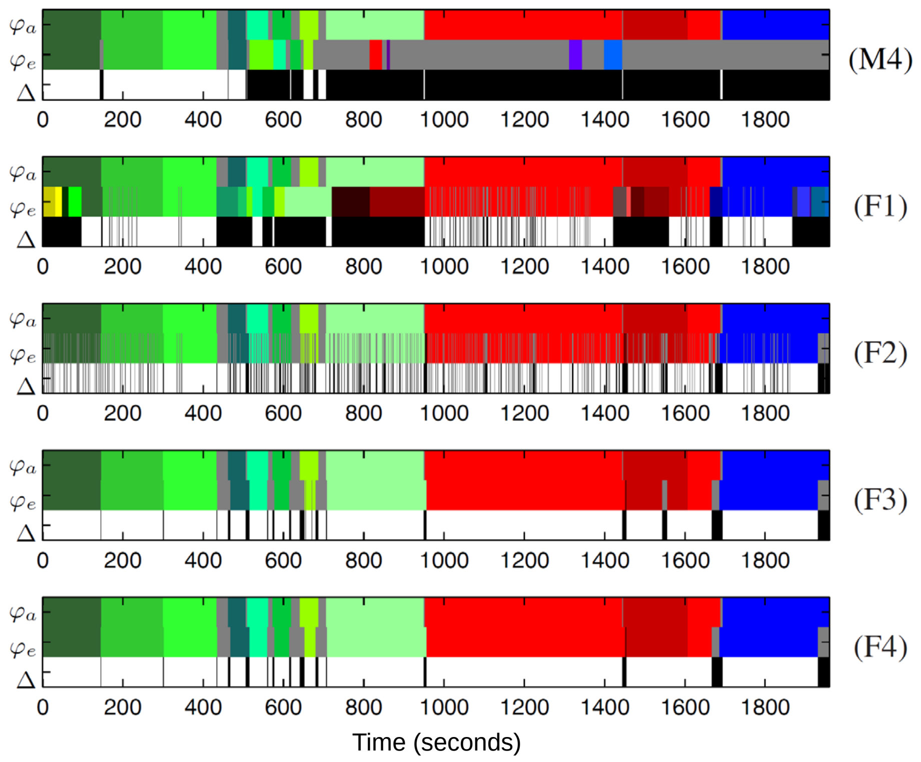


Figure 5.9. Segmentation results on `Kna1939` showing the ground truth label function φ_a, the induced label function φ_e, and the agreement sequence $\Delta := \Delta(\varphi_a, \varphi_e)$. White encodes an agreement and black a disagreement between φ_a and φ_e. **(M4),(F1),(F2),(F3),(F4):** See Section 5.3.6.

accuracy A_f is computed as following:

$$A_f := \frac{\sum_{k\in[0:K]} \left|{\varphi_a}^{-1}(k) \cap {\varphi_e}^{-1}(k)\right|}{\sum_{k\in[0:K]} \left|{\varphi_a}^{-1}(k)\right|}$$

We visualize the accuracy by means of an *agreement sequence* $\Delta(\varphi_a, \varphi_e)$ which we define as $\Delta(\varphi_a, \varphi_e)(m) := 1$ (white) if $\varphi_a(m) = \varphi_e(m)$ and $\Delta(\varphi_a, \varphi_e)(m) := 0$ (black) otherwise. The sequences $\Delta(\varphi_a, \varphi_e)$ visually correlates well with the values of the frame accuracy A_f, see Table 5.3 and the Figure 5.9. Note that in structural segmentation tasks, it is common to use different metrics such as the pairwise precision, recall, and f-measure [71]. These metrics disregard the absolute labeling of a frame sequence by relating equally labeled pairs of frames in an estimate to equally labeled frames in a ground truth sequence. However, in our scenario, we want to consider frames that are differently labeled in the ground truth and the induced label function as wrong. As the pairwise f-measure showed the same tendencies as the frame accuracy (which can be easily visualized), we decided to only present the frame accuracy values.

5.3.8 Qualitative Evaluation

In this section, we qualitatively discuss the results of our approach in more detail by considering the evaluation of the version `Kna1939`. For each of the five approaches, the results are visualized in a separate row of Figure 5.9, showing the ground truth φ_a, the induced label function φ_e and the agreement sequence $\Delta(\varphi_a, \varphi_e)$.

For `Kna1939`, the segment-level matching approach M4 does not work well. Only 28% of the frames are labeled correctly. The red segment, for example, at around 1500 seconds is not matched despite the fact that it has roughly the same overall duration as the corresponding segment in the reference version, see Figure 5.8. Under closer inspection, it becomes clear that it is performed slower than the corresponding segment in the reference version, and that some material was omitted at the start, in the middle and the end of the segment. The frame-level matching approach F1 leads to an improvement, having a frame accuracy of $A_f = 0.520$. However, there are still many frames wrongly matched. For example, the overture of the opera is missing in `Kna1939`, but frames from the overture (yellow) of the reference are matched into a segment from the first act (green), see Figure 5.9. Considering that the opera consists of many scenes with harmonically related material and that the partial matching allows for skipping frames at any point in the alignment, it sometimes occurs that not the semantically corresponding frames are aligned, but harmonically similar ones. This problem is better addressed in approach F2, leading to an improved frame accuracy of 0.788. The enhancement of path structures in the similarity matrix in this approach leads to an increased robustness of the partial matching. Now, all high similarity values are better concentrated in path structures of the similarity matrix.

As a result, the algorithm is more likely to follow sequences of harmonically similar frames, see also Figure 5.6. However, to follow paths that are not perfectly diagonal, the partial matching algorithm needs to skip frames in the alignment, which leads to a more scattered label function. This is approached by F3 which applies a mode filter on the label function from F2, resulting in an improved frame accuracy of 0.927. In F4, the remaining gaps in the label function of F3 are filled up, which leads to a frame accuracy of 0.934.

5.3.9 Quantitative Evaluation

In this section, we discuss the results of Table 5.3. Note that all abridged versions have less than 50% of the duration of the reference version (7763 seconds). From the mean frame accuracy values for all approaches, we can conclude that the segment-level matching (0.563) is not well suited for dealing with abridged versions, whereas the different strategies in the frame-level approaches F1 (0.687) – F4 (0.924) lead to a subsequent improvement of the frame accuracy. Using the segment-level approach, the frame accuracies for the versions `Ros1956` (0.887) and `Sch1994` (0.742) stand out compared to the other versions. The segments that are performed in

Version	*dur.*(*s*)	**M4**	F1	F2	F3	**F4**
`Kna1939`	1965	0.283	0.520	0.788	0.927	0.934
`Kri1933`	1417	0.390	0.753	0.777	0.846	0.870
`Mor1939`	1991	0.512	0.521	0.748	0.841	0.919
`Ros1956`	2012	0.887	0.749	0.817	0.850	0.908
`Sch1994`	2789	0.742	0.895	0.936	0.986	0.989
mean	2035	0.563	0.687	0.813	0.890	0.924

Table 5.3. Frame accuracy values on abridged versions. M4: Segment-level matching, F1: Frame-level segmentation, F2: Frame-level segmentation with path-enhanced similarity matrix, F3: Mode filtering with $L = 21$ seconds on F2. F4: Derived Segmentation on F4.

these versions are not shortened and therefore largely coincide with the segments of the reference version. This explains why the segment-level matching still performs reasonably well on these versions.

In Figure 5.10, we show the frame accuracy results for the approaches M4 and F4 obtained from an experiment on a set of systematically constructed abridged versions. The frame accuracy values at 100% correspond to a subset of 10 segments (out of 38) that were taken from a complete recording of the opera “Der Freischütz” recorded by Keilberth in 1958. From this subset, we successively removed 10% of the frames from each segment by removing 5% of the frames at the start, and 5% of the frames at the end sections of the segments. In the last abridged version, only 10% of each segment remains. This experiment further supports the conclusions that the segment-level approach is not appropriate for dealing with abridged versions, whereas the frame-level segmentation approach stays robust and flexible even in the case of strong abridgments.

5.3.10 Conclusions

In this section, we approached the problem of transferring the segmentation of a complete reference recording onto an abridged version of the same musical work. We compared the proposed frame-level segmentation approach based on partial matching with a segment-level matching strategy. In experiments with abridged recordings, we have shown that our frame-level approach is robust and flexible when enhancing the path structure of the used similarity matrix and applying a mode filter on the labeled frame sequence before deriving the final segmentation.

5.4 Further Notes

In this chapter, we presented two approaches for segmenting given audio versions of an opera into musically meaningful sections that have been specified by a domain expert.

In these approaches, we focused on audio matching using chroma-based audio features which are related to the harmonic progression of the music. For opera music, additional cues such

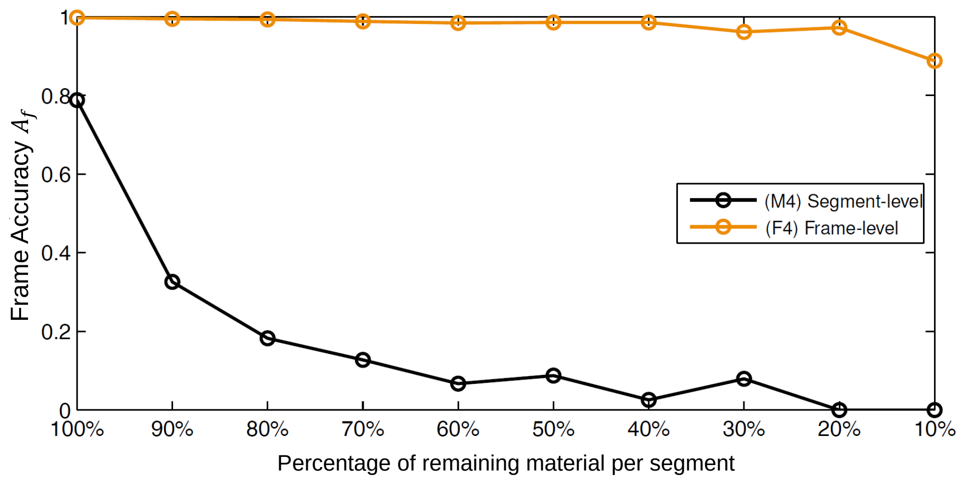


Figure 5.10. Performance of segment-level approach (M4) versus frame-level approach (F4) on constructed abridged versions. See Section 5.3.9.

as speech and singing voice detection, could further stabilize the matching results. In audio signal processing, the task of discriminating between speech and music signals is a well-studied problem [113, 117]. Most procedures for speech/music discrimination use machine learning techniques that automatically learn a model from example inputs (i.e., audio material labeled as speech and audio material labeled as music) in order to make data-driven predictions or decisions for unknown audio material [9]. The task of speech/music discrimination is an important step for automated speech recognition and general multimedia applications. Within our opera scenario, it is beneficial to also consider additional classes that correspond to applause and passages of silence. Such extensions have also been discussed extensively in the literature [93].

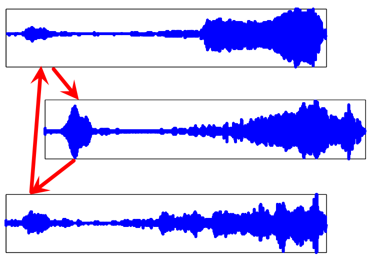
If a recording of “Der Freischütz” is complete and has no structural deviations, detecting the boundaries of the opera’s dialogues could already lead to the segment boundaries of our reference segmentation. Furthermore, repetitions in the form of different verses such as in No. 4 could be better differentiated by the matching procedure when incorporating features from speech processing. In future work, these issues should be addressed in an extended segmentation procedure.

In conclusion, our investigations showed that automated procedures may yield segmentation results with an accuracy of over 90%, even for versions with strong structural and acoustic variations. Still, for certain applications, segmentation errors in the order of 5% to 10% may not be acceptable. Here, we could demonstrate that automated procedures may still prove useful in semiautomatic approaches that also involve some manual intervention.

Chapter 6

Ground-Truth-Independent Analysis of Music Alignments

The work in this chapter is mainly based on our contribution in [101].



Music alignments are usually evaluated by comparing them to a manually generated ground-truth annotation. These annotations are not always available, and their creation is usually a very labor-intensive task. For many musical works, especially in classical music, there exists more than one version of the same piece of music. In this chapter, we introduce a concept for a ground-truth-independent analysis of music alignments which relies on the availability of at least three versions of the same piece of music. The main idea is to use pairwise alignments in a circular way: first transferring original time points of a fixed reference version to a second version, then transferring the resulting time points to a third version, and finally the third version's time points back to the reference version. A triple error is then computed by comparing the original time points with their circularly aligned version.

In the following, after introducing the basic idea of a ground-truth-independent evaluation, we formalize the concept of the triple error (Section 6.2) and discuss its theoretical limitations. We then perform experiments to illustrate that in practice, despite its theoretical limitations, the triple error can be a useful tool for analyzing and evaluating alignment results (Section 6.3). Related work will be discussed in the respective sections. At the end of the Chapter (Section 6.5), we put the triple approach into a larger context of approaches using multiple versions of the same piece of music for analysis and computation. These are also referred to as *cross-version* approaches.

6.1 Ground-Truth-Independent Analysis

In the field of Music Information Retrieval, music alignment algorithms are an important tool and active area of research [20, 57, 5, 128]. The goal of music alignment is to find corresponding time positions between different versions of the same piece of music. In an *audio-audio* alignment scenario, the task could be to align two different versions of the Mazurka Op. 63 No. 3 composed by Chopin, one performed by Rubinstein and the other by Cohen. To evaluate the quality of such an alignment, ground-truth annotations marking corresponding time positions in the two recordings are needed. These time positions are typically note onsets, beats, or measure positions. However, ground-truth annotations are not always available and their manual creation is very labor intensive. Furthermore, an evaluation is restricted to the time positions of the ground-truth annotations.

Often, especially in classical music, more than two versions of the same piece of music are available, sometimes even in different representations. For example, different audio recordings and a musical score representation might be available. More recently, multiple versions available for the same piece of music were used jointly, to improve and stabilize alignment methods [5, 128, 108, 4].

In this chapter, we exploit the availability of different versions for a ground-truth independent evaluation and analysis of alignment methods. For example, when considering a third recording of the Mazurka Op. 63 No. 3 performed by Ezaki, we can build a triplet of recordings of the same piece of music. Our main idea is to use the pairwise alignments between the versions in such a triplet in a circular way, to evaluate the alignments without the need for ground-truth annotations. An arbitrary given starting point on the time axis of a version V_1 is then circularly aligned back onto the time axis of V_1 through a version V_2 and a version V_3. In this way, a triple error can be computed by measuring the difference between the starting point and its circularly aligned version (see Figure 6.1). In the case of alignments that consistently align corresponding time positions in the different versions, we expect the triple error to be zero (see Figure 6.1a). In the presence of alignment errors, we expect the triple error to be greater than zero (see Figure 6.1b). However, note that the triple error might be higher or lower than a pairwise

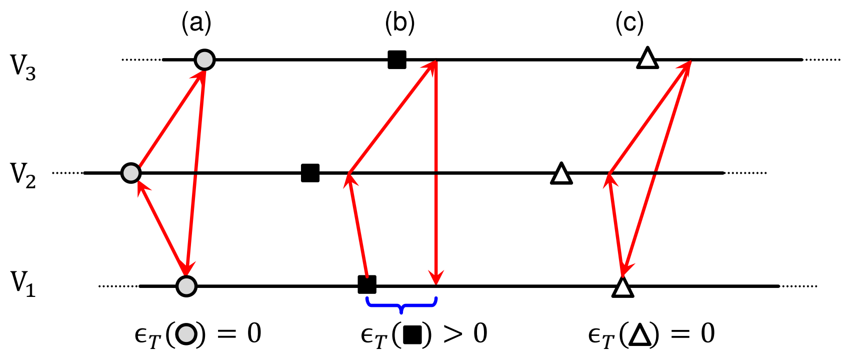


Figure 6.1. Illustration of the triple error on the time axis of three different versions V_1, V_2, and V_3. The red arrows indicate alignments between the different versions. The circles, squares and triangles mark corresponding ground-truth positions in the different versions. The triple error is denoted by ϵ_T. **(a)** Consistent alignment with a triple error of zero. **(b)** Inconsistent alignments with a triple error greater than zero. **(c)** Inconsistent alignments with zero triple error.

alignment error measured with ground-truth annotations. Errors from the different alignments can either accumulate or cancel out. The latter is illustrated in Figure 6.1c, where the triple error is zero, despite the fact that there are errors in the pairwise alignments. Hence, measuring a zero triple error is only a necessary condition for the alignment to be correct. However, a triple error greater than zero still implies errors in at least one of the alignments involved in the triple error computation.

6.2 Triple Error

Let V_1 and V_2 be two versions of the same piece of music with the corresponding time-continuous axes $[0, T_1]$ and $[0, T_2]$. An *alignment* $\mathcal{A}$ from V_1 to V_2 is a mapping of time points from V_1 onto corresponding time points of V_2. We model an alignment by the function

$$\mathcal{A}\colon [0, T_1] \to [0, T_2] \tag{6.1}$$

which is monotonous, i.e. $\mathcal{A}(s) \leq \mathcal{A}(t)$ for $s, t \in [0, T_1]$ with $s \leq t$. Furthermore, an alignment fulfills the boundary constraints $\mathcal{A}(0) = 0$ and $\mathcal{A}(T_1) = T_2$. Sometimes further constraints on the slope are required. Typically, an alignment is evaluated using pairs of manually specified ground-truth annotations that mark corresponding time positions in the two different versions. Each ground-truth pair is specified as

$$(g_1, g_2) \in [0, T_1] \times [0, T_2]. \tag{6.2}$$

Using these ground-truth pairs, the *pairwise alignment error* for a given alignment $\mathcal{A}$ between two versions is defined as $\epsilon_P\colon [0,T_1] \to \mathbb{R}$ with

$$\epsilon_P(g_1) := |\mathcal{A}(g_1) - g_2|\,. \tag{6.3}$$

The main drawback of this error measure is the need for ground-truth annotations which are often unavailable.

We now formalize our main idea, which exploits the fact that for many pieces, more than two versions are available. Let the triplet (V_1, V_2, V_3) be three versions with corresponding time axes $[0, T_i]$, where $i \in \{1, 2, 3\}$. Furthermore, let V_1 serve as a reference version. Assume the alignments $\mathcal{A}^{1\to2}$, $\mathcal{A}^{2\to3}$, $\mathcal{A}^{3\to1}$ are given, where $\mathcal{A}^{i\to j}$ denotes the alignment of version V_i to V_j. Using these alignments in a circular way, a time point $t \in [0, T_1]$ of version V_1 can be subsequently aligned to V_2, V_3 and back to V_1 by applying the alignments in a composition to compute $t' := \mathcal{A}^{3\to1}\left(\mathcal{A}^{2\to3}\left(\mathcal{A}^{1\to2}(t)\right)\right) \in [0, T_1]$. We can now measure the difference between the original time point t and its circularly aligned version t' by the *triple error* $\epsilon_T\colon [0,T_1] \to \mathbb{R}$ that is computed by

$$\epsilon_T := \left|\mathcal{A}^{3\to1}\left(\mathcal{A}^{2\to3}\left(\mathcal{A}^{1\to2}(t)\right)\right) - t\right|\,. \tag{6.4}$$

The main advantage of the triple error compared to the pairwise alignment error is its independence of ground-truth annotations. However, from a theoretical point of view, the triple error has to be considered with great care. First, note that the triple error is based on three pairwise alignments. Hence, when measuring a triple error we know that the pairwise alignment error in one of the alignments is at least one third of the measured triple error. However, the exact pairwise alignment errors cannot be inferred from the triple error. Second, a zero triple error does not necessarily imply that the alignments are error free. An error introduced by one alignment can be compensated by another (see Figure 6.1c). There are even alignments that always lead to a zero triple error. We can construct such an alignment by linearly scaling the durations of the different versions to be aligned. This leads to a triple error of zero for all time points, although the pairwise alignment error might be high. This proves that *a zero triple error is only a necessary condition* for a zero pairwise alignment error, but not a sufficient one. However, measuring a *triple error greater than zero is a sufficient condition* that there is an error in one of the pairwise alignments involved in the triplet. Hence, the triple error can still be a useful measure for the analysis of alignments. In practice, when using reasonable alignments, we expect the triple error to reflect the values of the pairwise alignment error in most of the cases.

ID	Composer	Piece	$\overline{\text{dur}}[s]$	Type	#(GT)	#(GT)/s
F06	Weber	Freischütz, No. 6	297	measures	150	0.51
F08	Weber	Freischütz, No. 8	540	measures	199	0.37
F09	Weber	Freischütz, No. 9	428	measures	200	0.47
M17-4	Chopin	Op. 17, No. 4	247	beats	396	1.60
M24-2	Chopin	Op. 24, No. 2	124	beats	360	2.90
M63-3	Chopin	Op. 63, No. 3	144	beats	229	1.59
M68-3	Chopin	Op. 68, No. 3	89	beats	181	2.03

Table 6.1. The three numbers of "Der Freischütz" and the four Chopin Mazurkas used in our experiments. For each piece, there are three recordings in our dataset with an average duration of $\overline{\text{dur}}[s]$. **#(GT):** number of annotated ground-truth positions, **Type:** type of annotations (measure or beat annotations), **#(GT)/s:** average number of ground-truth annotations per second.

6.3 Experiments

In this section, we first review the dataset (Section 6.3.1) and the alignment approach (Section 6.3.2) used in our experiments. Within the remaining sections, we perform experiments to compare the triple error with the pairwise alignment error. These experiments indicate that often, the same conclusions as drawn from a ground-truth based analysis can be made, by only using the ground-truth independent triple error. In particular, we discuss the potential of the triple error to identify problematic positions within an alignment (Section 6.3.3). Furthermore, we show how the triple error can be used to identify problematic versions (Section 6.3.4). Finally, we indicate its potential to compare different alignment methods (Section 6.3.5).

6.3.1 Dataset

For our experiments, we use a set of three excerpts of the opera "Der Freischütz" and four Chopin Mazurkas [112] (see Table 6.1). We use measure annotations (marking musical measure boundaries) for the opera excerpts and beat annotations for the Chopin Mazurkas (marking beat positions). For a given version, we denote all available ground-truth time positions as *ground-truth grid*. The Mazurkas are piano pieces that typically have clear note onsets, whereas the operas are composed of singing and orchestral music which typically has soft or blurred onsets. Using pieces with different instrumentations allows us to identify problems of an alignment method related to specific instrumentations. For each piece, we consider a triplet of recordings. For the opera excerpts the recordings were conducted by Bloemeke (2013), Kleiber (1973), and Furtwaengler (1954). The Mazurka recordings were performed by Ezaki (2006), Cohen (1997), and Rubinstein (1966). In our dataset, there is no canonical order of the different versions. We therefore compute all pairwise alignments $\mathcal{A}^{i \to j}$ with $(i, j) \in [1 : 3]^2$ and $i \neq j$. This leads to six possible pairs for which we compute six triple errors (two for each version as a reference) and

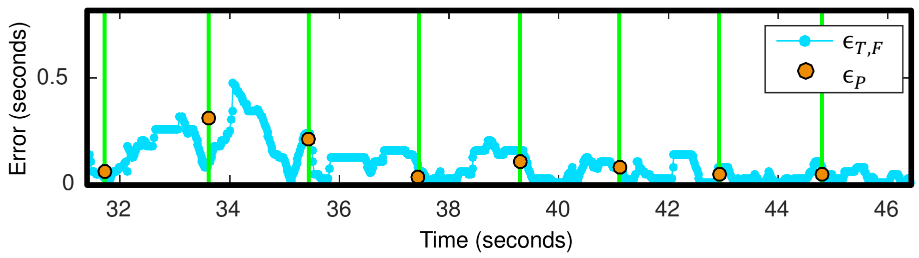


Figure 6.2. Example of frame-wise triple error $\epsilon_{T,F}$ (cyan curve) and pairwise alignment error ϵ_P (orange dots) on the piece F06 using the performance by Kleiber as reference. The solid green lines show the ground-truth grid.

six pairwise alignment errors.

6.3.2 Alignment Method

For the experiments in this chapter, we use the memory-restricted multiscale DTW alignment method introduced in Chapter 4. Remember, that the goal of DTW is to find an optimal alignment of two sequences under certain restrictions. In the case of two different versions of the same piece of music, one first needs to compute a suitable feature representation. Chroma features capture the coarse harmonic progression of the music and are commonly used in music alignments [57, 77, 36, 30]. To achieve a high temporal accuracy, we combine chroma-based features with features capturing note onset informations (DLNCO) as in [36], see also Section 3.3.4. Furthermore, we extend the approach to only align two given versions between the first and last ground-truth annotation for each recording. In this way, we exclude boundary artifacts that are caused by aligning silence or non-musical sounds such as applause or noise. All alignments were computed at a feature rate of 50 Hz.

Note that in Section 6.2, an alignment was modeled on a continuous time axis. The computed alignments, however, are time-discrete. A time-discrete alignment is a sequence of pairs containing corresponding time positions of the aligned versions. This makes it necessary to use interpolation to align arbitrary time positions across different versions [34].

In the following, we mainly discuss three error measures. First, the pairwise alignment error ϵ_P that is computed on the ground-truth grid using Equation (6.3). Second, the triple error ϵ_T that is evaluated only at the time positions of the ground-truth grid. And third, the frame-wise triple error $\epsilon_{T,F}$ that is evaluated on a 50 Hz frame grid which corresponds to the feature resolution of the alignments. The two last measures are computed with Equation (6.4).

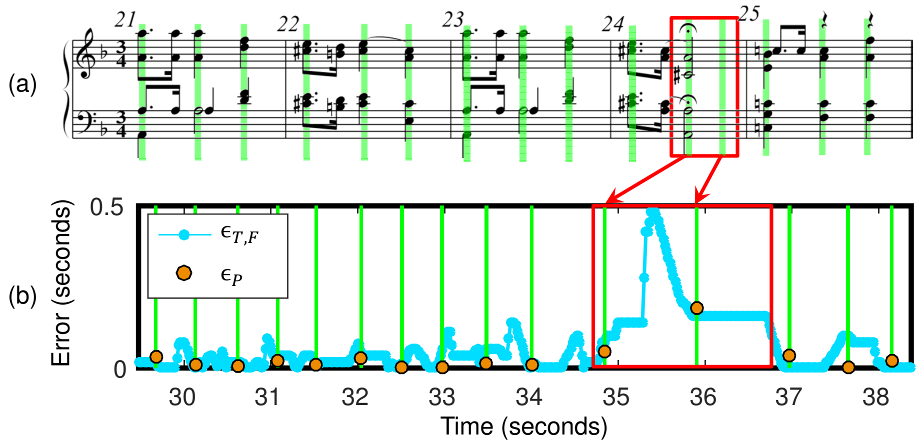


Figure 6.3. Excerpt showing frame-wise triple error $\epsilon_{T,F}$ and pairwise alignment error (solid cyan curve) ϵ_P (orange dots) on the piece M68-3 using the performance by Cohen as reference. The green lines mark the ground-truth grid. **(a)** Musical score excerpt of M68-3. **(b)** frame-wise triple error corresponding to the score excerpt shown in (a).

6.3.3 Identification of Alignment Errors

Figure 6.2 shows an example of the frame-wise triple error $\epsilon_{T,F}$ and the pairwise alignment error ϵ_P on an excerpt of the piece F06. The green solid lines indicate the ground-truth grid of musical measures at which ϵ_P is computed. In this example, we can clearly see that the pairwise alignment error and the triple error often show the same tendencies at the positions of the ground-truth grid. Note that the ground-truth grid marks characteristic positions in the recordings that usually coincide with note onsets. At these positions, the alignment is more likely to have lower errors. The positions in between the grid positions often correspond to steady notes. These lead to regions of homogenous feature values where the alignment typically exhibits higher error values.

Figure 6.3a+b shows an excerpt of the piece M68-3 with the corresponding musical score together with the pairwise alignment error ϵ_P and the frame-wise triple error $\epsilon_{T,F}$. Here again, the two error measures coincide well at the ground-truth grid, which marks beat positions in this example. For the same reason as above, the triple error is again much larger in between the positions of the ground-truth grid. Note that the beat annotations not always go along with onset positions and vice versa. There are *silent beats* that have no associated onset. For example, in Figure 6.3, the second beat of the half note within the red rectangles is silent. At the position of the silent beat, both the pairwise alignment error and the triple error exhibit high values. This illustrates that the higher errors between note onsets are not an artifact of the frame-wise triple error. Hence, the triple error can sometimes even lead to additional insights.

Figure 6.4 shows the mean of the triple errors $\epsilon_{T,F}$ and ϵ_T, and the mean of the pairwise alignment errors ϵ_P for all pieces in the dataset, see also Table 6.1. Note that the visualization

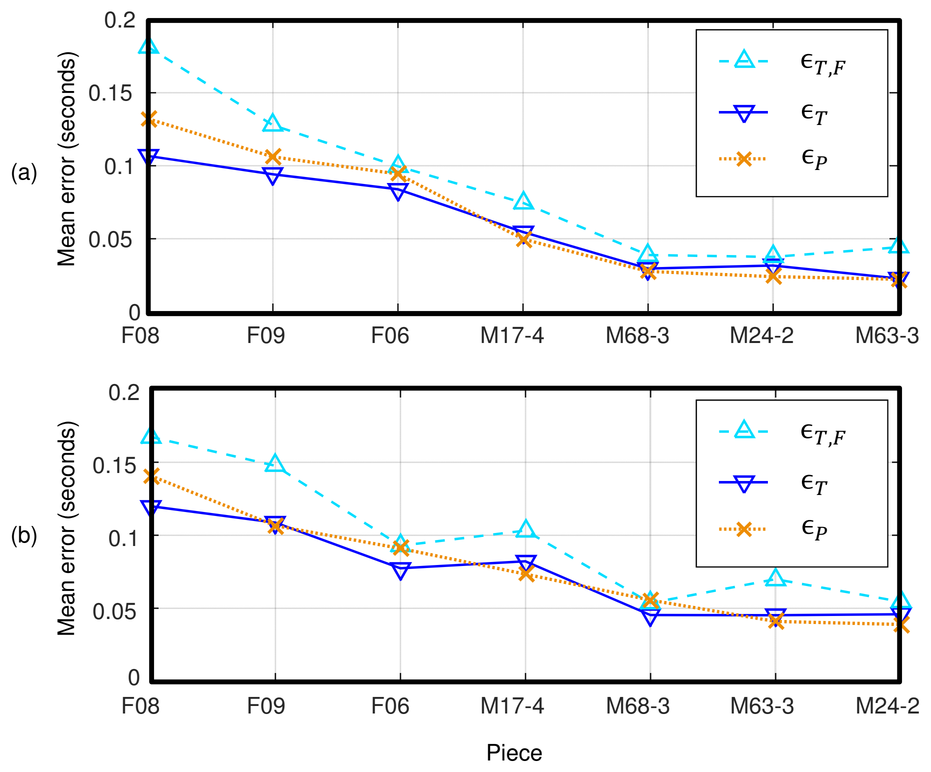


Figure 6.4. Mean values for frame-wise triple error $\epsilon_{T,F}$, triple error ϵ_T (computed on ground-truth grid), and pairwise alignment error ϵ_P for the pieces in our dataset. The pieces are sorted in descending order with respect to the mean pairwise alignment error. **(a)** Alignment method $\mathcal{A}_1$ with onset informations, **(b)** Alignment method $\mathcal{A}_2$ using only chroma features without onset information.

of the results is sorted by decreasing values of the pairwise alignment error. Overall, the triple errors and the pairwise alignment errors show similar tendencies. Especially the triple error ϵ_T that is evaluated on the ground-truth grid is very close to the pairwise alignment error in most cases. Furthermore, in Figure 6.4a, note that the frame-wise triple error $\epsilon_{T,F}$ is always higher than the triple error on the ground-truth grid ϵ_T caused by the larger errors in-between the ground-truth grid points.

Generally, the error measures are lower for the piano pieces than for the operas. This is due to the design of our alignment method, which has been optimized for music with strong onsets that are present in piano music, but not necessarily in operas. For F08 and F09, the relative difference between the two triple errors ϵ_T and $\epsilon_{T,F}$ is higher than for the other pieces. This can partly be explained by the density of ground-truth annotations. The measure ground-truth grid of the opera pieces is much coarser compared to the beat annotations of the piano pieces, see for example F08 (0.37 $\#(\mathrm{GT})/s$) compared to M24-2 (2.9 $\#(\mathrm{GT})/s$) in Table 6.1. Furthermore, F08 exhibits the highest error. The piece is also the one with slowest tempo (having also less note onsets), and is partly performed as a recitative. This gives a high degree of freedom to the singer in shaping the local tempo of the performance, making it particularly difficult to achieve

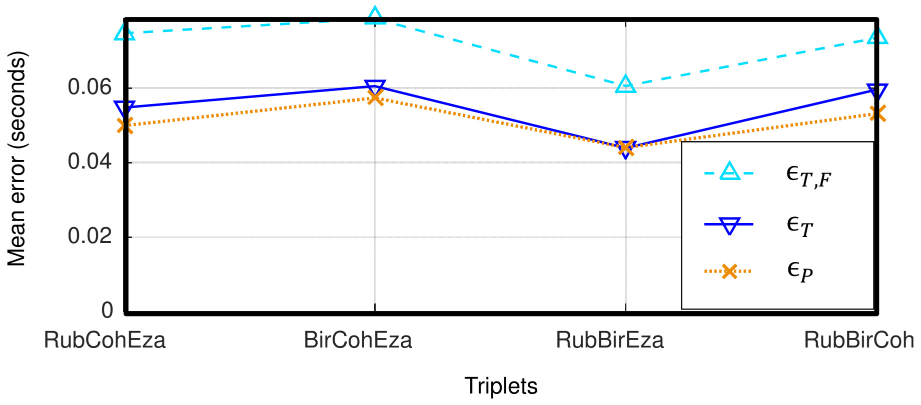


Figure 6.5. Mean of frame-wise triple error $\epsilon_{T,F}$, the triple error ϵ_T computed on the ground-truth grid and mean pairwise alignment error ϵ_P on M17-4. The labels on the x-axis denote which recordings are used in a triplet, using the recordings by Biret (Bir), Cohen (Coh), Ezaki (Eza), and Rubinstein (Rub).

accurate alignments.

6.3.4 Identification of Problematic Versions

To identify if a specific recording introduces higher errors in the pairwise alignments, we included another performance by Biret (1990) into the set of recording for M17-4. This way, four triplets can be formed, each excluding one of the performances. Figure 6.5 shows the triple errors and the alignment errors for each of the four triplets. The experiment reveals that the triplet excluding the recording by Cohen (RubBirEza in Figure 6.5) leads to a lower difference between the triple error ϵ_T and the pairwise alignment error ϵ_P. It is also worth noting, that both triple errors and the pairwise alignment errors have the minimum for the same triplet. By using this strategy with a larger set of recordings, one could use the frame-wise triple error to identify problematic versions that generally lead to higher alignment errors in their pairwise alignments.

6.3.5 Comparing different alignment methods

Let $\mathcal{A}_1$ be our previously used alignment procedure. Furthermore, let $\mathcal{A}_2$ denote an alignment procedure only using chroma features without onset information. Figure 6.4a+b shows the mean error measures for $\mathcal{A}_1$ and $\mathcal{A}_2$ respectively. Overall, by comparing Figure 6.4a and Figure 6.4b, we can see that using onset information in the features generally improves the alignments. This is reflected by most of the triple and the pairwise alignment errors. However, for the piece F08, although the pairwise alignment error ϵ_P is smaller for $\mathcal{A}_1$, the frame-wise triple error $\epsilon_{T,F}$ for $\mathcal{A}_1$ is larger than for $\mathcal{A}_2$. For the piano pieces, all error measures are considerably smaller for $\mathcal{A}_1$ compared to $\mathcal{A}_2$, which shows that onsets are an important aspect for aligning piano music. Figure 6.6 shows the *misalignment rates* for M17-4 for the two alignment methods. It is defined

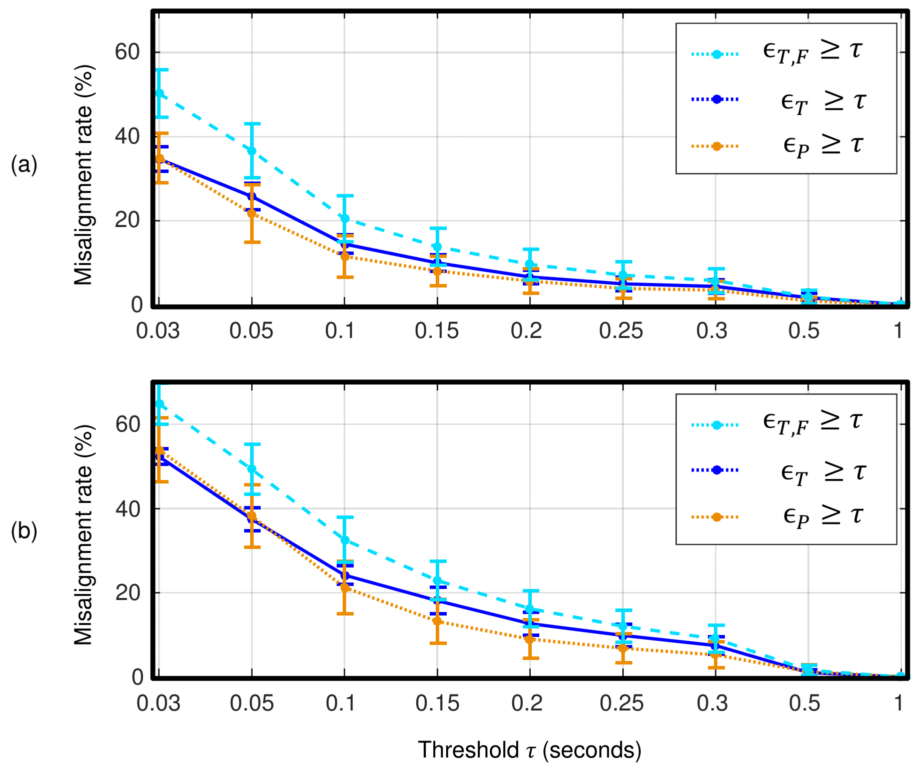


Figure 6.6. Misalignment rates on frame-wise triple error $\epsilon_{T,F}$, triple error ϵ_T, and the pairwise alignment error ϵ_P on M17-4. Error bars indicate the standard deviation. **(a)** Alignment method $\mathcal{A}_1$ using chroma with onset information. **(b)** Alignment method $\mathcal{A}_2$ using chroma without onset information.

as the percentage of time points in an alignment that have an error above a given threshold τ and is also commonly used to evaluate music alignment methods [5, 25]. The misalignment rate not only shows that $\mathcal{A}_1$ (Figure 6.6a) is more accurate than $\mathcal{A}_2$ (Figure 6.6b), but also reveals that the improvements are all below 0.3 seconds. Note that this conclusion can also be drawn by only considering the frame-wise triple error.

6.4 Conclusions

In this chapter, we introduced the triple error, a tool analyzing music alignment methods without the need for ground-truth annotations. It can be used when at least a triplet of recordings of the same piece of music is available. A large triple error indicates the existence of some misalignment, but not all misalignments necessarily lead to a large triple error. However, our experiments show similar tendencies for the triple error as for the pairwise alignment error based on ground-truth annotations. We demonstrated that the triple error computed on a much finer frame grid can sometimes even indicate problematic positions in alignments that cannot be captured by ground-truth annotations. Also, we have shown that the triple error can be used to identify versions that are problematic in the pairwise alignments. Finally, despite its theoretical limitations, we indicated that the triple error may be a useful tool for comparing and understanding different alignment methods.

6.5 Further Notes

The idea of using multiple versions of the same piece of music for music analysis or processing is also referred to as cross-version strategy. In recent years, the cross-version idea has been used in many different tasks, such as music alignments [128, 5, 4, 101], chord recognition [37, 62], novelty detection [89], singing voice detection [22], or in audio source separation [118, 48]. All these approaches have in common that they rely on an alignment step. Exploiting information from multiple versions can be done on different levels. In the case of music alignments, for example, the cross-version idea can be applied on the feature level [128] (early fusion), or on the alignment results (late fusion). In the following, we summarize the basic ideas for some of the tasks in which a cross-version strategy has been applied, and then illustrate a cross-version singing voice detection system in Section 6.5.1.

In [128], multiple performances of the same piece of music are aligned jointly. Inspired by progressive alignment methods [31], the basic idea in [128] is to align the different versions of a piece to an iteratively constructed template feature matrix. The authors in [5] stabilize an online alignment using multiple performances of the same piece of music. Their basic idea is to track the score position of a live performance by running parallel instances of an online alignment system with different audio versions of the same piece of music. These audio versions have been linked to the same musical score beforehand by using an offline alignment system. The resulting redundant output of the individual online alignment systems is then combined by median filtering. In this way, the authors both make the live tracking system more robust, as they do not have to rely on a single alignment, and improve the alignment accuracy of the online system.

The authors in [37] first identify reliable parts in music alignments by a late fusion of different alignment procedures. Subsequently, a cross-version analysis of different chord-labeling procedures is performed on the segments that were identified as reliable. In [62], the authors stabilize the results of a chord-labeling procedure by combining chord-labeling results obtained from different recordings.

In [89], the computation of tempo-based novelty curves from music recordings is approached by first computing individual novelty curves on different recordings, which are then combined into a fusion curve by averaging the individual curves on a common time axis.

In the context of audio source separation, multiple deformed references of the same piece of music have been used to support the separation process [118]. For example, multitrack recordings from cover versions of a piece have been used to initialize model parameters for the separation of individual instruments in the original stereo recordings of the piece [48].

6.5.1 Cross-Version Singing Voice Detection

In opera music, singing voice is one of the most important musical aspects. The task of *singing voice detection* aims at automatically segmenting a given music recording into vocal (where one or more singers are active) and non-vocal (only accompaniment or silence) sections [65, 73, 105]. Due to the huge variety of singing voice characteristics, as well as the simultaneous presence of other pitched musical instruments in the accompaniment, singing voice detection is generally considered a much harder problem than speech/music discrimination. For example, the singing voice may reveal complex temporal-spectral patterns, e.g., as a result of vibrato (frequency and amplitude modulations). Also, singing often exhibits a high dynamic range, from soft passages in a lullaby sung in pianissimo to dramatic passages sung by a heroic tenor. Furthermore, many other instruments with similar acoustic characteristics may interfere with the singing voice. This happens especially when the melody lines played by orchestral instruments are similar to the ones of the singing voice.

Most approaches for singing voice detection build upon extracting a set of suitable audio features and subsequently applying machine learning in the classification stage, see [65, 73, 105]. These approaches need extensive training material that reflects the acoustic variance of the classes to be learned. In particular, we used a state-of-the-art singing voice detection system that was originally introduced in [65]. This approach employs a classification scheme known as random forests to derive a time-dependent decision function (see Figure 6.7). The idea is that the decision function should assume large values close to one for time points with singing voice (vocal class) and small values close to zero otherwise (non-vocal class). In order to binarize the decision function, it is compared to a suitable threshold—only time instances where the decision function exceeds the threshold are classified as vocal.

In particular for popular music, annotated datasets for training and evaluation of singing voice detection algorithms are publicly available [105]. In the context of the Freischütz project, we looked at the singing voice detection problem for the case of classical opera recordings. Unsurprisingly, our first experiments showed that a straightforward application of previous approaches (trained on popular music) typically lead to poor classification for classical music. In [22, 24], we proposed various modifications of the system described in [65]. As one contribution, we described a bootstrapping procedure that helps to improve the results in the case that the training data does not match the unknown audio material to be classified. The main idea is to start with a classifier based on some initial training data set to compute a first decision function. Then, the audio frames that correspond to the largest values of this function are used to re-train the

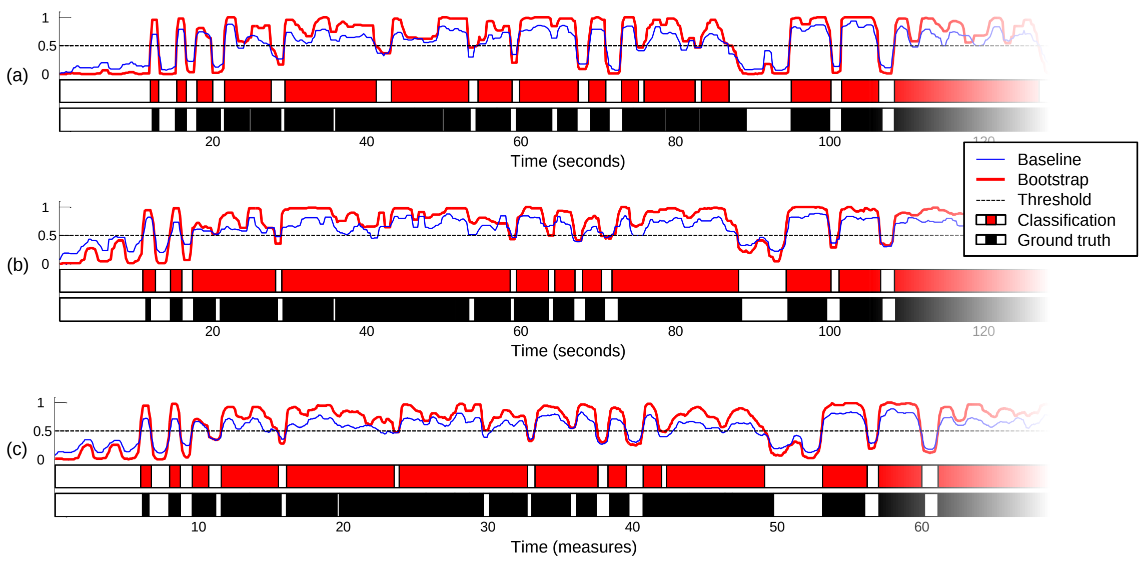


Figure 6.7. Illustration of the cross-version singing voice detection from [22] provided by Christian Dittmar. **(a)** Recording of the performance conducted by Karl-Heinz Bloemecke (2013). **(b)** Recording of the performance conducted by Carlos Kleiber (1973). **(c)** Cross-version results based on three performances (including Bloemecke and Kleiber) after temporal alignment to a common, measure-based time axis and subsequent averaging across the individual decision functions.

classifier. Our experiments showed that this adaptive classifier yields significant improvements for the singing voice detection task. As a final contribution, we showed that a cross-version approach, where one exploits the availability of different recordings of the same piece of music, can help to stabilize the detection results even further. Similarly as in [89], the decision functions computed on different versions of the same piece are averaged on a common time axis.

Figure 6.7 shows examples of decision curves from the singing voice detection system introduced in [22] along with the computed singing voice labels and the corresponding ground-truth labeling. Figure 6.7a and Figure 6.7b show the results for Bloemeke (2013) and Kleiber (1973) respectively. The curves and annotations are based on an excerpt corresponding to the first 80 measures of the duet No. 6 (Agathe and Ännchen): “Schelm! halt fest” from the opera “Der Freischütz” by Carl Maria von Weber. For each case, the decision functions of the baseline (blue thin curve) and bootstrap (red bold curve) classifiers are shown. The colored timelines below the decision curves show the automatically detected singing voice activity (red segments, derived from bootstrap decision) vs. the ground truth (black segments). Figure 6.7c shows the cross-version results obtained by averaging the decision functions from (a) and (b) on a common time axis.

Chapter 7

Interference Reduction in Multitrack Music Recordings

This chapter is based on our work introduced in [96] and [97] (in preparation).

When recording orchestra music, the microphones for capturing the different sources such as instrument groups or singing voice are usually not shielded from each other. In practice, each microphone not only records sound from its dedicated source, but also from all others in the room. This results in recordings that do not feature isolated signals, but rather mixtures of a predominant source with all others being audible through what is referred to as interference. This is also the case for the recordings in the "Freischütz Digital Multitrack Dataset" (Section 2.3, Appendix A, and Figure 2.7). Motivated by this scenario, we approach the task of reducing interference in multitrack music recordings in order to make these recordings more accessible, e.g. for pedagogical reasons or further processing tasks.

The remainder of the chapter is organized as follows: First, we give some background on interference reduction (Section 7.1). We then describe the model and the corresponding algorithm (Section 7.2). Subsequently, we explain the data generation and introduce the evaluation metrics (Section 7.3), and present the results of our experiments (Section 7.4). Finally, we conclude the chapter (Section 7.5) and describe applications of the proposed algorithm (Section 7.6).

7.1 Background

When recording musicians during a live performance, sound engineers typically set up a microphone for each *source*, which may be an instrument or an instrument group such a violin section or the singers, see Figure 7.1 for an illustration. In a typical professional recording setup for

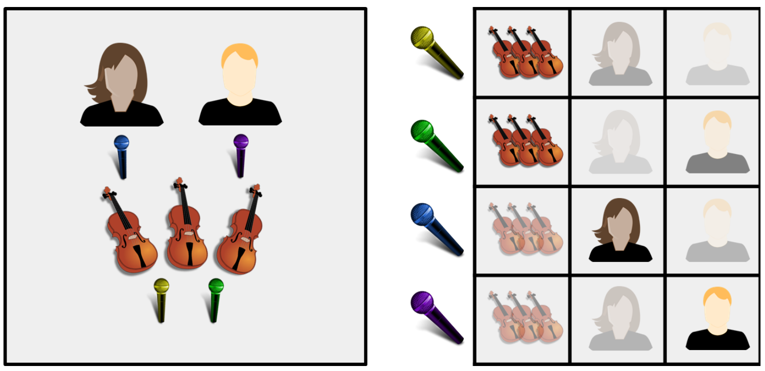


Figure 7.1. (Left) Illustration of setup with three sources (violin section, female singer, male singer). Each source is associated with at least one microphone. (Right) Diagram of the interference in each of the four microphones. Solid images represent the primary source in a microphone, faded images represent an interfering sources. For example, the first microphone (yellow) primarily picks up the violin section but also picks up interference from the female and male singers.

pop/rock music, the recording room is equipped with sound absorbing materials and acoustic shields to isolate all the sources as much as possible. However, complete acoustic isolation between the sources is often not possible. This is particularly the case for classical music recordings, where the interaction between musicians is essential. In practice and as depicted in Figure 7.1 and Figure 7.2, each microphone not only records sound from its dedicated source, but also from all other sound sources in the room. This results in recordings that do not feature isolated signals, but rather *mixtures* of a predominant source with all others being audible through what is referred to as *interference*, *bleeding*, *crosstalk*, or *leakage*. These interferences may be undesirable for several reasons. First, interferences greatly constrain the mixing possibilities for a sound engineer. Second, they severely reduce the accuracy of pitch trackers and other tools necessary for further audio processing and analysis. Furthermore, they prevent the removal or isolation of a source from the recording, which may be desirable, e.g. for pedagogical reasons. A natural question thus arises: is it possible to remove these interferences to get clean, isolated source signals?

Several studies have considered this interference reduction problem. In each of these studies, it is assumed that for each source there is at least one primary microphone and that the number of sources and their corresponding microphones are known. While some approaches are based on echo cancellation and adaptive filtering involving estimation of propagation filters between sources and microphones in the time domain [16, 124], others are based on Time-Frequency (TF) approaches, where the recordings are processed using adequate representations such as the short-time Fourier transform (STFT). For the latter cases, interference reduction is typically performed through Wiener filtering [60, 61], which has been shown to produce good results in terms of sound quality at a small computational cost. Interference reduction using Wiener filtering requires an estimate of the clean spectrogram for each source. While [60] simply assumes that the spectrogram of each recording is already a good estimate for its dedicated source signal, [61, 59] introduce further temporal constraints on the sources so as to better extract them from the mixture recordings. In [14], Non-Negative Matrix Factorization (NMF, see [15]) is used as a global parametric model for the source spectrograms. Multi-channel interference reduction is also important in many other fields, such as speech enhancement, hearing aid sound processing, or for telecommunications [46, 66, 119].

Interference Reduction (IR) is closely related to the problem of audio source separation, in which the objective is to separate a sound mixture into its constituent components [126]. Audio source separation in general is a very difficult problem where performance is highly dependent on the signals considered. However, recent studies demonstrate that separation methods can be very effective if prior information about the signals is available (see e.g. [38, 69] and references therein).

In this chapter, we introduce an algorithm for interference reduction in multitrack recordings called MIRA (Multitrack Interference Reduction Algorithm). For each source, MIRA iteratively estimates the fractional power spectral density, its spatial image, and the amount of its interference into each microphone. Thereby, similar to [61], we exploit the fact that each source is known to be predominant in its dedicated microphone. The power spectral densities and spatial images are estimated through Wiener filtering. The estimation of the interference of each source into each microphone is modeled in an interference matrix which is learned through non-negative matrix factorization (NMF) using the β-divergence [41]. MIRA is a modified version of our recently introduced Kernel Additive Modeling for Interference Reduction (KAMIR) [96].

As one main contributions of this chapter, we extend the model by using fractional spectrograms which use an exponent α on the magnitudes of the spectrograms as introduced in [68] (e.g. $\alpha = 1$ for magnitude spectrograms, or $\alpha = 2$ for power spectrograms). Furthermore, we simplify the model used in KAMIR by dropping the kernel filtering step and the frequency dependence of the interference matrix. As another contribution, we describe extensive experiments using the standard evaluation metrics for blind source separation [127] to evaluate the different parameter settings for MIRA on controlled multitrack mixtures. In particular, we test the interference reduction capabilities of MIRA and investigate the quality of the estimated interference matrix with respect to the parameters α and β. Finally, we compare the performance of MIRA in different mixing conditions (linear and convolutive mixtures) to the performance of independent component analysis (ICA) and other Wiener filter-based approaches for interference reduction [61].

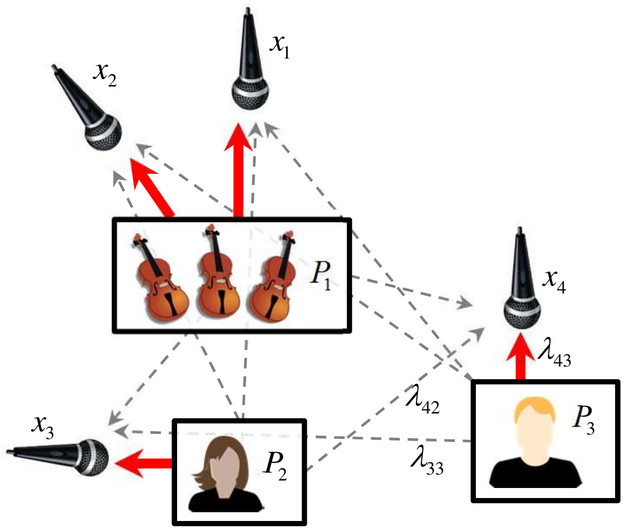


Figure 7.2. Illustration of setup with three sources (violin section, female singer, male singer). Each source is associated with at least one of the microphone. A solid line indicates a signal from a source to its corresponding microphone, and a dashed line indicates an interference signal from other sources into a microphone .

7.2 Model

7.2.1 Notations and model

Let I be the number of microphones and J be the number of sources. For $i \in [1:I]$, we refer to x_i as the signal received by microphone i. The sound produced by a particular source $j \in [1:J]$ may be picked up in any microphone i. We define the *image* y_{ij} as the contribution of the sound from source $j \in [1:J]$ in microphone $i \in [1:I]$. The microphone signal x_i is then simply taken as the sum of the corresponding source images:

$$x_i = \sum_{j=1}^{J} y_{ij}\,. \tag{7.1}$$

In the proposed procedure, we first compute the short-time Fourier transform (STFT) X_i of x_i. This yields I matrices of dimension $N_\omega \times N_t$, where N_ω and N_t are the number of frequency bands and the number of time frames, respectively. For each bin $(\omega, t) \in [1:N_\omega] \times [1:N_t]$ in the Time-Frequency (TF) domain, we then have:

$$X_i(\omega, t) = \sum_{j=1}^{J} Y_{ij}(\omega, t)\,, \tag{7.2}$$

where Y_{ij} denotes the $N_\omega \times N_t$ STFT of y_{ij}.

The goal of our procedure is to compute an estimate $\hat{Y}_{ij}$ of the image Y_{ij} for each source $j \in [1:J]$ in each microphone $i \in [1:I]$. A central assumption in our approach is that every source $j \in [1:J]$ is dominant in at least one microphone, and that this microphone is known. Let us assume that source $j \in [1:J]$ is dominant in microphone $i \in [1:I]$. For this reason, we consider X_i to be a good initial estimate for $\hat{Y}_{ij}$:

$$\hat{Y}_{ij} \approx X_i \,. \tag{7.3}$$

In many source separation tasks, either magnitude or power spectrograms are used to build ratio masks for extracting individual sources from a mixture signal. A common approach for generating ratio masks is Wiener filtering [68]. Recently, fractional spectrograms using an exponent $\alpha \in (0\ 2]$ on the magnitudes of the STFT have been introduced and theoretically justified for Wiener filtering [68]. We call $|Y_{ij}|^\alpha$ the fractional spectrogram of source $j \in [1:J]$ in microphone $i \in [1:I]$ [68]. In our experiments, we use $\alpha = 1$ and $\alpha = 2$ which correspond to the magnitude spectrogram and the power spectrogram, respectively. In this study, we assume that the fractional spectrograms $|Y_{ij}|^\alpha$ for a given $j \in [1:J]$ are all related to a latent fractional Power Spectral Density (αPSD) $P_j \geq 0$ up to a scaling factor λ_{ij} (see Figure 7.2):

$$|Y_{ij}\,(\omega, t)|^\alpha \approx \lambda_{ij} P_j\,(\omega, t)\ . \tag{7.4}$$

The quantities P_j are not known and their estimation is an essential step in our algorithm. In essence, for each source $j \in [1:J]$, the quantities $|Y_{ij}|^\alpha$ are equal to P_j up to a scaling factor. This model may be understood as only exploiting scale relationships, discarding any potential phase dependencies for a single source across different microphones. The scalar λ_{ij} specifies the amount of interference of source j into microphone i. Let

$$\Lambda = (\lambda_{ij})_{i\in[1:I],j\in[1:J]} \tag{7.5}$$

be the *interference matrix* containing the interference values for each source into all microphones. We describe how to initialize and learn Λ in Section 7.2.3.

If Λ is known, we can compute P_j from (7.3):

$$P_j(\omega,t) \approx \frac{1}{\lambda_{ij}} \left|\hat{Y}_{ij}(\omega,t)\right|^\alpha \tag{7.6}$$

Note that the above model breaks down when long reverberation times are present or when the delays between sources and microphones are longer than the window size used in the STFT. In this case, replacing (7.4) with a nonnegative convolutive model as in [52] could compensate for long reverberation times. However, good performance was observed with (7.4) even for real orchestral recordings [96]. For this reason, we stick to this simpler model for this study.

7.2.2 Separation method

If both the αPSDs P_j and the interference matrix Λ are known, the estimate $\hat{Y}_{ij}(\omega, t)$ of any $Y_{ij}(\omega, t)$ is given by α-Wiener filtering [68]:

$$\hat{Y}_{ij}(\omega, t) \quad = \quad W_{ij}(\omega, t) X_i(\omega, t) \tag{7.7}$$

with

$$W_{ij}(\omega, t) \quad := \quad \frac{\lambda_{ij} P_j(\omega, t)}{\sum_{j'=1}^{J} \lambda_{ij'} P_{j'}(\omega, t)}.$$

$W_{ij}(\omega, t)$ is called the *Wiener gain.* For $\alpha = 2$, (7.7) corresponds to classical Wiener filtering.

For a given source $j \in [1 : J]$, we are usually not interested in computing the estimates $\hat{Y}_{ij}$ for all microphone recordings $i \in [1 : I]$. Rather, we want to obtain $\hat{Y}_{ij}$ only for those microphones that were initially positioned to capture this source. By selecting the “most relevant” images for a specific source, we also significantly reduce computation time. We define the *microphone selection function* $\varphi(j) \subseteq [1 : I]$ that indicates which images of source j we want to recover. This is to account for possibly complex scenarios where several microphones are assigned to a single source signal, as in the case of a concert piano or the violin section of an orchestra (Figure 7.2). In the following, the microphone selection function φ is assumed to be known and given by the user (typically a sound engineer) who knows the recording setup and can easily provide this information. Furthermore, we assume that $\varphi(j) \neq \emptyset$ for all $j \in [1 : J]$, meaning that each source is predominant in at least one microphone.

Thus, the objective of interference reduction is to compute all $\{\hat{Y}_{ij}\}_{i \in \varphi(j)}$ for each source $j \in [1 : J]$. The resulting waveforms are recovered through an inverse STFT. Note that the phase for the image estimate of source $j \in [1 : J]$ in microphone $i \in [1 : I]$ is taken from the mixture STFT X_i.

7.2.3 Parameter estimation algorithm

In this section, we describe the MIRA algorithm for Interference Reduction based on the algorithm introduced in [96].[1] As input, MIRA takes the STFTs X_i of the recorded signals and the

[1] The main differences with [96] are the removal of frequency dependence in the interference matrix Λ and the omission of its normalization. Furthermore, we omit an optional kernel filtering step.

Algorithm MIRA

1. **Input**:
   - $X_i(\omega, t)$ for each microphone x_i
   - Microphone selection function $\varphi(j)$ for each source $j \in [1 : J]$
   - Initial interference ρ
   - Spectrogram exponent α
   - β for β-divergence.
   - Number of iterations N_{iter}.
2. **Initialization**
   - For each $i \in [1 : I]$, for each $j \in [1 : J]$: initialize λ_{ij} as in (7.8)
   - For each $j \in [1 : J]$, for each $i \in \varphi(j), \hat{Y}_{ij} \leftarrow X_i$
   - For each $j \in [1 : J]$, initialize P_j as in (7.9)
   - Set current iteration $n \leftarrow 0$.
3. **Separation step**
   For each $j \in [1 : J]$, for each $i \in \varphi(j)$, update $\hat{Y}_{ij}$ as in (7.7)
4. **If** $n < N_{\text{iter}}$, set $n \leftarrow n + 1$ and proceed with next step, otherwise go to step 7.
5. **Parameter fitting step**
   (a) For each $j \in [1 : J]$: update P_j as in (7.9)
   (b) Update Λ using (7.12).
6. For another iteration, return to step (3)
7. **Output**:
   $\hat{Y}_{ij}(\omega, t)$ for each $j \in [1 : J]$, for each $i \in \varphi(j)$.

microphone selection function φ, as described in the preceding section. It returns estimates for the STFTs of the desired clean signals $\{\hat{y}_{ij}\}_{i\in\varphi(j)}$ for each source $j \in [1 : J]$. To do this, it only needs to estimate the parameters for the α-Wiener filter (7.7): the αPSDs P_j of the sources and the interference matrix Λ.

In a nutshell, MIRA alternates between two distinct procedures in an iterative fashion. In a first *separation* step, the current parameters Λ and P_j for each $j \in [1 : J]$ are assumed known and fixed. Then, separation of the desired clean signals $\{\hat{Y}_{ij}\}_{i\in\varphi(j)}$ is performed for all sources $j \in [1 : J]$ through α-Wiener filtering (7.7). In a second *parameter fitting* stage, those separated signals are kept fixed and are assumed to be good estimates. The parameters P_j and Λ are then re-estimated. Finally, the whole procedure is repeated until a stopping criterion is met, which is usually the imposed number of iterations N_{iter}. The MIRA algorithm is summarized in the algorithm box. We now discuss initialization and re-estimation of the parameters P_j and Λ.

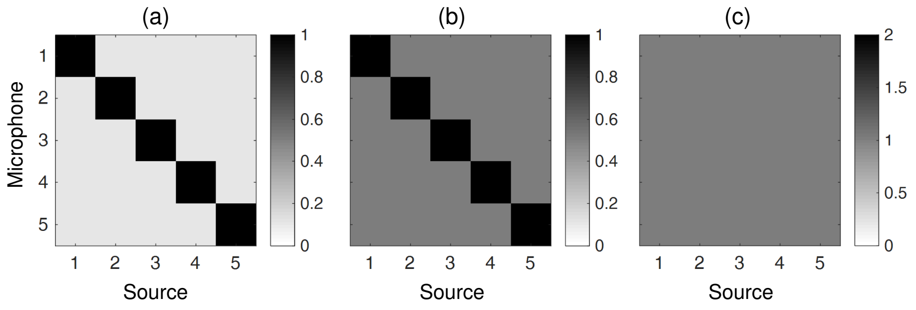


Figure 7.3. Example initializations of the interference matrix Λ with $I = 5$, $J = 5$ and **(a)** $\rho = 0.1$, **(b)** $\rho = 0.5$, **(c)** $\rho = 1$.

7.2.3.1 Initialization (Algorithm Step 2)

The elements of the interference matrices are initialized with:

$$\forall i \in [1 : I], j \in [1 : J],\ \lambda_{ij} = \begin{cases} 1 & \text{if } i \in \varphi(j) \\ \rho & \text{otherwise,} \end{cases} \tag{7.8}$$

where the parameter $\rho \in [0,\ 1]$ is called the *initial interference* (see Figure 7.3 for examples of interference matrices using different ρ values). The rationale for this initialization is the following: a source $j \in [1 : J]$ which is associated to a microphone $i \in [1 : I]$ is given the maximum interference value 1 as it should ideally be fully captured in this microphone. A source j which is not associated to microphone i is given the value ρ as it should only be marginally captured by this microphone. The estimate of the image $\hat{Y}_{ij}$ for each $j \in [1 : J]$ is initialized for $i \in \varphi(j)$ with the observed STFT X_i of microphone i.

Note that [60] can be understood as having $\varphi(j) = \{j\}$, setting $\lambda_{ij} = 1$ and directly recover the source estimates with (7.7). More specifically, when N_{iter} is set to zero, MIRA performs only one separation step using the initial values of the model parameters, and therefore returns the estimates $\hat{Y}_{ij}$ without performing a parameter fitting step.

7.2.3.2 Power spectral density P_j updates (Algorithm Step 5a)

Given the interference matrix Λ and image estimates $\hat{Y}_{ij}$, for a particular i we can approximate P_j with (7.6). Because we may have multiple microphones $i \in \varphi(j)$ for which the previous expression is a good estimate, we take the average of this approximation across all microphones in which source $j \in [1 : J]$ is known to be predominant, yielding:

$$P_j(\omega, t) \leftarrow \frac{1}{|\varphi(j)|} \sum_{i \in \varphi(j)} \frac{1}{\lambda_{ij}} \left| \hat{Y}_{ij}(\omega, t) \right|^{\alpha} \tag{7.9}$$

where $|\varphi(j)|$ denotes the number of microphones indicated by the selection function φ.

7.2.3.3 Interference matrix Λ updates (Algorithm Step 5b)

In this section, we describe a method to estimate the elements of the interference matrix Λ using NMF [15]. Using (7.4), we get:

$$|X_i(\omega,t)|^\alpha \approx \sum_{j=1}^{J} \lambda_{ij} P_j(\omega,t), \tag{7.10}$$

which is the foundation for our NMF formulation. The proposed approach to learn the interference matrix Λ is to enforce (7.10), by minimizing the discrepancies between the left and right-hand sides of (7.10):

$$\Lambda \leftarrow \underset{\Lambda}{\operatorname{argmin}} \sum_{t,\omega,i} d_\beta \left(|X_i(\omega,t)|^\alpha, \sum_{j=1}^{J} \lambda_{ij} P_j(\omega,t) \right), \tag{7.11}$$

where d_β stands for any appropriate cost-function such as the popular family of β-divergences, which notably includes Kullback-Leibler (KL) for $\beta = 1$ and Itakura-Saito (IS) for $\beta = 0$. Using IS with power spectra ($\alpha = 2$) and KL with magnitude spectra ($\alpha = 1$) are common choices [40].

However, depending on the task, experiments have shown that other choices of β are beneficial [42]. The main purpose of using NMF here is to update Λ in a multiplicative fashion, so as to enforce nonnegativity. In [41], NMF update rules for the β-divergence are derived. Applying these on (7.10), the corresponding update rule for each matrix element λ_{ij} is given by:

$$\lambda_{ij} \leftarrow \lambda_{ij} \cdot \frac{\sum_{\omega,t} \hat{V}_i(\omega,t)^{\beta-2} V_i(\omega,t) P_j(\omega,t)}{\sum_{\omega,t} \hat{V}_i(\omega,t)^{\beta-1} P_j(\omega,t)}, \tag{7.12}$$

where $V_i(\omega,t) := |X_i(\omega,t)|^\alpha$, and $\hat{V}_i(\omega,t) := \sum_{j=1}^{J} \lambda_{ij} P_j(\omega,t)$. Note that in most NMF applications [40], the matrix decomposition is used to parameterize the αPSDs of the signals. Here, the P_j are left constant and unconstrained, and NMF is only used to learn the interference matrix. An example of a learned interference matrix is shown in Figure 7.4b. In our previous work [96], we modeled Λ to be frequency-dependent. In this case, we would have one interference matrix per frequency ω. To adapt Equation 7.12 for frequency-dependence, we only need to drop the summations over ω, see [96] for details.

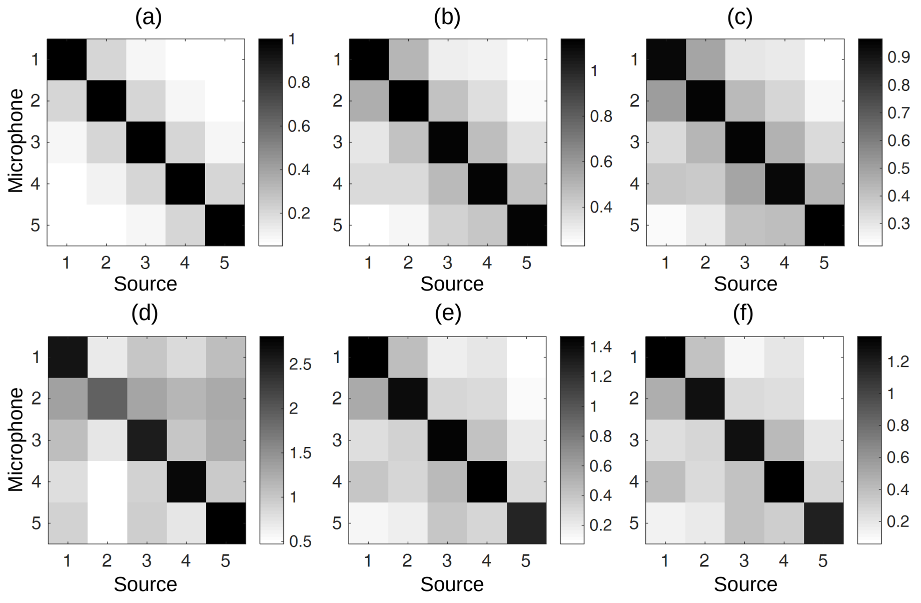


Figure 7.4. Interference Matrices for `MusicDelta_Beethoven`. Note that in the cases using $\alpha = 2$, we have taken the square root of the interference values for making the result comparable to (a). **(a)** Ground-truth mixing matrix. **(b-f)** Learned interference matrices (initialized with $\rho = 1$) after 5 iterations with **(b)** $\alpha = 1, \beta = 1$, **(c)** mixed model, see Section 7.4.4, **(d)** $\alpha = 2, \beta = 0$, **(e)** $\alpha = 2, \beta = 0.5$, **(f)** $\alpha = 2, \beta = 1$.

7.3 Data and Evaluation Metrics

In order to objectively and systematically evaluate the output of the algorithm, we run MIRA on controlled mixtures where the ideal outputs are known. Our controlled mixtures aims to simulate a realistic live recording setup to approximate the real-world use case as much as possible. The creation of this data is described in detail below. In the remainder of this section, we briefly introduce the dataset which is used (Section 7.3.1). Then we describe the generation of multitrack recordings containing artificial interference. First, we discuss the creation of linear mixes (Section 7.3.2), which are only found in specific studio recordings. And second, we describe the creation of convolutive mixes (Section 7.3.3), which simulate the acoustic setup with multiple microphones in a room. In both scenarios, we assume an equal number of sources and microphones ($I = J$).

7.3.1 Raw Material

The audio material created for our experiments was generated using multitrack recordings from the MedleyDB dataset [11]. We used 10 of the dataset's multitracks that did not contain bleed, see Table 7.1 for an overview. Each multitrack in the MedleyDB dataset contains raw audio (unprocessed direct microphone input corresponding to a musical source) and stems (pro-

No	MedleyDB ID	Stems
01	`ID-AimeeNorwich_Child`	singer (m), bass, drums, ac. guitar, el. guitar
02	`ID-AlexanderRoss_GoodbyeBolero`	singer (f), el. guitar, harmonica, brass section, viola section
03	`ID-Creepoid_OldTree`	singer (m), singer (m), dist. el. guitar, vocalists, bass
04	`ID-HeladoNegro_MitadDelMundo`	singer (m), synthesizer, synthesizer, bass, drum machine
05	`ID-LizNelson_Rainfall`	singer (f), ac. guitar, singer (f), clean electric guitar, singer(f)
06	`ID-MusicDelta_Beethoven`	oboe, flute, clarinet, trumpet, french horn
07	`ID-MusicDelta_ChineseDrama`	erhu, dizi, guzheng, yangqin, zhongruan
08	`ID-MusicDelta_Gospel`	singer (f), drums, bass, clean electric guitar, clean electric guitar
09	`ID-MusicDelta_Vivaldi`	violin section, viola section, violin section, cello, double bass
10	`ID-StevenClark_Bounty`	singer (m), bass, piano, synthesizer, synthesizer

Table 7.1. Pieces from MedleyDB used in our experiments. The stems column shows the voice/instrument used in the mixing.

cessed signals corresponding to a musical source). One stem in a multitrack may be generated from many raw audio tracks, e.g. each containing a recording of a different part from a drum set. In the case of a singer, for example, each raw track possibly corresponds to different sections of a song. We sum the raw recordings corresponding to a stem to estimate the source's direct sound. In the following, we refer to the sources by $s_1, s_2, ..., s_J$.

Throughout our experiments, we limit the number of sources to $J = 5$. We select the top J stems in each multitrack first by their melodic ranking (see the MedleyDB documentation for details), and then by the percentage of the song that they are active in descending order. Each sound source s_j with $j \in [1 : J]$ is normalized by the maximum over its absolute amplitude values. This normalization was chosen to roughly equalize the volumes of each of the sources. By normalizing in this way the sources will have different overall energies, which happens frequently in a real-world scenario, where instruments may play at different volumes.

7.3.2 Linear Mixes

Given I microphones and J sound sources with $I = J$, let g_{ij} be the gain factor with which source s_j is mixed into microphone $i \in [1 : I]$. We define the *linear image* of source $j \in [1 : J]$ in microphone $i \in [1 : I]$ as:

$$y_{ij} = g_{ij} \cdot s_j \,. \tag{7.13}$$

The linear mixture recorded at microphone i is defined as the sum of the source's images as defined in (7.1). The gain factors define the *mixing matrix* $(g_{ij})_{i\in[1:I],j\in[1:J]}$. In the following, we assume that the mixing matrix is normalized such that the gain factor of a source $j \in [1 : J]$ in its dedicated microphone equals to one, i.e. $g_{ij} = 1$ if $i = j$ and $g_{ij} \in [0, 1]$ otherwise. The ground-truth interference matrix Λ is computed as the ratio of the mean fractional magnitudes of an image y_{ij} of a given source $j \in [1 : J]$ in microphone i and the image $y_{i'j}$ in its dedicated

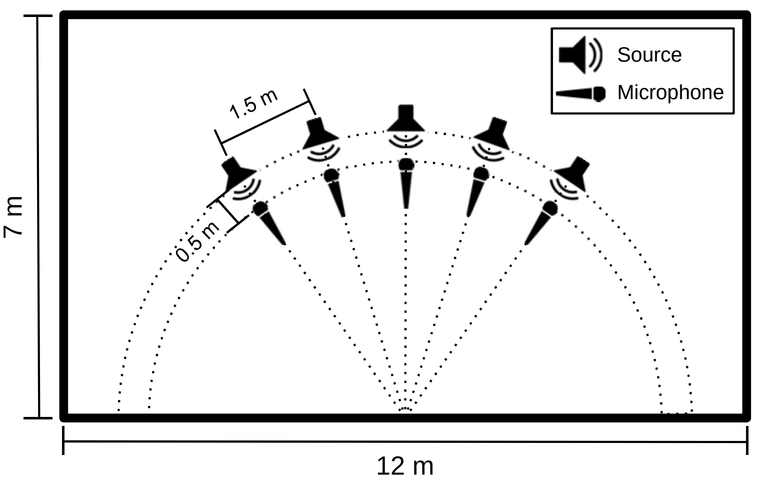


Figure 7.5. Virtual recording setup for $J = 5$ sources.

microphone $i' := j$:

$$\lambda_{ij} = \frac{\sum_n |y_{ij}(n)|^\alpha}{\sum_n |y_{i'j}(n)|^\alpha}\,, \tag{7.14}$$

where $n \in [1 : N]$ is the sample index of the image signals. For $\alpha = 1$, the elements of the interference matrix are magnitude ratios, whereas for $\alpha = 2$, they are energy ratios. In the case of linear mixtures computed from a normalized mixing matrix, (7.14) simplifies to

$$\lambda_{ij} = g_{ij}^\alpha\,. \tag{7.15}$$

This relationship lets us compare interference matrices that were computed with different α values. For example, when comparing an interference matrix computed with $\alpha = 2$ to one computed with $\alpha = 1$, we only need to take the square root of the first interference matrix to make them comparable. Note that this is not true for the convolutive mixes in the next section.

7.3.3 Convolutive Mixes

In this section, we describe how ground-truth data for spatial images is created using a virtual room as depicted in Figure 7.5. Given I microphones and J virtual sound sources $s_1, s_2, ..., s_J$, let h_{ij} denote the impulse response for source j in microphone i. The *spatial image* of source j in microphone i is computed as:

$$y_{ij} = h_{ij} * s_j \tag{7.16}$$

where the operator $*$ denotes convolution. The signal x_i received by the i-th microphone is computed by (7.1).

To create the impulse responses h_{ij}, we use the virtual recording setup illustrated in Figure 7.5. The space is modeled as a rectangular, medium-sized recording studio (12m x 7m x 3m), where the J sources are each placed with a single cardiod microphone pointing directly toward them. We place the sound sources in an arc near the center of the room with each musician seated 1.5 meters apart. The I virtual microphones are placed 0.5 meters in front of each sound source. We compute the impulse responses for each microphone position with each source position. To compute the corresponding spatial images of each source in each microphone, we convolve the impulse responses with the source's direct sound. These spatial images are summed across sources for each microphone as in (7.1), creating a simulation of the sound x_i that would be picked up by the microphones. The impulse responses are calcluated using an implementation of [1], which is available online.[2] Additional parameters required by the implementation were set as follows: $\texttt{nsample} = 4096$, $\texttt{beta} = 0.2$, $\texttt{c} = 340$, and $\texttt{fs} = 44100$.

We use each x_i as input to MIRA, and the spatial images $y_{i'j}$ with $i' \in \varphi(j)$ for the source $j \in [1 : J]$ as the ground-truth target signals in the evaluation. The elements λ_{ij} of the ground-truth interference matrix Λ are computed as in (7.14) using the convolutive spatial images from (7.16).

7.3.4 Evaluation Metrics

For assessing the performance of the algorithm we use the Blind Source Separation Evaluation metrics (BSS-eval) *Signal to Interference ratio (SIR)*, *Signal to Distortion ratio (SDR)*, and the *Signal to Artifacts ratio (SAR)* introduced in [127]. The SIR measures how much other sources are interfering with a signal. In our case, we use the SIR to measure how well the removal of interfering sources was achieved. SAR measures the amount of artifacts in the signal which are introduced by the Wiener filtering. SDR is often used to get an impression of the overall performance, as it both includes interferences and artifacts. All measures are given in dB and higher values indicate better performance.

Furthermore, we use the mixture signals in each microphone as baseline for the SIR. As an upper performance limit for the Wiener filter, we use the ground-truth source images to filter the mixture in their primary microphones. This is done by building a filter from the ratio of the spectrograms from the image of source j and the mixture in its associated microphone $i = \varphi(j)$:

$$\hat{Y}_{ij}(\omega, t) = \frac{|Y_{ij}(\omega, t)|^{\alpha}}{|X_i(\omega, t)|^{\alpha}} X_i(\omega, t) \tag{7.17}$$

In the following experiments we will use $\alpha \in \{1, 2\}$. For $\alpha = 1$, the filter in (7.17) is also called *ideal ratio mask*, and for $\alpha = 2$ it results in a Wiener filter [33].

[2] `www.audiolabs-erlangen.de/fau/professor/habets/software/rir-generator`

7.4 Parameter Setting and Experiments

In this section, we evaluate the output of MIRA with different parameter settings using the BSS-eval metrics described in Section 7.3.4. The evaluations were performed on ten songs with $I = 5$ microphones and $J = 5$ sources generated as described in Section 7.3.3, see Table 7.1 for an overview. Unless otherwise stated, the following default parameters were used: Number of iterations $N_{\text{iter}} = 10$, initial interference $\rho = 1.0$, and learning=`True`.[3] Furthermore, as a standard setting, we used $\beta = 0$ (IS) if $\alpha = 2$ and $\beta = 1$ (KL) if $\alpha = 1$. If not stated otherwise we use the linear mixes as created in Section 7.3.2. To make the linear and convolutive scenarios comparable, the linear mixing matrix was computed as the average of the ground-truth convolutive interference matrices (the values were floored at the first non-zero digit).

In the remainder of this section, we first evaluate the choice of the window size and overlap in the STFT computation (Section 7.4.1). We then discuss the influence of the choice of α (spectrogram model) and the number of iterations on the algorithm (Section 7.4.2). Then, we discuss how the learning influences the interference reduction by testing different initializations of Λ with varying number of iterations (Section 7.4.3). We investigate how the spectrogram model (α) and the choice of the divergence (β) influence the interference reduction and the learning of the interference matrix in Section 7.4.4. Finally, we compare MIRA to baseline algorithms (Section 7.4.5).

7.4.1 Spectrogram Parameters

Using long windows leads to good frequency resolution, but a poor localization in time. This can be compensated to a certain degree by using large overlaps between the windows. We initally observed that larger windows lead to better results, so we performed a systematic parameter sweep over the window length N_ω and overlap parameters of the STFT. The sample rate of our audio recordings is 44.1 kHz. We evaluated the usage of hamming windows [90] with window lengths $N_\omega \in \{512, 1024, 2048, 4096, 8152, 16384, 32768, 65536\}$ given in samples, and for each of these window lengths we evaluated a 50% and 90% overlap between spectrogram frames. Furthermore, we used $\alpha = 2$, $N_{\text{iter}} = 2$, and a constant interference matrix (no learning) with $\lambda_{ij} = 1$ for all $i \in [1 : I]$ and $j \in [1 : J]$. The results are shown in Figure 7.6.

We found that with increasing window lengths up to 8192 samples, the BSS-eval metrics were better on average. For larger window sizes, the metrics dropped again. The overlap percentage mainly influences the SDR and SAR. Generally, larger overlaps between windows leads to a slight increase of SDR and SAR. The increase of the SDR is mainly due to a decrease in artifacts, as

[3]Note that the parameter 'learning' is not part of the algorithm box, but is an optional parameter in our implementation. We use this parameter to test the influence of learning Λ on the interference reduction.

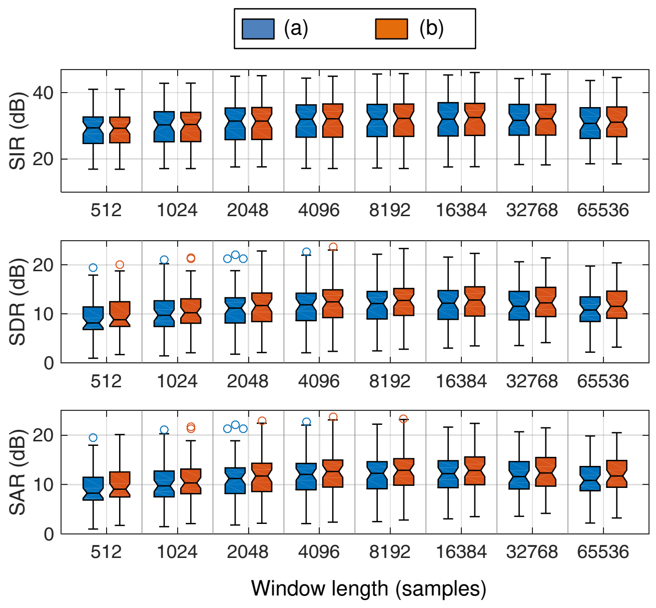


Figure 7.6. Evaluation measures for different values of the window length N_ω and frame overlaps of **(a)** 50% and **(b)** 90%. Circles in the boxplots denote outliers.

both SDR and SAR increase when using larger overlaps whereas the SIR stays constant. This makes sense, as larger overlaps lead to a better time resolution. We therefore set the window length to $N_\omega = 8192$ and the overlap to 90%. The optimal choice of the window size is also dependent on the type of signals considered. For example, when only harmonic signals like string instruments are processed, larger windows will improve the results. However, when also percussive signals as drums are considered, one needs a compromise between time and frequency resolution.

7.4.2 Spectrogram Model α and Iterations

In this experiment, we investigated how the spectrogram model exponent $\alpha \in \{1, 2\}$ and the number of iterations N_{iter} influence the performance of the algorithm. To this end, we keep Λ fixed, skipping Step 5b in the algorithm. First, we used the ground-truth interference matrix, as computed in Section 7.3.2. For $\alpha = 1$, we used the absolute value of the mixing matrix as the interference matrix, whereas for $\alpha = 2$ we used the square of the mixing matrix. We can see the tendency that SIR, and SDR both increase with the number of iterations, see Figure 7.7. The SAR follows an inverse tendency—it is more likely to introduce artifacts by a stronger suppression in the spectrogram. Furthermore, using power spectrograms ($\alpha = 2$), leads to slightly better values, especially in the SIR. Intuitively, because of the square, the power spectrogram already enhances dominant and suppresses weak components in the spectrogram.

In a second experiment, we used constant interference matrices with $\rho \in \{\sqrt{0.1}, \sqrt{0.5}, 1.0\}$ for

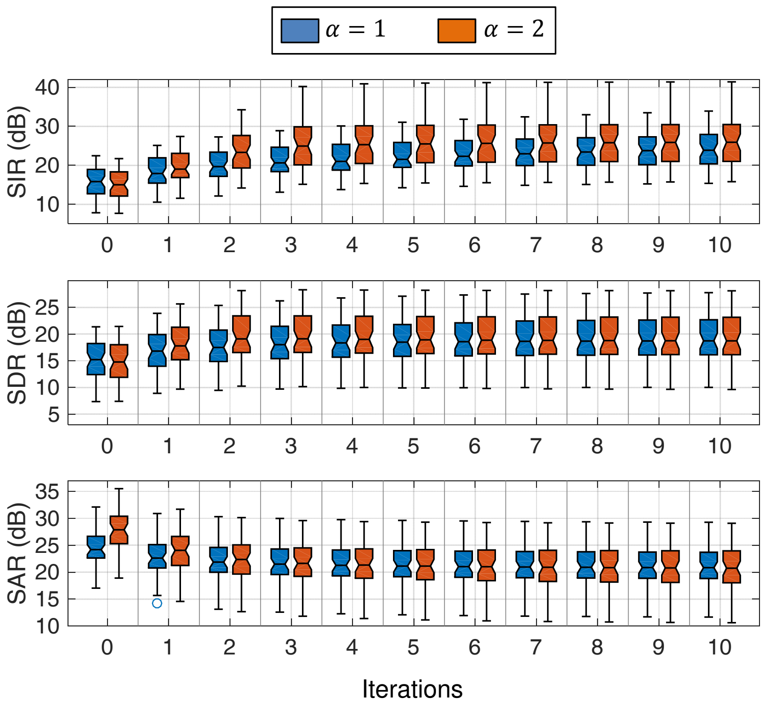


Figure 7.7. Evaluation measures for magnitude spectrograms ($\alpha = 1$) and power spectrograms ($\alpha = 2$) using the ground-truth interference matrices in each iteration (no learning). Note that for $\alpha = 1$, the interference matrix corresponds to the mixing matrix (provided all entries are non-negative), whereas for $\alpha = 2$, the interference matrix corresponds to the square of the mixing matrix.

$\alpha = 1$ and $\rho \in \{0.1, 0.5, 1\}$ for $\alpha = 2$ (see Figure 7.3), such that the values of ρ used with the different α are comparable, see also Section 7.3.2. The BSS-eval metrics for $\alpha = 1$ and $\alpha = 2$ are shown in Figure 7.8a and Figure 7.9a respectively. Using these constant interference matrices leads to a better SIR compared to using the ground-truth interference matrices from the previous experiment (Figure 7.7). Both SDR and SAR follow an inverse tendency. It might look surprising that using a constant interference matrix leads to better SIR values than using the ground-truth interference matrix. However, the values in the ground-truth interference matrices (see Figure 7.4a) are all lower or equal than the values used in the constant interference matrices. The constant interference matrices are therefore overestimating the actual interference which leads to an over-suppression. Thus, although it leads to higher SIR values, it also leads to more distortions and artifacts.

7.4.3 Influence of Learning Λ on Interference Reduction

In this section, we investigate how the learning of Λ influences the interference reduction. We applied MIRA with activated learning using the initial interference $\rho \in \{\sqrt{0.1}, \sqrt{0.5}, 1.0\}$ for initialization with $\alpha = 1$ and $\rho \in \{0.1, 0.5, 1.0\}$ for initialization with $\alpha = 2$, see Figure 7.8b and Figure 7.9b.

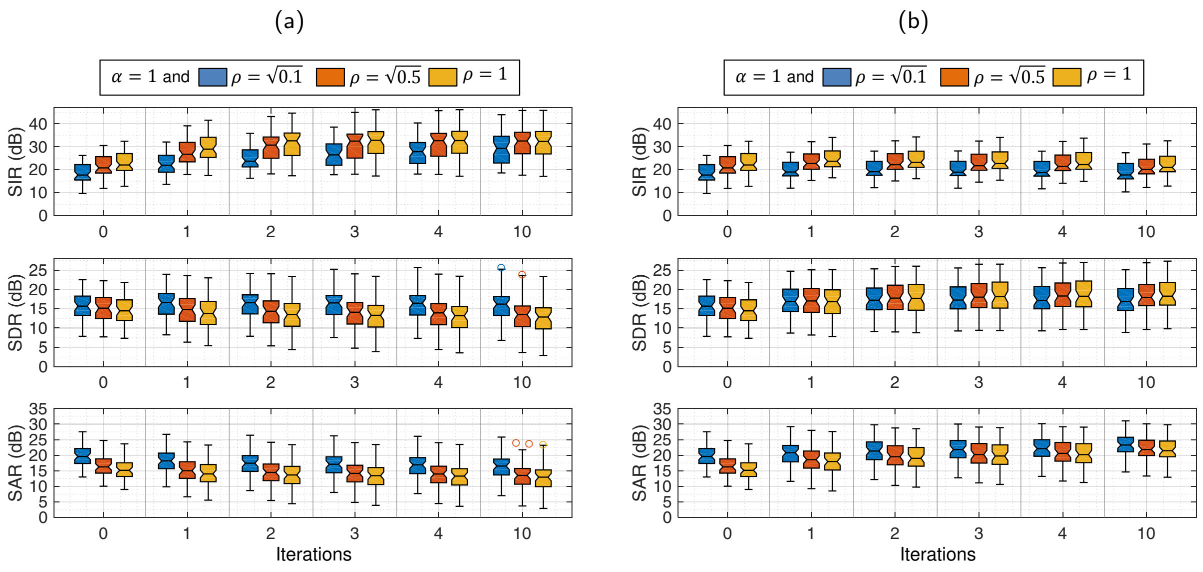


Figure 7.8. Evaluation measures for MIRA with magnitude spectrograms ($\alpha = 1$). **(a)** Using constant interference matrix. **(b)** Updating interference matrix in each iteration using $\beta = 1$ (KL divergence).

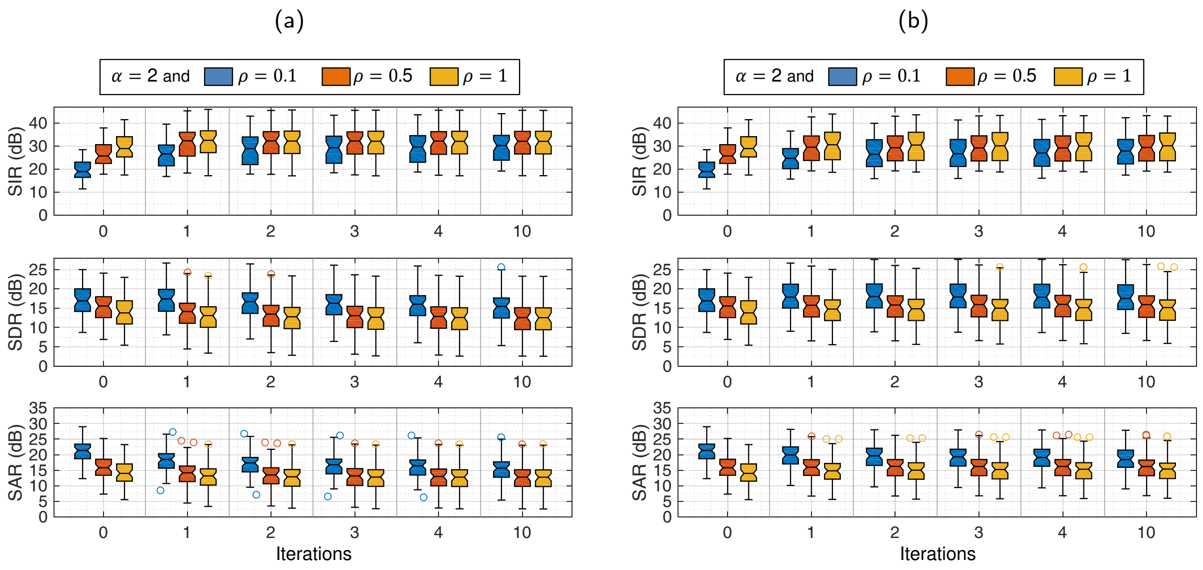


Figure 7.9. Evaluation measures for MIRA with power spectrograms ($\alpha = 2$). **(a)** Using constant interference matrix. **(b)** Updating interference matrix in each iteration using $\beta = 0$ (IS divergence). Circles in the boxplots are outliers.

For $\alpha = 1$, the SIR values increase in the first iterations, and then slightly decrease again after iteration one. Note that in iteration zero, the initial interference matrices are applied. From the next iterations on, the interference matrices obtained from the learning step (step 5b in the algorithm) are used. An example of a learned interference matrix after 5 iterations with $\alpha = 1$ and $\rho = 1$ is shown in Figure 7.4b.

For $\alpha = 2$, both SIR and SDR tend to increase with the number of iterations, see Figure 7.9b. The improvements in SIR over the iterations are very small for $\rho = 1$, we can even observe a decrease of 0.43 dB in the median value when comparing iteration 1 to iteration 10. This makes sense, as we already start with a high amount of suppression with the initial interference matrix. The values for $\rho = 0.5$ evolve similarly. However, the SIR is smaller, whereas SDR and SAR are higher. For $\rho = 0.1$, we observe a clear increase of the SIR values over the iterations, whereas SAR follows an inverse tendency. The SDR increases until iteration 2, and then slightly decreases again. Again, we can observe the tendency that smaller values of ρ lead to better SDR and SAR, whereas larger values lead to better SIR. After a few iterations, see e.g. iteration 4 in Figure 7.9b, the SIR values corresponding to the initilization with $\rho = 1$ and $\rho = 0.5$ are nearly the same. However, the initialization with $\rho = 0.5$ shows slightly better SDR and SAR values. Comparing these results to the case without learning the interference matrix in Figure 7.9a, we can see the tendency that learning the interference matrix with $\alpha = 2$ leads to similar SIR as using a constant matrix (the median of the SIR values differ by about 2 dB), but also improves SDR and SAR values. As can be seen from the previous results, the largest improvements are within the first few iterations. Usually, above 5 iterations, the results do not change much. We therefore fix the number of iterations to $N_{\text{iter}} = 5$ in the following experiments for computational reasons.

An example of a learned interference matrix after 5 iterations with $\alpha = 2$ and $\rho = 1$ is shown in Figure 7.4d. Comparing this matrix to the one learned with $\alpha = 1$ (Figure 7.4b), we can see that the interference matrix learned with $\alpha = 1$ is visually more similar to the ground-truth matrix (Figure 7.4a). Furthermore, using $\alpha = 2$ leads to higher values in the interference matrix. We will further investigate this in Section 7.4.4.

7.4.4 Mixed Model, Influence of β on Separation and Learning Λ

From the previous experiments, we observed that using power spectrograms ($\alpha = 2$) improved the SIR compared to magnitude spectrograms ($\alpha = 1$). However, the learning of the interference matrix visually appeared more stable when using magnitude spectrograms ($\alpha = 1$), see Figure 7.4b and Figure 7.4d, respectively, and Figure 7.4a for the ground-truth mixing matrix. This lead us to the idea to build a *mixed model* using $\alpha = 1$ for the learning step (with $\beta = 1$), and $\alpha = 2$ in the remainder of the algorithm.[4] An example of a learned Λ from the mixed model is shown in Figure 7.4c. Figure 7.10 shows the BSS-eval results for the mixed model. The SIR values are lower, but comparable to the models using $\alpha = 2$. Again, we can see the tendency, that the SIR values minimally decrease after iteration one. For SDR and SAR, we observe slight improvements.

Recall that we used $\beta = 0$ (IS) for power spectrograms and $\beta = 1$ (KL) for magnitude spectro-

[4]To implement this idea, we only need to modify the input of the learning step to use magnitude spectrograms ($\alpha = 1$). To this end, we need to take the square root of the elements of Λ and the PSDs P_j before updating Λ, and squaring them after the update step.

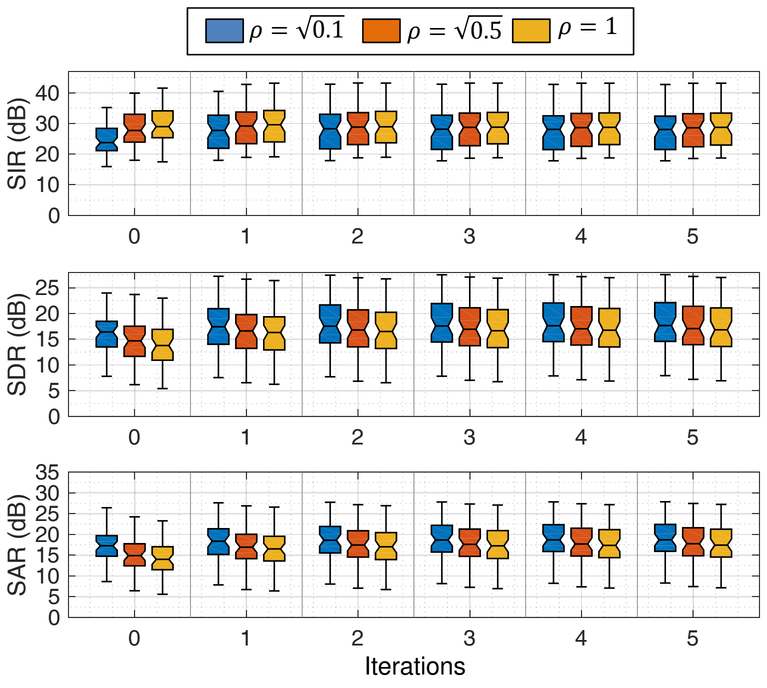


Figure 7.10. Results for mixed model, see Section 7.4.4.

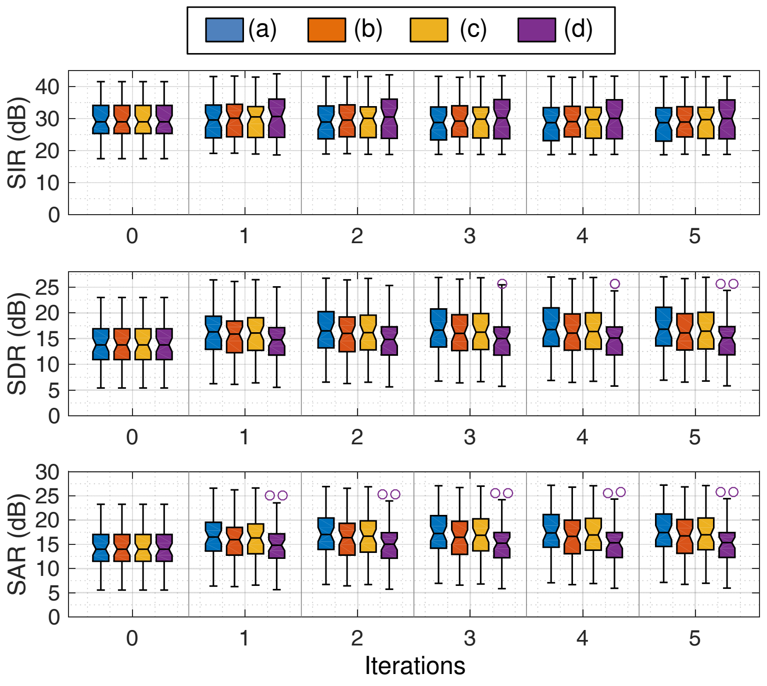


Figure 7.11. Evaluation measures for mixed model compared to model using power spectrograms ($\alpha = 2$) with different β values all using $\rho = 1$. **(a)** Mixed model with $\beta = 1$. **(b)** $\alpha = 2$, $\beta = 1$ (KL). **(c)** $\alpha = 2$, $\beta = 0.5$. **(d)** $\alpha = 2$, $\beta = 0$ (IS).

grams. To test whether the learning is generally more stable when using the mixed model, or if it is rather dependent on the choice of β, we additionally computed the results for $\alpha = 2$ with

$\beta \in \{0, 0.5, 1\}$ and compared them to the results for $\alpha = 1$ with $\beta = 1$ and the mixed model, both in terms of a quantitative evaluation of the interference matrices against the ground-truth mixing matrix (see Table 7.2) and the BSS-eval metrics (see Figure 7.11). As an error measure for evaluating Λ, we computed the *mean absolute difference* of the elements in the ground-truth mixing matrix $(g_{ij})_{i\in[1:I],j\in[1:J]}$ and the mixing matrix computed from Λ. Note that for linear mixes, the estimated mixing matrix can be computed from Λ by using (7.15). For the values in brackets, Λ has been first normalized to $[0, 1]$ by dividing each row by its maximum value before computing the error measure, which we will refer to by the *normalized mean absolute difference*. This normalization uses the prior knowledge that in our scenario, each microphone has a dominant source which has a mixing weight of one.

We now discuss some representative examples from Table 7.2. On average, using $\beta = 0$ leads to a mean absolute difference of 0.497 (0.137) which is the highest among all parameters tested. However, for piece No 01, $\beta = 0$ exhibits a lower mean absolute difference (0.109) than $\beta \in \{0.5, 1\}$ with $\alpha = 2$ (0.179 and 0.134). Generally, with increasing β, both error values decrease, with the lowest errors produced by $\alpha = 1$ with $\beta = 1$ resulting in an error of 0.066 (0.015), and the mixed model resulting in an error of 0.025 (0.019). On average, the mean absolute difference for $\alpha = 1$ with $\beta = 1$ is higher than for the mixed model, which is inverse for the normalized mean absolute difference. Hence, the mixed model computes interference matrices that are closer to the ground-truth in terms of absolute values, whereas the model using $\alpha = 1$ with $\beta = 1$ is closer when we use a normalization that exploits some prior knowledge. Overall, we can conclude that using $\alpha = 1$ stabilizes the learning of Λ. Interestingly, for piece No 06 (excerpt from Beethoven's "Ode to joy"), we observe the highest errors in the interference matrix computation. The instruments in this piece highly overlap in their melodic content which might be an explanation for the higher estimation error.

Concerning the BSS-eval metrics, the mixed model has the smallest SIR and the SIR is highest for $\alpha = 2$ with $\beta = 0$ (after five iterations). The higher SIR for the latter might be explained by the higher values in the learned interference matrix, see Figure 7.4d for an example. However, for the SDR and SAR, the results follow an inverse tendency. We conclude that the mixed model outputs a more stable interference matrix and leads to a good compromise between interference reduction and distortions/artifacts. Furthermore, we have seen that for $\beta \in \{0.5, 1\}$, the learning of the interference matrix improves and leads to similar results as the mixed model. This shows the importance of choosing an appropriate divergence measure.

7.4.5 Performance limits of MIRA and baseline comparisons

In this section, we compare MIRA to different algorithms and baselines both on linear mixes as computed in Section 7.3.2, as well as on the convolutive mixes from Section 7.3.3. As lower

No	$\alpha = 2$ $\beta = 0$	$\alpha = 2$ $\beta = 0.5$	$\alpha = 2$ $\beta = 1$	$\alpha = 1$ $\beta = 1$	mixed model $\beta = 1$
01	0.109 (0.107)	0.179 (0.053)	0.134 (0.048)	0.068 (0.013)	0.031 (0.022)
02	0.105 (0.075)	0.075 (0.020)	0.050 (0.014)	0.062 (0.014)	0.013 (0.009)
03	0.320 (0.081)	0.128 (0.045)	0.102 (0.042)	0.056 (0.019)	0.026 (0.019)
04	0.477 (0.114)	0.116 (0.032)	0.073 (0.023)	0.064 (0.014)	0.015 (0.011)
05	0.478 (0.125)	0.098 (0.030)	0.046 (0.012)	0.065 (0.014)	0.015 (0.012)
06	0.951 (0.211)	0.212 (0.065)	0.165 (0.064)	0.072 (0.015)	0.041 (0.029)
07	0.641 (0.150)	0.179 (0.056)	0.111 (0.042)	0.069 (0.016)	0.032 (0.027)
08	0.486 (0.139)	0.136 (0.040)	0.079 (0.027)	0.068 (0.016)	0.020 (0.015)
09	0.708 (0.159)	0.161 (0.046)	0.089 (0.030)	0.068 (0.012)	0.025 (0.019)
10	0.697 (0.203)	0.201 (0.071)	0.153 (0.061)	0.065 (0.020)	0.035 (0.029)
∅	0.497 (0.137)	0.149 (0.046)	0.100 (0.036)	0.066 (0.015)	0.025 (0.019)

Table 7.2. Mean absolute difference (normalized mean absolute difference) of learned Λ with $\alpha = 2$ after $N_{\text{iter}} = 5$ for each piece in the dataset.

performance limit for the SIR, we use the unprocessed mixture signals (Figure 7.12a). As an upper limit for Wiener filter-based approaches, we provide the upper performance limits for $\alpha = 1$ and $\alpha = 2$ (Figure 7.12f+g) as described Section 7.3.4.

As baseline approach, we compare to an algorithm for interference reduction developed by Kokkinis et al. [61]—in the following referred to as Kokkinis2012—which is also based on Wiener filtering. The algorithm uses a similar concept as the interference matrix which the authors call weighting coefficients. For Kokkinis2012, we use two parameter settings. First, in Kokkinis-GT (Figure 7.12b), we set the weighting coefficients to the ground-truth interference matrix. And second, in Kokkinis2012-UI (Figure 7.12), we use uniform weighting coefficients similar to the interference matrix initialized with $\rho = 1$. As further baseline approach, we compare to independent component analysis (ICA) which aims to recover independent components from linear mixtures. We used the implementation from [47][5] of the FastICA algorithm [55] (Figure 7.12h), which is an ICA variant.

As for MIRA, we provide the results for the $\alpha = 2$ with $\beta = 1$ (Figure 7.12d) and the mixed model (Figure 7.12e).

On the linear mixes, ICA outperforms all other approaches in terms of BSS-eval metrics. However, ICA has some outliers (circles under the boxes). These result from extracted signals in which some sources still highly overlap. For example, when considering the piece `AimeeNorwich_Child`, ICA perfectly extracts the singing voice (SIR = 53.2) and the drums (SIR = 38.4), but fails in separating the bass and guitar sources (SIR $\approx$ 0). In the case of the singing voice and the drums, the ICA results are even above the range of our upper limit soft masking with $\alpha = 2$ (SIR = 41.2). Interestingly, the SAR value of ICA is in the range of the unprocessed mixture signals. On the convolutive mixes, unsurprisingly, ICA practically fails,

[5] We set the number of components '`numOfIC`' to 5, '`approach`' to '`symm`', and the nonlinearity '`g`' to '`gauss`'.

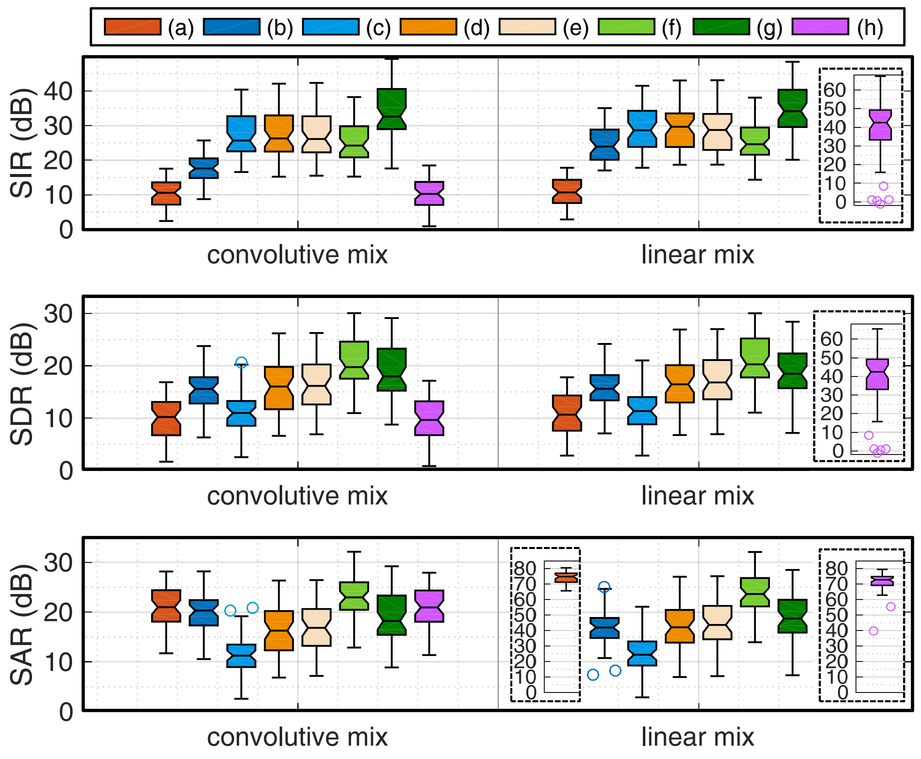


Figure 7.12. Comparison of performance measures on convolutive and linear mixtures. **(a)** Lower baseline for SIR. Note that this is not a lower limit for SDR and SAR. **(b)** Kokkinis2012-GT. **(c)** Kokkinis2012-UI. **(d)** $\alpha = 2$ with $\beta = 1$. **(e)** Mixed model. **(f)** Upper performance limit for $\alpha = 1$. **(g)** Upper performance limit for $\alpha = 2$. **(h)** FastICA.

having BSS-eval values similar to our unprocessed mixes.

MIRA and Kokkinis2012 perform comparable in terms of SIR both on the linear mixtures and the convolutive mixtures. Kokkinis2012-GT has the lowest SIR which is comparable to the zeroth iteration of MIRA using the ground-truth interference matrices, see also Figure 7.7. The SAR of Kokkinis2012-GT is comparable to the unprocessed mixture signals (Figure 7.12a). The SDR results of MIRA are comparable or even higher than for the Kokkinis2012 approaches. While Kokkinis-UI has a comparable SIR to MIRA, its SDR and SAR results are worse.

When considering the upper limit with $\alpha = 2$, there is still some room for improvement for Wiener filter-based approaches. Comparing the two upper limits, we can see that using power spectrograms ($\alpha = 2$) leads to more distortions and artifacts than using magnitude spectrograms ($\alpha = 1$), but a higher interference reduction (Figure 7.12f+g). The same findings on the influence of α on interference and artifacts have been reported in [23], where different α values have been evaluated using the Perceptual Evaluation Methods for Audio Source Separation (PEASS) [32, 125].

7.5 Conclusions

In this chapter, we introduced MIRA, an algorithm for interference reduction in multitrack music recordings based on [96]. MIRA iteratively estimates the PSD of each source and learns the strength of each source in each microphone via NMF. We extended the algorithm to use fractional spectrograms for which we evaluated the use of power spectrograms and magnitude spectrograms. Experiments have shown that interference reduction for power spectrograms and magnitude spectrograms is comparable when the interference matrix is kept fixed through the iterations of the algorithm. When the learning is activated, power spectrograms have shown to perform better. Furthermore, we observed a strong influence of the parameter β on the quality of the learned interference matrices. Finally, experiments show that MIRA's interference reduction is comparable to other Wiener filter-based approaches.

7.6 Further Notes

In this section, we describe potential applications and use cases of the MIRA algorithm. In Section 7.6.1, we describe MIRA as a pre-processing step to simplify music annotation procedures. Furthermore, we point to the applications of MIRA in remixing and source separation (Section 7.6.2 and Section 7.6.3 respectively).

7.6.1 Annotation

The evaluation of MIR algorithms typically requires annotated music. Generating annotations for instrument activations, fundamental frequency, or onsets is generally more complex on mixture signals containing many sources than on multitrack recording where each track contains an isolated source. In the latter case, algorithms designed for processing monophonic signals can be used to assist the annotation process. Often, however, the individual tracks in a multitrack recording contain interference from other sources. Using MIRA in a pre-processing step to generate cleaner multitrack recordings can drastically reduce the work required in the annotation process. In [97], we performed an experiment using the monophonic pitch tracker YIN [21] on clean multitrack signals, the same multitracks containing artificial (convolutional) interference as described in Section 7.3.3, and the output of MIRA on these multitracks. The experiments have shown that the pitch tracker performs virtually as good on the multitracks processed by MIRA as on the clean multitracks. Further results and an extended discussion will be presented in [97] and [10].

7.6.2 Remixing

Interference greatly reduces the mixing possibilities for a sound engineer and it prevents the removal or isolation of a voice from the recording, which may be desirable, e.g. for pedagogical reasons or "music minus one" applications (mixtures where a particular voice has been removed). An instrument equalizer provides the possibility to adjust the volume of an individual instrument in a recording without affecting the volume of the other instruments. In [96], we processed multitrack recordings from the "Freischütz Digital Multitrack Dataset" (Appendix A) to reduce the interference in the individual tracks.[6] We exemplarily show the use of the processed signals in an instrument equalizer where the volume of each instrument or voice can be changed separately without affecting the volume of others [102]. For example, when studying a specific melody line of the violins and the flutes, an instrument equalizer enables a user to raise the volume for these two voices and to lower it for the other instruments.

The demo can be found at

`https://www.audiolabs-erlangen.de/resources/MIR/FreiDi-InstrumentEqualizer/`.

7.6.3 Source Separation

In cover song based source separation [48], a multitrack recording of a cover song is used to support the separation of the original song by using the multitracks to inform the source separation model. In the case of classical live recordings, the multitracks usually have bleed. Using MIRA as a pre-processing could help to create cleaner tracks for the initialization of a source separation model. In the context of audio source separation, multiple deformed reference recordings of the same piece of music have been used to support the separation process [118]. For example, multitrack recordings from cover versions of a piece have been used to initialize model parameters for the separation of individual instruments in the original stereo recordings of the piece [48].

[6]Sound examples can be found at `http://www.audiolabs-erlangen.de/resources/MIR/2015-ICASSP-KAMIR/`

Chapter 8

Summary and Future Work

Nun frisch' noch einmal das Ende des Liedchens!

Ännchen, "Der Freischütz" [114]

In this thesis, we discussed content-based audio processing tasks that were motivated by the interdisciplinary project "Freischütz Digital" (FreiDi, Chapter 2). The opera "Der Freischütz" served as a challenging real-world music scenario. Among others, we considered 28 recordings of different "Der Freischütz" interpretations exhibiting acoustic and structural variations. For example, the recordings differ in duration, tuning, length and realized text of the dialogues, language, and sound quality. In addition to 23 full recordings of the opera with an average duration of over two hours, our dataset also included 5 abridged recordings that structurally differed by omitting varying parts of the opera. The FreiDi scenario also includes multitrack recordings from the "Freischütz Digital Multitrack Dataset" (Appendix A). Motivated by the complex music scenario in the FreiDi project, we developed novel automated music processing approaches for music synchronization, segmentation, and interference reduction on a comprehensive dataset of opera music. Furthermore, we indicated how the introduced methods can help to access, analyze, and compare different audio recordings. We showed how automated methods may be useful for systematically revealing and understanding the inconsistencies and variations across the music recordings.

We developed a method to consistently segment all 28 opera recordings in our dataset according to a given reference segmentation (Chapter 5). By systematically adapting the segmentation procedure, we not only improved the segmentation results, but also revealed how much the tuning of a recording, structural variations, and harmonic similarities between segments influence the segmentation results (Section 5.2). Furthermore, we introduced a segmentation approach to accommodate abridged versions. This involved solving the problem of transferring a labeled reference segmentation from a complete recording of the opera onto recordings where segments are entirely or partly omitted. Experiments have shown that our method works robustly even when the recordings are abridged (Chapter 5.3.9). With our segmentation procedures, we sometimes see confusion of highly repetitive passages such as the different verses of a song. This can be explained by the fact that the chroma-based features used in our segmentation approach capture the harmonic content of the signal only—so in passages containing musical repetitions, these features are very similar and thus difficult to distinguish. In future work, modeling the lyrics, e.g. by using features from speech processing [70], could help to resolve some of these confusions. Further improvements could be achieved by using additional cues from an audio classification system, e.g. detecting segments with singing voice, speech, silence, or applause [22].

When synchronizing two full recordings of an opera like "Der Freischütz" (each having a duration of over two hours), memory consumption becomes an issue. We therefore developed a novel memory-efficient music synchronization algorithm (see Chapter 4) in which the memory requirements can be kept constant. Experiments showed that our proposed algorithm basically yields the same alignments as classical DTW, even when restricting the memory consumption to eight megabytes (compared to one terabyte when using the full DTW to synchronize a complete recording of the opera). We have shown different application scenarios for music synchronization such as for tempo extraction (Chapter 4.6.2) and navigation purposes (Synchronization & Interpretation Switcher in Section E.1). Ground-truth data for evaluating music synchronization is often unavailable and difficult to generate. To address this issue, we introduced the triple error, a novel method to analyze alignment results which can be applied when ground-truth data is not available (Chapter 6). The triple error can be computed when at least three recordings of the same music piece are available. We showed that using this method often permits drawing conclusions similar to those derived from the classical pairwise alignment error (Section 6.3). In first experiments, we also indicated that the triple error might be useful to detect recordings that are likely to cause a higher rate of alignment errors (Section 6.3.4). A large-scale analysis using the triple error on datasets containing many versions of the same piece of music would be a natural continuation of this work. Further investigations include experiments on using the triple-analysis to identify problematic versions causing the alignment method to fail, detecting errors in ground-truth annotations, and analyzing whether alignment methods can be optimized or combined by using the triple error.

Motivated by the "Freischütz Digital Multitrack Dataset" (Appendix A), we developed MIRA, an iterative algorithm to reduce interference in multitrack music recordings (Chapter 7). MIRA simultaneously learns the PSD of each source and its contribution to each microphone signal using Wiener filtering and NMF. In controlled experiments, we showed that the algorithm can be applied both to linear mixes and convolutive mixes. Iterating the algorithm leads to a good compromise between interference reduction and artifacts in the resulting signals. Furthermore, we indicated possible application scenarios of MIRA in the creation of ground-truth pitch annotations, mixing, and source separation (Section 7.6). A promising research direction is to combine music synchronization with interference reduction. For example, the interference-reduced multitrack recordings may be warped onto historic recordings to support the extraction of instruments or singing voice from historical recordings, similar to the approaches introduced in [48, 118]. If available, the multitrack recordings in this approach could also be replaced by a musical score as proposed in score-informed source separation approaches [35, 69].

Many of our procedures were motivated by and tested within the FreiDi scenario. However, we want to emphasize that the methods introduced in this thesis are applicable beyond the FreiDi scenario, which was demonstrated by using diverse datasets in our experiments, including classical music and popular music.

Appendix A

Freischütz Digital Multitrack Dataset

In the project FreiDi project, three numbers (No. 06, 08, 09) of the opera “Der Freischütz” from Carl Maria von Weber have been recorded by the Erich-Thienhaus-Institute (HfM Detmold)[1]. The recording was carried out by the Tonmeister students Stefan Antonin (No. 6), Florian Bitzer (No. 8), and Matthias Kieslich (No. 9) under the supervision of Prof. Dipl-Tonm. Bernhard Güttler and Prof. Dipl-Tonm. Michael Sandner. Further information on the recording sessions and audio production can be found on the Freischütz Digital website[2][3]

The recordings are publicly accessible as the *Freischütz Digital Multitrack Dataset* on the following website:

`https://www.audiolabs-erlangen.de/resources/MIR/FreiDi/MultitrackDataset`.

For each of the recorded numbers (No. 6, 8, 9), the following data is provided:

Stereo Mixes. An artistic stereo mix including all voices and microphones. This is the kind of mix one usually finds on professionally produced CD recordings.

Mono Tracks RAW. These are the signals as they were recorded by the microphones without further processing.

[1]`http://www.eti.hfm-detmold.de`

[2] Background informations on the recording session (in German) `http://freischuetz-digital.de/audio-recording-2013.html`

[3]Finalization of the audio production (in German) `http://freischuetz-digital.de/audio-production-2014.html`

Mono Tracks EQDEL. The raw mono tracks were delayed with respect to the main microphones (AA,BB). Furthermore, individual equalizers (EQs) have been applied to the signals. These tracks have been used in the production of the stereo mixes.

Stereo Instrument Mixes Additional stereo mixes were produced for each individual voice.

Stereo Group Mixes. Additional group mixes were produced including the woodwinds (bassoon, fute, clarinet, oboe), the strings (violin 1, violin 2, viola, cello, double bass), and the singers.

Automation Tracks. During the recordings, the volumes of the singing voice has been continuously adapted. To make this process clear, we provide the automation tracks for these tracks.

The recordings are provided as WAV files sampled at 48 kHz/24 bit.

Figure 2.7b shows the microphone setup used in the recording session. The setup involved 25 microphones, involving two main microphones for recording a stereo image and at least one spot microphone for each instrument section. For each string section, a spot microphone at the front (Nf) and at the rear (Nr) position was used. Additionally, clip microphones (C) were used for principal musicians of the string sections.

The first violin section, for example, was recorded with three microphones: one at the front position, one at the rear position, and a clip microphone attached to the principal musician's instrument. Usually, a voice is captured by its spot microphones before it arrives at the main microphones which are positioned further away. Therefore, it is important to compensate for different runtimes by delaying the spot microphones such that their signals are synchronized to the main microphones. This avoids unwanted reverberation or artifacts (caused by phase interference) in the mixing process. Furthermore, individual equalizers are applied to each of the microphones to suppress frequencies that are outside of the range of their associated voice. Note that in a typical professional setup, the recording room is equipped with sound absorbing materials and acoustic shields to isolate all the voices as much as possible. However, especially for orchestra music, complete acoustic isolation between the voices is often not possible. In practice, each microphone not only records sound from its dedicated voice, but also from all others in the room. This results in recordings that do not feature isolated signals, but rather mixtures of a predominant voice with all other voices being audible through what is referred to as interference, bleeding, crosstalk, or leakage. The recordings in the Freischütz Digital Multitrack Dataset all have interference between the different microphones.

Mic No.	Mic String	Description	Delay	No. 6	No. 8	No. 9
01	aa-O	Main microphone 1	0	x	x	x
02	bb-O	Main microphone 2	0	x	x	x
03	vn1-Nf	Violin 1 (front)	268	x	x	x
04	vn1-Nr	Violin 1 (rear)	473	x	x	x
05	vn1-C	Violin 1 (clip)	0	x	x	x
06	vc-Nf	Cello (front)	403	x	x	x
07	vc-Nr	Cello (rear)	552	x	x	x
08	vc-C	Cello (clip)	0	x	x	x
09	va-Nf	Viola (front)	342	x	x	x
10	va-Nr	Viola (rear)	601	x	x	x
11	va-C	Viola (clip)	0	x	x	x
12	vn2-Nf	Violin 2 (front)	329	x	x	x
13	vn2-Nr	Violin 2 (rear)	601	x	x	x
14	vn2-C	Violin 2 (clip)	0	x	x	x
15	db-O	Bass (omni)	832	x	x	x
16	db-S	Bass	861	x	x	x
17	fl-N	Flute	648	x	x	x
18	ob-N	Oboe	659	-	x	-
19	cl-N	Clarinet	790	x	x	x
20	bn-N	Bassoon	825	x	x	x
21	hn-Nf	Horn (front)	922	x	x	x
22	hn-Nr	Horn (rear)	953	x	x	x
23	s1-S	Singer 1 (Ännchen)	160	x	-	x
24	s2-S	Singer 2 (Max)	160	-	-	x
25	s3-S	Singer 3 (Agathe)	160	x	x	x

Table A.1. Overview of the microphone used in the recordings. Entries marked with a dash '-' in the last three columns specify that an instrument/voice is not present in the respective number. If the microphone string (Mic String) is followed by a dash, then the first character specifies the type of microphone (N: cardioid, S: super-cardioid, O: omnidirectional, C: clip microphone). An optional second character specifies the microphone position (f: front, r: rear). The delay with respect to the main microphones (AA,BB) is specified in samples.

Table A.1 shows an overview of the microphones, their associated voices, and the delay with respect to the main microphones. All tracks have been recorded with cardioid microphones (N) except:

- Microphone 23-25 (Singers): super-cardioid (S)
- Microphone 15 (Bass): omnidirectional (O)
- Microphone 05,08,11,14 (Strings): clip microphone (C).

Appendix B

Freischütz Recordings

Table B.1 shows the recordings that have been used in the synchronization and segmentation experiments in Chapter 4 and Chapter 5. Parts of the presented information has been taken from the collection at

`http://www.webergesellschaft.de/wp-content/uploads/2016/03/Diskographie2015.pdf`

containing a compilation of recordings of pieces by Carl-Maria von Weber. At this point, I want to thank Frank Ziegler for compiling this overview.

ID	Dur. (sec)	Conductor	Publisher
Ack1951	6905	Otto Ackermann	Decca LXT 2597-9 (ARL 837-42)
Boe1972	7772	Karl Böhm	Arkadia CDMP 457.2
Bru1957	7439	Wilhelm Brückner-Rüggeberg	Hamburger Archiv für Gesangskunst 30057
Dav1990	8198	Sir Colin Davis	Philips 426 319-2
Elm1944	7082	Karl Elmendorff	Cantus Classics CACD 5.00125 F
Fur1954	9122	Wilhelm Furtwängler	EMI 567419 2
Gui1957	6911	Vittorio Gui	IMD Music Distributing S.A. ANDRCD 5032
Har1995	8045	Nikolaus Harnoncourt	Arthaus Musik 107011
Hau1985	8240	Dieter Hauschild	Denon CM 7433/35
Heg1969	7437	Robert Heger	EMI 153-28351-3
Jan1994	7843	Marek Janowski	RCA 09026 62538-2
Joc1960	7178	Eugen Jochum	DG 2727 003
Kei1958	8043	Joseph Keilberth	EMI 137-290696-3
Kle1973	7763	Carlos Kleiber	DG 2720 071
Kle1955	7459	Erich Kleiber	Hunt 34033
Kub1979	8045	Rafael Kubelik	Decca 6.35 504
Leo1972	7726	Leopold Ludwig	Gala 100.729
Mat1967	8309	Lovro Matačić	RCA 74321 25287 2
Mue1950	7560	Hans Müller-Kray	Walhall 0027
Orl1946	7369	Alexander Orlov	AQVR 356-2
Pen1998	7768	Jean-Paul Penin	L'empreinte digitale 13100/101
Saw1972	6871	Wolfgang Sawallisch	Myto 061.322
Wei2001	7220	Bruno Weil	DHM 05472 77536 2
Kna1939	1965	Hans Knappertsbusch	Koch Schwann 3-1467-2
Kri1933	1417	Josef Krips	Koch Schwann 3-1453-2
Mor1939	1991	Rudolf Moralt	Koch Schwann 3-1463-2
Ros1956	2012	Mario Rossi	On stage! 4707
Sch1994	2789	Wolfgang Schmid	Musikhochschule Graz MHS G 1

Table B.1. Overview of the recordings used for the segmentation and synchronization experiments with their total duration in seconds (Dur.). The upper part of the table shows the complete recordings, and the lower part shows the abridged recordings.

Appendix C

Sync Toolbox

The Sync Toolbox—a collection of MATLAB tools for music synchronization and its application—is a long-term project by Meinard Müller and his research group. It includes a MIDI toolbox that was developed by Frank Kurth and extended by Sebastian Ewert. The first music synchronization approaches in the Sync Toolbox were realized using standard DTW procedures and chroma-based features [85, 84]. Then, first efficient multiscale implementations (MsDTW) were implemented in [86]. This approach has been extended in [36] to combine chroma features with onset-based chroma features to realize a high-resolution music synchronization algorithm.

A main contribution of this work was a revision of the Sync Toolbox to include our proposed MrMsDTW algorithm [98]. The toolbox was restructured such that it relies on a single efficient DTW implementation in MATLAB and C++ written by Thomas Helten. The MrMsDTW calls this implementation as its working horse.

Furthermore, the toolbox contains utility functions for sonification that were adapted from old revisions of the toolbox developed by Meinard Müller, Sebastian Ewert, and Peter Grosche, as well as new tools for time scale modifications (TSM) [27] implemented by Jonathan Driedger.

C.1 Organization

Figure C.1 shows the directory structure of the Sync Toolbox and a short description of the subfolder contents. The folder `SyncCore/` contains the core functionality of the toolbox. A short description of the most important functions of this folder is listed in Table C.1.

Furthermore, the Sync Toolbox contains the Chroma Toolbox [78] (providing essential feature extractors) and the TSM Toolbox [27] (for utitily functions). The root folder of the Sync Toolbox contains a set of demo scripts that illustrate the main functionalities of the toolbox, see Ta-

```
MATLAB_SyncToolbox_1.0
├── data_music/ .................................................. Audio and MIDI data
├── data_output/ .......... Computed output data (features, warping paths, sonifications)
├── DTW_TH/ ............................ DTW functions implemented by Thomas Helten
├── Feature/ ..................................................... Feature extractors
│   └── MATLAB-Chroma-Toolbox_2.0/ ............................... Chroma toolbox [78]
├── MATLAB_TSM-Toolbox_1.0/ . Time Scale Modification Toolbox (used for sonification) [27]
├── MIDI/ .......................................... MIDI read/write and utility functions
├── SyncCore/ ...................................... Synchronization related functions
├── SyncUtil/ ...................................... Utility functions, e.g. sonification
├── demo_01_feature_highRes.m
├── demo_01_feature_standard.m
├── demo_02_sync_DTW_highRes.m
├── demo_02_sync_DTW_standard.m
├── demo_02_sync_DTW_toy.m
├── demo_02_sync_MrMsDTW_highRes.m
├── demo_02_sync_MrMsDTW_standard.m
├── demo_02_sync_MrMsDTW_toy.m
├── demo_03_app_generateWebsite.m
├── demo_03_app_sonifyMIDI_sinusoidals.m
├── demo_03_app_sonifyMIDI_timidity.m
├── demo_03_app_warpAnnotation.m
├── demo_03_app_warpAudio.m
├── demo_03_app_warpMIDI.m
└── demoPipeline_highResSynchronization.m
```

Figure C.1. Directory structure of Sync Toolbox

ble C.2. Demo scripts including a number in their name need to be evoked in the correct running order. An illustrative example for this is given by `demoPipeline_highResSynchronization.m` in Section C.2.1. Section C.2.2 shows the demo script illustrating how the high resolution synchronization is computed using our proposed MrMsDTW algorithm.

C.2 Code examples

C.2.1 `demoPipeline_highResSynchronization.m`

The following script demonstrates how the different demo scripts are evoked.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Name: demoPipeline_highResSynchronization.m
% Date of Revision: 2016-09
% Programmer: Thomas Praetzlich
%
% Description:
```

```
% Demo for high resolution MrMsDTW pipeline using
% pitch and pitch onset features.
% Furthermore, the pipeline invokes:
%  - warping of MIDI
%  - warping of audio
%  - sonification of MIDI via sinusoidals
%  - sonification of MIDI via timidity
%  - website containing the synchronization and
%    sonification results.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Compute features
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
demo_01highRes_feature

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Run synchronization
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
demo_02highRes_sync_MrMsDTW

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% warp MIDI
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
demo_03_warpMIDI

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Sonification
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% warp audio to MIDI time axis via TSM
demo_03_app_warpAudio

% sonify MIDI with sinusoidals
demo_04_app_sonifyMIDI_sinusoidals

% sonify MIDI with sinusoidals
demo_04_app_sonifyMIDI_timidity


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Generate Website
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
demo_05_app_generateWebsite
```

C.2.2 demo_02highRes_sync_MrMsDTW.m

Example for high resolution synchronization demo script.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Name: demo_02highRes_sync_MrMsDTW
% Date of Revision: 2016-09
% Programmer: Thomas Praetzlich
%
% Description:
% Demo for high resolution synchronization using
% MrMsDTW (Memory-restriced multiscale DTW).
% Shows how to compute high resolution alignments
% based on a combination of pitch features and pitch
% onset features using MrMsDTW.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear;
close all;
%init;
addpath(fullfile(pwd,'DTW_TH'));
addpath(fullfile(pwd,'Feature'));
addpath(fullfile(pwd,'Feature/MATLAB-Chroma-Toolbox_2.0'));
addpath(fullfile(pwd,'SyncCore'));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Specify data
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
pathData = 'data_output/';
pathOutput = 'data_output/';

cell_alignPairs = {
    'Chopin_Op010-03-Measures1-8_Igoshina','Chopin_Op010-03-Measures1-8
        _Varsi';
    'Chopin_Op010-03-Measures1-8_Varsi','Chopin_Op010-03-Measures1-8
        _Igoshina';
    'Chopin_Op010-03-Measures1-8_Igoshina','Chopin_Op010-03-Measures1-8
        _MIDI'
    'Chopin_Op010-03-Measures1-8_Varsi','Chopin_Op010-03-Measures1-8_MIDI'
    };

for k = 1:size( cell_alignPairs,1 )

    name1 = cell_alignPairs{k,1};
    name2 = cell_alignPairs{k,2};

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %% Load and compute features
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fprintf('Loading pitch features ...\n');
l = load([pathData name1 '_pitch_882.mat']);
f_pitch1 = l.f_pitch;
pitch_sideinfo1 = l.sideinfo;

l = load([pathData name2 '_pitch_882.mat']);
f_pitch2 = l.f_pitch;
pitch_sideinfo2 = l.sideinfo;

l = load([pathData name1 '_pitchOnsetPeaks.mat']);
f_peaks1 = l.f_peaks;
peaks_sideinfo = l.sideinfo;

l = load([pathData name2 '_pitchOnsetPeaks.mat']);
f_peaks2 = l.f_peaks;
shiftFB1 = 0;
shiftFB2 = 0;
if isfield(pitch_sideinfo1.pitch,'shiftFB')
    shiftFB1 = pitch_sideinfo1.pitch.shiftFB;
end
if isfield(pitch_sideinfo2.pitch,'shiftFB')
    shiftFB2 = pitch_sideinfo2.pitch.shiftFB;
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Synchronize
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fprintf('Synchronizing via sync_pitchOnset_pitchOnset_via_MrMsDTW \n')
    ;

clear paramSync ;
paramSync.winLenSmooth    =  [201 101 21 1];
paramSync.downsampSmooth  =  [50 25 5 1];
paramSync.implementation  = 'mex';
% paramSync.implementation  = 'matlab';
paramSync.stepSizes =   [1 0 1; ...
    0 1 1];
paramSync.stepWeights = [1.5 1.5 2];
paramSync.thresholdRec = 10^6;
paramSync.visualizeSteps = true;
paramSync.chromaTranspositions = 0:11;
% paramSync.chromaTranspositions =  0;
paramSync.parallelComputing = 0;

```

```
    start_time = tic;
    syncinfos = sync_pitchOnset_pitchOnset_via_MrMsDTW( f_pitch1, f_peaks1
        , f_pitch2, f_peaks2, paramSync);
    fprintf('Computation time of sync_pitchOnset_pitchOnset_via_MrMsDTW: %
        f seconds\n', toc(start_time));

    % Add filterbank shifts
    syncinfos.shiftFB1 = shiftFB1;
    syncinfos.shiftFB2 = shiftFB2;

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %% Save
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    % Save old syncinfos
    save([pathOutput name1 '--' name2 '_syncinfos.mat'],'syncinfos');

    % Save warpingpath as anchor alignment
    A = indices_to_seconds( syncinfos.warpingpath, syncinfos.featureRate )
        ;

    alignment = [];
    alignment.chromaShift = syncinfos.chromaShift;
    alignment.shiftFB1 = syncinfos.shiftFB1;
    alignment.shiftFB2 = syncinfos.shiftFB2;
    alignment.A        = A;

    % save alignment as .mat file
    save([pathOutput name1 '--' name2 '_alignment.mat'],'alignment');

    % and also as text file
    dlmwrite([pathOutput name1 '--' name2 '.txt'],A','precision','%08.03f'
        );

end
```

Filename	Main parameters	Description
`sync_pitchOnset_pitchOnset_via_MrMsDTW.m`	thresholdRec, winLenSmooth, downsampSmooth, stepSizes, stepWeights	MrMsDTW implementation for high resolution synchronization taking pitch and pitch onset features from different versions of a music piece as input.
`sync_seqFeat_seqFeat_via_MrMsDTW.m`	thresholdRec, winLenSmooth, downsampSmooth, stepSizes, stepWeights	MrMsDTW implementation that works with arbitrary feature sequences as input.
`buildPath_fromWPCell.m`	wpCell, A	Builds a path from a given cell of warping paths and an alignment. The indices of the original warping paths are adapted such that they fullfil the alignment conditions.
`computeWP_viaDTW.m`	stepSizes, stepWeights, implementation	Applies DTW on cost matrix C.
`computeWPCell.m`	Ccell, stepSizes, stepWeights, implementation	Computes a path via DTW on each matrix in Ccell.
`computeC.m`	distanceMeasure, cosMeasMin, cosMeasMax, l2MeasMin, l2MeasMax	Computes a cost matrix **C** from two feature sequences.
`computeC_HighRes.m`	distanceMeasure, cosMeasMin, cosMeasMax, l2MeasMin, l2MeasMax	Computes a cost matrix **C** from chroma and DLNCO features.
`computeCCell.m`	distanceMeasure, cosMeasMin, cosMeasMax, l2MeasMin, l2MeasMax	Compute cell of cost matrices from chroma features.
`computeCCell_HighRes.m`	distanceMeasure, cosMeasMin, cosMeasMax, l2MeasMin, l2MeasMax	Compute cell of cost matrices from chroma and DLNCO features.
`sizeDTWCell.m`	DTWCell	Gives information about the dimensionality of a DTW matrix given in form of a cell matrix.
`DTWCellMatrices_to_DTWMatrices.m`	–	Converts DTW cell matrices to DTW matrices.
`visualize_C.m`		Visualization of cost matrix **C**.
`visualize_CCell.m`		Visualization of cost matrices.
`visualizeContraintRectangles.m`		Visualization of constraint regions used in MrMsDTW.

Table C.1. Overview of the core MATLAB functions and parameters of the Sync Toolbox contained in `SyncCore/`.

Filename	Description
`demo_01highRes_feature.m`	Computes pitch and pitch onset features from audio and MIDI files in `data_music/` with a resolution of 50 Hz needed for high resolution synchronization.
`demo_01standard_feature.m`	Computes pitch features from audio and MIDI files in `data_music/` with a resolution of 10 Hz needed for the standard synchronization.
`demo_02highRes_sync_DTW.m`	Shows how to compute high resolution (50 Hz) alignments based on a combination of pitch and pitch onset features DTW
`demo_02highRes_sync_MrMsDTW.m`	Shows how to compute high resolution (50 Hz) alignments based on a combination of pitch features and pitch onset features using MrMsDTW.
`demo_02standard_sync_DTW.m`	Shows how to compute alignments (10 Hz) based on CENS features (derived from the pitch features computed in previous demo scripts) using DTW.
`demo_02standard_sync_MrMsDTW.m`	Shows how to compute alignments (10 Hz) based on CENS features (derived from the pitch features computed in previous demo scripts) using MrMsDTW.
`demo_03_app_warpMIDI.m`	Shows how a MIDI file can be warped according to a computed synchronization result.
`demo_03_app_warpAudio.m`	Shows how two audio signals can be warped with respect to a given anchor alignment, using time-scale modification algorithms.
`demo_04_app_sonifyMIDI_sinusoidals.m`	Shows how a MIDI file can be sonified with simple sine tones. Additionally, a stereo audio file containing the original audio on the left channel, and the sine wave sonification on the right channel is generated.
`demo_04_app_sonifyMIDI_timidity.m`	Shows how a MIDI file can be sonified with timidity. Additionally, a stereo audio file containing the original audio on the left channel, and the sine wave sonification on the right channel is generated.
`demo_05_app_generateWebsite.m`	Shows how one can generate a HTML website presenting the synchronization and sonification results.
`demoPipeline_highResSynchronization.m`	Invokes all demo scripts (from feature computation to applications) for the high resolution synchronization.
`demoPipeline_standardSynchronization.m`	Invokes all demo scripts (from feature computation to applications) for the standard synchronization.
`demo_sync_DTW_toy.m`	Toy example aligning two manually defined feature sequences via DTW.
`demo_sync_DTWwithAnchor.m`	Dempo Script for constraing a DTW alignment with anchor alignment.
`demo_sync_MrMsDTW_toy.m`	Toy example aligning two manually defined feature sequences using MrMsDTW.
`demo_warpMIDI_sonifyMIDI.m`	Shows how a MIDI file can be warped and sonified according to a manually specified synchronization result.

Table C.2. Overview of demo scripts in the Sync Toolbox. Demo scripts that have a number need to be evoked in the correct order, e.g. before starting one of the scripts with the prefix `demo_02highRes`, features need to be computed with `demo_01highRes_feature.m`. An explicit example for this is given in `demoPipeline_highResSynchronization.m`.

Appendix D

Overview Segmentation

Table D.1 shows an overview of the segmentation that was used in Chapter 5.

No	Name	page in score	#measures	Tracks (all)	Tracks (music)	start [measure]	end [measure]	ID
0	Ouverture	7	342	1	1	1.000	36.999	Weber_Freischuetz-00-01M
				2	2	37.000	278.999	Weber_Freischuetz-00-02M
				3	3	279.000	342.999	Weber_Freischuetz-00-03M
1	Introduction	23	138	4	4	1.000	65.999	Weber_Freischuetz-01-01M
				5	5	66.000	98.999	Weber_Freischuetz-01-02M
				6	6	99.000	138.999	Weber_Freischuetz-01-03M
				7	▨			***Weber_Freischuetz-01-04D***
2	Terzett/Chor	35	217	8	7	1.000	62.999	Weber_Freischuetz-02-01M
				9	8	63.000	128.833	Weber_Freischuetz-02-02M
				10	9	128.833	217.999	Weber_Freischuetz-02-03M
				11	▨			***Weber_Freischuetz-02-04D***
3	Scene, Walzer, Arie	53	252	12	10	1.000	57.999	Weber_Freischuetz-03-01M
				13	11	58.000	136.999	Weber_Freischuetz-03-02M
				14	12	137.000	164.999	Weber_Freischuetz-03-03M
				15	13	165.000	252.499	Weber_Freischuetz-03-04M
				16	▨			***Weber_Freischuetz-03-05D***
4	Lied	64	31	17	14	1.000	31.999	Weber_Freischuetz-04-01M
				18	▨			***Weber_Freischuetz-04-02D***
				19	15	5.000	31.999	Weber_Freischuetz-04-03M
				20	▨			***Weber_Freischuetz-04-04D***
				21	16	5.000	31.999	Weber_Freischuetz-04-05M
				22	▨			***Weber_Freischuetz-04-06D***
5	Arie: Schweig!	66	118	23	17	1.000	118.999	Weber_Freischuetz-05-01M
6	Duett: Schelm	76	149	24	18	1.000	149.999	Weber_Freischuetz-06-01M
				25	▨			***Weber_Freischuetz-06-02D***
7	Ariette	86	113	26	19	1.000	113.999	Weber_Freischuetz-07-01M
				27	▨			***Weber_Freischuetz-07-02D***
8	Scene + Arie	92	198	28	20	1.000	198.999	Weber_Freischuetz-08-01M
				29	▨			***Weber_Freischuetz-08-02D***
9	Terzett	104	199	30	21	1.000	141.749	Weber_Freischuetz-09-01M
				31	22	141.750	199.999	Weber_Freischuetz-09-02M
10	Finale	118	430	32	23	1.000	109.999	Weber_Freischuetz-10-01M
				33	24	110.000	260.999	Weber_Freischuetz-10-02M
				34	▨			***Weber_Freischuetz-10-03D***
				35	25	261.000	372.999	Weber_Freischuetz-10-04M
				36	26	373.000	430.999	Weber_Freischuetz-10-05M
11	Entre-Acte	148	94	37	27	0.500	94.999	Weber_Freischuetz-11-01M
				38	▨			***Weber_Freischuetz-11-02D***
12	Cavatina	153	66	39	28	1.000	66.999	Weber_Freischuetz-12-01M
				40	▨			***Weber_Freischuetz-12-02D***
13	Romanze + Arie	156	169	41	29	1.000	51.749	Weber_Freischuetz-13-01M
				42	30	51.750	169.999	Weber_Freischuetz-13-02M
				43	▨			***Weber_Freischuetz-13-03D***
14	Volkslied	165		44	31	1.000	28.999	Weber_Freischuetz-14-01M
				45	▨			***Weber_Freischuetz-14-02D***
				46	32	29.000	34.999	Weber_Freischuetz-14-03M
				47	▨			***Weber_Freischuetz-14-04D***
				48	33	35.000	61.999	Weber_Freischuetz-14-05M
15	Jägerchor	169	77	49	34	0.750	77.749	Weber_Freischuetz-15-01M
				50	▨			***Weber_Freischuetz-15-02D***
16	Finale	174	422	51	35	1.000	113.999	Weber_Freischuetz-16-01M
				52	36	114.000	256.749	Weber_Freischuetz-16-02M
				53	37	256.750	367.999	Weber_Freischuetz-16-03M
				54	38	368.000	422.999	Weber_Freischuetz-16-04M

Table D.1. Overview of the segmentation of the "Freischütz" opera used in this thesis. The individual segments are defined by their start and end position given in measures. Dialogues have no measure number and are defined by their surrounding musical segments.

Appendix E

Demos

The project "Freischütz Digital"[1] (funded by the German Federal Ministry of Education and Research), was a close cooperation between musicologists and computer scientists to explore new digital ways for analyzing and presenting music-related data in critical editions. The opera "Der Freischütz" by Carl Maria von Weber served as challenging example scenario in the project. It offers a large number of (historical) sources including different versions of the musical score, the libretto, and (multitrack) audio recordings. One major task of the project was to adapt music synchronization techniques to align different versions of the same piece of music [19, 100, 77]. Such alignments can then be used to realize extended music audio players, that display various types of music-related information while playing. In this chapter, we present three audio-related web demos [102] providing simple navigation and interaction possibilities with the music material.

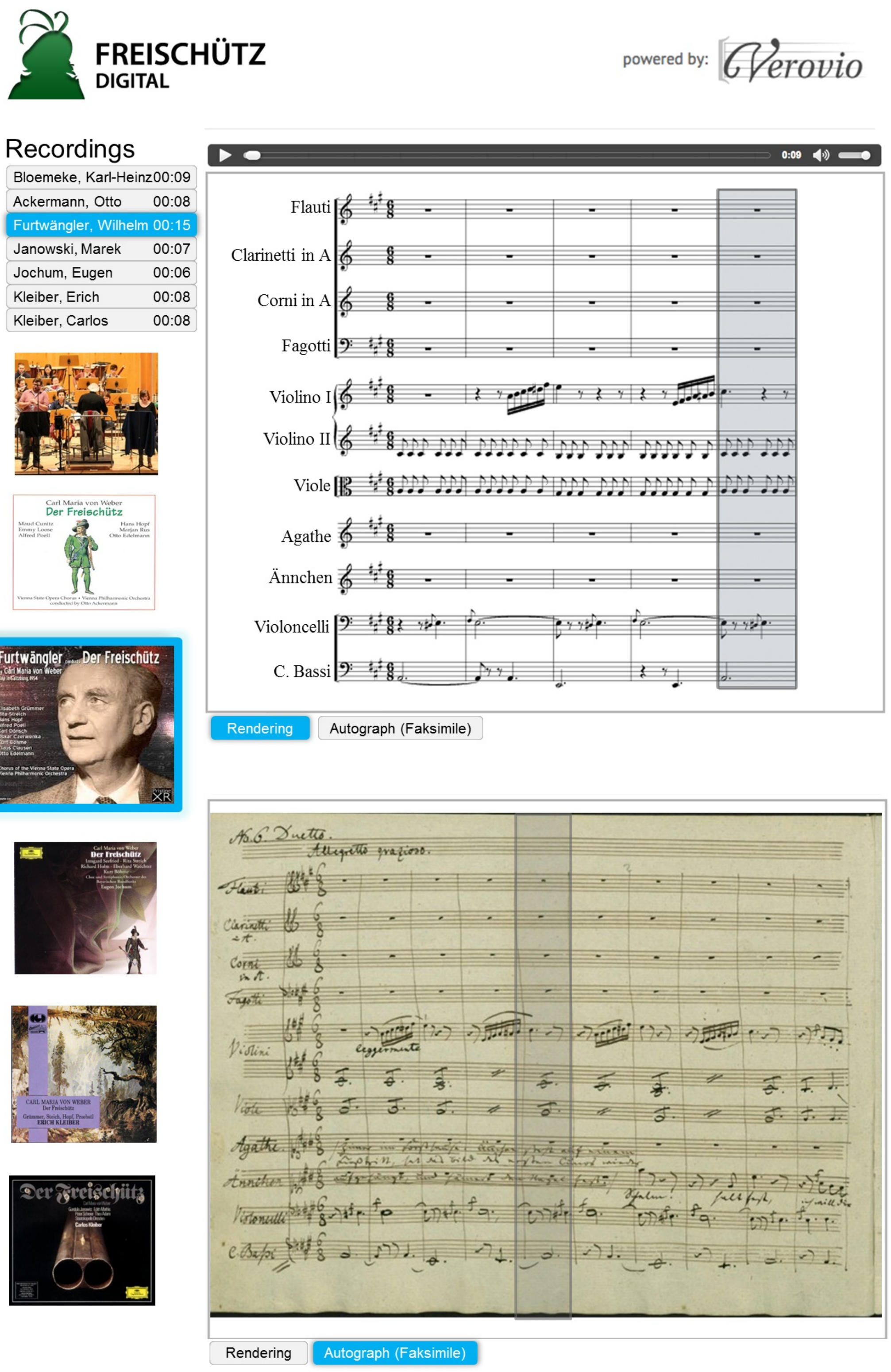
E.1 Demo: Score Follower & Interpretation Switcher

The goal in score following [19] is to retrieve the current score position in a sheet music representation while playing back a corresponding music recording. Similarly, the goal in interpretation switching is to to retrieve the time positions in different recordings that correspond to the current playback position of a given music recording.

Figure E.1 shows a web-based audio player providing score following and interpretation switching functionalities.[2] During the playback of a recording, the corresponding position in the musical score is highlighted (blue rectangle). The user can choose between scans of historical scores

[1]`http://www.freischuetz-digital.de`

[2]This demo has been developed by Benjamin Bohl, Laurent Pugin, and Thomas Prätzlich. It can be accessed at `http://freischuetz-digital.de/demos/syncPlayer/`



Bloemeke, Karl-Heinz	00:09
Ackermann, Otto	00:08
Furtwängler, Wilhelm	00:15
Janowski, Marek	00:07
Jochum, Eugen	00:06
Kleiber, Erich	00:08
Kleiber, Carlos	00:08

Figure E.1. Score Follower & Interpretation Switcher Interface. Top: Score Rendering, Bottom: Autograph.

(facsimiles) or a rendering of a digitally encoded score in MEI.[3] To display the MEI score, we use the library *Verovio*.[4] Furthermore, a list of available audio versions is shown. Simultaneously to the playback of the active audio version (marked in blue on the left side), the corresponding time positions in all available audio versions is displayed. When selecting another audio version, the player switches seamlessly to the chosen version and starts the playback at the position that corresponds to the time position of the previously active audio version. The score following and interpretation switching is realized by providing offline alignments between the different audio versions and the musical score.

E.2 Demo: Single Microphone Switcher

Within the Freischütz Digital project, three numbers (No. 6, 8, and 9) of the opera "Der Freischütz" have been produced at the Erich-Thienhaus-Institute (HfM Detmold). The main purpose for the recording sessions was to produce royalty free audio material that can be used for demonstration and research purposes. Besides a professional stereo mix of the three numbers, the dataset provides the raw multitrack recordings from the individual microphones as well as individual group mixes that emphasize different voices or instrument sections.

Figure E.2 shows the Single Microphone Switcher—an audio player interface that sketches the microphone setup during the recording of No. 6. The interface provides the possibility to listen to the individual microphone recordings. Furthermore, a matrix visualization displays which instruments are currently active (black) or inactive (white). This information has been extracted from a digital score representation that was aligned to the audio recordings.

E.3 Demo: Instrument Equalizer

An instrument equalizer provides the possibility to adjust the volume of an individual instrument in a recording without affecting the volume of the other instruments. Figure E.3 shows a multitrack audio player with an individual track for each voice/instrument of No. 6 from "Der Freischütz". When studying a specific melody line of the violins and the flutes, for example, the instrument equalizer enables a user to raise the volume for these two voices and to lower it for the others, see Figure E.3.

Note that, when recording orchestra music, the microphones for capturing the different voices are usually not shielded from each other. In practice, each microphone not only records sound from its dedicated voice or instrument, but also from all others in the room. This results in

[3]`http://music-encoding.org/`
[4] `http://www.verovio.org/`

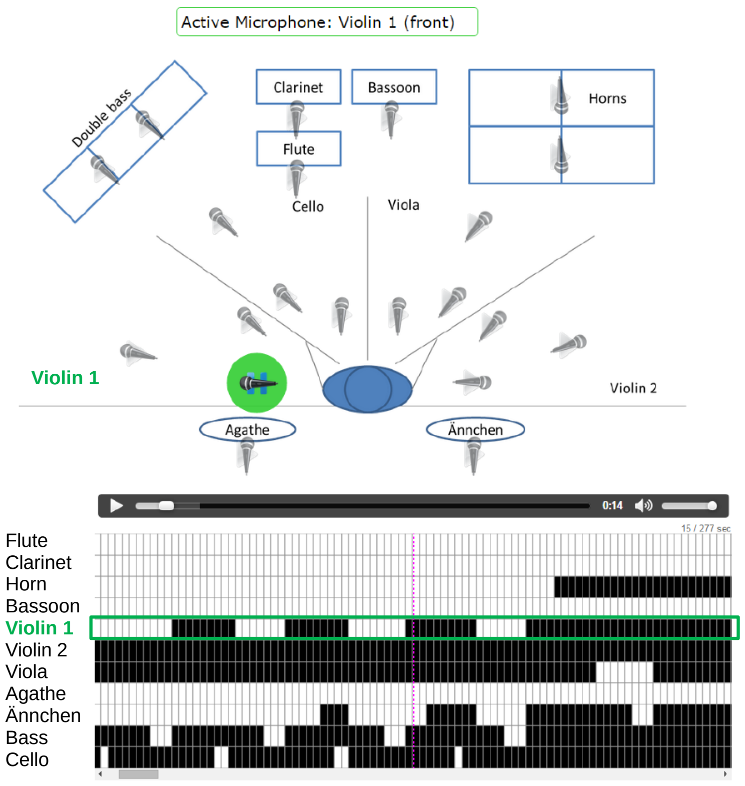


Figure E.2. Single Microphone Switcher.

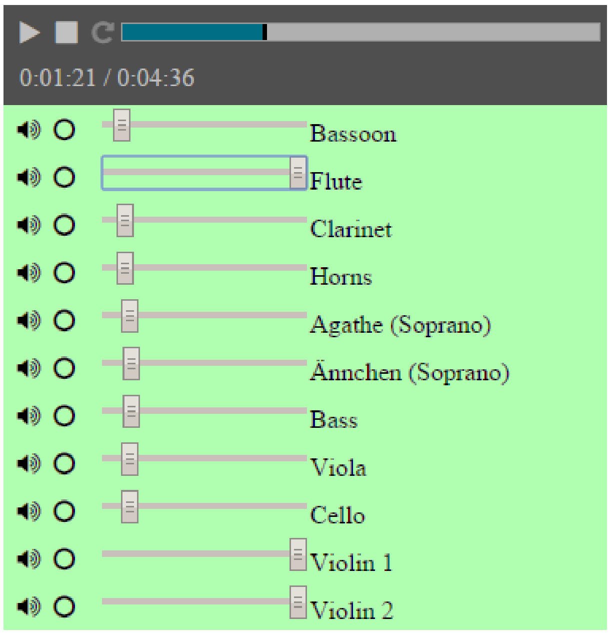


Figure E.3. Instrument Equalizer Interface.

recordings that do not feature isolated signals, but rather mixtures of a predominant voice with all others being audible through what is referred to as interference. In [96], we presented a method for reducing interferences in multitrack recordings. A way to use multitrack recordings suffering from interference in an instrument equalizer is to apply an interference reduction first.

All demos as well as the multitrack recordings discussed in this contribution are available at: `http://freischuetz-digital.de/demos.html`

Bibliography

[1] Jont B. Allen and David A. Berkley. Image method for efficiently simulating small-room acoustics. *Journal of the Acoustic Society of America (JASA)*, 65(4):943–950, Apr 1979.

[2] Vlora Arifi, Michael Clausen, Frank Kurth, and Meinard Müller. Synchronization of music data in score-, MIDI- and PCM-format. *Computing in Musicology*, 13:9–33, 2004.

[3] Andreas Arzt, Sebastian Böck, and Gerhard Widmer. Fast identification of piece and score position via symbolic fingerprinting. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 433–438, Porto, Portugal, 2012.

[4] Andreas Arzt, Harald Frostel, Thassilo Gadermaier, Martin Gasser, Maarten Grachten, and Gerhard Widmer. Artificial intelligence in the Concertgebouw. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2424–2430, Buenos Aires, Argentina, 2015.

[5] Andreas Arzt and Gerhard Widmer. Real-time music tracking using multiple performances as a reference. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 357–363, Málaga, Spain, 2015.

[6] Andreas Arzt, Gerhard Widmer, and Simon Dixon. Adaptive distance normalization for real-time music tracking. In *Proceedings of the European Signal Processing Conference (EUSIPCO)*, pages 2689–2693, Bucharest, Romania, 2012.

[7] Stefan Balke, Lukas Lamprecht, Vlora Arifi-Müller, Thomas Prätzlich, and Meinard Müller. Automatisierte Identifikation von Audioaufnahmen anhand symbolisch codierter musikalischer Themen. In *Proceedings of the Deutsche Jahrestagung für Akustik (DAGA)*, Nuremberg, Germany, 2015.

[8] Juan Pablo Bello and Jeremy Pickens. A robust mid-level representation for harmonic content in music signals. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 304–311, London, UK, 2005.

[9] Christopher M. Bishop. *Pattern recognition and machine learning.* Springer, New York, 2006.

[10] Rachel M. Bittner. *Data driven fundamental frequency estimation.* Phd thesis, New York University, 2017 (in preparation).

[11] Rachel M. Bittner, Justin Salamon, Mike Tierney, Matthias Mauch, Chris Cannam, and Juan Pablo Bello. Medleydb: A multitrack dataset for annotation-intensive MIR research. In *Proceedings of the*

International Society for Music Information Retrieval Conference (ISMIR), pages 155–160, Taipei, Taiwan, 2014.

[12] Sebastian Böck, Florian Krebs, and Gerhard Widmer. A multi-model approach to beat tracking considering heterogeneous music styles. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 603–608, Taipei, Taiwan, 2014.

[13] Benjamin W. Bohl, Thomas Prätzlich, Meinard Müller, and Joachim Veit. Der Gang durch die Domänen. In *Jahrestagung der Digital Humanities im deutschsprachigen Raum: „Von Daten zu Erkenntnissen: Digitale Geisteswissenschaften als Mittler zwischen Information und Interpretation“*, Graz, Austria, 2015.

[14] Julio J. Carabias-Orti, Maximo Cobos, Pedro Vera-Candeas, and Francisco J. Rodriguez-Serrano. Nonnegative signal factorization with learnt instrument models for sound source separation in close-microphone recordings. *EURASIP Journal on Advances in Signal Processing*, 2013.

[15] Andrzej Cichocki, Rafal Zdunek, Anh Huy Phan, and Shun-ichi Amari. *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-Way Data Analysis and Blind Source Separation.* John Wiley and Sons, 2009.

[16] Alice Clifford and Joshua Reiss. Microphone interference reduction in live sound. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, 2011.

[17] Arshia Cont. A coupled duration-focused architecture for real-time music-to-score alignment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(6):974–987, 2010.

[18] David Damm, Christian Fremerey, Verena Thomas, Michael Clausen, Frank Kurth, and Meinard Müller. A digital library framework for heterogeneous music collections: from document acquisition to cross-modal interaction. *International Journal on Digital Libraries: Special Issue on Music Digital Libraries*, 12(2-3):53–71, 2012.

[19] Roger B. Dannenberg and Ning Hu. Polyphonic audio matching for score following and intelligent audio editors. In *Proceedings of the International Computer Music Conference (ICMC)*, pages 27–34, San Francisco, USA, 2003.

[20] Roger B. Dannenberg and Christopher Raphael. Music score alignment and computer accompaniment. *Communications of the ACM, Special Issue: Music Information Retrieval*, 49(8):38–43, 2006.

[21] Alain de Cheveigné and Hideki Kawahara. YIN, a fundamental frequency estimator for speech and music. *Journal of the Acoustical Society of America (JASA)*, 111(4):1917–1930, 2002.

[22] Christian Dittmar, Bernhard Lehner, Thomas Prätzlich, Meinard Müller, and Gerhard Widmer. Cross-version singing voice detection in classical opera recordings. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 618–624, Málaga, Spain, 2015.

[23] Christian Dittmar, Jouni Paulus, Jonathan Driedger, and Meinard Müller. An experimental approach to generalized Wiener filtering in music source separation. In *Proceedings of the European Signal Processing Conference (EUSIPCO)*, pages 1743–1747, Budapest, Hungary, 2016.

[24] Christian Dittmar, Thomas Prätzlich, and Meinard Müller. Towards cross-version singing voice detection. In *Proceedings of the Jahrestagung für Akustik (DAGA)*, Nuremberg, Germany, 2015.

[25] Simon Dixon and Gerhard Widmer. MATCH: A music alignment tool chest. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 492–497, London, UK, 2005.

[26] Jonathan Driedger, Harald Grohganz, Thomas Prätzlich, Sebastian Ewert, and Meinard Müller. Score-informed audio decomposition and applications. In *Proceedings of the ACM International Conference on Multimedia (ACM-MM)*, pages 541–544, Barcelona, Spain, 2013.

[27] Jonathan Driedger and Meinard Müller. TSM Toolbox: MATLAB implementations of time-scale modification algorithms. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, pages 249–256, Erlangen, Germany, 2014.

[28] Jonathan Driedger, Thomas Prätzlich, and Meinard Müller. Let It Bee – Towards NMF-inspired audio mosaicing. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 350–356, Málaga, Spain, 2015.

[29] Zhiyao Duan, Jinyu Han, and Bryan Pardo. Harmonically informed multi-pitch tracking. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 333–338, Kobe, Japan, 2009.

[30] Zhiyao Duan and Bryan Pardo. A state space model for online polyphonic audio-score alignment. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 197–200, Prague, Czech Republic, 2011.

[31] Richard Durbin, Sean R. Eddy, Anders Krogh, and Graeme Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, New York, USA, 1999.

[32] Valentin Emiya, Emmanuel Vincent, Niklas Harlander, and Volker Hohmann. Subjective and Objective Quality Assessment of Audio Source Separation. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(7):2046–2057, 2011.

[33] Hakan Erdogan, John R. Hershey, Shinji Watanabe, and Jonathan Le Roux. Phase-sensitive and recognition-boosted speech separation using deep recurrent neural networks. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 708–712, 2015.

[34] Sebastian Ewert and Meinard Müller. Refinement strategies for music synchronization. In *Proceedings of the International Symposium on Computer Music Modeling and Retrieval (CMMR)*, volume 5493 of *Lecture Notes in Computer Science*, pages 147–165, Copenhagen, Denmark, 2008.

[35] Sebastian Ewert and Meinard Müller. Using score-informed constraints for NMF-based source separation. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 129–132, Kyoto, Japan, 2012.

[36] Sebastian Ewert, Meinard Müller, and Peter Grosche. High resolution audio synchronization using chroma onset features. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 1869–1872, Taipei, Taiwan, 2009.

[37] Sebastian Ewert, Meinard Müller, Verena Konz, Daniel Müllensiefen, and Geraint Wiggins. Towards cross-version harmonic analysis of music. *IEEE Transactions on Multimedia*, 14(3):770–782, 2012.

[38] Sebastian Ewert, Bryan Pardo, Meinard Müller, and Mark Plumbley. Score-informed source separation for musical audio recordings: An overview. *IEEE Signal Processing Magazine*, 31(3):116–124, April 2014.

[39] Sebastian Ewert, Bryan Pardo, Meinard Müller, and Mark D. Plumbley. Score-informed source separation for musical audio recordings: An overview. *IEEE Signal Processing Magazine*, 31(3):116–124, May 2014.

[40] Cédric Févotte. Itakura-saito nonnegative factorizations of the power spectrogram for music signal decomposition. *Machine Audition: Principles, Algorithms and Systems*, pages 266–296, 2010.

[41] Cédric Févotte and Jérôme Idier. Algorithms for nonnegative matrix factorization with the β-divergence. *Neural Computation*, 23(9):2421–2456, 2011.

[42] Derry FitzGerald, Matt Cranitch, and Eugene Coyle. On the use of the beta divergence for musical source separation. In *Proceedings Irish Signals and Systems Conference (ISSC)*, 2008.

[43] Hiromasa Fujihara and Masataka Goto. Lyrics-to-audio alignment and its application. In Meinard Müller, Masataka Goto, and Markus Schedl, editors, *Multimodal Music Processing*, volume 3 of *Dagstuhl Follow-Ups*, pages 23–36. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2012.

[44] Takuya Fujishima. Realtime chord recognition of musical sound: A system using common lisp music. In *Proceedings of the International Computer Music Conference (ICMC)*, pages 464–467, Beijing, 1999.

[45] Dennis Gabor. Theory of communication. *Journal of the Institution of Electrical Engineers (IEE)*, 93(26):429–457, 1946.

[46] Sharon Gannot, David Burshtein, and Ehud Weinstein. Signal enhancement using beamforming and nonstationarity with applications to speech. *IEEE Transactions on Signal Processing*, 49(8):1614–1626, 2001.

[47] Hugo Gävert, Jarmo Hurri, Jaakko Särelä, and Aapo Hyvärinen. The FastICA MATLAB package version 2.5. `http://research.ics.aalto.fi/ica/fastica/`, 2005.

[48] Timothée Gerber, Martin Dutasta, Laurent Girin, and Cédric Févotte. Professionally-produced music separation guided by covers. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 85–90, Porto, Portugal, 2012.

[49] Emilia Gómez. *Tonal Description of Music Audio Signals*. PhD thesis, UPF Barcelona, 2006.

[50] Maarten Grachten, Martin Gasser, Andreas Arzt, and Gerhard Widmer. Automatic alignment of music performances with structural differences. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 607–612, Curitiba, Brazil, 2013.

[51] Andrew Hankinson, Perry Roland, and Ichiro Fujinaga. The music encoding initiative as a document-encoding framework. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 293–298, Miami, Florida, USA, 2011.

[52] Romain Hennequin and François Rigaud. Long-term reverberation modeling for under-determined audio source separation with application to vocal melody extraction. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 206–210, New York, NY, USA, 2016.

[53] André Holzapfel and Yannis Stylianou. Beat tracking using group delay based onset detection. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 653–658, Philadelphia, USA, 2008.

[54] Ning Hu, Roger B. Dannenberg, and George Tzanetakis. Polyphonic audio matching and alignment for music retrieval. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, New Paltz, NY, USA, 2003.

[55] Aapo Hyvärinen. Fast and robust fixed-point algorithms for independent component analysis. *IEEE Transactions on Neural Networks*, 10(3):626–634, 1999.

[56] Nancy M. Ide and Christopher Michael Sperberg-McQueen. The TEI: History, goals, and future. *Computers and the Humanities*, 29(1):5–15, 1995.

[57] Cyril Joder, Slim Essid, and Gaël Richard. A conditional random field framework for robust and scalable audio-to-score matching. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(8):2385–2397, 2011.

[58] Johannes Kepper, Solveig Schreiter, and Joachim Veit. Freischütz analog oder digital–Editionsformen im Spannungsfeld von Wissenschaft und Praxis. In *Editio: Internationales Jahrbuch für Editionswissenschaft*, volume 28, pages 127–150, 2014.

[59] Elias Kokkinis, Alexandros Tsilfidis, Thanos Kostis, and Kostas Karamitas. A new DSP tool for drum leakage suppression. In *Audio Engineering Society Convention*, 2013.

[60] Elias K. Kokkinis and John Mourjopoulos. Unmixing acoustic sources in real reverberant environments for close-microphone applications. *Journal of the Audio Engineering Society*, 58(11):907–922, 2010.

[61] Elias K. Kokkinis, Joshua D. Reiss, and John Mourjopoulos. A Wiener filter approach to microphone leakage reduction in close-microphone applications. *IEEE Transactions on Audio, Speech and Language Processing*, 20(3):767–779, 2012.

[62] Verena Konz and Meinard Müller. A cross-version approach for harmonic analysis of music recordings. In Meinard Müller, Masataka Goto, and Markus Schedl, editors, *Multimodal Music Processing*, volume 3 of *Dagstuhl Follow-Ups*, pages 53–72. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2012.

[63] Frank Kurth and Meinard Müller. Efficient index-based audio matching. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(2):382–395, 2008.

[64] Frank Kurth, Meinard Müller, Christian Fremerey, Yoon ha Chang, and Michael Clausen. Automated synchronization of scanned sheet music with audio recordings. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 261–266, Vienna, Austria, 2007.

[65] 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)*, pages 7480–7484, Florence, Italy, 2014.

[66] Harry Levitt. Noise reduction in hearing aids: a review. *Journal of rehabilitation research and development*, 38(1):111–121, 2001.

[67] Cynthia C. S. Liem, Meinard Müller, Douglas Eck, George Tzanetakis, and Alan Hanjalic. The need for music information retrieval with user-centered and multimodal strategies. In *Proceedings of the International ACM Workshop on Music Information Retrieval with User-centered and Multimodal Strategies (MIRUM)*, pages 1–6, 2011.

[68] Antoine Liutkus and Roland Badeau. Generalized Wiener filtering with fractional power spectrograms. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 266–270, Brisbane, Australia, 2015.

[69] Antoine Liutkus, Jean-Louis Durrieu, Laurent Daudet, and Gaël Richard. An overview of informed audio source separation. In *Proceedings of the International Workshop on Image and Audio Analysis for Multimedia Interactive Services (WIAMIS)*, pages 1–4, Paris, France, 2013.

[70] Beth Logan. Mel frequency cepstral coefficients for music modeling. In *Proceedings of the International Symposium on Music Information Retrieval (ISMIR)*, Plymouth, Massachusetts, USA, 2000.

[71] Hanna Lukashevich. Towards quantitative measures of evaluating song segmentation. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 375–380, Philadelphia, USA, 2008.

[72] Robert Macrae and Simon Dixon. Accurate real-time windowed time warping. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 423–428, Utrecht, The Netherlands, 2010.

[73] Matthias Mauch, Hiromasa Fujihara, Kazuyoshii Yoshii, and Masataka Goto. Timbre and melody features for the recognition of vocal activity and instrumental solos in polyphonic music. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, pages 233–238, Miami, Florida, USA, 2011.

[74] Nicola Montecchio, Emanuele Di Buccio, and Nicola Orio. An efficient identification methodology for improved access to music heritage collections. *Journal of Multimedia*, 7(2):145–158, 2012.

[75] Nicola Montecchio and Arshia Cont. A unified approach to real time audio-to-score and audio-to-audio alignment using sequential Montecarlo inference techniques. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 193–196, Prague, Czech Republic, 2011.

[76] Meinard Müller. *Information Retrieval for Music and Motion.* Springer Verlag, 2007.

[77] Meinard Müller. *Fundamentals of Music Processing.* Springer Verlag, 2015.

[78] Meinard Müller and Sebastian Ewert. Chroma Toolbox: MATLAB implementations for extracting variants of chroma-based audio features. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 215–220, Miami, Florida, USA, 2011.

[79] Meinard Müller, Masataka Goto, and Markus Schedl, editors. *Multimodal Music Processing*, volume 3 of *Dagstuhl Follow-Ups*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany, 2012.

[80] Meinard Müller, Nanzhu Jiang, and Harald Grohganz. SM Toolbox: MATLAB implementations for computing and enhancing similarity matrices. In *Proceedings of the Audio Engineering Society (AES) Conference on Semantic Audio*, London, UK, 2014.

[81] Meinard Müller and Verena Konz. Automatisierte Methoden zur Unterstützung der Interpretationsforschung. In Heinz von Loesch and Stefan Weinzierl, editors, *Gemessene Interpretation*, volume 4 of *Klang und Begriff*, pages 1–12. Schott Verlag, 2011.

[82] Meinard Müller, Verena Konz, Andi Scharfstein, Sebastian Ewert, and Michael Clausen. Towards automated extraction of tempo parameters from expressive music recordings. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 69–74, Kobe, Japan, 2009.

[83] Meinard Müller, Frank Kurth, and Michael Clausen. Audio matching via chroma-based statistical features. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 288–295, London, UK, 2005.

[84] Meinard Müller, Frank Kurth, and Michael Clausen. Chroma-based statistical audio features for audio matching. In *Proceedings of the Workshop on Applications of Signal Processing (WASPAA)*, pages 275–278, New Paltz, New York, USA, 2005.

[85] Meinard Müller, Frank Kurth, and Tido Röder. Towards an efficient algorithm for automatic score-to-audio synchronization. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 365–372, Barcelona, Spain, 2004.

[86] Meinard Müller, Henning Mattes, and Frank Kurth. An efficient multiscale approach to audio synchronization. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 192–197, Victoria, Canada, 2006.

[87] Meinard Müller, Thomas Prätzlich, Benjamin Bohl, and Joachim Veit. Freischütz Digital: a multimodal scenario for informed music processing. In *Proceedings of the International Workshop on Image and Audio Analysis for Multimedia Interactive Services (WIAMIS)*, pages 1–4, Paris, France, 2013.

[88] Meinard Müller, Thomas Prätzlich, and Christian Dittmar. Freischütz Digital: When computer science meets musicology. In Kristina Richts and Peter Stadler, editors, *„Ei, dem alten Herrn zoll' ich Achtung gern'“: Festschrift für Joachim Veit zum 60. Geburtstag*, pages 551–574. Allitera Verlag, München, 2016.

[89] Meinard Müller, Thomas Prätzlich, and Jonathan Driedger. A cross-version approach for stabilizing tempo-based novelty detection. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 427–432, Porto, Portugal, 2012.

[90] Alan V. Oppenheim and Ronald W. Schafer. *Discrete-Time Signal Processing.* Pearson Higher Education, 2010.

[91] Alan V. Oppenheim, Alan S. Willsky, and Hamid Nawab. *Signals and Systems.* Prentice Hall, 1996.

[92] Nicola Orio, Serge Lemouton, and Diemo Schwarz. Score following: State of the art and new developments. In *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*, pages 36–41, Montreal, Canada, 2003.

[93] Yorgos Patsis and Werner Verhelst. A speech/music/silence/garbage/classifier for searching and indexing broadcast news material. In *International Conference on Database and Expert Systems Application (DEXA)*, pages 585–589, Turin, Italy, 2008.

[94] Jouni Paulus, Meinard Müller, and Anssi P. Klapuri. Audio-based music structure analysis. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 625–636, Utrecht, The Netherlands, 2010.

[95] Geoffroy Peeters. Deriving musical structure from signal analysis for music audio summary generation: "sequence" and "state" approach. In *Computer Music Modeling and Retrieval*, volume 2771 of *Lecture Notes in Computer Science*, pages 143–166. Springer Berlin / Heidelberg, 2004.

[96] Thomas Prätzlich, Rachel Bittner, Antoine Liutkus, and Meinard Müller. Kernel additive modeling for interference reduction in multi-channel music recordings. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Brisbane, Australia, 2015.

[97] Thomas Prätzlich, Rachel M. Bittner, Antoine Liutkus, Juan P. Bello, and Meinard Müller. Interference reduction for multitrack music recordings. *IEEE Transactions on Audio, Speech, and Language Processing*, 2016, in preparation.

[98] Thomas Prätzlich, Jonathan Driedger, and Meinard Müller. Memory-restricted multiscale dynamic time warping. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 569–573, Shanghai, China, 2016.

[99] Thomas Prätzlich and Meinard Müller. Freischütz Digital: a case study for reference-based audio segmentation of operas. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 589–594, Curitiba, Brazil, 2013.

[100] Thomas Prätzlich and Meinard Müller. Frame-level audio segmentation for abridged musical works. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 307–312, Taipei, Taiwan, 2014.

[101] Thomas Prätzlich and Meinard Müller. Triple-based analysis of music alignments without the need of ground-truth annotations. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 266–270, Shanghai, China, 2016.

[102] Thomas Prätzlich, Meinard Müller, Benjamin W. Bohl, and Joachim Veit. Freischütz Digital: Demos of audio-related contributions. In *Demos and Late Breaking News of the International Society for Music Information Retrieval Conference (ISMIR)*, Málaga, Spain, 2015.

[103] Laurent Pugin, Johannes Kepper, Perry Roland, Maja Hartwig, and Andrew Hankinson. Separating presentation and content in MEI. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 505–510, Porto, Portugal, 2012.

[104] Lawrence Rabiner and Bing-Hwang Juang. *Fundamentals of Speech Recognition.* Prentice Hall Signal Processing Series, 1993.

[105] Mathieu Ramona, Gäel Richard, and Bertrand David. Vocal detection in music with support vector machines. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 1885–1888, Las Vegas, Nevada, USA, 2008.

[106] Christopher Raphael. A probabilistic expert system for automatic musical accompaniment. *Journal of Computational and Graphical Statistics*, 10(3):487–512, 2001.

[107] Christopher Raphael. A hybrid graphical model for aligning polyphonic audio with musical scores. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 387–394, Barcelona, Spain, 2004.

[108] Hannah Ilea Robertson. *Testing a new tool for alignment of musical recordings.* Master's thesis, McGill University, Montreal, Canada, 2013.

[109] Perry Roland and Christiane Siegert. Process-oriented notation in MEI. *Die Tonkunst. Magazin für klassische Musik und Musikwissenschaft*, 5(3):305–309, 2011.

[110] Daniel Röwenstrunk, Thomas Prätzlich, Thomas Betzwieser, Meinard Müller, Gerd Szwillus, and Joachim Veit. Das Gesamtkunstwerk Oper aus Datensicht – Aspekte des Umgangs mit einer heterogenen Datenlage im BMBF-Projekt "Freischütz Digital". *Datenbank-Spektrum*, 15(1):65–72, 2015.

[111] Stan Salvador and Philip Chan. FastDTW: Toward accurate dynamic time warping in linear time and space. In *Proceedings of the KDD Workshop on Mining Temporal and Sequential Data*, 2004.

[112] Craig Stuart Sapp. Hybrid numeric/rank similarity metrics. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 501–506, Philadelphia, USA, 2008.

[113] John Saunders. Real-time discrimination of broadcast speech/music. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 993–996, 1996.

[114] Solveig Schreiter. *Friedrich Kind & Carl Maria von Weber - Der Freischütz. Kritische Textbuch-Edition.* Allitera Verlag, 2007.

[115] Joan Serrà, Emilia 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*, 16:1138–1151, 2008.

[116] Alexander Sheh and Daniel P.W. Ellis. Chord segmentation and recognition using EM-trained hidden Markov models. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 185–191, Baltimore, Maryland, USA, 2003.

[117] Reinhard Sonnleitner, Bernhard Niedermayer, Gerhard Widmer, and Jan Schlüter. A simple and effective spectral feature for speech detection in mixed audio signals. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, 2012.

[118] Nathan Souviraà-Labastie, Anaik Olivero, Emmanuel Vincent, and Frédéric Bimbot. Multi-channel audio source separation using multiple deformed references. *IEEE Transactions on Audio, Speech, and Language Processing*, 23(11):1775–1787, 2015.

[119] Peter Stavroulakis. *Interference Analysis and Reduction for Wireless Systems*. Artech House mobile communications series. Artech House, 2003.

[120] Sebastian Stober, Thomas Prätzlich, and Meinard Müller. Brain Beats: Tempo extraction from EEG data. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 276–282, New York City, NY, USA, 2016.

[121] TJ Tsai, Thomas Prätzlich, and Meinard Müller. Known-artist live song id: A hashprint approach. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 427–433, New York City, NY, USA, 2016.

[122] Robert J. Turetsky and Daniel P.W. Ellis. Ground-truth transcriptions of real music from force-aligned MIDI syntheses. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 135–141, Baltimore, Maryland, USA, 2003.

[123] George Tzanetakis and Perry Cook. Multifeature audio segmentation for browsing and annotation. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 103–106, New Paltz, NY, USA, 1999.

[124] Christian Uhle and Josh Reiss. Determined source separation for microphone recordings using IIR filters. In *Proceedings of the Audio Engineering Society Convention(AES)*, 2010.

[125] Emmanuel Vincent. Improved perceptual metrics for the evaluation of audio source separation. In *Proceedings of the International Conference on Latent Variable Analysis and Signal Separation (LVA/ICA)*, pages 430–437, Tel Aviv, Israel, 2012.

[126] Emmanuel Vincent, Nancy Bertin, Rémi Gribonval, and Frédéric Bimbot. From blind to guided audio source separation: How models and side information can improve the separation of sound. *IEEE Signal Processing Magazine*, 31(3):107–115, 2014.

[127] Emmanuel Vincent, Rémi Gribonval, and Cédric Févotte. Performance measurement in blind audio source separation. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(4):1462–1469, 2006.

[128] Siying Wang, Sebastian Ewert, and Simon Dixon. Robust joint alignment of multiple versions of a piece of music. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 83–88, Taipei, Taiwan, 2014.

[129] John Warrack. *Carl Maria von Weber*. Cambridge University Press, 1976.

[130] Christof Weiß, Vlora Arifi-Müller, Thomas Prätzlich, Rainer Kleinertz, and Meinard Müller. Analyzing measure annotations for Western classical music recordings. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 517–523, New York City, NY, USA, 2016.

[131] Gerhard Widmer. Using AI and machine learning to study expressive music performance: project survey and first report. *AI Communications*, 14(3):149–162, 2001.

[132] Gerhard Widmer. Machine discoveries: A few simple, robust local expression principles. *Journal of New Music Research*, 31(1):37–50, 2003.

[133] Gerhard Widmer, Simon Dixon, Werner Goebl, Elias Pampalk, and Asmir Tobudic. In search of the Horowitz factor. *AI Magazine*, 24(3):111–130, 2003.

[134] John Woodruff, Bryan Pardo, and Roger B. Dannenberg. Remixing stereo music with score-informed source separation. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 314–319, Victoria, Canada, 2006.

[135] José R. Zapata, Matthew E. P. Davies, and Emilia Gómez. Multi-feature beat tracking. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(4):816–825, 2014.