Processing Music Signals Using Audio Decomposition Techniques

Verarbeitung von Musiksignalen unter Verwendung von Zerlegungstechniken für Audiodaten

Dissertation

Der Technischen Fakultät

der Friedrich-Alexander-Universität Erlangen-Nürnberg

zur

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

vorgelegt von

Jonathan Driedger

aus

Annweiler am Trifels

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

Tag der mündlichen Prüfung:	9. Mai 2016
Vorsitzende des Promotionsorgans:	Prof. Dr. Peter Greil
1. Gutachter:	Prof. Dr. Meinard Müller
2. Gutachter:	Prof. Dr. Michael Clausen

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, 29.02.2016

Jonathan Driedger

Abstract

Music signals are complex. When musicians play together, their instruments' sounds superimpose and form a single complex sound mixture. Furthermore, even the sound of a single instrument may already comprise sound components of harmonic, percussive, noise-like, and transient nature, among others. The complexity of music signal processing tasks such as *time-scale modification*—the task of stretching or compressing the duration of a music signal—or *music source separation*—the task of separating a music recording into signals that correspond to the individual instruments—is therefore often directly derived from the complexity of music signals themselves.

In this thesis, our goal is to explore novel ways of approaching music signal processing tasks. One of our core ideas is to reduce a task's complexity by decomposing a given music signal into a set of two or more *mid-level components* and then process these components individually. Depending on the audio decomposition technique, a mid-level component may reflect certain aspects of the music signal, such as its harmonic or percussive sounds. This explicit interpretation often allows us to apply more specialized methods for processing the mid-level components. In a last step, the processed component signals are recombined to form a global result.

As part of our contributions, we propose various novel audio decomposition techniques for splitting a music signal into mid-level components. For example, we present a method for decomposing a signal into three components that contain the signal's harmonic-, percussive-, and noise-like sounds, respectively. Furthermore, we apply the general strategy described previously to approach different tasks in the fields of digital signal processing and music information retrieval. In particular, we propose novel procedures for time-scale modification, singing voice separation, vibrato analysis, and audio mosaicing. Built upon these methods, we additionally present various prototype user interfaces and tools for analyzing, modifying, editing, and synthesizing music signals.

Zusammenfassung

Musiksignale sind typischerweise hoch komplexe Klanggemische, die sich aus der Überlagerung von einzelnen, miteinander interagierenden Instrumentalstimmen ergeben. Sogar der Klang eines einzelnen Instruments kann sich bereits aus vielen unterschiedlichen Klangkomponenten zusammensetzen, zum Beispiel aus harmonischen, perkussiven, rauschartigen und transienten Anteilen. Diese Komplexität macht die automatisierte Verarbeitung von Musiksignalen, etwa für Aufgabenstellungen wie *Time-Stretching* (Stauchung oder Streckung der Länge einer Musikaufnahme) oder *Quellentrennung* (Zerlegung einer Musikaufnahme in Anteile die zu den einzelnen Instrumentalstimmen korrespondieren), zu einem äußerst schwierigen Problem.

Eine Kernidee dieser Arbeit besteht darin, die Verarbeitung von komplexen Musikaufnahmen zu erleichtern, indem man die Aufnahme zunächst in zwei oder mehrere *mid-level* Klangkomponenten zerlegt und diese Teilsignale anschließend separat weiterverarbeitet. Da die extrahierten Komponenten gewisse, von der Zerlegungstechnik vorgegebene Eigenschaften haben, lassen sich für deren Verarbeitung oft spezialisierte Techniken verwenden. In einem letzten Schritt werden die verarbeiteten Komponenten wieder zusammengeführt.

In dieser Arbeit stellen wir zunächst verschiedene neuartige Zerlegungstechniken für Audiodaten vor. Eines dieser Verfahren zerlegt beispielsweise eine Musikaufnahme in drei mid-level Komponenten, die zu den harmonischen, den perkussiven und den rauschartigen Klanganteilen der Aufnahme korrespondieren. Diese und weitere Zerlegungstechniken werden dann verwendet um neuartige Verfahren für Aufgabenstellungen aus den Bereichen der digitalen Audiosignalverarbeitung und des *Music Information Retrievals* zu entwickeln, beispielsweise zum Time-Stretching, zur Abtrennung der Singstimme aus polyphonen Musikaufnahmen, zur Analyse von Vibrato und für das *Audio Mosaicing*. Weiterhin stellen wir mehrere prototypische Benutzerschnittstellen und Werkzeuge zur Analyse, Modifikation, Editierung und Synthese von Musikaufnahmen vor.

Contents

Abstract **iii**

Zusammenfassung **v**

1 Introduction **3**

1.1 Structure of this Thesis 5
1.2 Contributions 8
1.3 Main Publications 8
1.4 Additional Publications 9
1.5 Acknowledgments 10

2 Fundamentals of Music Signal Processing **13**

2.1 Music Representations 14
2.2 Fourier Transform and Spectrogram Representation 18
2.3 Pitch- and Chroma-Based Audio Features 20
2.4 Audio Features for Melody Tracking 23
2.5 Novelty-Based Audio Features 27

3 Extensions to Harmonic-Percussive Separation of Music Signals **29**

3.1 Basic Procedure 32
3.2 Separation Factor 33
3.3 Harmonic-Percussive-Residual (HPR) Separation 35
3.4 Evaluation 40
3.5 Applications of HPR Separation 42
3.6 Conclusions and Further Notes 44

4 Contributions to Time-Scale Modification of Music Signals **47**

4.1 Fundamentals on Time-Scale Modification (TSM) 48
4.2 TSM Based on Harmonic-Percussive Separation (HP-TSM) 65
4.3 TSM Based on Cascading (Cascade-TSM) 70
4.4 Applications of TSM 77
4.5 Conclusions and Further Notes 83

5 Cascaded Music Signal Decomposition for Singing Voice Extraction 85
5.1 Characteristics of Singing Voice and Accompaniment 87
5.2 Cascaded Audio Decomposition 91
5.3 Evaluation 97
5.4 Conclusions and Further Notes 102

6 Informed Music Signal Parametrization 103
6.1 Score-Informed Parametrization of Note Events 104
6.2 Template-Informed Parametrization of Vibrato 113
6.3 Conclusions and Further Notes 123

7 NMF-Inspired Audio Mosaicing for Music Signals 127
7.1 Basic NMF-Inspired Audio Mosaicing 129
7.2 Learning Sparse Diagonal Activations 132
7.3 Experiments and Examples 137
7.4 Conclusions and Further Notes 141

8 Summary and Future Work 143

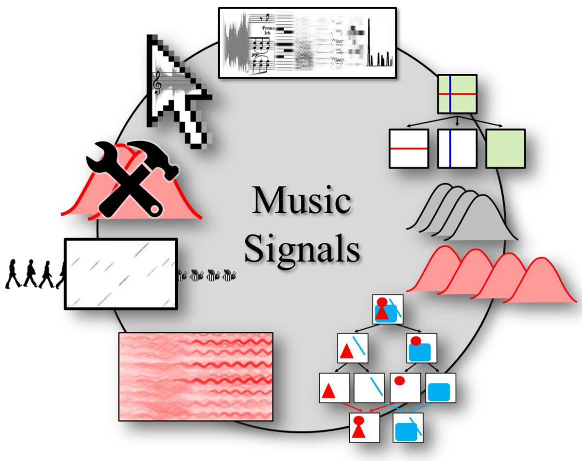
A TSM Toolbox 147
A.1 Code Example 148
A.2 Additional Functionality and Content 151

B User Interfaces for Music Signal Analysis and Editing 153
B.1 F0-Annotation Tool 154
B.2 Source Separation Audio Player 156
B.3 Voice Separation Audio Player 157

Bibliography 159

Chapter 1

Introduction



The digital revolution has strongly influenced the way we consume and interact with music. Digitized music recordings make it possible to enjoy music everywhere and anytime. We can store large music collections on our mobile playing devices or access them via streaming services, while music production software turns our home computers into fully functional recording studios.

The step from analog to digital music has also opened up novel ways of music research, leading to the advent of new scientific disciplines. While early computer-based studies on music were mainly focused on music in symbolic form—such as MIDI files—the increasing availability of digital music recordings as well as the leap in computational power made it possible to delve deeper into *music signal processing*—digital signal processing with a focus on music recordings [146]. In the year 2000, the International Society for Music Information Retrieval (ISMIR) was formed, a community dedicated to the research of computer-based analysis, processing, and retrieval topics in the context of music [32]. Since then, this field has blossomed into a variety of sub-fields, each dealing with a wide range of research questions.

An example for such a research direction is *time-scale modification* (TSM) [29] of music signals. Here, the goal is to stretch or compress the duration of a music signal without altering the signal's pitch. Methods for TSM are frequently used by DJs and music producers, for example to synchronize the tempi of two recordings in order to create a music remix. Since music signals may be very complex—comprising harmonic, percussive, and transient components, among others—preserving the entirety of a signal's qualities in a time-scale modified version is a non-trivial task. Stretching the harmonic sound of a violin usually requires a different TSM approach than modifying a recording of percussive castanets. To this end, there is no single TSM procedure that can cope with all kinds of music signals equally well.

Another example is *music source separation* [65]. Music recordings are usually complex superpositions of sound sources. For example, a recording of a rock band may consist of singing voice, guitar, bass, and drums. When the musicians play together, the sounds of their instruments superimpose and form a single complex music signal. Given this signal, one goal of source separation is to derive source signals that correspond to the individual instruments such that one obtains a singing voice, a guitar, a bass, and a drum signal. Ideally, each of the source signals then sounds as if the respective instrument were being played isolatedly. In general, source separation can be considered an extremely difficult task. This is especially the case when singing voice is involved, due to its highly complex nature. Therefore, music source separation is a problem far from being solved.

As indicated by the two examples described above, the complexity of music processing tasks often stems from the complexity of the music signals themselves. A common method to approach this kind of complexity is to apply time-frequency analysis tools, such as the *short-time Fourier transform* [158] or the *wavelet transform* [106]. These tools can be interpreted as the decomposition of a given music signal into a set of component signals, each of them less complex than the music signal as a whole. For example, in the case of Fourier analysis, the components are sinusoidal signals of different frequency, amplitude, and phase. However, despite the individual component signals' reduced complexity, it is seldom obvious how to utilize this kind of decomposition to approach a given processing task, since the components typically lack a musically meaningful interpretation.

Instead, a novel approach that we explore in this thesis is to use *audio decomposition techniques* in order to split a given music signal into a small set of *mid-level* components with an explicit semantic interpretation. The core idea is that the task at hand is easier to perform on each of the mid-level components than on the original signal. This is the case since a mid-level component's explicit interpretation may serve as prior knowledge for the subsequent signal processing step—thus allowing the use of more specialized processing methods. Consider TSM as an example. As described above, stretching harmonic sound events requires conceptually different algorithmic approaches than stretching percussive ones—rendering the modification of a music signal consisting of both harmonic and percussive sound components a difficult task. However, decomposing the music signal into two mid-level components—a harmonic component and a percussive component—reduces the problem's complexity. Knowing a priori that the harmonic component contains exclusively harmonic sounds allows us to apply a specialized TSM procedure that preserves the component's characteristics. Similarly, the percussive component can be modified with a different specialized TSM procedure. In the last step, the two modified component signals are combined to form a global result.

Therefore, the main questions that we approach in this thesis are: Given a signal processing task, is it possible to decompose a music signal into two or more mid-level component signals such that the desired task is easier to perform on the components than on the original? What are suitable audio decomposition techniques for the given task? How can the processed component signals be combined to form a global result? And, does the decomposition process itself yield deeper insights into the task at hand?

1.1 Structure of this Thesis

This thesis is structured in six main chapters.

We start in Chapter 2 by reviewing fundamentals of music signal processing. Following [146], we first introduce various music representations. In particular, we embed music signals, the core subject of this thesis, in a framework that allows us to define subsequent processing techniques in a mathematically rigorous way. We proceed with reviewing the famous short-time Fourier transform—one of the most important tools in music signal processing. Furthermore, we discuss various feature representations for music signals such as log-frequency spectrograms, chromagrams, and novelty curves.

In Chapter 3 we introduce novel audio decomposition techniques. In recent years, methods to decompose an audio signal into a harmonic and a percussive component have received a lot of interest and are frequently applied as a processing step in a variety of scenarios. One problem is that the computed components are almost never of purely harmonic or percussive nature, but also contain noise-like sounds that are neither clearly harmonic nor percussive. Furthermore, depending on the parameter settings, one often can observe a leakage of harmonic sounds into the percussive component and vice versa. In this chapter, we present two extensions to a state-of-the-art harmonic-percussive separation procedure to target these problems. First, we introduce a *separation factor* parameter into the decomposition process that allows for tightening separation results and forcing the components to be clearly harmonic or percussive. As a second contribution, inspired by the classical sines+transients+noise (STN) audio model [133], this novel concept is exploited to add a third *residual* component to the decomposition which captures the sounds that lie in between the clearly harmonic and percussive sounds of the audio signal.

Chapter 4 is dedicated to time-scale modification (TSM)—the task of stretching or compressing the time-scale of an audio signal without changing its pitch. In digital music production, TSM has become an indispensable tool, which is nowadays integrated in a wide range of music production software. A major problem in TSM of music signals is that many approaches degrade the perceptual quality of percussive transients. To prevent this degradation, some TSM procedures try to explicitly identify transients in the input signal and to handle them in a special way. However, such approaches are problematic for two reasons. First, errors in the transient detection have an immediate influence on the final TSM result and, secondly, a perceptually transparent preservation of transients is by far not a trivial task. However, there exist methods that can preserve the transient nature of signals without an explicit transient detection, but only in very limited scenarios. Our contributions in this chapter are twofold. First, we aim to foster a better understanding of the capabilities and limitations of several conceptually different TSM approaches by reviewing fundamental TSM methods. Second, we propose two novel TSM procedures that improve over the fundamental methods by combining different TSM approaches with decomposition techniques as introduced in Chapter 3. This strategy allows us to generalize the aforementioned procedures with implicit transient handling to arbitrary TSM scenarios.

Next, in Chapter 5, we deal with the task of singing voice extraction from music recordings. This source separation problem has received increasing research interest in recent years. Many proposed decomposition techniques are based on one of the following two strategies. The first approach is to directly decompose a given music recording into one component for the singing voice and one for the accompaniment by exploiting knowledge about specific characteristics of singing voice. Procedures following the second approach disassemble the recording into a large set of fine-grained components, which are classified as either singing voice or accompaniment, and are reassembled afterwards to yield the desired source estimates. In this chapter, we propose a novel approach that combines the strengths of both strategies. We first apply different audio decomposition techniques in a cascaded fashion to disassemble the music recording into a set of mid-level components. This decomposition is fine enough to model various characteristics of singing voice, but coarse enough to keep an explicit semantic meaning of the components. These properties allow us to directly reassemble the singing voice and the accompaniment from the components. Our objective and subjective evaluations show that this strategy can compete with state-of-the-art singing voice separation techniques and yields perceptually appealing results.

In Chapter 6, we explore novel ways of parameterizing signal components in music recordings. In this chapter's first part, we introduce a framework for parameterizing a music recording with respect to note events as they are specified in a given musical score of that recording. This parametrization is then used to decompose the recording into note-wise audio events which serve as elementary building blocks. In particular, we introduce an interface that employs the additional score information to provide a natural way for a user to interact with these audio events. By simply selecting arbitrary note groups within the score, a user can access, modify, or analyze corresponding events in the music recording. In this way, our framework not only opens up new ways for audio editing applications, but also serves as a valuable tool for evaluating and better understanding the results of source separation procedures.

In the second part of Chapter 6, we aim to parameterize vibrato—a musical voice's periodic frequency modulation—in complex music signals. A common strategy to analyze vibrato is to proceed in a two-step fashion. First, a fundamental frequency (F0) trajectory for the musical voice that is likely to exhibit vibrato is estimated. In a second step, the trajectory is then analyzed with respect to periodic frequency modulations. As a major drawback, however, such a method cannot recover from errors made in the inherently difficult first step, which severely limits the performance during the second step. To this end, we present a novel vibrato analysis approach that avoids the first error-prone F0-estimation step. Our core idea is to perform the analysis directly on a signal's spectrogram representation where vibrato is evident in the form of characteristic spectro-temporal patterns. We detect and parameterize these patterns by locally comparing the spectrogram with a predefined set of vibrato templates. Our systematic experiments indicate that this approach is more robust than F0-based strategies.

Finally, in Chapter 7, we discuss our contributions to the field of audio mosaicing. Given a target and a source recording, the goal of audio mosaicing is to generate a mosaic recording that conveys musical aspects—like melody and rhythm—of the target, using sound components taken from the source. In this chapter, we propose a novel approach for automatically generating audio mosaics with the objective to preserve the source's timbre in the mosaic. Inspired by audio decomposition techniques based on non-negative matrix factorization (NMF), our idea is to use update rules to learn an activation matrix that, when multiplied with the spectrogram of the source recording, resembles the spectrogram of the target recording. However, when applying the original NMF procedure, the resulting mosaic may not adequately reflect the source's timbre. As our main technical contribution in this chapter, we propose an extended set of update rules for the iterative learning procedure that supports the development of sparse diagonal structures in the activation matrix. The learning procedure therefore tries to balance two conflicting goals: On the one hand, the sparse diagonal structures better retain the source's timbral characteristics in the resulting mosaic. On the other hand, the procedure still aims to approximate the target's spectrogram as well as possible.

1.2 Contributions

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

- Two novel techniques for decomposing an audio recording into mid-level components that correspond to harmonic sounds, percussive sounds, and sounds that are of neither harmonic nor percussive nature (Section 3.3.1 and Section 3.3.3).
- A detailed review of time-scale modification of music signals, including illustrative descriptions of the most important fundamental TSM procedures (Section 4.1).
- Two novel TSM procedures that are based on the combination of different TSM approaches with harmonic-percussive audio decomposition techniques (Section 4.2 and Section 4.3).
- A novel approach for singing voice extraction from music signals based on a cascade of different audio decomposition techniques (Chapter 5).
- An extension to a score-informed audio parametrization technique that allows for the decomposition of music recordings into note-wise audio events with applications to audio editing (Section 6.1).
- A template-based method for the detection and parametrization of vibrato in polyphonic music recordings (Section 6.2).
- A novel set of update rules for non-negative matrix factorization (NMF) that supports the development of sparse diagonal structures in activation matrices. These structures have shown to be beneficial in the context of NMF-based audio mosaicing (Chapter 7).
- The *TSM toolbox*, a MATLAB toolbox that includes implementations of various TSM procedures, example applications, and a dataset for the evaluation of TSM methods (Appendix A).
- Three different MATLAB user interfaces. One for the interactive estimation of fundamental frequency trajectories in complex music recordings, one for the note-wise decomposition, analysis and editing of piano recordings, and one for the editing of singing voice in complex music recordings (Appendix B.1, Appendix B.2, and Appendix B.3).

1.3 Main Publications

The main contributions of this thesis are based on the following publications, which appeared in conference proceedings and journal articles in the field of audio signal processing and music information retrieval.

[46] Jonathan Driedger, Meinard Müller, and Sascha Disch. Extending harmonic-percussive separation of audio signals. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 611–616, Taipei, Taiwan, October 2014.

[42] Jonathan Driedger and Meinard Müller. Harmonisch-Perkussiv-Rest Zerlegung von Musiksignalen. In *Proceedings of the Deutsche Jahrestagung für Akustik (DAGA)*, pages 1421–1424, Nuremberg, Germany, 2015.

[44] Jonathan Driedger and Meinard Müller. A review on time-scale modification of music signals. *Applied Sciences*, 6(2):57–82, February 2016.

[48] Jonathan Driedger, Meinard Müller, and Sebastian Ewert. Improving time-scale modification of music signals using harmonic-percussive separation. *IEEE Signal Processing Letters*, 21(1):105–109, 2014.

[40] 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.

[41] Jonathan Driedger and Meinard Müller. Extracting singing voice from music recordings by cascading audio decomposition techniques. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 126–130, Brisbane, Australia, 2015.

[37] 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.

[50] 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, Malaga, Spain, 2015.

[43] Jonathan Driedger and Meinard Müller. Verfahren zur Schätzung der Grundfrequenzverläufe von Melodiestimmen in mehrstimmigen Musikaufnahmen. In Wolfgang Auhagen, Claudia Bullerjahn, and Richard von Georgi, editors, *Musikpsychologie – Anwendungsorientierte Forschung*, volume 25 of *Jahrbuch Musikpsychologie*, pages 55–71. Hogrefe-Verlag, 2015.

1.4 Additional Publications

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

[72] Richard Füg, Andreas Niedermeier, Jonathan Driedger, Sascha Disch, and Meinard Müller. Harmonic-percussive-residual sound separation using the structure tensor on spectrograms. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Shanghai, China, 2016.

[26] Christian Dittmar, Jonathan Driedger, and Meinard Müller. A separate and restore approach to score-informed music decomposition. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, New Paltz, NY, USA, October 2015.

[148] Meinard Müller, Jonathan Driedger, and Sebastian Ewert. Notentext-informierte Quellentrennung für Musiksignale. In *Proceedings of 43th GI Jahrestagung*, pages 2928–2942, Koblenz, Germany, 2013.

[154] 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.

[147] Meinard Müller and Jonathan Driedger. Data-driven sound track generation. In Meinard Müller, Masataka Goto, and Markus Schedl, editors, *Multimodal Music Processing*, volume 3 of *Dagstuhl Follow-Ups*, pages 175–194. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2012.

[171] 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)*, Shanghai, China, 2016.

1.5 Acknowledgments

This thesis reflects the results of my work and research conducted in the group of Prof. Dr. Meinard Müller over the last four years. The majority of this time I was financed by the DFG-project SIAMUS[1] (DFG MU 2686/6-1) for which I would like to thank the German Research Foundation. It was an intense, but nevertheless amazing time that I would definitely miss. By following Meinard around various stations of his academic career (namely from Saarbrücken to Bonn and finally to Erlangen), I got to know many great people who not only influenced my work, but also my personal life. At this point, I would like to take the opportunity to express my gratitude to them.

First of all, I want to thank Meinard himself. Not only was he the one who ignited my interest in music signal processing, but he also became a good friend and mentor. If it weren't for him, I probably would have never considered starting a PhD. Thank you for convincing me!

Also, Meinard's group always had the tendency to consist of great people without exception. I want to deeply thank the other current group members Thomas Prätzlich, Stefan Balke, Christian Dittmar, Patricio López-Serrano, Vlora Arifi-Müller, and Christof Weiß, as well as the former members Harald G. Grohganz, Nanzhu Jiang, Sebastian Ewert, Peter Grosche, Verena Konz, Andreas Baak, and Thomas Helten for all their support and advice over the years as well as for the great time that we shared.

Next I would like to thank Prof. Dr. Michael Clausen for the great time that I spent in Bonn with his group, as well as for reviewing this thesis.

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

The major part of my PhD studies I stayed at the International Audio Laboratories Erlangen[2] which is not only one of the most renowned institutions in the field of audio signal processing, but also offers an incredibly good working and research environment. I want to express my deepest gratitude to all the people who made my stay so comfortable. In particular, I would like to thank the administrative staff: Stefan Turowski, Elke Weiland, Tracy Harris, Day-See Riechmann, Carola Seibold, Jennifer Kohlhage, and Claudia Erbel for their enormous help in all possible and impossible administrative matters, as well as the AudioLabs professors Bernd Edler, Tom Bäckström, Emanuël Habets, and Jürgen Herre for encouraging such an open and inspiring atmosphere.

During my PhD studies, I had the opportunity to do a very interesting internship at the Fraunhofer IIS. To this end, I would like to thank Albert Heuberger, Harald Popp, Bernhard Grill, Oliver Hellmuth, Harald Fuchs, Frederick Nagel, Christian Uhle, Sascha Disch, Jouni Paulus, Falko Ridderbusch, Andreas Walther, and Alexandra Craciun for this experience and the great collaborative work.

I also want to deeply thank Mark Sandler, who invited me for a two-week research stay with his group at Queen Mary University of London, as well as Sebastian Ewert for the great collaborative work that we did during that time.

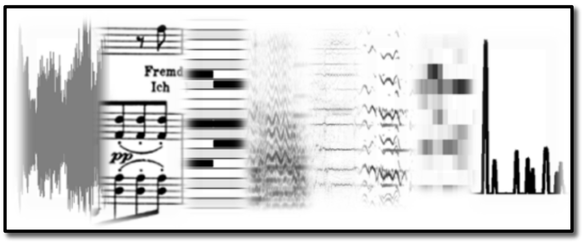
Furthermore, I would like to express my gratitude to all other people and colleagues who made my last four years such a good time. Many of them also helped me in the context of this thesis with constructive suggestions for improvements, collaborations, participation in experiments, proofreading, or just by spending time with me having a good cup of premium coffee. I will confine myself to only mentioning their names in alphabetical order: Alexander Adami, Stefan Bayer, Christian Borß, Rachel Bittner, Sebastian Braun, Soumitro Chakrabarty, Esther Fee Feichtner, Thorsten Feichtner, Johannes Fischer, Christian Fischer Pedersen, Richard Füg, Annika Gräwe, Christian Helmrich, Tobias Jähnel, Lena Krauss, Verena Kriesel, Moritz Küntzler, Lukas Lamprecht, Bernhard Lehner, Antoine Liutkus, Heinrich Löllmann, María Luis Valero, Christian Helmrich, Matthias Mauch, Brian McFee, Oriol Nieto, Po-Yao Hsiao, Colin Raffel, Jean-Baptiste Rollande, Mi Tian, Mattheo Turcoli, Justin Salamon, Maximilian Schäfer, Nadja Schinkel-Bielefeld, Michael Schöffler, Hendrik Schreiber, Fabian-Robert Stöter, Maja Taseska, Armin Taghipour, Nils Werner, and Frank Zalkow.

Last but not least, I thank my family and friends who shared my ups and downs with me throughout my time as a PhD student. In particular, I want to thank my fiancée Lisa from the bottom of my heart for never letting me down and all the support that she gave me.

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

Chapter 2

Fundamentals of Music Signal Processing



In this chapter, we introduce notions, transforms, and feature representations that will be used throughout this thesis. Rather than giving all technical details, our goal is to present them in an illustrative way while fixing some mathematical notation. For a more elaborate discussion of these topics we refer to [146]. We start with introducing three different music representations, namely *audio signals* (Section 2.1.1), *sheet music* (Section 2.1.2), and the symbolic *MIDI representation* (Section 2.1.3). Afterwards, we review the famous *Fourier transform*—a tool of fundamental importance for music signal processing—as well as its short-time formulation (Section 2.2). We then proceed with discussing various feature representations of music signals: the *pitch-based log-frequency spectrogram* and *chroma features* (Section 2.3), the *refined log-frequency spectrogram* and *salience spectrogram* (Section 2.4), and finally *novelty curves* (Section 2.5).

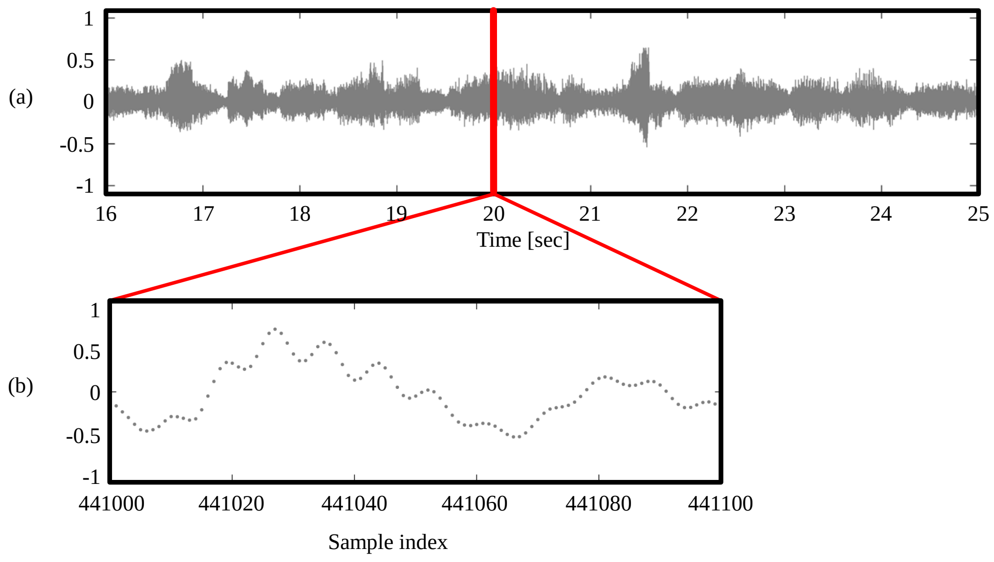


Figure 2.1: Excerpt of a recording of Franz Schubert's "Gute Nacht". The signal is sampled at a sampling rate of $F_s = 22050$ Hz. **(a):** The signal plotted on a time axis given in seconds. **(b):** A short excerpt of the signal, plotted on a time axis given in samples.

2.1 Music Representations

2.1.1 Audio Signals

A music signal can be seen as an encoding of the time-varying air pressure deviations that we, ultimately, can hear as sound. Following [146, Section 2.2.1], we define a *continuous-time* (CT) signal or *analog* signal to be a function $f : \mathbb{R} \to \mathbb{R}$, that assigns an amplitude value $f(t) \in \mathbb{R}$ to each point in time $t \in \mathbb{R}$. This definition makes it possible to reflect infinitesimally small changes in both the amplitude and the time and therefore allows for representing arbitrary sounds. However, when using digital technology, only a finite number of parameters can be stored and processed. To this end, analog signals need to be converted into finite representations—a process commonly referred to as *digitization*. One step that is often applied in an analog-to-digital conversion is known as *equidistant sampling*. Given an analog signal $f : \mathbb{R} \to \mathbb{R}$ and a positive real number $T > 0$, one defines a function $x : \mathbb{Z} \to \mathbb{R}$ by

$$x(r) = f(r \cdot T) \tag{2.1}$$

for $r \in \mathbb{Z}$. Since x is only defined on a discrete set of points in time, it is also referred to as *discrete-time* (DT) signal. The value $x(r)$ for some *sample index* $r \in \mathbb{Z}$ is called a *sample* taken at time $t = r \cdot T$ of the original analog signal f. This procedure is also known as T-*sampling*, where the number T is referred to as the *sampling period.* Its inverse

$$F_\mathrm{s} = 1/T \tag{2.2}$$

is also called the DT-signal's *sampling rate* that specifies the number of samples per second and is measured in Hertz (Hz). Figure 2.1 shows an example of a DT-signal, visualized as a *waveform* (a time-air pressure plot). Here, we see an excerpt from a recording of Franz Schubert's "Gute Nacht", a piece from the famous *Winterreise* song cycle for singing voice and piano. Figure 2.1a comprises nine seconds of audio material. Since the sampling rate of this DT-signal is $F_\mathrm{s} = 22050$ Hertz, resulting in 198450 samples overall, it is not possible to see individual samples at this scale. Figure 2.1b shows a magnified excerpt of 0.0045 seconds from the waveform (100 samples) that is plotted on a sample-based time axis.

Note that the above definition still does not allow for storing a DT-signal with a finite set of parameters due to two reasons: first, the amplitude of a DT-signal is still defined to be in $\mathbb{R}$. Since it is not possible to store an arbitrary real number with finite memory, a DT-signal's range is usually restricted by *quantization* for storage, see [146, Section 2.2.2.2]. However, we omit this step at this point for the sake of simplicity. Second, in the above definition, a DT-signal x has an infinite length (it is defined on $\mathbb{Z}$). For a digital music recording, one only has a finite number of samples, say $x(0), x(1), \ldots, x(L-1)$, where $L \in \mathbb{N}$ is the recording's *length.* To avoid boundary considerations, we typically set $x(r) = 0$ for $r \in \mathbb{Z} \setminus [0 : L-1]$ where $[0 : L-1] = \{0, 1, \ldots, L-1\}$, thus resulting in a signal $x : \mathbb{Z} \to \mathbb{R}$.

As a final remark, note that the term "DT-signal" does, in general, not imply any relation to music or even to audio. Focusing on music signal processing, we will therefore often use terms like "audio signal", "music signal", or "music recording" instead of "DT-signal" to make the signal's nature more explicit.

2.1.2 Sheet Music

A music recording captures a specific performance of a piece of music. In contrast, sheet music, also referred to as *musical score*, describes the musical work on a more abstract level. Loosely speaking, the musical score is an instruction for a musician that tells him or her what to play and how to play it. Figure 2.2 depicts an excerpt of the sheet music, given in Western music notation, of our Winterreise example. The score is segmented into three *staffs* that run horizontally.

The first staff is related to the singing voice. In addition to *note symbols* (♪,♫) and *rests* (𝄽,𝄾) it also shows the *lyrics* that should be sung by the singer. The vertical positions of the notes' heads with respect to the staff's five lines indicate the pitches of the tones to be sung by the singer, while different note symbols and ornaments encode the notes' durations, specified in musical

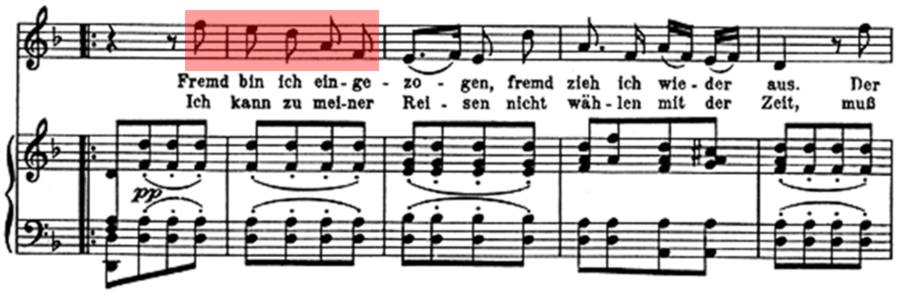

Fremd bin ich ein-ge - zo - gen, fremd zieh ich wie-der aus. Der
Ich kann zu mei-ner Rei - sen nicht wäh-len mit der Zeit, muß
pp

Figure 2.2: Excerpt of the sheet music for Franz Schubert's "Gute Nacht" [188] (measures six to ten).

time (given in beats or measures). In Western music, pitches are typically specified with respect to the *equal-tempered scale*. Here, one *octave* (the pitch interval between two tones where one tone's frequency is half or double the frequency of the other) is divided into twelve *semitones* that are denoted by a *chroma* C, C$^{\#}$, D,. . .,B in combination with a number that specifies the octave (for a detailed discussion on cognitive aspects of musical pitch, we refer to [122]). The first note of the singing voice is an eighth note F4, being half a beat long, followed by the eighth notes E4, D4, A3, and F3 (marked in red). Notes of other durations are, for example, the half note 𝅗𝅥, having a duration of two beats, or the quarter note with a duration of one beat.

The second and third staff specify what should be played by the piano. Here, note symbols are stacked on top of each other, indicating that the respective notes should be played simultaneously by the pianist. The notes are distributed over two staffs to loosely indicate which ones should be played by the piano player's left and right hand, respectively. We also see additional performance instructions like ***pp***, which stands for *pianissimo* and instructs the musician to play "very softly".

Unlike music recordings where information about notes that are played by musicians or the lyrics that are sung by the singer are implicitly encoded in the recording's waveform, sheet music specifies this kind of information explicitly. This explicit information can be very valuable in the context of music signal processing since it can be used to guide the processing of the audio material in various applications. However, sheet music is typically not computer-readable and furthermore lacks a connection between the musical time specified in the score (given in beats and measures) and the physical time of a music recording (given in seconds or sample indices). For further details about sheet music we refer to [146, Section 1.1].

2.1.3 MIDI Representation

Symbolic music representations, such as the *Musical Instrument Digital Interface* (MIDI) standard, bridge the gap between the worlds of music recordings and sheet music. MIDI is a digital protocol that is commonly used to control electronic music instruments [107, 192, 205]. The format allows

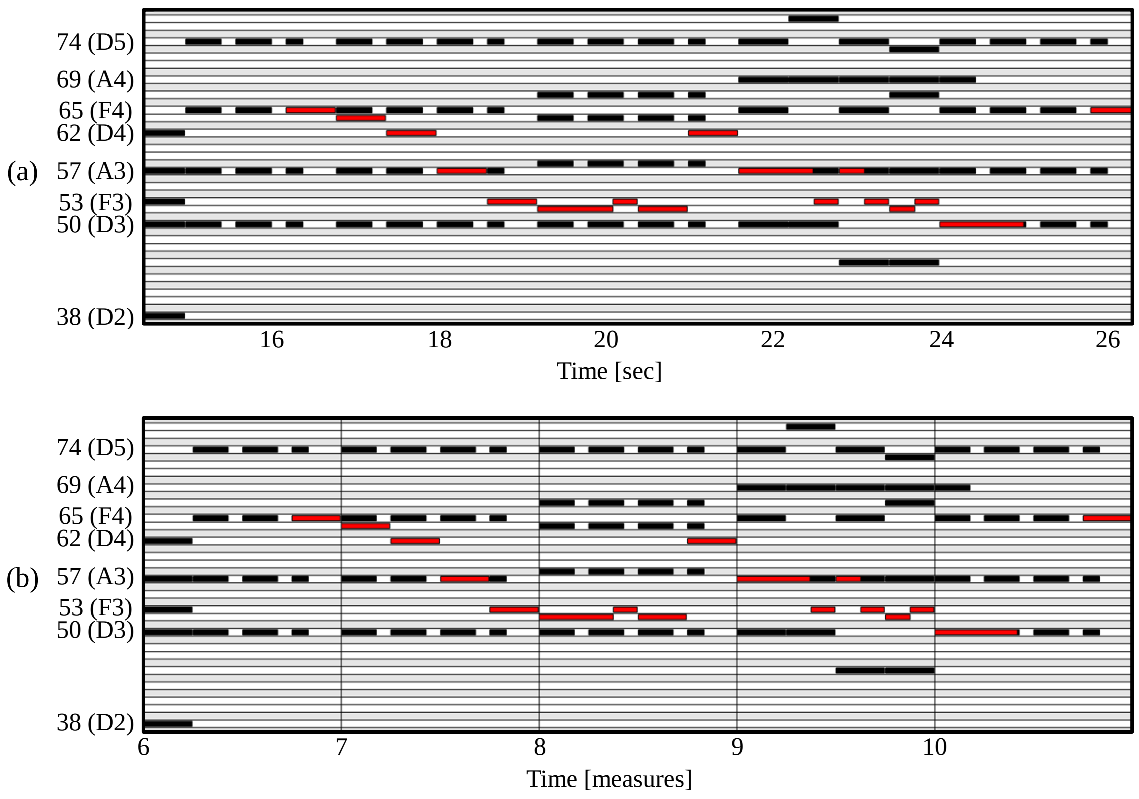


Figure 2.3: Excerpt of a MIDI representation of Franz Schubert's "Gute Nacht". Individual notes are represented by rectangles. The singing voice's track is indicated in red. **(a):** Plotted on a physical time axis. **(b):** Plotted on a musical time axis.

to sequence large amounts of control parameters and to store them in so called *MIDI files*. In a MIDI file, the control parameters are grouped in *tracks* that commonly correspond to the different voices of a musical piece. Besides other information, each track encodes the individual notes' pitches as well as their on- and offset times. Figure 2.3a shows a time-pitch plot, sometimes also referred to as a *piano roll representation*, of our running example's MIDI file. Notes are indicated by rectangles where a rectangle's vertical position, horizontal localization, and length encode the note's pitch, onset time, and duration, respectively. Furthermore, the different musical voices are color-coded. Here, red rectangles correspond to the singing voice and black rectangles to the piano. This example illustrates that a MIDI file may deliver similar information as a musical score.

In MIDI, a pitch is encoded by a number $p \in [0:127]$, also called a *MIDI pitch*, that corresponds to the physical frequency

$$F_{\text{pitch}}(p) = 2^{(p-69)/12} \cdot 440 \, , \tag{2.3}$$

given in Hertz. This assignment reflects the equal-tempered scale as introduced in the previous section. A note's on- and offset may be specified both in musical as well as in physical units.

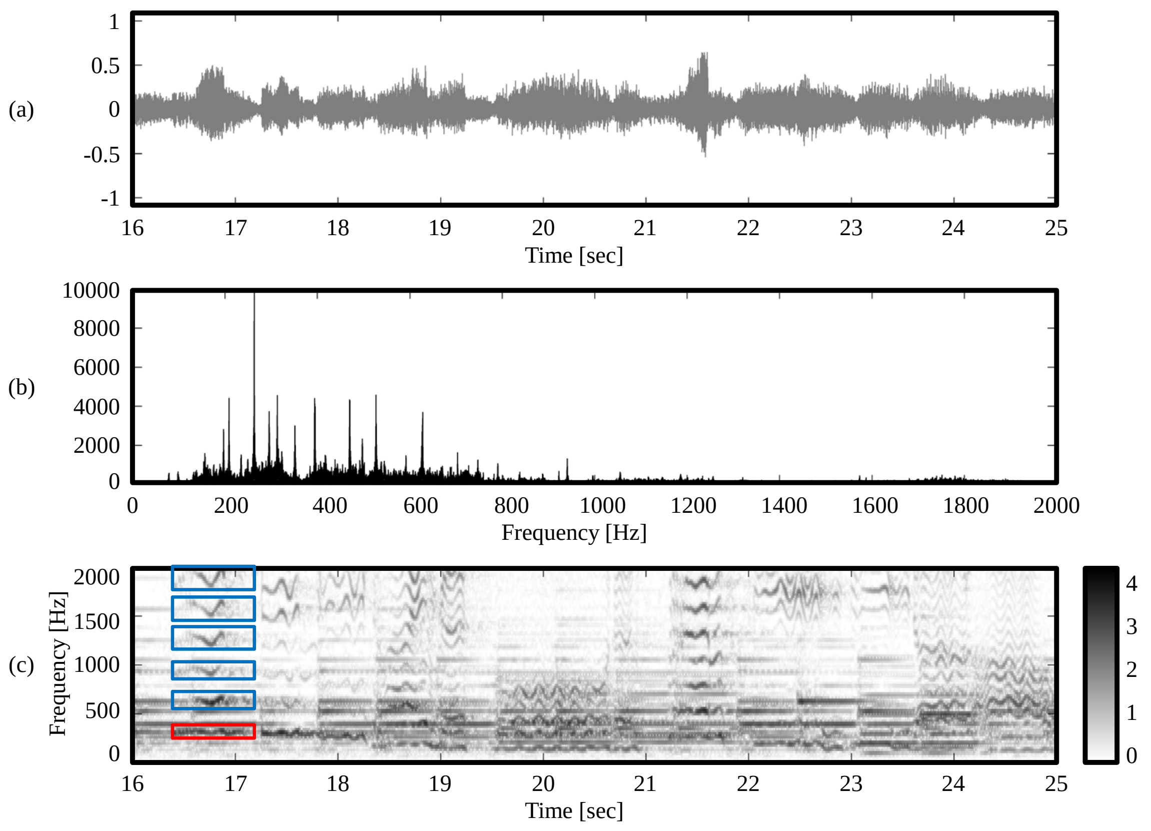


Figure 2.4: Signal representations. **(a):** Waveform. **(b):** Magnitude Fourier spectrum. **(c):** Spectrogram. The fundamental frequency of the singer's first note is indicated in red while its overtones are marked in blue.

This is shown in Figure 2.3 where the MIDI file is once visualized on a time axis (Figure 2.3a) and once on an axis (Figure 2.3b). This feature of MIDI is particularly interesting in the context of music signal processing since MIDI can therefore link the abstract world of sheet music with the concrete world of music recordings. To this end, we commonly use MIDI files when our goal is to incorporate information from musical scores into music signal processing techniques.

2.2 Fourier Transform and Spectrogram Representation

Music signals can be complex mixtures consisting of a multitude of different sound components. A first step in better understanding a given signal is to decompose it into building blocks that are better accessible for the subsequent processing steps. In the case that these building blocks consist of exponential functions, such a process is also called *Fourier analysis*. The famous Fourier transform maps a time-dependent signal to a frequency-dependent function which reveals the *spectrum* of frequency components that compose the original signal. While there are also Fourier transform formulations for CT-signals, we exclusively deal with the Fourier transform for DT-signals—also known as *discrete Fourier transform* (DFT)—in the context of this thesis. Given a DT-signal x of length L, the DFT X of x is defined by

$$X(k) = \sum_{r=0}^{L-1} x(r) \cdot \exp(-2\pi ikr/L) \ , \tag{2.4}$$

for $k \in [0 : L-1]$. The complex *Fourier coefficient* $X(k)$ encodes the magnitude and phase of the given signal's sinusoidal component with frequency

$$F_{\mathrm{coef}}(k) = \frac{k \cdot F_{\mathrm{s}}}{L} \ . \tag{2.5}$$

Summing these sinusoidal components yields the original signal again, a process known as *inverse discrete Fourier transform*:

$$x(r) = \frac{1}{L} \sum_{k=0}^{L-1} X(k) \cdot \exp(2\pi ikr/L) \ , \tag{2.6}$$

for $r \in [0 : L-1]$. In Figure 2.4a/b we see our running example's waveform and the magnitude of its Fourier transform. While in the recording's waveform we can see *when* sound events are occurring (at least roughly), the Fourier transform reveals *which* sound events can be heard (in terms of their frequencies). Loosely speaking, a signal and its Fourier transform are two sides of the same coin. On the one side, the signal displays the time information and hides the information about frequencies. On the other side, the Fourier transform reveals information about frequencies and hides the time information [106].

To obtain back the hidden time information, Dennis Gabor introduced in the year 1946 a modified version of the Fourier transform, now known as the *short-time Fourier transform* (STFT) [74]. This transform is a compromise between a time- and a frequency-based representation. Its core idea is to apply the Fourier transform to local sections of a signal, so called *frames*, rather than to the whole signal at once. This way, the STFT does not only reveal which frequencies are contained in a given signal signal, but also at which points of times—or, to be more precise, in which time frames—these frequencies appear. A signal's STFT is defined by

$$X(m,k) = \sum_{r=-N/2}^{N/2-1} w(r) \cdot x(r + mH) \cdot \exp(-2\pi ikr/N) \ , \tag{2.7}$$

where $m \in \mathbb{Z}$ is the frame index, $k \in [0 : N-1]$ is the frequency index, N is the frame length, w is a window function and H is the hopsize that defines the number of samples between two consecutive frames. A Fourier coefficient $X(m,k)$ is also referred to as *time-frequency bin* where the time index m corresponds to the physical time

$$T_{\text{coef}}(m) = \frac{m \cdot H}{F_{\text{s}}} \,, \tag{2.8}$$

given in seconds, and the frequency index k corresponds to the frequency band with center frequency

$$F_{\text{coef}}(k) = \frac{k \cdot F_{\text{s}}}{N} \,, \tag{2.9}$$

given in Hertz. An STFT's magnitude is also called a *spectrogram* which we denote by Y:

$$Y = |X| \,. \tag{2.10}$$

Figure 2.4c shows our running example's spectrogram for $N = 1024$ and $H = 512$. Here, the color-coded magnitude values reveal time- and frequency-dependent information about the recording. In particular, we can see wave-like structures that stem from vibrato in the singing voice as well as horizontal patterns originating from the notes played on the piano. The spectrogram also reveals the harmonic nature of both the piano as well as the singing voice. A single sung note does not only have energy at the frequency corresponding to its pitch (also known as the pitch's *fundamental frequency* or *F0*), but also at integer multiples (known as *overtones*). In Figure 2.4c the fundamental frequency of the singer's first note is indicated in red while its overtones are marked in blue.

This example already indicates the STFT's great potential in the context of music signal processing. The STFT will serve as fundamental decomposition and analysis tool throughout this thesis.

For further information on Fourier analysis and its mathematical foundations we refer to classical textbooks such as [158, 172, 146].

2.3 Pitch- and Chroma-Based Audio Features

A common task in music signal processing is to analyze a music recording with respect to musical pitches occurring in the recording or its rough harmonic progression. Feature representations that capture this kind of information in a given music recording are therefore often helpful. In this section, we first introduce the *pitch-based log-frequency spectrogram* (Section 2.3.1) that represents a recording's energy distribution with respect to the 128 pitches defined by Equation (2.3). Afterwards, we show how it can be converted into a *chromagram*, a feature representation that reflect harmony-based properties of the recording (Section 2.3.2).

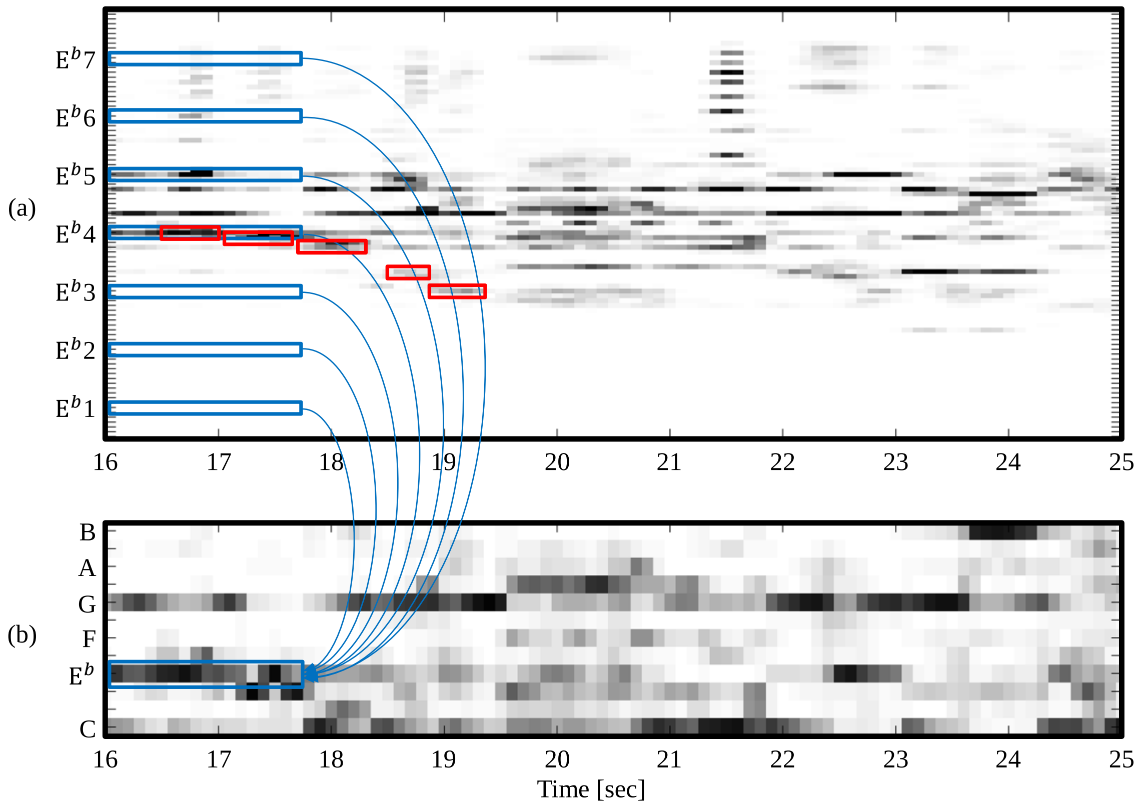


Figure 2.5: Feature representations derived from the STFT X. **(a):** Pitch-based log-frequency spectrogram Y_{LF}. The pitches of the singing voice's first five notes are indicated by red rectangles (note that the recording is transposed and the first note is therefore an E^{b}4 rather than an F4). Furthermore, pitch bands corresponding to the chroma E^b are marked in blue. **(b):** Chromagram C. The pooling process is visualized in blue for the chroma E^b.

2.3.1 Pitch-Based Log-Frequency Spectrogram

In Section 2.2 we introduced the STFT, a tool that can be used to derive a time-frequency representation of a given signal. As indicated by Equation (2.9), the frequency axis of such an STFT representation is linearly sampled, meaning that the center frequencies of two neighbored frequency bands always differ by the same constant amount. However, in order to represent the signal's pitch content, it is desirable to have a logarithmically spaced frequency axis that reflects the logarithmic nature of musical pitches. There exist various methods to derive so called *log-frequency* spectrograms as, for example, the *Constant-Q Transform* [13, 187], filter bank approaches [145], or approaches based on *instantaneous frequency estimation* [33, 183]. When the goal is to have a frequency axis where each frequency band corresponds to a MIDI pitch as defined by Equation (2.3), a particularly simple approach is described in [146, Section 3.1]. Given a music signal's STFT X as defined in Equation (2.7), the idea is to assign each Fourier coefficient $X(m, k)$ to the MIDI pitch p whose frequency $F_{\mathrm{pitch}}(p)$ is closest to the coefficient's center frequency $F_{\mathrm{coef}}(k)$. More precisely, we define

$$P(p) = \{k : F_{\mathrm{pitch}}(p - 0.5) \leq F_{\mathrm{coef}}(k) < F_{\mathrm{pitch}}(p + 0.5)\} , \tag{2.11}$$

$p \in [0 : 127]$, to be the set of frequency indices that are pooled in the pitch band p. Using this definition we then define the *pitch-based log-frequency spectrogram* $Y_{\mathrm{LF}} : \mathbb{Z} \times [0 : 127] \to \mathbb{R}$ by

$$Y_{\mathrm{LF}}(m,p) = \sum_{k \in P(p)} |X(m,k)|^2 . \tag{2.12}$$

Figure 2.5a shows Y_{LF} for our running example. Similar as the MIDI representation shown in Figure 2.3a, the shown visualization is a time-pitch plot. When looking carefully at this representation, one can even recognize the notes sung by the singer (marked in red). This investigation reveals that the musicians actually transposed the musical piece downwards by two semitones for their performance (the first note sung by the singer is a E^b rather than an F as notated in the musical score shown in Figure 2.2). This simple example demonstrates that the pitch-based log-frequency spectrogram can deliver musically meaningful information about a music recording.

2.3.2 Chroma Features

When the task is to capture the recording's rough harmonic progression one possibility is to derive *chroma features* (sometimes also referred to as *pitch class profiles*) that capture the recording's energy with respect to the twelve chroma classes [5, 79, 145]. They can be easily derived from the pitch-based log-frequency spectrogram Y_{LF} by pooling pitch bands that correspond to the same chroma. Formally, we define a chromagram C by

$$C(m,c) = \sum_{\{p \in [0:127] : p \bmod 12 = c\}} Y_{\mathrm{LF}}(m,p) , \tag{2.13}$$

for $c \in [0 : 11]$. Figure 2.5b shows the chromagram derived from the pitch-based log-frequency spectrogram shown in Figure 2.5a. Here we see, for example, that in the first second of the observed excerpt most of the energy in the recording is contained in the chroma bands C, E^b, and G, indicating a C minor harmony. Considering that the recording was transposed by the musicians, this corresponds to the D minor chord that is notated in the musical score to be played by the piano in this excerpt's first measure (see Figure 2.2). Here, the chromagram therefore reflects the recording's harmonic content well.

At around second 20 the chromagram is rather noisy, which is caused by multiple effects such as strong vibrato in the singing voice, breathing sounds, and dense overtone distributions. While the presented simple procedure for deriving chroma features yields a suboptimal chroma representation

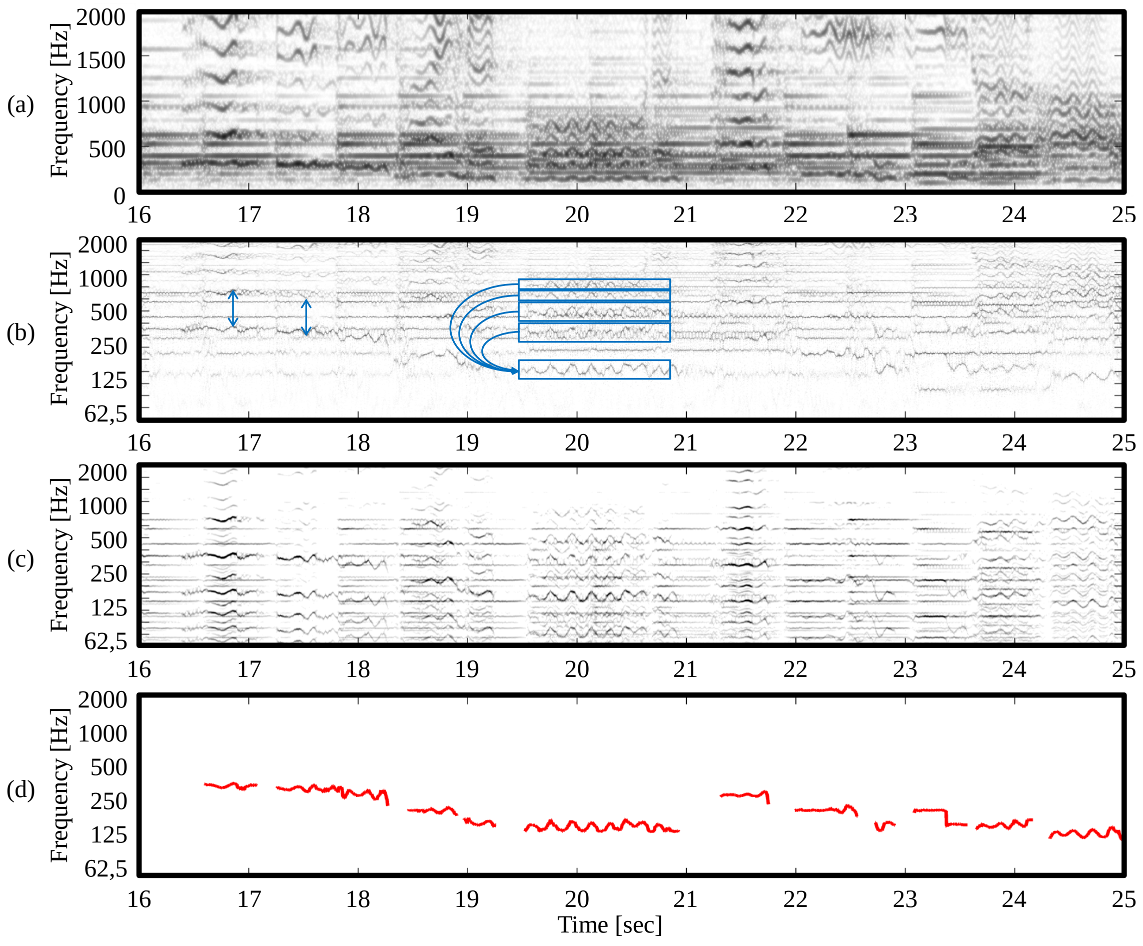


Figure 2.6: Spectrogram representations. **(a):** Spectrogram with linear frequency axis Y. **(b):** Refined log-frequency spectrogram $Y_{\mathrm{LF}}^{\mathrm{IF}}$. **(c):** Salience spectrogram Z. **(d):** Manual annotation of the melody track.

in this case, there exists various methods in the literature to enhance chroma representations of music signals. Some of these approaches are, for example, based on weighting schemes in the pooling process of STFT coefficients [60, 79], enhancing low energy values by logarithmic compression [118] or related spectral whitening techniques [202], considering short-term statistics over the energy distributions [153], or making the features timbre invariant [150].

Note that there exist various publically available implementations of both log-frequency spectrograms [187, 151, 59] as well as chroma features [58, 151, 134].

2.4 Audio Features for Melody Tracking

In the previous section, we have seen the concept of a log-frequency spectrogram that reflects the logarithmic nature of pitch better than a spectrogram with linearly sampled frequency axis.

The pitch-based log-frequency spectrogram has one frequency band per semitone. For tasks like *melody extraction* or *predominant $F0$ estimation*, where the objective is to estimate the sequence of frequency values that correspond to the main melody [80, 82, 149, 167, 183, 184] (the melody track of our running example's melody can be see in Figure 2.6d), this coarse resolution does, however, not suffice. Here it is necessary to have a much finer frequency resolution such that aspects like vibrato or glissando in the melody can be represented appropriately. In this section, closely following [146, Section 8.2], we therefore first discuss the *refined log-frequency spectrogram* that can have an arbitrary frequency resolution (Section 2.4.1). Afterwards, we introduce the *salience spectrogram*, a feature representation that is derived from the refined log-frequency spectrogram and particularly designed for the task of melody tracking (Section 2.4.2).

2.4.1 Refined Log-Frequency Spectrogram

Recall that the core idea of the pitch-based log-frequency spectrogram was to assign the center frequencies of the STFT's frequency bands to pitch bands by Equation (2.11). One can show that Equation (2.11) can be reformulated by

$$P(p) = \{k : \mathrm{Bin}(F_{\mathrm{coef}}(k)) = p\}\ , \tag{2.14}$$

with

$$\mathrm{Bin}(\omega) = \left\lfloor 12 \cdot \log_2\left(\frac{\omega}{440}\right) + 69.5 \right\rfloor\ . \tag{2.15}$$

Inspired by the MIDI pitch numbers as defined in Equation (2.3), this assignment is designed such that the center frequency 440 Hertz serves as reference and is assigned to the index 69. Furthermore, one octave is subdivided into 12 bands. We can generalize this assignment in the following way. Assume that our goal is to construct a log-frequency representation with $B \in \mathbb{N}$ bands where each band corresponds to R cents (a semitone is subdivided into 100 cents). Furthermore, let ω_{ref} be a reference frequency such that index b_{ref} of our representation corresponds to this frequency. The bin assignment can then be defined by

$$\mathrm{Bin}(\omega) = \left\lfloor \frac{1200}{R} \cdot \log_2\left(\frac{\omega}{\omega_{\mathrm{ref}}}\right) + b_{\mathrm{ref}} + 0.5 \right\rfloor\ . \tag{2.16}$$

This definition allows us to construct a grid for a log-frequency representation with arbitrary frequency resolution. The problem at this point is that the frequency resolution of the STFT itself is rather coarse—we can only associate a Fourier coefficient $X(m,k)$ with the center frequency $F_{\mathrm{coef}}(k)$. However, it is possible to refine frequency estimates for the STFT by using techniques for *instantaneous frequency estimation* [1, 28, 69, 125] (we discuss instantaneous frequency estimation in detail in Section 4.1.4.3). These techniques yield refined frequency estimates $F^{\mathrm{IF}}_{\mathrm{coef}} : \mathbb{Z} \times [0 : N-1] \to \mathbb{R}$ for all Fourier coefficients of an STFT. We can therefore define a refined log-frequency spectrogram by

$$Y_{\mathrm{LF}}^{\mathrm{IF}}(m,b) = \sum_{k \in P^{\mathrm{IF}}(m,b)} |X(m,k)|^2 \,, \tag{2.17}$$

with $b \in [0 : B-1]$ and

$$P^{\mathrm{IF}}(m,b) = \{k : \mathrm{Bin}(F_{\mathrm{coef}}^{\mathrm{IF}}(m,k)) = b\} \,. \tag{2.18}$$

In this representation the frequency index $b \in [0 : B-1]$ corresponds to the physical frequency

$$F_{\mathrm{LF}}^{\mathrm{IF}}(b) = 2^{(b-b_{\mathrm{ref}})R/1200} \cdot \omega_{\mathrm{ref}} \,, \tag{2.19}$$

given in Hertz. Figure 2.6b shows the refined log-frequency spectrogram of our running example. Here, we defined each band to correspond to $R = 10$ cents, set the reference frequency to $\omega_{\mathrm{ref}} = 62.5$ Hertz and the reference index $b_{\mathrm{ref}} = 0$. When comparing this representation to the spectrogram with linear frequency axis shown in Figure 2.6a, one can see that in the refined log-frequency spectrogram spectral structures are much sharper due to the refined frequency resolution. Furthermore, the multiplicative relation between a fundamental frequency and its overtones translates into an additive one in the log-frequency domain. Therefore, a fundamental frequency and its n^{th} overtone always have the same distance, irrespective of the fundamental frequency itself. This is indicated in Figure 2.6b by blue arrows. We will exploit this property in the next section in order to derive a spectrogram representation that visually enhances the underlying music recording's melody track.

2.4.2 Salience Spectrogram

A main step in procedures for melody extraction is the computation of a *salience function* or *salience spectrogram*. A salience spectrogram is a time-frequency representation of a given music recording in which time-frequency bins that are likely to be part of a recording's melody track are salient [82, 183, 117]. One approach to derive a salience spectrogram is *harmonic summation* [117, 183]. This strategy is based on the observation that melody instruments, unlike accompanying instruments, tend to have large amounts of energy not only in their fundamental frequency but also in their overtones. This can also be seen in Figure 2.6a where the singing voice's overtones are quite dominant while the piano's overtones tend to be weaker. The idea of the harmonic summation strategy is to compute a salience spectrogram Z by assigning each time-frequency bin $Z(m,b)$ not only the energy $Y_{\mathrm{LF}}^{\mathrm{IF}}(m,b)$, but also the energy of the time-frequency bins corresponding to the first H overtones of the fundamental frequency $F_{\mathrm{LF}}^{\mathrm{IF}}(b)$. Formally, we

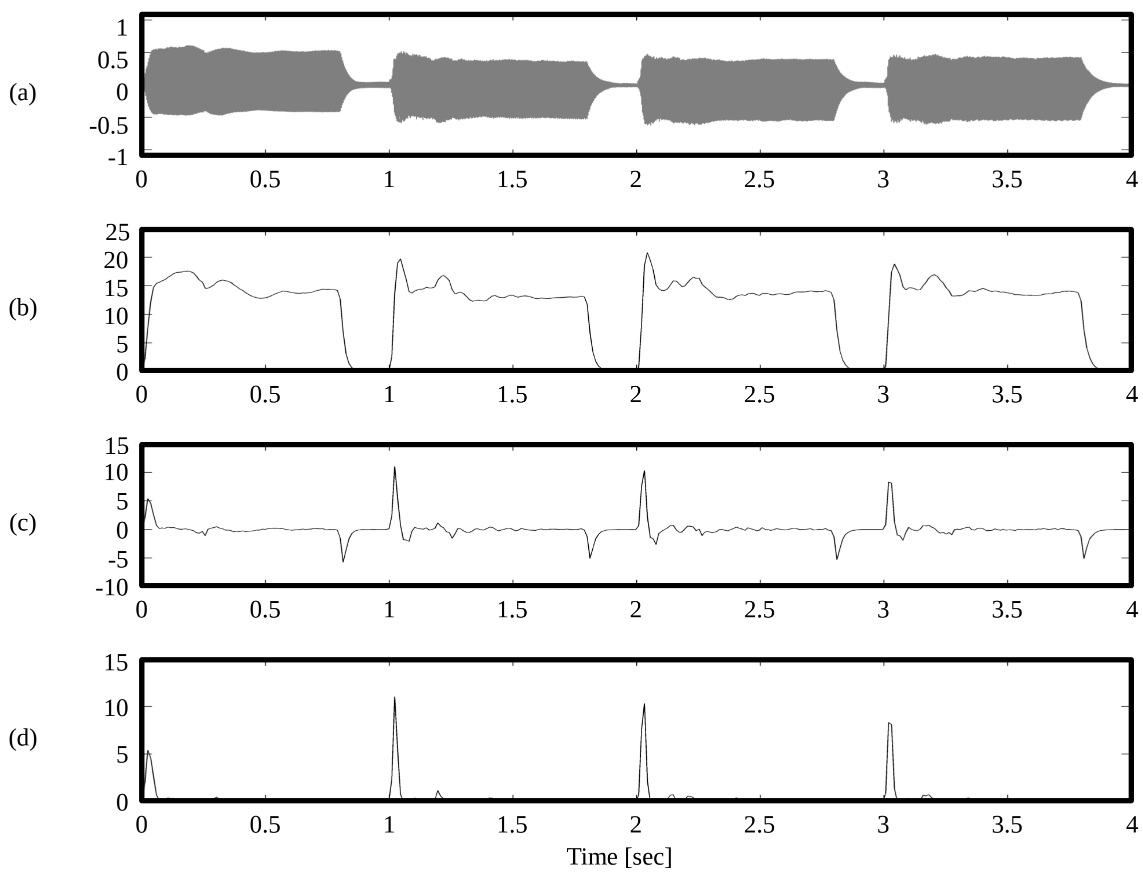


Figure 2.7: Energy-based novelty. **(a):** Audio signal of four consecutive notes played on a clarinet x. **(b):** Local energy function E. **(c):** Discrete derivative. **(d):** Novelty curve Δ_{Energy}.

can define the harmonic summation by

$$Z(m, b) = \sum_{h=1}^{H+1} Y_{\text{LF}}^{\text{IF}} \left(m, \text{Bin}\left(h \cdot F_{\text{LF}}^{\text{IF}}(b)\right)\right) \ . \tag{2.20}$$

This process is indicated in Figure 2.6b for the first $H = 4$ overtones. When comparing this representation to the manual annotation of this excerpt's melody in Figure 2.6d, one can see that in Z this track is, as intended, rather salient. Still, the automatic extraction of the melody from this representation is a difficult task that attracts high research interest [3, 82, 117, 183]. In this thesis, we present a user interface in Appendix B.1 that allows for approaching this problem in a user-guided fashion.

2.5 Novelty-Based Audio Features

An important concept in music signal processing is *novelty*—the sudden change of some musical or physical property in a music recording. Such changes may be measured by *novelty curves* which represent the "degree of change" over time with respect to some audio feature that captures the property of interest. On a coarse level, novelty may be related to musical structure. A new musical segment can be indicated by, for example, a change in instrumentation, key, tempo, and so on [70]. On a finer level, novelty is often related to the onsets of individual notes or transient sound events. To this end, novelty curves of high time-resolution are commonly used for *onset detection* [6, 21, 27]—a task that often serves as a first step to approach more complex problems such as *tempo estimation* or *beat tracking* [88, 90, 165, 87, 86].

A very basic and simple method for computing novelty curves in the context of onset detection is described in [146, Section 6.1] and visualized in Figure 2.7. Figure 2.7a shows an audio signal x consisting of four consecutive notes played on a clarinet. One can see that the note onsets go along with a sudden increase of the audio signal's energy. Therefore, a first step in computing a novelty curve for the given audio signal is to derive its *local energy*

$$E(r) = \sum_{\ell=-N/2}^{N/2-1} |x(r+\ell) \cdot w(\ell)|^2 = \sum_{\ell \in \mathbb{Z}} |x(\ell) \cdot w(\ell - r)|^2 \,, \tag{2.21}$$

where $r \in \mathbb{Z}$, w is a centered window function, and N is the window function's length. The local energy of our example signal is shown in Figure 2.7b, where we can see four segments of high energy that correspond to the four notes. Since we want to detect the notes' onsets, we aim to detect the sudden energy increases at the segments' beginnings. We therefore compute the local energy's discrete derivative as it can be seen in Figure 2.7c. In this representation, the note onsets go along with clear peaks while the sudden energy decreases at the notes' offsets cause dips. Since we are only interested in energy increases, not decreases, we only keep the positive derivative values while setting the negative values to zero. This operation is known as *half-wave rectification* that can be formally defined by

$$|r|_{\geq 0} = \frac{r + |r|}{2} = \begin{cases} r, & \text{if } r \geq 0, \\ 0, & \text{otherwise,} \end{cases} \tag{2.22}$$

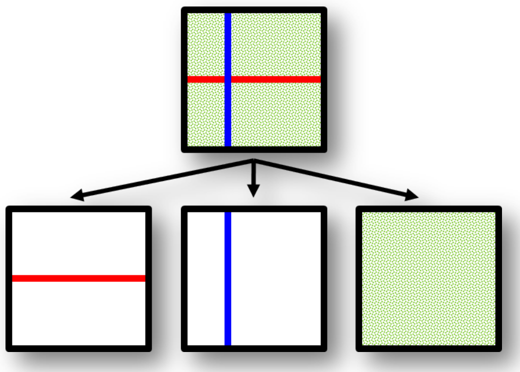
for $r \in \mathbb{R}$. The novelty curve is computed by

$$\Delta_{\text{Energy}}(r) = |E(r+1) - E(r)|_{\geq 0} \,. \tag{2.23}$$

In our example's novelty curve, which is plotted in Figure 2.7d, we can see four clear peaks that indicate the note's onsets. Although the presented energy-based approach is sufficient to detect the onsets in this toy example, it is not very robust in general. In particular instruments with soft note onsets, such as the violin, do not cause sudden energy increases at onset positions. To detect this kind of onsets, much more involved procedures for novelty curve computation are needed. Approaches proposed in the literature are based, for example, on analyzing the signal's spectral content [6, 66, 100, 229], pitch [22, 100, 229], harmony [62, 81], or phase [6, 99, 100]. There also exist various methods that combine different cues [25, 66, 100, 211, 229] and supervised approaches [66, 123]. Finally, note that freely available implementations of onset detectors are included, for example, in *LibROSA* [135], or in the *Tempogram Toolbox* [89, 87].

Chapter 3

Extensions to Harmonic-Percussive Separation of Music Signals



In this chapter, we discuss audio decomposition techniques that we originally proposed in [46, 42]. These methods will serve as fundamental tools that we use to approach other music signal processing tasks in subsequent chapters.

The task of decomposing an audio signal into its harmonic and its percussive component has received large interest in recent years. This is mainly because for many applications it is useful to consider just the harmonic or the percussive portion of an input signal. Harmonic-percussive separation has been applied, for example, for audio remixing [156], improving the quality of chroma features [213], tempo estimation [76], or time-scale modification [55, 48]. Several decomposition algorithms have been proposed. In [54], the percussive component is modeled by

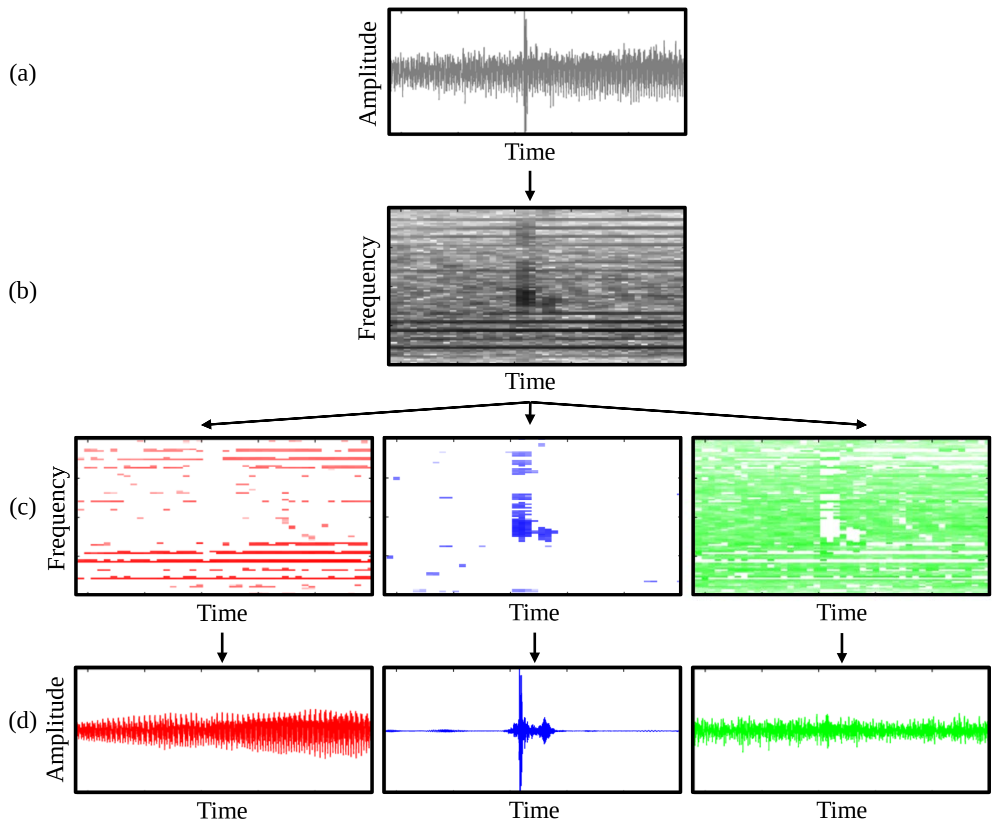


Figure 3.1: **(a):** Input audio signal x. **(b):** Magnitude spectrogram $|X|$. **(c):** Magnitude spectrograms of the harmonic component $|X_\mathrm{h}|$ (left), the percussive component $|X_\mathrm{p}|$ (middle) and the residual component $|X_\mathrm{r}|$ (right). **(d):** Waveforms of the harmonic component x_h (left), the percussive component x_p (middle) and the residual component x_r (right).

detecting portions in the input signal which have a rather noisy phase behavior. The harmonic component is then computed by the difference of the original signal and the computed percussive component.

A crucial observation that many procedures are based on is that harmonic sounds have a horizontal structure in a spectrogram representation of the input signal, while percussive sounds form vertical structures, see [14, 67, 77, 157, 163, 206]. These properties can then be exploited to separate portions of an audio signal's spectrogram belonging to harmonic sounds from portions belonging to percussive sounds. For example, in [157] the harmonic and percussive structures are first enhanced by iteratively diffusing the spectrogram once in horizontal and once in vertical direction, respectively. The two enhanced representations are then compared, and entries in the original spectral representation are assigned to either the harmonic or the percussive component according to the dominating enhanced spectrogram. Finally, the two components are transformed back to the time-domain. Following the same idea, Fitzgerald [67] replaces the diffusion step by a much simpler median filtering strategy, which turns out to yield similar results while having a much lower computational complexity.

A drawback of the aforementioned approaches is that the computed decompositions are often not very *tight* in the sense that the harmonic and percussive components may still contain some non-harmonic and non-percussive residues, respectively. This is mainly because of two reasons. First, sounds that are neither of clearly harmonic nor of clearly percussive nature such as applause, rain, or the sound of a heavily distorted guitar are often more or less randomly distributed among the two components. Second, depending on the parameter setting, harmonic sounds often leak into the percussive component and the other way around. Acquiring satisfactory results for the task at hand often involves either finding suitable parameters that yield an acceptable trade-off between a leakage in one or the other direction, or post-processing the resulting decomposition to remove the leaked sounds again, see [209].

In this chapter, we propose two extensions to [67] that lead towards more flexible and refined decompositions. First, we introduce the concept of a *separation factor* (Section 3.2). This novel parameter allows for *tightening* decomposition results by enforcing the harmonic and percussive component to contain just the clearly harmonic and percussive sounds of the input signal, respectively, and therefore to attenuate the aforementioned problems. Second, we exploit this concept to add a third *residual* component that captures all sounds in the input audio signal which are neither clearly harmonic nor percussive[1] (see Figure 3.1). This kind of decomposition is inspired by the classical *sines+transients+noise* (STN) audio model [133, 166, 217, 219, 194] that aims at resynthesizing a given audio signal in terms of a parameterized set of sine waves, transient sounds, and shaped white noise.

While a first methodology to compute such a decomposition follows rather straightforward from the concept of a separation factor (Section 3.3.1), we also propose a more involved iterative decomposition procedure. Building on concepts first proposed in [207, 113], this procedure allows for a more refined adjustment of the decomposition results (Section 3.3.3). Finally, we evaluate our proposed procedures based on objective evaluation measures as well as subjective listening tests (Section 3.4). Note that this chapter has an accompanying website at [45] where one can find all audio examples that are used in this chapter.

[1]Noticeably, Jeong and Lee proposed a technique for the task of *singing voice extraction* (see Chapter 5) based on similar ideas in [113] at the same time that we proposed this work in [46]. In their work, instead of building on the harmonic-percussive separation technique proposed by Fitzgerald [67] as we did, they extended the work by Ono et al. from [157].

3.1 Basic Procedure

Our proposed procedures extend Fitzgerald's method for harmonic-percussive separation [67], which we review in this section. The input of this procedure is an audio signal as shown in Figure 3.2a, where we see a synthetic mixture of a tone played on a violin (harmonic) and a single click of castanets (percussive). The procedure's first step is to compute the STFT X of the given audio signal x as defined in Equation (2.7). A critical observation is that, in the spectrogram $Y = |X|$, harmonic sounds form structures in the time direction, while percussive sounds yield structures in the frequency direction. In the spectrogram shown in Figure 3.2b we can see horizontal lines, reflecting the harmonic sound of the violin, as well as a single vertical line in the middle of the spectrogram, stemming from the click of the castanets. By applying a median filter to Y—once horizontally and once vertically—we get a horizontally enhanced spectrogram $\tilde{Y}_\mathrm{h}$ and a vertically enhanced spectrogram $\tilde{Y}_\mathrm{p}$:

$$\tilde{Y}_\mathrm{h}(m,k) = \mathrm{median}\left(Y(m-\ell_\mathrm{h},k),\ldots,Y(m+\ell_\mathrm{h},k)\right) \tag{3.1}$$

$$\tilde{Y}_\mathrm{p}(m,k) = \mathrm{median}\left(Y(m,k-\ell_\mathrm{p}),\ldots,Y(m,k+\ell_\mathrm{p})\right) \tag{3.2}$$

for $\ell_\mathrm{h}, \ell_\mathrm{p} \in \mathbb{N}$, where $2\ell_\mathrm{h}+1$ and $2\ell_\mathrm{p}+1$ are the lengths of the median filters, respectively (Figure 3.2c). A time-frequency bin of the original STFT $X(m,k)$ is assumed to be part of the harmonic component if $\tilde{Y}_\mathrm{h}(m,k) > \tilde{Y}_\mathrm{p}(m,k)$ and of the percussive component if $\tilde{Y}_\mathrm{p}(m,k) \geq \tilde{Y}_\mathrm{h}(m,k)$. Using this principle, one can define binary masks $\mathcal{M}_\mathrm{h}$ and $\mathcal{M}_\mathrm{p}$ for the harmonic and the percussive components (Figure 3.2d):

$$\mathcal{M}_\mathrm{h}(m,k) = \begin{cases} 1, & \text{if } \tilde{Y}_\mathrm{h}(m,k) > \tilde{Y}_\mathrm{p}(m,k), \\ 0, & \text{otherwise,} \end{cases} \tag{3.3}$$

$$\mathcal{M}_\mathrm{p}(m,k) = \begin{cases} 1, & \text{if } \tilde{Y}_\mathrm{p}(m,k) \geq \tilde{Y}_\mathrm{h}(m,k), \\ 0, & \text{otherwise.} \end{cases} \tag{3.4}$$

Applying these masks to the original STFT X yields modified STFTs corresponding to the harmonic and percussive components (Figure 3.2e):

$$X_\mathrm{h}(m,k) = X(m,k) \cdot \mathcal{M}_\mathrm{h}(m,k)\,, \tag{3.5}$$

$$X_\mathrm{p}(m,k) = X(m,k) \cdot \mathcal{M}_\mathrm{p}(m,k)\,. \tag{3.6}$$

These spectrograms can then be transformed back to the time-domain by applying an "inverse" short-time Fourier transform, see [84]. This yields the desired signals x_h and x_p, respectively. As we can see in Figure 3.2f, the harmonic component x_h contains the tone played by the violin, while the percussive component x_p shows the waveform of the castanets' click.

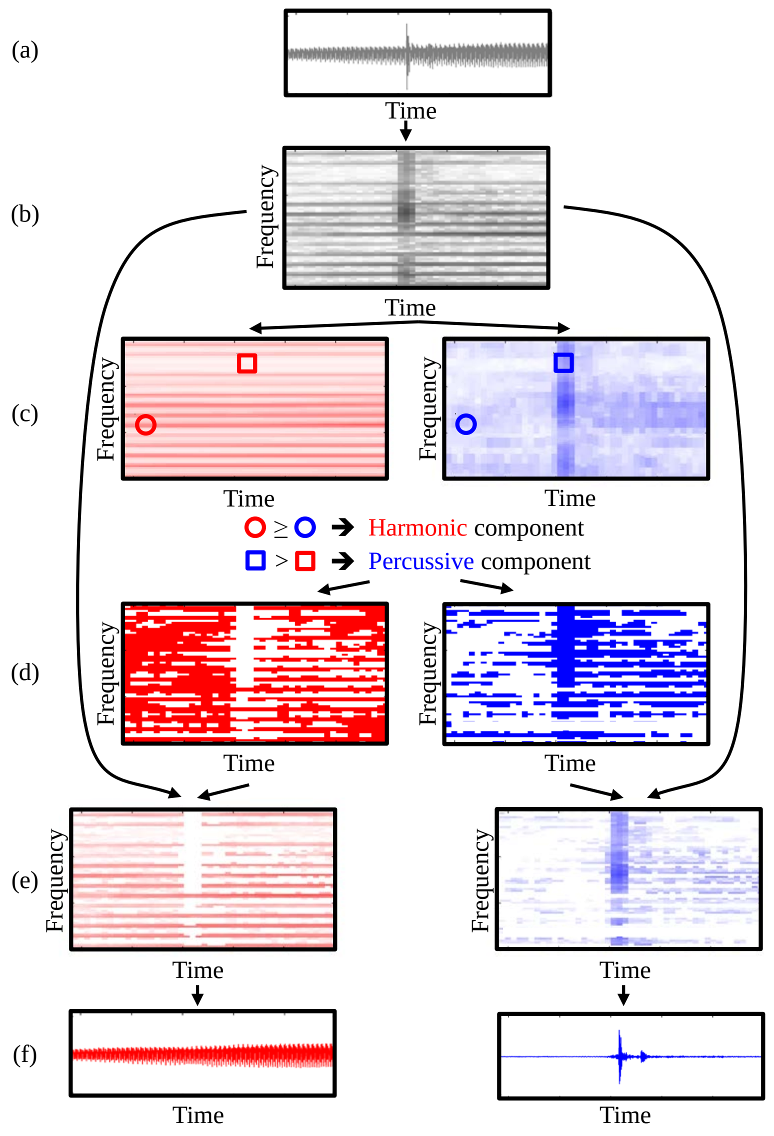


Figure 3.2: The harmonic-percussive separation procedure presented in [67]. **(a)**: Input audio signal x. **(b)**: STFT X of x. **(c)**: Horizontally and vertically enhanced spectrograms $\tilde{Y}_\mathrm{h}$ and $\tilde{Y}_\mathrm{p}$. **(d)**: Binary masks $\mathcal{M}_\mathrm{h}$ and $\mathcal{M}_\mathrm{p}$. **(e)**: Spectrograms of the harmonic and the percussive component X_h and X_p. **(f)**: Harmonic and percussive components x_h and x_p.

3.2 Separation Factor

In this section we present our first extension to Fitzgerald's fundamental harmonic-percussive separation procedure, that we discussed in the previous section. Given an input audio signal x, our goal is to compute a tight harmonic component x_h and a tight percussive component x_p such that x_h and x_p contain only the clearly harmonic and percussive sounds of x, respectively. The first steps of our proposed procedure are the same as in Fitzgerald's method. To this end, we compute the horizontally and vertically enhanced spectrograms $\tilde{Y}_\mathrm{h}$ and $\tilde{Y}_\mathrm{p}$ as defined in Equations (3.1)

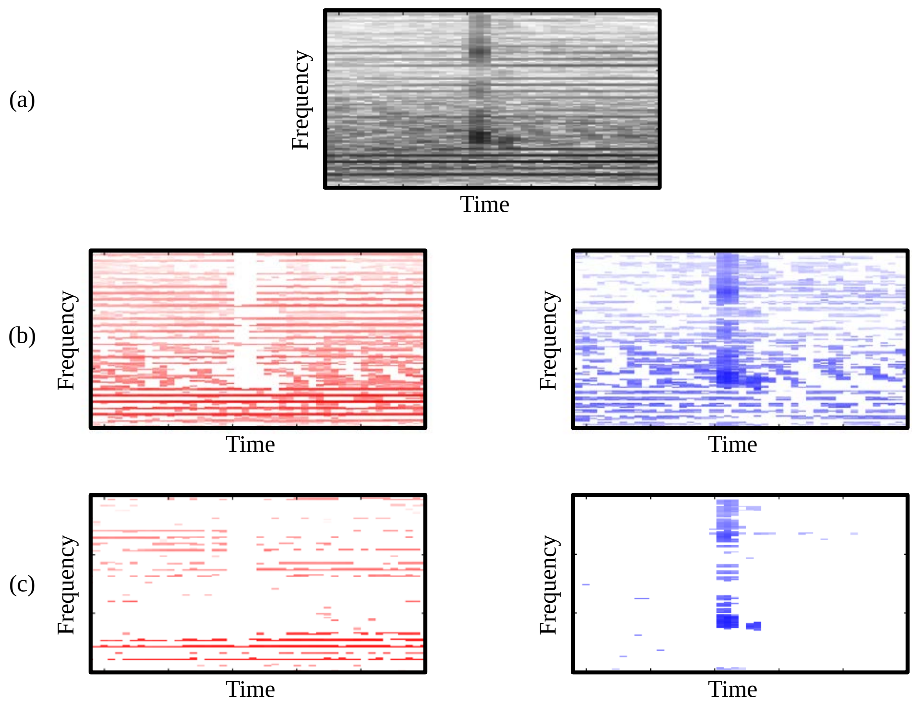


Figure 3.3: (a): Original magnitude spectrogram $|X|$ of the sound mixture of a violin, castanets, and applause. **(b):** Magnitude spectrograms $|X_{\mathrm{h}}|$ (left) and $|X_{\mathrm{p}}|$ (right) for $\beta = 1$. **(c):** Magnitude spectrograms $|X_{\mathrm{h}}|$ (left) and $|X_{\mathrm{p}}|$ (right) for $\beta = 3$.

and (3.2), respectively. Then, extending [67], we introduce an additional parameter $\beta \in \mathbb{R}$, $\beta \geq 1$, called the *separation factor*. We assume an entry of the original spectrogram $X(m, k)$ to be part of the clearly harmonic or percussive component if $\tilde{Y}_{\mathrm{h}}(m,k) > \beta \cdot \tilde{Y}_{\mathrm{p}}(m,k)$ or $\tilde{Y}_{\mathrm{p}}(m,k) \geq \beta \cdot \tilde{Y}_{\mathrm{h}}(m,k)$, respectively. Intuitively, for a sound to be included in the harmonic component it is required to stand out from the percussive portion of the signal by at least a factor of β, and vice versa for the percussive component. We therefore define the binary masks $\mathcal{M}_{\mathrm{h}}$ and $\mathcal{M}_{\mathrm{p}}$ by

$$\mathcal{M}_{\mathrm{h}}(m,k) = \begin{cases} 1, & \text{if } \tilde{Y}_{\mathrm{h}}(m,k) > \beta \cdot \tilde{Y}_{\mathrm{p}}(m,k), \\ 0, & \text{otherwise}, \end{cases} \tag{3.7}$$

$$\mathcal{M}_{\mathrm{p}}(m,k) = \begin{cases} 1, & \text{if } \tilde{Y}_{\mathrm{p}}(m,k) \geq \beta \cdot \tilde{Y}_{\mathrm{h}}(m,k), \\ 0, & \text{otherwise}. \end{cases} \tag{3.8}$$

Applying these masks to the original STFT X and transforming the resulting modified STFTs back to the time-domain yields the desired signals x_{h} and x_{p}. Choosing a separation factor $\beta > 1$ tightens the separation result of the procedure by preventing sounds which are neither clearly harmonic nor percussive to be included in the components. To demonstrate this, we extend the sound mixture used in Figure 3.2 of a violin (clearly harmonic) and castanets (clearly percussive) with applause (noise-like, and neither harmonic nor percussive). Figure 3.3a shows the spectrogram of this mixture. Again, the sound of the violin manifests itself as clear horizontal structures, while the click of the castanets is visible as a clear vertical structure in the middle of the spectrogram. However, the sound of the applause does not form any kind of directed structure and is spread all over the spectrum. When decomposing this audio signal with a separation factor of $\beta = 1$, which basically yields the procedure proposed in [67], the applause is more or less equally distributed among the harmonic and the percussive component, see Figure 3.3b. However, when choosing $\beta = 3$, only the clearly horizontal and vertical structures are preserved in X_{h} and X_{p}, respectively, and the applause is no longer contained in the two components, see Figure 3.3c.

3.3 Harmonic-Percussive-Residual (HPR) Separation

In the previous section we have seen how harmonic-percussive separation can be tightened such that the harmonic and the percussive components only contain the clearly harmonic and percussive sounds, respectively. Sounds that are neither of clearly harmonic nor percussive nature are therefore excluded from the resulting components. In this section we extend the tightened harmonic-percussive separation procedure with a third *residual* component that captures the excluded sounds. We first show a rather straight forward extension approach in Section 3.3.1. Afterwards, in Section 3.3.2, we discuss how the parameters of the proposed procedure influence the decomposition results. Finally, in Section 3.3.3, we present an iterative decomposition procedure which allows for a more flexible adjustment of the decomposition results.

3.3.1 Basic HPR Separation

The concept presented in Section 3.2 allows us to extend the decomposition procedure with a third component x_{r}, called the *residual component*. It contains the portion of the input signal x that is neither part of the harmonic component x_{h} nor the percussive components x_{p}. To compute x_{r}, we define the binary mask

$$\mathcal{M}_{\mathrm{r}}(m,k) \quad = \quad 1 - \big(\mathcal{M}_{\mathrm{h}}(m,k) + \mathcal{M}_{\mathrm{p}}(m,k)\big), \tag{3.9}$$

apply it to X, and transform the resulting spectrogram X_{r} back to the time-domain (note that the masks $\mathcal{M}_{\mathrm{h}}$ and $\mathcal{M}_{\mathrm{p}}$ are disjoint). This decomposition into three components is inspired by the STN audio model [133, 166, 217] and related formulations [175]. Here, an audio signal is analyzed to yield parameters for sinusoidal, transient, and noise components which can then be used to approximately resynthesize the original signal. While the main application of the STN model lies in the field of low bitrate audio coding, the estimated parameters can also be used to synthesize just the sinusoidal, the transient, or the noise component of the approximated signal. The harmonic, the percussive, and the residual component resulting from our proposed decomposition procedure are often perceptually similar to the STN components. However, our proposed procedure is conceptually different. STN modeling aims for a *parametrization* of the given audio signal. While the estimated parameters constitute a compact approximation of the input signal, this approximation and the original signal are not necessarily equal. Our proposed approach yields a *decomposition* of the signal. The three components always add up to the original signal again. The separation factor β hereby constitutes a flexible handle to adjust the sound characteristics of the components.

3.3.2 Influence of the Parameters

The main parameters of our decomposition procedure are the length of the median filters, the frame size N used to compute the STFT, and the separation factor β. Intuitively, the length of the filters specify the minimal sizes of horizontal and vertical structures which should be considered as harmonic and percussive sounds in the STFT of x, respectively. Our experiments have shown that the filter lengths actually do not influence the decomposition too much as long as no extreme values are chosen, see also [45]. The frame size N on the other hand pushes the overall energy of the input signal towards one of the components. For large frame sizes, the short percussive sounds lose influence in the spectral representation and more energy is assigned to the harmonic component. This results in a leakage of some percussive sounds to the harmonic component. Vice versa, for small frame sizes the low frequency resolution often leads to a blurring of horizontal structures, and harmonic sounds tend to leak into the percussive component. The separation factor β shows a different behavior to the previous parameters. The larger its value, the clearer becomes the harmonic and percussive nature of the components x_{h} and x_{p}. At the same time, also the portion of the signal that is assigned to the residual component x_{r} increases. To illustrate this behavior, let us first consider a synthetic example where we apply our proposed procedure to the mixture of a violin (clearly harmonic), castanets (clearly percussive), and applause (neither harmonic nor percussive), all sampled at 22050 Hertz and having the same energy. In Figure 3.4, we visualized the relative energy distribution of the three components for varying frame sizes N and separation factors β, while fixing the length of the median filters to be always equivalent to 200 milliseconds in horizontal direction and 500 Hertz in vertical direction, see also [45]. Since the energy of all three signals is normalized, potential leakage between the components is indicated by components that have either more or less than a third of the overall energy assigned. Considering Fitzgerald's procedure [67] as a baseline ($\beta = 1$), we can investigate its behavior by looking at the first columns of the matrices in Figure 3.4. While the residual component has zero energy in this setting, one can observe by listening that the applause is more or less equally distributed between the harmonic and the

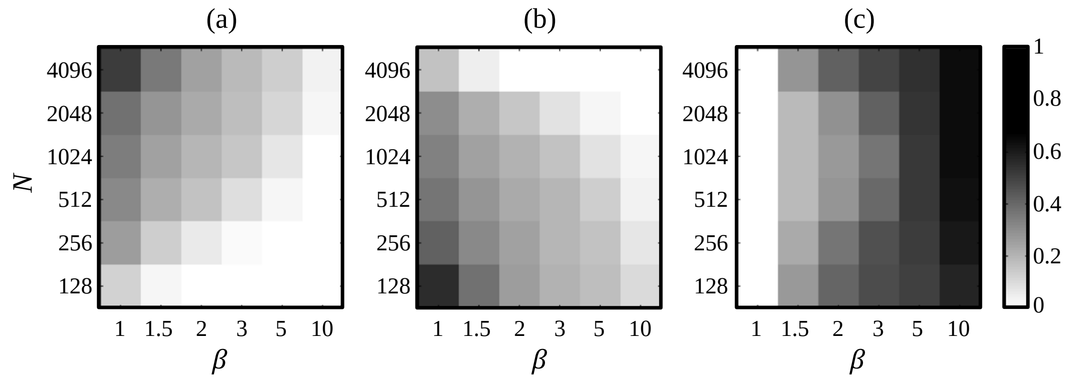


Figure 3.4: Energy distribution between the harmonic, percussive, and residual components for different frame sizes N and separation factors β. **(a):** Harmonic components. **(b):** Percussive components. **(c):** Residual components.

percussive component for medium frame sizes. This is also reflected in Figure 3.4a/b by the energy being split up roughly into equal portions. For very large N, most of the signal's energy moves towards the harmonic component (value close to one in Figure 3.4a for $\beta = 1, N = 4096$), while for very small N, the energy is shifted towards the percussive component (value close to one in Figure 3.4b for $\beta = 1$, $N = 128$). With increasing β, one can observe how the energy contained in the harmonic and the percussive component flows towards the residual component (decreasing values in Figure 3.4a/b and increasing values in Figure 3.4c for increasing β). Listening to the decomposition results shows that the harmonic and the percussive component thereby become more and more extreme in their respective characteristics. For medium frame sizes, this allows us to find settings that lead to decompositions in which the harmonic component contains the violin, the percussive component contains the castanets, and the residual contains the applause. This is reflected by Figure 3.4, where for $N = 1024$ and $\beta = 2$ the three sound components all hold roughly one third of the overall energy. For very large or very small frame sizes it is not possible to get such a clear separation. For example, considering $\beta = 1$ and $N = 4096$, we already observed that the harmonic component holds most of the signal's energy and also contains some of the percussive sounds. However, already for small $\beta > 1$ these percussive sounds are shifted towards the residual component (see the large amount of energy assigned to the residual in Figure 3.4c for $\beta = 1.5$, $N = 4096$). Furthermore, also the energy from the percussive component moves towards the residual. The large frame size therefore results in a very clear harmonic component while the residual holds both the percussive as well as all other non-harmonic sounds, leaving the percussive component virtually empty. For very small N the situation is exactly the other way around. This observation can be exploited to define a refined decomposition procedure which we discuss in the next section.

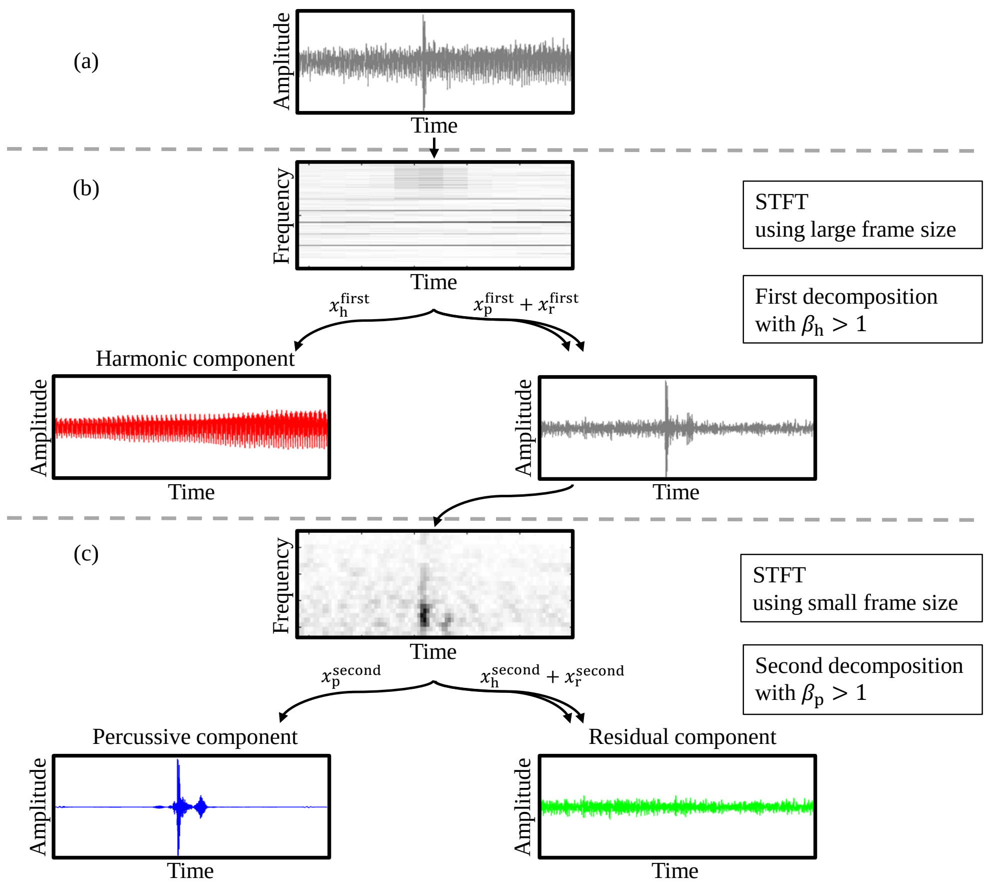

(a)
Amplitude
Time
(b)
Frequency
Time
STFT using large frame size
x_h^{first}
$x_p^{first} + x_r^{first}$
First decomposition with $\beta_h > 1$
Harmonic component
Amplitude
Time
Amplitude
Time
(c)
Frequency
Time
STFT using small frame size
x_p^{second}
$x_h^{second} + x_r^{second}$
Second decomposition with $\beta_p > 1$
Percussive component
Residual component
Amplitude
Time
Amplitude
Time

Figure 3.5: Overview of the refined procedure. **(a):** Input signal x. **(b):** First run of the decomposition procedure using a large frame size N_h and a separation factor β_h. **(c):** Second run of the decomposition procedure using a small frame size N_p and a separation factor β_p.

3.3.3 Iterative HPR Separation

In [207], Tachibana et al. described a method for the extraction of human singing voice from music recordings. In this algorithm, the singing voice is estimated by iteratively applying the harmonic-percussive decomposition procedure described in [156] first to the input signal and afterwards again to one of the resulting components. This yields a decomposition of the input signal into three components, one of which containing the estimate of the singing voice. The core idea of this algorithm is to perform the two harmonic-percussive separations on spectrograms with two different time-frequency resolutions. In particular, one of the spectrograms is based on a large frame size and the other on a small frame size. Using this idea, we now extend our proposed harmonic-percussive-residual separation procedure presented in Section 3.3.1. Although it is possible to find a good combination of N and β for this procedure such that both the harmonic as well as the percussive component represent the respective characteristics of the input signal well (see Section 3.3.2), the computation of the two components is still coupled. It

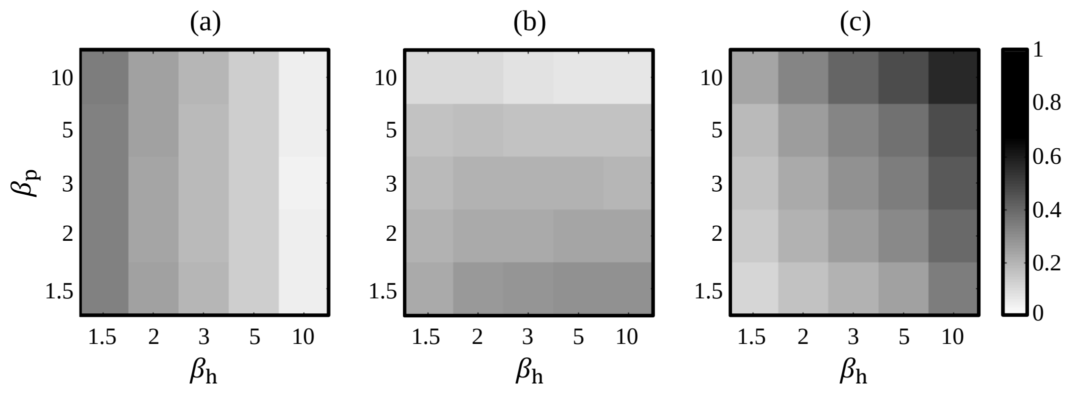


Figure 3.6: Energy distribution between the harmonic, percussive, and residual components for different separation factors β_h and β_p. **(a):** Harmonic components. **(b):** Percussive components. **(c):** Residual components.

is therefore not clear how to adjust the content of the harmonic and the percussive component independently. Having made the observation that large N lead to good harmonic but poor percussive/residual components for $\beta > 1$, while small N lead to good percussive components but poor harmonic/residual components for $\beta > 1$, we build on the idea from Tachibana et al. [207] and compute the decomposition in two iterations. Here, the goal is to decouple the computation of the harmonic component from the computation of the percussive component. First, the harmonic component is extracted by applying our basic procedure with a large frame size N_h and a separation factor $\beta_\mathrm{h} > 1$, yielding $x_\mathrm{h}^\mathrm{first}$, $x_\mathrm{p}^\mathrm{first}$, and $x_\mathrm{r}^\mathrm{first}$. In a second run, the procedure is applied again to the sum $x_\mathrm{p}^\mathrm{first} + x_\mathrm{r}^\mathrm{first}$, this time using a small frame size N_p and a second separation factor $\beta_\mathrm{p} > 1$. This yields the components $x_\mathrm{h}^\mathrm{second}$, $x_\mathrm{p}^\mathrm{second}$, and $x_\mathrm{r}^\mathrm{second}$. Finally, we define the output components of the procedure to be

$$x_\mathrm{h} = x_\mathrm{h}^\mathrm{first}\,, \tag{3.10}$$

$$x_\mathrm{p} = x_\mathrm{p}^\mathrm{second}\,, \tag{3.11}$$

$$x_\mathrm{r} = x_\mathrm{h}^\mathrm{second} + x_\mathrm{r}^\mathrm{second}\,. \tag{3.12}$$

For an overview of the procedure see Figure 3.5. While fixing the values of N_h and N_p to a small and a large frame size, respectively (in our experiments we chose $N_\mathrm{h} = 4096$ and $N_\mathrm{p} = 256$), the separation factors β_h and β_p yield handles that give simple and independent control over the harmonic and percussive component. Figure 3.6, which is based on the same audio example as Figure 3.4, shows the energy distribution among the three components for different combinations of β_h and β_p, see also [45]. For the harmonic components (Figure 3.6a) we see that the portion of the signals energy contained in this component is independent of β_p and can be controlled purely by β_h. This is a natural consequence from the fact that in our proposed procedure the harmonic component is always computed directly from the input signal x and β_{p} does not influence its computation at all. However, we can also observe that the energy contained in the percussive component (Figure 3.6b) is fairly independent of β_{h} and can be controlled almost solely by β_{p}. Listening to the decomposition results confirms these observations. Our proposed iterative procedure therefore allows to adjust the harmonic and the percussive component almost independently what significantly simplifies the process of finding an appropriate parameter setting for a given input signal. Note that in principle it would also be possible to choose $\beta_{\mathrm{h}} = \beta_{\mathrm{p}} = 1$, resulting in an iterative application of Fitzgerald's method [67]. However, as discussed in Section 3.3.2, Fitzgerald's method suffers from component leakage when using very large or small frame sizes. Therefore, most of the input signal's energy will be assigned to the harmonic component in the first iteration of the algorithm, while most of the remaining portion of the signal is assigned to the percussive component in the second iteration. This leads to a very weak, although not empty, residual component.

3.4 Evaluation

In a first experiment, we applied objective evaluation measures to our running example. Assuming that the violin, the castanets, and the applause signal represent the characteristics that we would like to capture in the harmonic, the percussive, and the residual components, respectively, we treated the decomposition task of this mixture as a source separation problem. In an optimal decomposition the harmonic component would contain the original violin signal, the percussive component the castanets signal, and the residual component the applause. To evaluate the decomposition quality, we computed the *source to distortion ratios* (SDR), the *source to interference ratios* (SIR), and the *source to artifacts ratios* (SAR) [220] for the decomposition results of the following procedures.

As a baseline (BL), we simply considered the original mixture as an estimate for all three sources. Furthermore, we applied the standard harmonic-percussive separation procedure by Fitzgerald [67] (HP) with the frame size set to $N = 1024$, the HP method applied iteratively (HP-I) with $N_{\mathrm{h}} = 4096$ and $N_{\mathrm{p}} = 256$, the proposed basic harmonic-percussive-residual separation procedure (HPR) as described in Section 3.3.1 with $N = 1024$ and $\beta = 2$, and the proposed iterative harmonic-percussive-residual separation procedure (HPR-I) as described in Section 3.3.3 with $N_{\mathrm{h}} = 4096$, $N_{\mathrm{p}} = 256$, and $\beta_{\mathrm{h}} = \beta_{\mathrm{p}} = 2$. As a final method, we also considered HPR-I with separation factor $\beta_{\mathrm{h}} = 3$ and $\beta_{\mathrm{p}} = 2.5$, which were optimized manually for the task at hand (HPR-IO). The filter lengths in all procedures were always fixed to be equivalent to 200 milliseconds in time direction and 500 Hertz in frequency direction. Decomposition results for all procedures can be found at [45].

	SDR			**SIR**			**SAR**		
	Violin	Castanets	Applause	Violin	Castanets	Applause	Violin	Castanets	Applause
BL	−3.10	−2.93	−3.04	−3.10	−2.93	−3.04	274.25	274.25	274.25
HP	−5.85	3.58	—	−5.09	6.06	—	8.33	8.14	—
HP-I	0.08	2.86	−7.03	1.08	10.45	14.69	9.44	4.07	−6.85
HPR	8.23	8.29	4.25	17.69	22.34	8.41	8.82	8.49	6.95
HPR-I	7.65	9.14	4.93	14.58	20.66	12.80	8.78	9.50	5.93
HPR-IO	8.85	9.28	5.00	21.65	24.41	9.04	9.11	9.44	7.69

Table 3.1: Objective evaluation measures. All values are given in dB.

The results are listed in Table 3.1. All values are given in dB and higher values indicate better results. As expected, BL yields rather low SDR and SIR values for all components, while the SAR values are excellent since there are no artifacts present in the original mixture. The method HP yields low evaluation measures as well. However, these values are to be taken with care since HP decomposes the input mixture in just a harmonic and a percussive component. The applause is therefore not estimated explicitly and, as also discussed in Section 3.2, randomly distributed among the harmonic and percussive component. It is therefore clear that especially the SIR values are low in comparison to the other procedures since the applause heavily interferes with the remaining two sources in the computed components. When looking at HP-I, the benefit of having a third component becomes clear. Although here the residual component does not capture the applause very well (SDR of −7.03 dB) this already suffices to yield SDR and SIR values clearly above the baseline for the estimates of the violin and the castanets. The separation quality further improves when considering the results of our proposed method HPR. Here, the evaluation yields high values for all measures and components. The very high SIR values are particularly noticeable since they indicate that the three sources are separated very clearly with very little leakage between the components. This confirms our claim that our proposed concept of a separation factor allows for *tightening* decomposition results as described in Section 3.2. The results of HPR-I are very similar to the results for the basic procedure HPR. However, listening to the decomposition reveals that the harmonic and the percussive component still contain some slight residue sounds of the applause. Slightly increasing the separation factors to $\beta_\mathrm{h} = 3$ and $\beta_\mathrm{p} = 2.5$ (HPR-IO) eliminates these residues and further increases the evaluation measures. This straight-forward adjustment is possible since the two separation factors β_h and β_p constitute independent handles to adjust the content of the harmonic and percussive component, what demonstrates the flexibility of our proposed procedure.

Item name	**Description**
CastanetsViolinApplause	Synthetic mixture of a violin, castanets and applause.
Heavy	Recording of heavily distorted guitars, a bass and drums.
Stepdad	Excerpt from *My Leather, My Fur, My Nails* by the band *Stepdad*.
Bongo	Regular beat played on bongos.
Glockenspiel	Monophonic melody played on a glockenspiel.
Winterreise	Excerpt from "Gute Nacht" by Franz Schubert which is part of the *Winterreise* song cycle. It is a duet of a male singer and piano.

Table 3.2: List of audio excerpts.

The above described experiment constitutes a first case study for the objective evaluation of our proposed decomposition procedures, based on an artificially mixed example. To also evaluate these procedures on real-world audio data, we additionally performed an informal subjective listening tests with several test participants. To this end, we applied our procedures to the set of audio excerpts listed in Table 3.2. Among the excerpts are complex sound mixtures as well as purely percussive and harmonic signals, see also [45]. Raising the question whether the computed harmonic and percussive components meet the expectation of representing the clearly harmonic or percussive portions of the audio excerpts, respectively, the performed listening test confirmed our hypothesis. It furthermore turned out that $\beta_\mathrm{h} = \beta_\mathrm{p} = 2$, $N_\mathrm{h} = 4096$ and $N_\mathrm{p} = 256$ seems to be a setting for our iterative procedure which robustly yields good decomposition results, rather independent of the input signal. Regarding the residual component, it was often described to sound like a *sound texture* by the test participants, which is a very interesting observation. Although there is no clear definition of what a sound texture exactly is, literature states "sound texture is like wallpaper: it can have local structure and randomness, but the characteristics of the fine structure must remain constant on the large scale" [182]. In our opinion this is not a bad description of what one can hear in residual components.

3.5 Applications of HPR Separation

Concerning applications of our procedure, we see a potential use for example in extracting enhanced audio features. Such approaches have already been used, where harmonic-percussive separation is applied to improve chroma features [213] or tempo features [76]. As an illustrative example we computed chroma features (standard harmony-based features, see Section 2.3) and a simple energy-based novelty curve (as they can be used for onset detection and beat tracking,

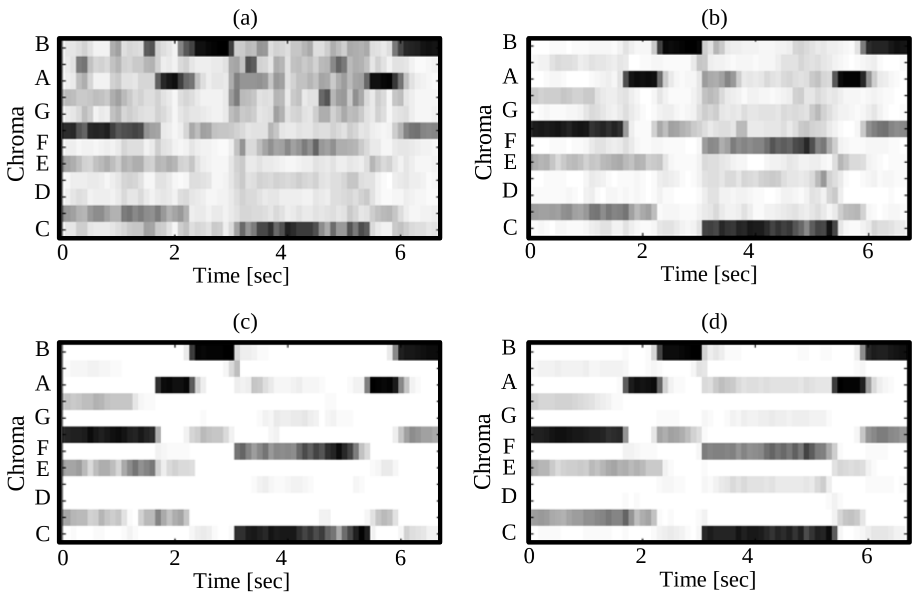


Figure 3.7: Chroma features. **(a):** Original mixture signal. **(b):** Harmonic component of the procedure proposed in [67]. **(c):** Harmonic component of our refined procedure. **(d):** Original violin signal.

see Section 2.5) for our running example (the mixture of a violin, castanets and applause). The results are visualized in Figure 3.7 and 3.8. In Figure 3.7a, one can see the chroma sequence computed for the mixture signal. Note that the presence of the castanets as well as the applause in the signal lead to rather noisy features. Applying Fitzgerald's method and computing the features on just the harmonic component only slightly denoises the features since a part of the applause is still present in this component, see Figure 3.7b. However, utilizing the separation factor $\beta_\mathrm{h} = 3.5$ yields a much cleaner feature sequence, see Figure 3.7c. As comparison, we also computed the chroma features on the pure violin signal which is visualized in Figure 3.7d. One can see that this sequence actually does not differ too much from the sequence computed on the harmonic component of our procedure.

Figure 3.8a shows an excerpt of the novelty curve computed for the mixture signal. On can see that, although each annotated castanet onset—visualized by a vertical gray line—is aligned with a peak of the curve, there are further spurious peaks present. The same is true for the novelty curve that is computed on the percussive component of Fitzgerald's procedure visualized in Figure 3.8b. Again, this is due to the fact that this component still contains residual sounds of the applause. When applying our proposed refined decomposition procedure with $\beta_\mathrm{p} = 5$, these spurious peaks do not occur any more, similar as in the novelty curve which was computed on the pure castanets signal, see Figure 3.8c/d. Of course, these examples only indicate tendencies and a detailed evaluation, e.g., in the context of chord recognition and beat tracking, is left as future work.

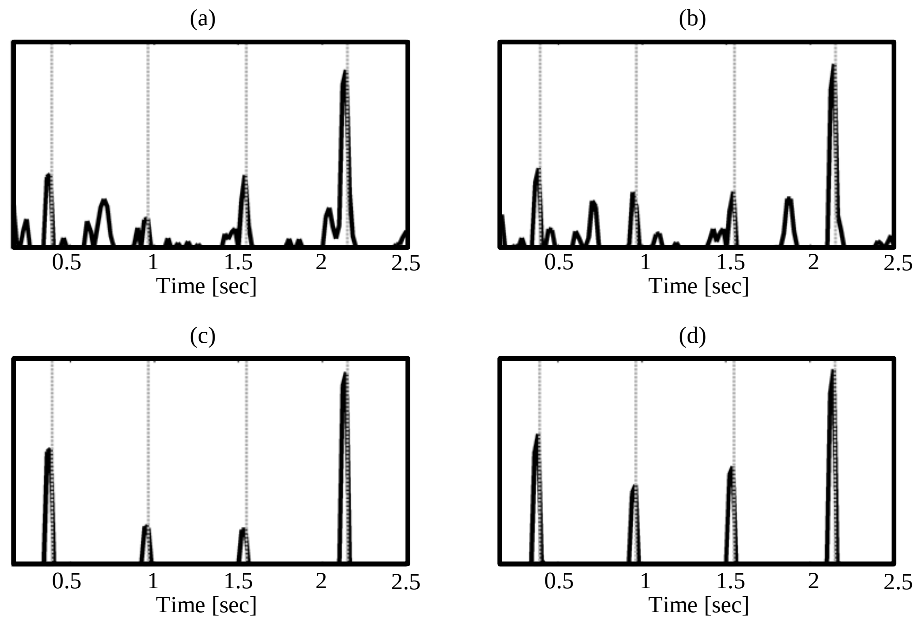


Figure 3.8: Novelty curves. Gray vertical lines indicate the ground truth onset positions. **(a):** Original mixture signal. **(b):** Percussive component of the procedure proposed in [67]. **(c):** Percussive component of our refined procedure. **(d):** Original castanets signal.

A third application lies in the field of *time-scale modification* (TSM) of audio signals. A problem of many basic time-scale modification algorithms is that they can process signals of purely harmonic or purely percussive nature well, while introducing noticeable artifacts to signals of the opposite class. In Chapter 4 we show how harmonic-percussive decomposition techniques can be utilized to improve the quality of TSM algorithms by first decomposing a given audio signal into a set of component signals and processing the components separately afterwards.

Finally, it has shown that our proposed decomposition technique can also be applied in the context of *singing voice extraction*, see Chapter 5. The task of decomposing a given music recording of a singer accompanied by musical instruments into one component that holds the singing voice and second component containing the accompaniment is very difficult in general. However, similar as for the TSM problem, it turned out that it is beneficial to first decompose a given recording with the presented harmonic-percussive-residual decomposition technique and then process the three components individually.

3.6 Conclusions and Further Notes

In this chapter, we introduced novel methods for decomposing music signals into harmonic, percussive, and residual sound components. Our proposed procedures are based on the observation that harmonic sounds tend to have horizontal structures in spectrogram representations (in

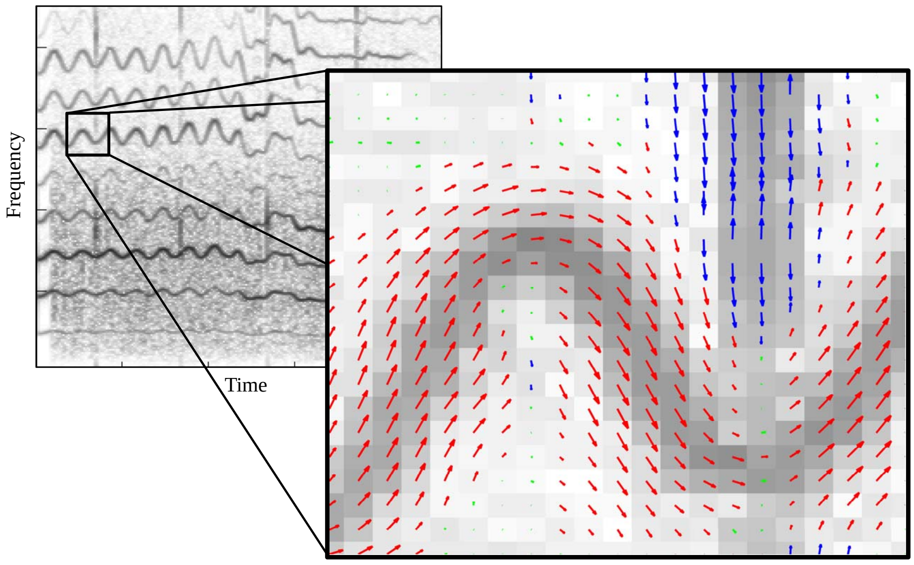


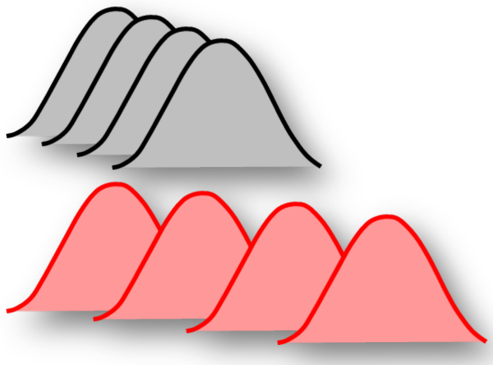
Figure 3.9: HPR Decomposition using the Structure Tensor. Based on the estimated orientation of structures (indicated by arrows), time-frequency bins are assigned to either the harmonic (red), the percussive (blue) or the residual (green) component. Example taken from [72].

time direction), while percussive sounds usually form vertical structures (in frequency direction). However, this is not always true. For example, a tone sung with strong vibrato (periodic frequency fluctuation) or glissando (continuous frequency fluctuation) is usually considered to be a harmonic sound. The magnitude spectrogram of singing voice, superimposed with clicks of castanets and applause, can be seen in Figure 3.9. Here we can observe that the singing voice does not form horizontal, but wave-like structures due to the vibrato. Our proposed HPR decomposition procedures would therefore not assign it to the harmonic component since this component captures—by design—only clearly horizontal structures.

In the context of standard harmonic-percussive separation, this problem has been recently approached by Park and Lee in [163] with a strategy based on non-negative matrix factorization. Another interesting approach was proposed by Füg et al. who extended our HPR Decomposition method [72]. Their core idea is to use the Structure Tensor [8, 119] known from image processing to derive information about the orientation of structures in magnitude spectrograms. This information is visualized in Figure 3.9 by colored arrows. Dependent on the computed orientation (orientation of arrows) and confidence measures (length of arrows), time-frequency bins are assigned to either the harmonic (red), the percussive (blue), or residual (green) component. This allows the method to also capture vibrato or glissando sounds in the harmonic component. Sound examples for the HPR decomposition based on the Structure Tensor can be found at [71].

Chapter 4

Contributions to Time-Scale Modification of Music Signals



This chapter is based on our review article on *time-scale modification* (TSM) of music signals [44] as well as our contributions published in [48, 40].

Time-scale modification procedures are digital signal processing methods for stretching or compressing the duration of a given audio signal. Ideally, the time-scale modified signal should sound as if the original signal's content was performed at a different tempo while preserving properties such as pitch and timbre. TSM procedures are applied in a wide range of scenarios. For example, they simplify the process of creating music remixes. Music producers or DJs apply TSM to adjust the durations of music recordings, enabling synchronous playback [19, 110]. Nowadays TSM is built into music production software as well as hardware devices. A second application scenario is adjusting an audio stream's duration to that of a given video clip. For example, when generating a slow motion video, it is often desirable to also slow down the tempo of the associated audio stream. Here, TSM can be used to synchronize the audio material with the video's visual content [143].

A main challenge for TSM procedures is that music signals are complex sound mixtures, consisting of a wide range of different sounds. As an example, imagine a music recording consisting of a violin playing together with castanets. When modifying this music signal with a TSM procedure, both the harmonic sound of the violin as well as the percussive sound of the castanets should be preserved in the output signal. To keep the violin's sound intact, it is essential to maintain its pitch as well as its timbre. On the other hand, the clicking sound of the castanets does not have a pitch—it is much more important to maintain the crisp sound of the single clicks, as well as their exact relative time positions, in order to preserve the original rhythm. Retaining these contrasting characteristics usually requires conceptually different TSM approaches. For example, classical TSM procedures based on waveform similarity overlap-add (WSOLA) [216] or on the phase vocoder (PV-TSM) [69, 168, 128] are capable of preserving the perceptual quality of harmonic signals to a high degree, but introduce noticeable artifacts when modifying percussive signals. However, as we show in this chapter, it is possible to substantially reduce artifacts by combining different TSM approaches.

This chapter is structured as follows. In Section 4.1 we first discuss fundamental challenges and algorithmic approaches in the field of TSM, review well-known TSM methods, and discuss their respective advantages and drawbacks. Afterwards, we propose two novel TSM procedures that are both based on combining different TSM strategies. In Section 4.2 we introduce the first procedure that improves on the fundamental approaches by first decomposing a given audio signal into a harmonic and a percussive sound component. The two components are then modified separately. The second method, discussed in Section 4.3, extends a fundamental TSM procedure that is capable of scaling music signals with high quality, but not by arbitrary factors. Here, the idea is to cascade the procedure with our proposed method based on harmonic-percussive separation in order to allow for arbitrary modifications. Finally, we point out different application scenarios for TSM such as music synchronization and pitch-shifting in Section 4.4 and close this chapter in Section 4.5.

4.1 Fundamentals on Time-Scale Modification (TSM)

4.1.1 Basic TSM Strategy

As mentioned above, a key requirement for time-scale modification procedures is that they change the time-scale of a given audio signal without altering its pitch content. To achieve this goal,

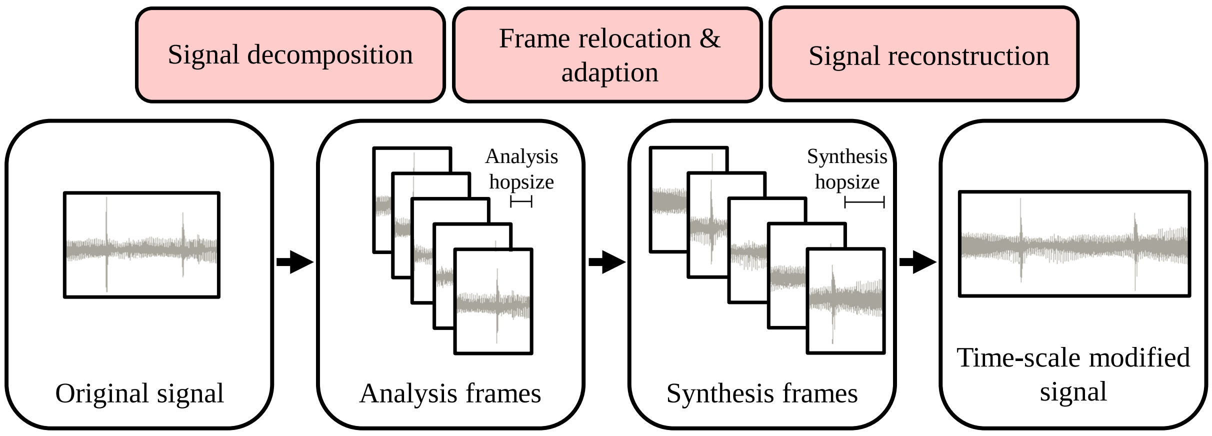


Figure 4.1: Generalized processing pipeline of TSM procedures.

many TSM procedures follow a common fundamental strategy which is sketched in Figure 4.1. The core idea is to decompose the input signal into short *frames*. Having a fixed length, usually in the range of 50 to 100 milliseconds of audio material, each frame captures the local pitch content of the signal. The frames are then relocated on the time axis to achieve the actual time-scale modification, while, at the same time, preserving the signal's pitch.

More precisely, this process can be described as follows. The input of a TSM procedure is a discrete-time audio signal $x : \mathbb{Z} \to \mathbb{R}$, equidistantly sampled at a sampling rate of F_s as described in Chapter 2. Recall that although audio signals typically have a finite length of $L \in \mathbb{N}$ samples $x(r)$ for $r \in [0 : L-1]$, for the sake of simplicity, we model them to have an infinite support by defining $x(r) = 0$ for $r \in \mathbb{Z} \setminus [0 : L-1]$. The first step of the TSM procedure is to split x into short *analysis frames* x_m, $m \in \mathbb{Z}$, each of them having a length of N samples (in the literature, the analysis frames are sometimes also referred to as *grains*, see [230]). The analysis frames are spaced by an *analysis hopsize* $H_\mathrm{a} \in \mathbb{N}$:

$$x_m(r) = \begin{cases} x(r + mH_\mathrm{a}), & \text{if } r \in [-N/2 : N/2-1], \\ 0, & \text{otherwise.} \end{cases} \tag{4.1}$$

In a second step, these frames are relocated on the time axis with regard to a specified *synthesis hopsize* $H_\mathrm{s} \in \mathbb{N}$. This relocation accounts for the actual modification of the input signal's time-scale by a *stretching factor* $\alpha = H_\mathrm{s}/H_\mathrm{a} \in \mathbb{Q}^+$. Since it is often desirable to have a specific overlap of the relocated frames, the synthesis hopsize H_s is often fixed (common choices are $H_\mathrm{s} = N/2$ or $H_\mathrm{s} = N/4$) while the analysis hopsize is given by $H_\mathrm{a} = H_\mathrm{s}/\alpha$. However, simply superimposing the overlapping relocated frames would lead to undesired artifacts such as phase discontinuities at the frame boundaries and amplitude fluctuations. Therefore, prior to signal reconstruction, the analysis frames are suitably adapted to form *synthesis frames* y_m. In the

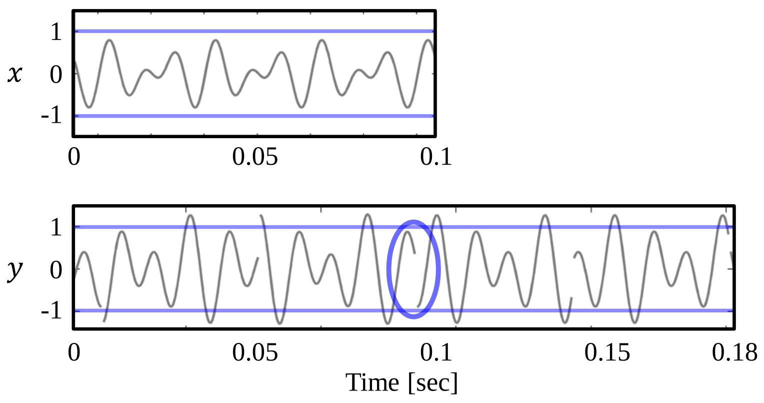


Figure 4.2: Typical artifacts that occur when choosing the synthesis frames y_m to be equal to the analysis frames x_m. The input signal x is stretched by a factor of $\alpha = 1.8$. The output signal y shows discontinuities (blue oval) and amplitude fluctuations (indicated by blue lines).

final step, the synthesis frames are superimposed in order to reconstruct the actual time-scale modified output signal $y : \mathbb{Z} \to \mathbb{R}$ of the TSM procedure:

$$y(r) = \sum_{m \in \mathbb{Z}} y_m(r - mH_\mathrm{s}) \ . \tag{4.2}$$

Although this fundamental strategy seems straightforward at a first glance, there are many pitfalls and design choices that may strongly influence the perceptual quality of the time-scale modified output signal. The most obvious question is how to adapt the analysis frames x_m in order to form the synthesis frames y_m. There are many ways to approach this task, leading to conceptually different TSM procedures. In the following, we discuss several strategies.

4.1.2 TSM Based on Overlap-Add (OLA)

4.1.2.1 The Procedure

In the general scheme described in the previous section, a straightforward approach would be to simply define the synthesis frames y_m to be equal to the unmodified analysis frames x_m. This strategy, however, immediately leads to two problems which are visualized in Figure 4.2. First, when reconstructing the output signal by using Equation (4.2), the resulting waveform typically shows discontinuities—perceivable as clicking sounds—at the unmodified frames' boundaries. Second, the synthesis hopsize H_s is usually chosen such that the synthesis frames are overlapping. When superimposing the unmodified frames, this typically leads to an undesired increase of the output signal's amplitude.

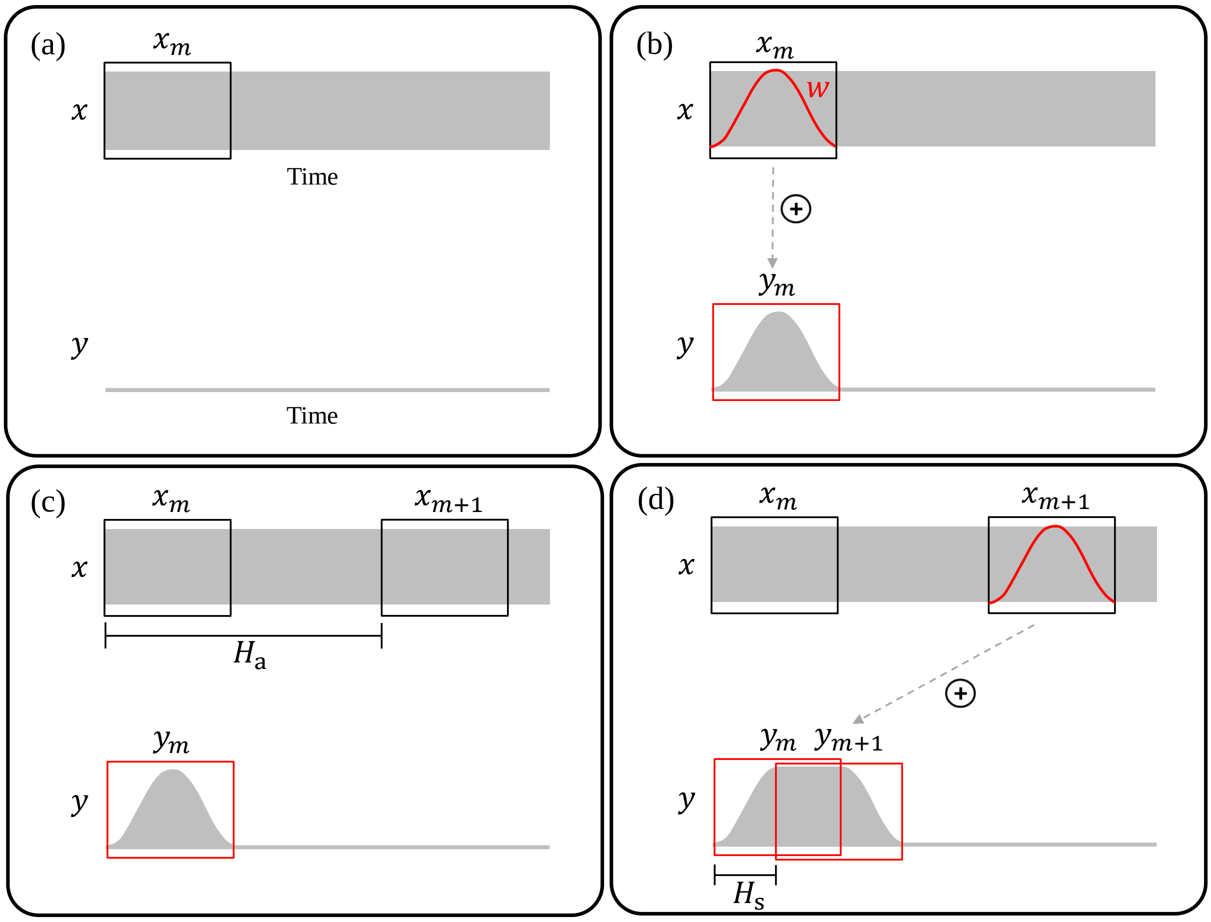


Figure 4.3: The principle of TSM based on overlap-add (OLA). **(a)**: Input audio signal x with analysis frame x_m. The output signal y is constructed iteratively. **(b)**: Application of Hann window function w to the analysis frame x_m resulting in the synthesis frame y_m. **(c)**: The next analysis frame x_{m+1} having a specified distance of H_a samples from x_m. **(d)**: Overlap-add using the specified synthesis hopsize H_s.

A basic TSM procedure should both enforce a smooth transition between frames as well as compensate for unwanted amplitude fluctuations. The idea of the *overlap-add* (OLA) TSM procedure is to apply a window function w to the analysis frames, prior to the reconstruction of the output signal y. The task of the window function is to remove the abrupt waveform discontinuities at the the analysis frames' boundaries. A typical choice for w is a *Hann window* function

$$w(r) = \begin{cases} 0.5 \cdot \left(1 - \cos\left(\frac{2\pi(r+N/2)}{N-1}\right)\right), & \text{if } r \in [-N/2 : N/2-1], \\ 0, & \text{otherwise.} \end{cases} \tag{4.3}$$

The Hann window has the nice property that

$$\sum_{n\in\mathbb{Z}} w\left(r - n\frac{N}{2}\right) = 1\ , \tag{4.4}$$

for all $r \in \mathbb{Z}$. The principle of the iterative OLA procedure is visualized in Figure 4.3. For the frame index $m \in \mathbb{Z}$, we first use Equation (4.1) to compute the m^{th} analysis frame x_m (Figure 4.3a). Then, we derive the synthesis frame y_m by

$$y_m(r) = \frac{w(r) \cdot x_m(r)}{\sum_{n \in \mathbb{Z}} w(r - nH_\mathrm{s})} \, . \tag{4.5}$$

The nominator of Equation (4.5) constitutes the actual windowing of the analysis frame by multiplying it pointwise with the given window function. The denominator normalizes the frame by the sum of the overlapping window functions, which prevents amplitude fluctuations in the output signal. Note that, when choosing w to be a Hann window and $H_\mathrm{s} = N/2$, the denominator always reduces to one by Equation (4.4). This is the case in Figure 4.3b where the synthesis frame's amplitude is not scaled before being added to the output signal y. Proceeding to the next analysis frame x_{m+1}, (Figure 4.3c), this frame is again windowed, overlapped with the preceding synthesis frame, and added to the output signal (Figure 4.3d). Note that Figure 4.3 visualizes the case where the original signal is compressed ($H_\mathrm{a} > H_\mathrm{s}$). Stretching the signal ($H_\mathrm{a} < H_\mathrm{s}$) works in exactly the same fashion. In this case, the analysis frames overlap to a larger degree than the synthesis frames.

OLA is an example of a *time-domain* TSM procedure where the modifications to the analysis frames are applied purely in the time-domain. In general, time-domain TSM procedures are not only efficient but also preserve the timbre of the input signal to a high degree. On the downside, output signals produced by OLA often suffer from other artifacts, as we explain next.

4.1.2.2 Artifacts

The OLA procedure is in general not capable of preserving local periodic structures that are present in the input signal. This is visualized in Figure 4.4 where a periodic input signal x is stretched by a factor of $\alpha = 1.8$ using OLA. When relocating the analysis frames, the periodic structures of x may not align any longer in the superimposed synthesis frames. In the resulting output signal y, the periodic patterns are distorted. These distortions are also known as *phase jump artifacts*. Since local periodicities in the waveforms of audio signals correspond to harmonic sounds, OLA is not suited to modify signals that contain harmonic components. When applied to harmonic signals, the output signals of OLA have a characteristic *warbling* sound, which is a kind of periodic frequency modulation [128]. Since most music signals contain at least some harmonic sources (as for example singing voice, piano, violins, or guitars), OLA is usually not suited to modify music.

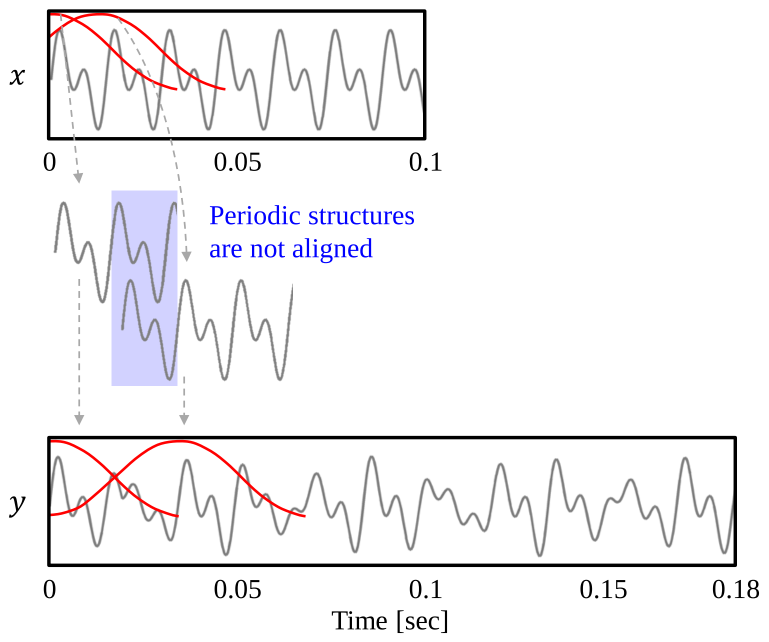


Figure 4.4: Illustration of a typical artifact for an audio signal modified with OLA. In this example, the input signal x is stretched by a factor of $\alpha = 1.8$. The OLA procedure is visualized for two frames indicated by window functions. The periodic structure of x is not preserved in the output signal y.

4.1.2.3 Tricks of the Trade

While OLA is unsuited for modifying audio signals with harmonic content, we found out in [48] that it delivers high quality results for purely percussive signals, as is the case with drum or castanet recordings. This is because audio signals with percussive content seldom have local periodic structures, but are more of noise-like or transient nature. The phase jump artifacts introduced by OLA are therefore not noticeable in the output signals. In this scenario, it is important to choose a very small frame length N (corresponding to roughly 10 milliseconds of audio material) in order to reduce the effect of *transient doubling*, an artifact that we discuss at length in Section 4.1.3.2. Although transients are slightly stretched in length when modified with OLA, they are still perceptually appealing and crisp for reasonable α (we experimented with $\alpha \in [0.5, 3]$). The conclusion is that OLA can be used to modify noise-like signals, while preserving transients without any explicit transient detection step.

4.1.3 TSM Based on Waveform Similarity Overlap-Add (WSOLA)

4.1.3.1 The Procedure

One of OLA's problems is that it lacks signal sensitivity: the windowed analysis frames are copied from fixed positions in the input signal to fixed positions in the output signal. In other words,

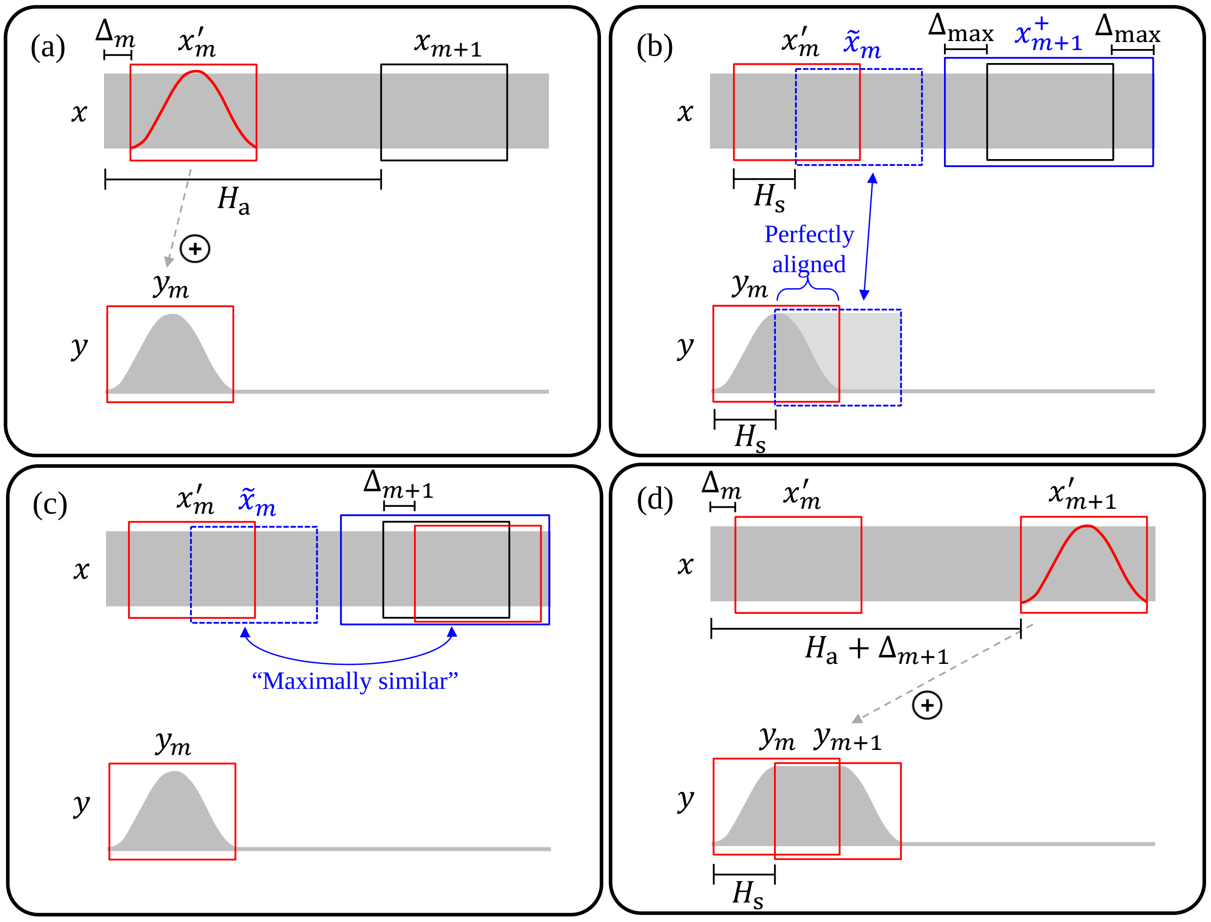


Figure 4.5: The principle of WSOLA. **(a)**: Input audio signal x with the adjusted analysis frame x'_m. The frame was already windowed and copied to the output signal y. **(b)/(c)**: Retrieval of a frame from the extended frame region x^+_{m+1} (solid blue box) that is as similar as possible to the natural progression $\tilde{x}_m$ (dashed blue box) of the adjusted analysis frame x'_m. **(d)**: The adjusted analysis frame x'_{m+1} is windowed and copied to the output signal y.

the input signal has no influence on the procedure. One time-domain strategy to reduce phase jump artifacts as caused by OLA is to introduce some flexibility in the TSM process, achieved by allowing some tolerance in the analysis or synthesis frames' positions. The main idea is to adjust successive synthesis frames in such a way that periodic structures in the frames' waveforms are aligned in the overlapping regions. Periodic patterns in the input signal are therefore maintained in the output. In the literature, several variants of this idea have been described, such as *synchronized OLA* (SOLA) [180], *time-domain pitch-synchronized OLA* (TD-PSOLA) [144], or autocorrelation-based approaches [126].

Another well-known procedure is *waveform similarity-based OLA* (WSOLA) [216]. This method's core idea is to allow for slight shifts (of up to $\pm\Delta_\mathrm{max} \in \mathbb{Z}$ samples) of the analysis frames' positions. Figure 4.5 visualizes the principle of WSOLA. Similar to OLA, WSOLA proceeds in an iterative fashion. Assume that in the m^{th} iteration the position of the analysis frame x_m was shifted by $\Delta_m \in [-\Delta_{\max} : \Delta_{\max}]$ samples. We call this frame the *adjusted analysis frame* x'_m (Figure 4.5a):

$$x'_m(r) = \begin{cases} x(r + mH_\mathrm{a} + \Delta_m), & \text{if } r \in [-N/2 : N/2-1], \\ 0, & \text{otherwise.} \end{cases} \tag{4.6}$$

The adjusted analysis frame x'_m is windowed and copied to the output signal y as in OLA (Figure 4.5a). Now, we need to adjust the position of the next analysis frame x_{m+1}. This task can be interpreted as a constrained optimization problem. Our goal is to find the optimal shift index $\Delta_{m+1} \in [-\Delta_{\max} : \Delta_{\max}]$ such that periodic structures of the adjusted analysis frame x'_{m+1} are optimally aligned with structures of the previously copied synthesis frame y_m in the overlapping region when superimposing both frames at the synthesis hopsize H_s. The *natural progression* $\tilde{x}_m$ of the adjusted analysis frame x'_m (dashed blue box in Figure 4.5b) would be an optimal choice for the adjusted analysis frame x'_{m+1} in an unconstrained scenario:

$$\tilde{x}_m(r) = \begin{cases} x(r + mH_\mathrm{a} + \Delta_m + H_\mathrm{s}), & \text{if } r \in [-N/2 : N/2-1], \\ 0, & \text{otherwise.} \end{cases} \tag{4.7}$$

This is the case, since the adjusted analysis frame x'_m and the synthesis frame y_m are essentially the same (up to windowing). As a result, the structures of the natural progression $\tilde{x}_m$ are perfectly aligned with the structures of the synthesis frame y_m when superimposing the two frames at the synthesis hopsize H_s (Figure 4.5b). However, due to the constraint $\Delta_{m+1} \in [-\Delta_{\max} : \Delta_{\max}]$, the adjusted frame x'_{m+1} must be located inside of the *extended frame region* x^+_{m+1} (solid blue box in Figure 4.5b):

$$x^+_{m+1}(r) = \begin{cases} x\left(r + (m+1)H_\mathrm{a}\right), & \text{if } r \in [-N/2 - \Delta_{\max} : N/2 - 1 + \Delta_{\max}], \\ 0, & \text{otherwise.} \end{cases} \tag{4.8}$$

Therefore, the idea is to retrieve the adjusted frame x'_{m+1} as the frame in x^+_{m+1} whose waveform is most similar to $\tilde{x}_m$. To this end, we must define a measure that quantifies the similarity of two frames. One possible choice for this metric is the *cross-correlation*

$$c(q, p, \Delta) = \sum_{r \in \mathbb{Z}} q(r) \cdot p(r + \Delta) \tag{4.9}$$

of a signal q and a signal p shifted by $\Delta \in \mathbb{Z}$ samples. We can then compute the optimal shift index Δ_{m+1} that maximizes the cross-correlation of $\tilde{x}_m$ and x^+_{m+1} by

$$\Delta_{m+1} = \underset{\Delta \in [-\Delta_{\max} : \Delta_{\max}]}{\operatorname{argmax}} \; c(\tilde{x}_m, x^+_{m+1}, \Delta) \,. \tag{4.10}$$

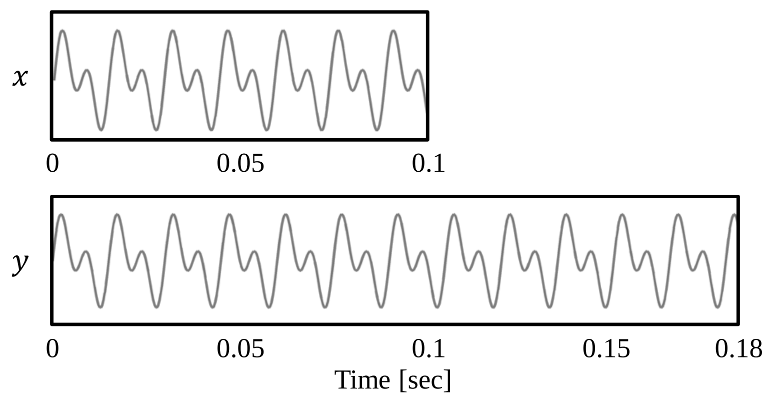


Figure 4.6: Preservation of periodic structures by WSOLA. The input signal x is the same as in Figure 4.4, this time modified with WSOLA (time-stretch with $\alpha = 1.8$).

The shift index Δ_{m+1} defines the position of the adjusted analysis frame x'_{m+1} inside the extended frame region x^{+}_{m+1} (Figure 4.5c). Finally, we compute the synthesis frame y_{m+1}, similarly to OLA, by

$$y_{m+1}(r) = \frac{w(r)\; x(r + (m+1)H_\mathrm{a} + \Delta_{m+1})}{\sum\limits_{n\in\mathbb{Z}} w(r - nH_\mathrm{s})} \tag{4.11}$$

and use Equation (4.2) to reconstruct the output signal y (Figure 4.5d). In practice, we start the iterative optimization process at the frame index $m = 0$ and assume $\Delta_0 = 0$.

4.1.3.2 Artifacts

WSOLA can improve on some of the drawbacks of OLA as illustrated by Figure 4.6 (WSOLA) and Figure 4.4 (OLA). However, it still causes certain artifacts that are characteristic for time-domain TSM procedures.

A prominent problem with WSOLA-like methods is known as *transient doubling* or *stuttering*. This artifact is visualized in Figure 4.7, where we can see a single transient in the input signal x that is contained in the overlapping region of two successive adjusted analysis frames x'_m and x'_{m+1}. When the frames are relocated and copied to the output signal, the transient is duplicated and can be heard two times in quick succession. A related artifact is *transient skipping* where transients get lost in the modification process since they are not contained in any of the analysis frames. While transient doubling commonly occurs when stretching a signal ($\alpha > 1$), transient skipping usually happens when the signal is compressed ($\alpha < 1$). These artifacts are particularly noticeable when modifying signals with percussive components, such as recordings of instruments with strong onsets (e.g. drums, piano).

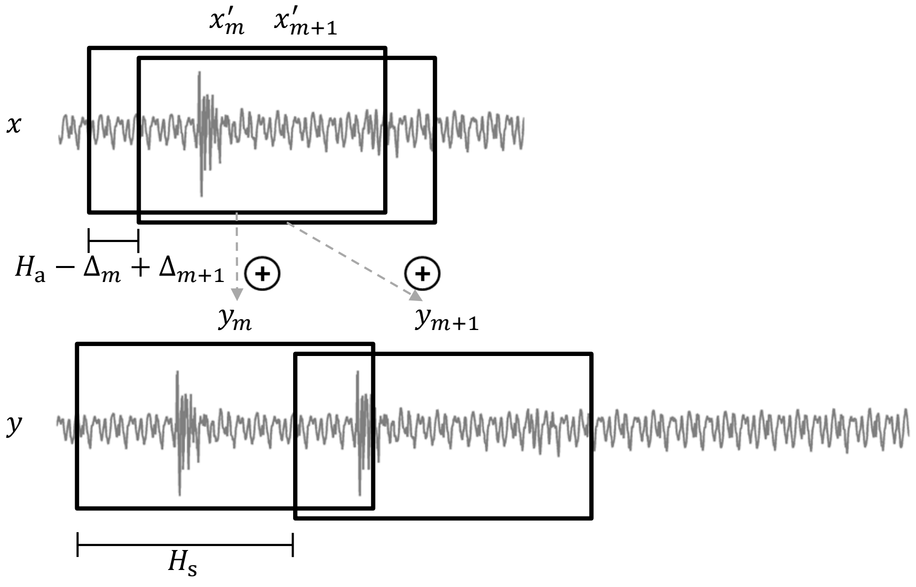


Figure 4.7: Transient doubling artifact as it typically occurs in signals modified with WSOLA.

Furthermore, WSOLA has particular problems when modifying polyphonic input signals such as recordings of orchestral music. For these input signals, the output often still contains considerable warbling. The reason for this is that WSOLA can, by design, only preserve the most prominent periodic pattern in the input signal's waveform. Therefore, when modifying recordings with multiple harmonic sound sources, only the sound of the most dominant source is preserved in the output, whereas the remaining sources can still cause phase jump artifacts. While WSOLA is well-suited to modify monophonic input signals, this is often not the case with more complex audio.

4.1.3.3 Tricks of the Trade

In order to assure that WSOLA can adapt to the most dominant periodic pattern in the waveform of the input signal, one frame must be able to capture at least a full period of the pattern. Also, the tolerance parameter $\Delta_{\max}$ must be large enough to allow for an appropriate adjustment. Therefore, it should be set to at least half a period's length. Assuming that the lowest frequency that can be heard by humans is roughly 20 Hertz, a common choice is a frame length N corresponding to 50 milliseconds and a tolerance parameter of 25 milliseconds.

One possibility to reduce transient doubling and skipping artifacts in WSOLA is to apply a *transient preservation* scheme. In [85], the idea is to first identify the temporal positions of transients in the input signal by using a transient detector. Then, in the WSOLA process, the analysis hopsize is temporarily fixed to be equal to the synthesis hopsize whenever an analysis frame is located in a neighborhood of an identified transient. This neighborhood, including the transient, is therefore copied without modification to the output signal, preventing WSOLA from doubling or skipping it. The deviation in the global stretching factor is compensated dynamically in the regions between transients.

4.1.4 TSM Based on the Phase Vocoder (PV-TSM)

4.1.4.1 Overview

As we have seen, WSOLA is capable of maintaining the most prominent periodicity in the input signal. The next step to improve the quality of time-scale modified signals is to preserve the periodicities of all signal components. This is the main idea of *frequency-domain* TSM procedures, which interpret each analysis frame as a weighted sum of sinusoidal components with known frequency and phase. Based on these parameters, each of these components is manipulated individually to avoid phase jump artifacts across all frequencies in the reconstructed signal.

A fundamental tool for the frequency analysis of the input signal is the *short-time Fourier transform* [74] (Section 4.1.4.2). However, depending on the chosen discretization parameters, the resulting frequency estimates may be inaccurate. To this end, the *phase vocoder*[1] technique [69, 168] is used to improve on the the short-time Fourier transform's coarse frequency estimates by deriving the sinusoidal components' *instantaneous frequencies* (Section 4.1.4.3). In TSM procedures based on the phase vocoder (PV-TSM), these improved estimates are used to update the phases of an input signal's sinusoidal components in a process known as *phase propagation* (Section 4.1.4.4).

4.1.4.2 The Short-Time Fourier Transform

The most important tool of PV-TSM is the short-time Fourier transform (STFT) that we introduced in Section 2.2. For the sake of convenience we reformulate its definition at this point to match our general TSM framework. In this context, the STFT can be seen as the application of the Fourier transform to every analysis frame of a given input signal. The STFT X of a signal x is therefore given by

$$X(m,k) = \sum_{r=-N/2}^{N/2-1} x_m(r)\ w(r)\ \exp(-2\pi ikr/N)\ , \tag{4.12}$$

[1]Although the term "phase vocoder" refers to a specific technique for the estimation of instantaneous frequencies, it is also frequently used in the literature as the name of the TSM procedure itself.

where $m \in \mathbb{Z}$ is the frame index, $k \in [0 : N-1]$ is the frequency index, N is the frame length, x_m is the m^{th} analysis frame of x as defined in Equation (4.1), and w is a window function. Given the input signal's sampling rate F_s, the frame index m of $X(m,k)$ is associated to the physical time

$$T_\text{coef}(m) = \frac{m \cdot H_\text{a}}{F_\text{s}} \tag{4.13}$$

given in seconds, and the frequency index k corresponds to the physical frequency

$$F_\text{coef}(k) = \frac{k \cdot F_\text{s}}{N} \tag{4.14}$$

given in Hertz. The complex number $X(m,k)$, also called a *time-frequency bin*, denotes the k^{th} Fourier coefficient for the m^{th} analysis frame. It can be represented by a magnitude $|X(m,k)| \in \mathbb{R}^+$ and a phase $\varphi(m,k) \in [0,1)$ as

$$X(m,k) = |X(m,k)| \cdot \exp(2\pi i \, \varphi(m,k)) \; . \tag{4.15}$$

The magnitude of an STFT X is also called a *spectrogram* which is denoted by Y:

$$Y = |X| \; . \tag{4.16}$$

In the context of PV-TSM, it is necessary to reconstruct the output signal y from a modified STFT X^Mod. Note that a modified STFT is in general not invertible [146]. In other words, there might be no signal y that has X^Mod as its STFT. However, there exist methods that aim to reconstruct a signal y from X^Mod whose STFT is close to X^Mod with respect to some distance measure. Following the procedure described in [84], we first compute time-domain frames x_m^Mod by using the inverse Fourier transform.

$$x_m^\text{Mod}(r) = \frac{1}{N} \sum_{k=0}^{N-1} X^\text{Mod}(m,k) \cdot \exp(2\pi i k r / N) \; . \tag{4.17}$$

From these frames, we then derive synthesis frames

$$y_m(r) = \frac{w(r) \cdot x_m^\text{Mod}(r)}{\sum\limits_{n \in \mathbb{Z}} w(r - nH_\text{s})^2} \tag{4.18}$$

and reconstruct the output signal y by Equation (4.2). It can be shown that, when computing the synthesis frames by Equation (4.18), the STFT of y (when choosing $H_\text{a} = H_\text{s}$) minimizes a squared error distance measure defined in [84].

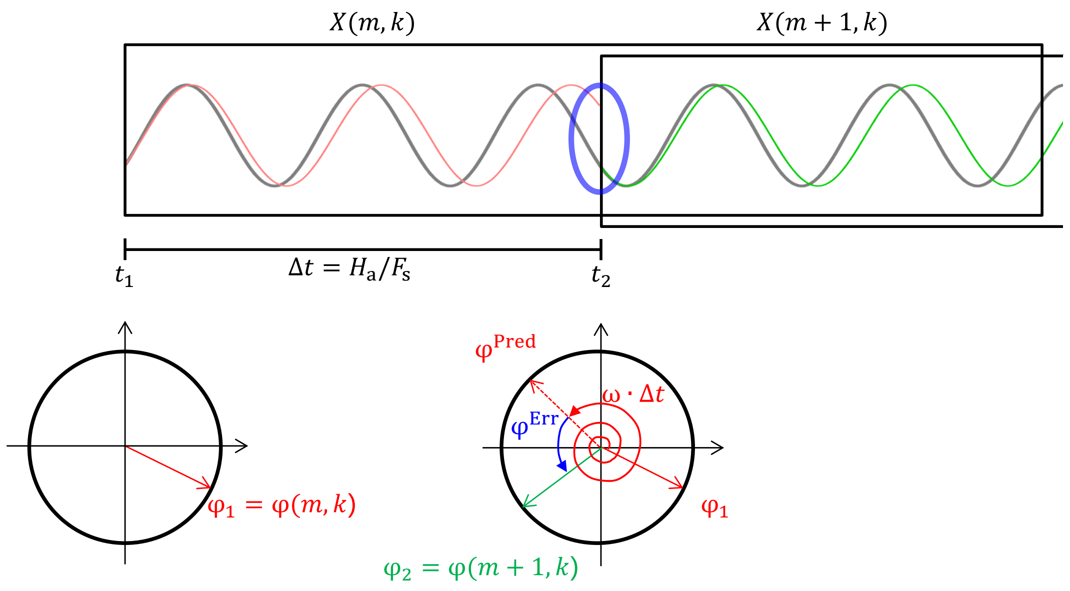


Figure 4.8: Principle of the phase vocoder.

4.1.4.3 The Phase Vocoder

Each time-frequency bin $X(m,k)$ of an STFT can be interpreted as a sinusoidal component with amplitude $|X(m,k)|$ and phase $\varphi(m,k)$ that contributes to the m^{th} analysis frame of the input signal x. However, the Fourier transform yields coefficients only for a discrete set of frequencies that are sampled linearly on the frequency axis, see Equation (4.14). The STFT's frequency resolution therefore does not suffice to assign a precise frequency value to this sinusoidal component. The phase vocoder is a technique that refines the STFT's coarse frequency estimate by exploiting the given phase information.

In order to understand the phase vocoder, let us have a look at the scenario shown in Figure 4.8. Assume we are given two phase estimates $\varphi_1 = \varphi(m,k)$ and $\varphi_2 = \varphi(m+1,k)$ at the time instances $t_1 = T_{\mathrm{coef}}(m)$ and $t_2 = T_{\mathrm{coef}}(m+1)$ of a sinusoidal component for which we only have a coarse frequency estimate $\omega = F_{\mathrm{coef}}(k)$. Our goal is to estimate the sinusoid's "real" instantaneous frequency $\mathrm{IF}(\omega)$. Figure 4.8 shows this sinusoidal component (gray) as well as two sinusoids that have a frequency of ω (red and green). In addition, we also see phase representations at the time instances t_1 and t_2. The red sinusoid has a phase of φ_1 at t_1 and the green sinusoid a phase of φ_2 at t_2. One can see that the frequency ω of the red and green sinusoids is slightly lower than the frequency of the gray sinusoid. Intuitively, while the phases of the gray and the red sinusoid match at t_1, they diverge over time, and we can observe a considerable discrepancy after

$\Delta t = t_2 - t_1$ seconds[2] (blue oval). Since we know the red sinusoid's frequency, we can compute its *unwrapped phase advance*, i.e. the number of oscillations that occur over the course of Δt seconds:

$$\omega \cdot \Delta t \, . \tag{4.19}$$

Knowing that its phase at t_1 is φ_1, we can predict its phase after Δt seconds:

$$\varphi^{\mathrm{Pred}} = \varphi_1 + \omega \cdot \Delta t \, . \tag{4.20}$$

At t_2 we again have a precise phase estimate φ_2 for the gray sinusoid. We therefore can compute the *phase error* φ^{Err} (also called the *heterodyned phase increment* [28]) between the phase actually measured at t_2 and the predicted phase when assuming a frequency of ω:

$$\varphi^{\mathrm{Err}} = \Psi(\varphi_2 - \varphi^{\mathrm{Pred}}) \, . \tag{4.21}$$

Here, $\Psi : \mathbb{R} \to [-0.5, 0.5)$ is the *principal argument function*:

$$\Psi(a) = a + \gamma \, , \tag{4.22}$$

with $a \in \mathbb{R}$ and $\gamma \in \mathbb{Z}$ such that $a + \gamma \in [-0.5, 0.5)$. Note that by mapping φ^{Err} into the range $[-0.5, 0.5)$, we assume that the number of oscillations of the gray and red sinusoids differ by at most half a period. In the context of instantaneous frequency estimation, this means that the coarse frequency estimate ω needs to be close to the actual frequency of the sinusoid, and that the interval Δt should be small. The unwrapped phase advance of the gray sinusoid can then be computed by the sum of the unwrapped phase advance of the red sinusoid with frequency ω (red spiral arrow) and the phase error (blue curved arrow):

$$\omega \cdot \Delta t + \varphi^{\mathrm{Err}} \, . \tag{4.23}$$

This gives us the number of oscillations of the gray sinusoid over the course of Δt seconds. From this we can derive the instantaneous frequency of the gray sinusoid by

$$\mathrm{IF}(\omega) = \frac{\omega \cdot \Delta t + \varphi^{\mathrm{Err}}}{\Delta t} = \omega + \frac{\varphi^{\mathrm{Err}}}{\Delta t} \, . \tag{4.24}$$

The frequency $\varphi^{\mathrm{Err}}/\Delta t$ can be interpreted as the small offset of the gray sinusoid's actual frequency from the rough frequency estimate ω.

[2]Note that Figure 4.8 does not represent the phase coming from an STFT, where the red and green sinusoids would be phase-shifted to globally match the gray signal within the respective frame (see also [146, Chapter 8]). Furthermore, according to the STFT's definition from Equation (4.12) phases are not measured at the at a frame's beginning but at its center. However, this only causes a relative phase offset and does not influence the illustration of the phase vocoder's core principle.

We can use this approach to refine the coarse frequency resolution of the STFT by computing instantaneous frequency estimates $F^{\mathrm{IF}}_{\mathrm{coef}}(m,k)$ for all time-frequency bins $X(m,k)$:

$$F^{\mathrm{IF}}_{\mathrm{coef}}(m,k) = \mathrm{IF}(\omega) = \omega + \frac{\Psi\left(\varphi_2 - (\varphi_1 + \omega\cdot\Delta t)\right)}{\Delta t}\ . \tag{4.25}$$

With $\omega = F_{\mathrm{coef}}(k)$, $\Delta t = H_{\mathrm{a}}/F_{\mathrm{s}}$ (the analysis hopsize given in seconds), $\varphi_1 = \varphi(m,k)$, and $\varphi_2 = \varphi(m+1,k)$ we finally get

$$F^{\mathrm{IF}}_{\mathrm{coef}}(m,k) = F_{\mathrm{coef}}(k) + \Psi\left(\varphi(m+1,k) - \left(\varphi(m,k) + F_{\mathrm{coef}}(k)\frac{H_{\mathrm{a}}}{F_{\mathrm{s}}}\right)\right)\frac{F_{\mathrm{s}}}{H_{\mathrm{a}}}\ . \tag{4.26}$$

For further details, we refer to [146, Chapter 8].

4.1.4.4 PV-TSM

The principle of PV-TSM is visualized in Figure 4.9. Given an input audio signal x, the first step of PV-TSM is to compute the STFT X. Figure 4.9a shows the two successive frequency spectra of the m^{th} and $(m+1)^{\mathrm{th}}$ analysis frames, denoted here by $X(m)$ and $X(m+1)$, respectively. Our goal is to compute a modified STFT X^{Mod} with adjusted phases φ^{Mod} from which we can reconstruct a time-scale modified signal without phase jump artifacts:

$$X^{\mathrm{Mod}}(m,k) = |X(m,k)|\cdot\exp(2\pi i\,\varphi^{\mathrm{Mod}}(m,k))\ . \tag{4.27}$$

We compute the adjusted phases φ^{Mod} in an iterative process that is known as phase propagation. Assume that the phases of the m^{th} frame have already been adjusted (Figure 4.9b). As indicated by Figure 4.9b, overlapping the two frames at the synthesis hopsize H_{s} may lead to phase jumps. Knowing the instantaneous frequencies $F^{\mathrm{IF}}_{\mathrm{coef}}$ derived by the phase vocoder, we can predict the phases of the sinusoidal components in the m^{th} frame after a time interval corresponding to H_{s} samples. To this end, we set $\varphi_1 = \varphi^{\mathrm{Mod}}(m,k)$, $\omega = F^{\mathrm{IF}}_{\mathrm{coef}}(m,k)$, and $\Delta t = H_{\mathrm{s}}/F_{\mathrm{s}}$ (the synthesis hopsize given in seconds) in Equation (4.20). This allows us to replace the phases of the $(m+1)^{\mathrm{th}}$ frame with the predicted phase:

$$\varphi^{\mathrm{Mod}}(m+1,k) = \varphi^{\mathrm{Mod}}(m,k) + F^{\mathrm{IF}}_{\mathrm{coef}}(m,k)\frac{H_{\mathrm{s}}}{F_{\mathrm{s}}} \tag{4.28}$$

for $k\in[0:N-1]$. Assuming that the estimates of the instantaneous frequencies $F^{\mathrm{IF}}_{\mathrm{coef}}$ are correct, there are no phase jumps any more when overlapping the modified spectra at the synthesis hopsize H_{s} (Figure 4.9c). In practice, we start the iterative phase propagation with the frame index $m=0$ and set

$$\varphi^{\mathrm{Mod}}(0,k) = \varphi(0,k)\ , \tag{4.29}$$

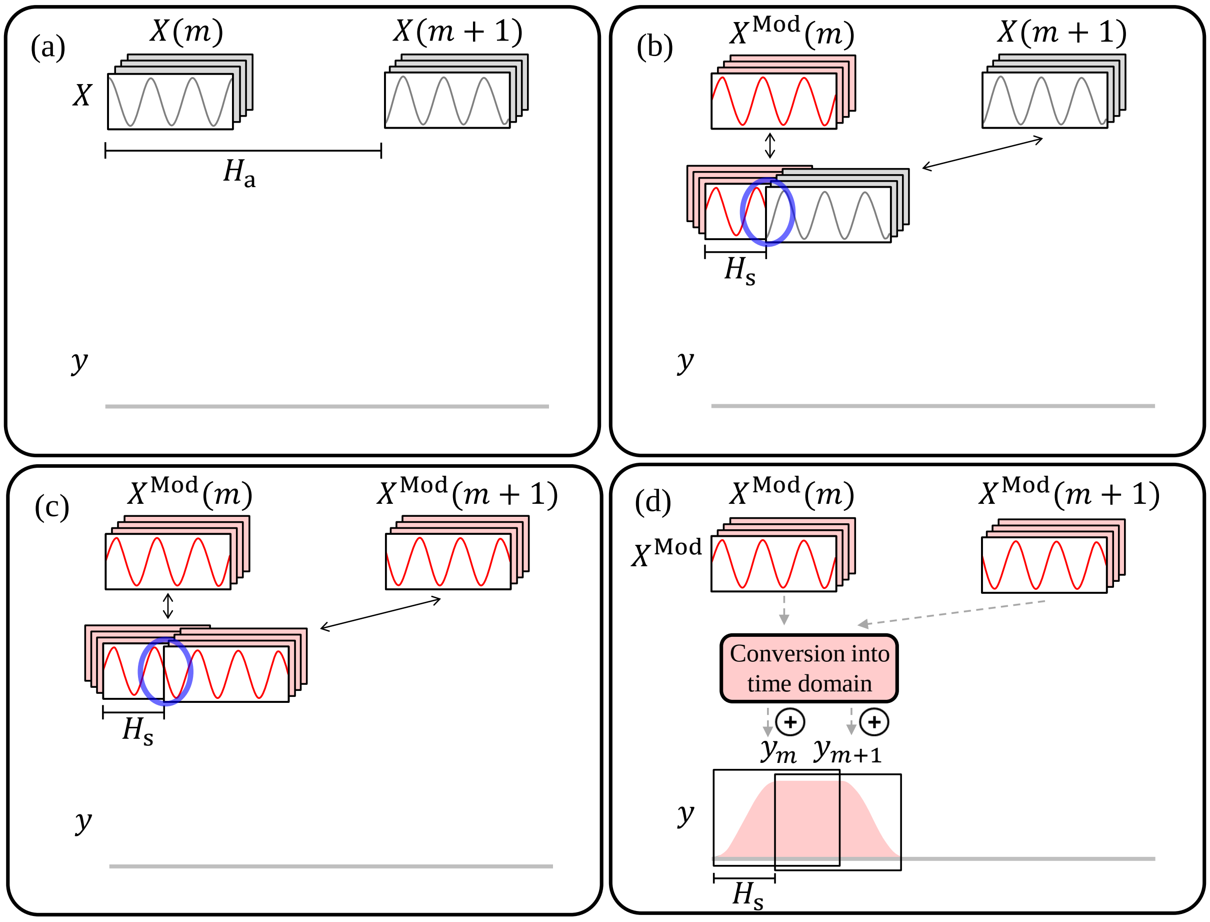


Figure 4.9: The principle of PV-TSM. **(a)**: STFT X. **(b)**: Using the original phase of $X(m+1,k)$ leads to phase jumps when overlapping the frames at the synthesis hopsize H_s (blue oval). **(c)**: Update of the sinusoids' phases via phase propagation. Phase jumps are reduced (blue oval). **(d)**: Signal reconstruction from the modified STFT X^Mod.

for all $k \in [0 : N-1]$. Finally, the output signal y can be computed using the signal reconstruction procedure described in Section 4.1.4.2 (Figure 4.9d).

4.1.4.5 Artifacts

By design, PV-TSM can achieve phase continuity for all sinusoidal components contributing to the output signal. This property, also known as *horizontal phase coherence*, ensures that there are no artifacts related to phase jumps. However, the *vertical phase coherence*, i.e., the phase relationships of sinusoidal components within one frame, is usually destroyed in the phase propagation process. The loss of vertical phase coherence affects the time localization of events such as transients. As a result, transients are often *smeared* in signals modified with PV-TSM (Figure 4.10). Furthermore, output signals of PV-TSM often have a very distinct sound coloration

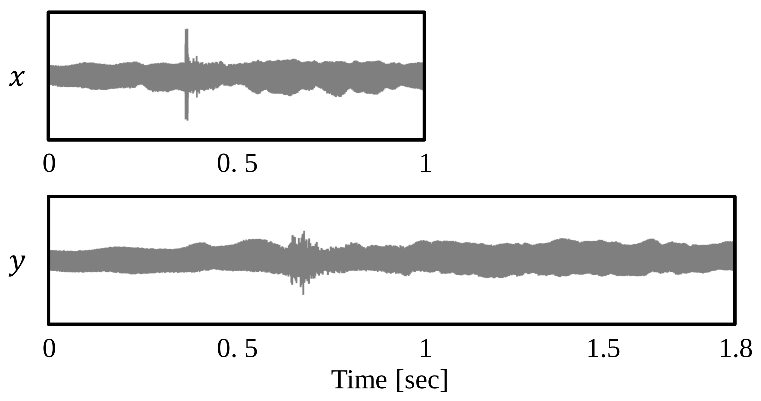


Figure 4.10: Illustration of a smeared transient as a result of applying PV-TSM. The input signal x has been stretched by a factor of $\alpha = 1.8$ to yield the output signal y.

known as *phasiness* [127], an artifact also caused by the loss of vertical phase coherence. Signals suffering from phasiness are often described as being reverberant, having "less bite," or a "loss of presence" [128].

4.1.4.6 Tricks of the Trade

The phase vocoder technique highly benefits from frequency estimates that are already close to the instantaneous frequencies as well as from a small analysis hopsize (resulting in a small Δt). In PV-TSM, the frame length N is therefore commonly set to a relatively large value. In practice, N typically corresponds to roughly 100 milliseconds in order to achieve a high frequency resolution of the STFT.

To reduce the loss of vertical phase coherence in the phase vocoder, Laroche and Dolson proposed a modification to the standard PV-TSM in [128]. Their core observation is that a sinusoidal component may affect multiple adjacent time-frequency bins of a single analysis frame. Therefore, the phases of these bins should not be updated independently, but in a joint fashion. A peak in a frame's magnitude spectrum is assumed to be representative of a particular sinusoidal component. The frequency bins with lower magnitude values surrounding the peak are assumed to be affected by the same component as well. In the phase propagation process, only the time-frequency bins with a peak magnitude are updated in the usual fashion described in Equation (4.28). The phases of the remaining frequency bins are *locked* to the phase of the sinusoidal component corresponding to the closest peak. This procedure allows to locally preserve the signal's vertical phase coherence. This technique, also known as *identity phase locking*, leads to reduced phasiness artifacts and less smearing of transients.

A second remarkable finding of Laroche and Dolson is that in case the stretching factor $\alpha \in \mathbb{N}$ there exists a very simple version of PV-TSM that even reduces phasiness artifacts and transient smearing. In this case, instead of using the phase propagation procedure defined in Equation (4.28) one can simply compute the modified phase by

$$\varphi^{\mathrm{Mod}}(m,k) = \alpha \cdot \varphi(m,k) \; . \tag{4.30}$$

We discuss this version in more detail in Section 4.3.

Another approach to reduce phasieness artifacts is the *phase vocoder with synchronized overlap-add* (PVSOLA) and similar formulations [142, 121, 31, 30]. The core observation of these methods is that the original vertical phase coherence is usually lost gradually over the course of phase-updated synthesis frames. Therefore, these procedures "reset" the vertical phase coherence repeatedly after a fixed number of frames. The resetting is done by copying an analysis frame unmodified to the output signal and adjusting it in a WSOLA-like fashion. After having reset the vertical phase coherence, the next synthesis frames are again computed by the phase propagation strategy.

To reduce the effect of transient smearing, other approaches include transient preservation schemes, similar to those for WSOLA. In [155], transients are identified in the input signal, cut out from the waveform, and temporarily stored. The remaining signal, after filling the gaps using linear prediction techniques, is modified using PV-TSM. Finally, the stored transients are relocated according to the stretching factor and reinserted into the modified signal.

4.2 TSM Based on Harmonic-Percussive Separation (HP-TSM)

4.2.1 The Procedure

As we have seen in the previous sections, percussive sound components and transients in the input signal cause issues with most fundamental TSM procedures. In this section, we present a simple, yet effective method that avoids an explicit detection of transients, but is still capable of preserving them to a high degree. This contribution was originally presented in [48]. Our procedure's core principle is visualized in Figure 4.11. The shown input signal (Figure 4.11a) consists of a tone played on a violin (harmonic) superimposed with a single click of castanets halfway through the signal (percussive). The procedure's core idea is to first decompose the input signal into a harmonic and a percussive component (Figure 4.11b), using the computationally efficient algorithm by Fitzgerald [67] that we already discussed in Section 3.1. Here, one main observation is that the percussive component contains, besides other noise-like sounds, all the transients. The two components are then modified with two different TSM procedures (Figure 4.11c). The harmonic component is processed with PV-TSM that yields good TSM results for harmonic signals but smears sudden events like transients noticeably (see Section 4.1.4).

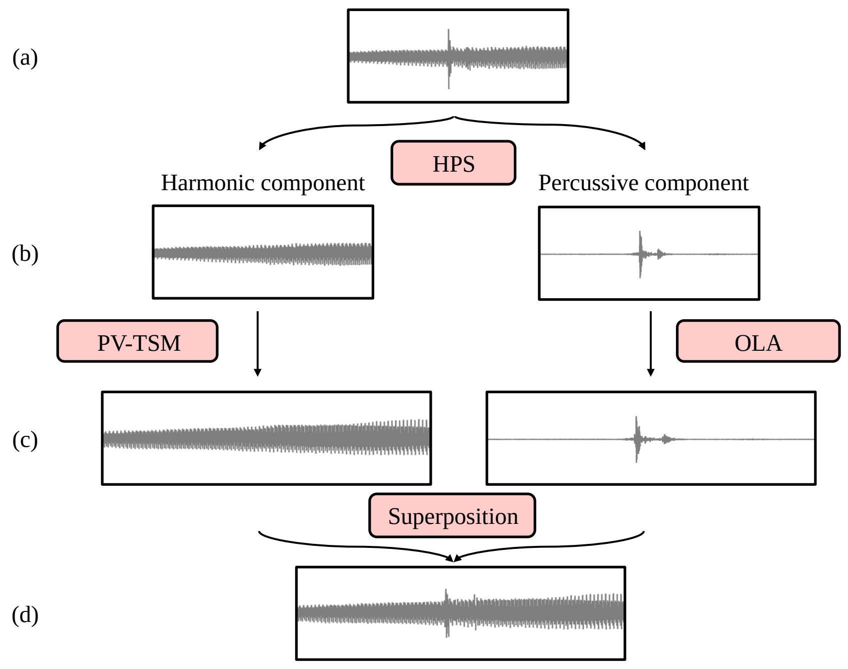


Figure 4.11: Overview of the proposed TSM approach based on harmonic-percussive separation. **(a):** Input audio signal x. **(b):** Separation in harmonic component x_h (left) and percussive component x_p (right). **(c):** TSM results for the harmonic component using the PV-TSM (left) and for the percussive component using OLA (right). **(d):** Superposition of the output signals.

As for the percussive component, we use OLA with a very small frame size. Although OLA is considered a poor TSM technique in particular for harmonic sounds, it is capable of preserving the sound of transients without any explicit transient detection (see Section 4.1.2). Finally, the two modified components are superimposed to form the output of our procedure [3] (Figure 4.11d). Even though OLA may introduce artifacts in case some harmonic sounds remain in the percussive component due to an imperfect decomposition, those artifacts are often perceptually masked by the TSM result of the harmonic component and vice versa.

To evaluate our proposed method, we performed a listening experiment (see Section 4.2.3). The results suggest that our relatively simple approach is almost always superior to PV-TSM or WSOLA and can also compete with a commercial state-of-the-art TSM algorithm.

[3]A conceptually similar strategy for TSM has been proposed by Verma and Meng in [218] where they use a *sines+transients+noise* signal model as discussed in Section 3.3.1 to parameterize a given signal's sinusoidal, transient, and noise-like sound components. The estimated parameters are then modified in order to synthesize a time-scaled output signal. However, their procedure still relies on an explicit transient detection in order to estimate appropriate parameters for the transient sound components.

4.2.2 Tricks of the Trade

Concerning parameters settings, there are a few points to notice. As discussed in Chapter 3, the most crucial parameter to influence the harmonic-percussive separation procedure's result is the frame length N of the STFT. The larger N, the larger the portion of the signal that is assigned to the harmonic component x_h. This is the case since a transient is an event of very short temporal duration and its influence on the structure of the magnitude spectrogram Y therefore decreases when N becomes larger. The length of the median filters $2\ell_\mathrm{h} + 1$ and $2\ell_\mathrm{p} + 1$ on the other hand do not influence the separation too much as long as no extreme values are chosen (see Section 3.3.2). Our experiments have shown that setting $N = 1024$ and $\ell_\mathrm{h} = \ell_\mathrm{p} = 5$ when using a sampling rate of 22050 Hz for the original music signals yields decompositions in which the percussive component contains the transients while showing only very few harmonic residual sounds. In further experiments it showed that for obtaining such results only the ratio between the sampling rate and the frame size is important. For example, we obtain perceptually similar decompositions by setting $N = 2048$ for signals sampled at 44100 Hz. Systematic experiments on choosing optimal values for N and ℓ are left as future work and may further improve the TSM results of our proposed procedure.

For the phase vocoder, it is important to use large frame sizes N to get a good frequency resolution in the STFT computation (see Section 4.1.4.6). We therefore use a frame size of 4096 samples for signals sampled at 22050 Hz. Furthermore, we implemented the identity phase locking strategy presented in [128] to increase the overall quality of the technique.

Finally, as mentioned above, for OLA it is crucial to use a very small frame length N to ensure the accurate temporal location and sound preservation of the percussive signal elements. In our implementation we set N to 256 samples.

4.2.3 Evaluation

To evaluate our proposed method, we conducted an online listening experiment (all sources of the experiment can be found at [47]) where we compared five TSM algorithms: our proposed method based on harmonic-percussive separation (HP-TSM), the commercial *élastique* algorithm [231] (EL-TSM) which is included in a wide range of music software nowadays and can be considered one of the important state-of-the-art algorithms, the phase vocoder with identity phase locking [128] (PV-TSM), the phase vocoder with transient preservation by Nagel and Walther as described in [155] (NW-TSM), and WSOLA [216]. As evaluation dataset we used the ten short audio excerpts listed in Table 4.1. The set was compiled to cover a wide range of aspects in music signals. Among the excerpts are purely percussive signals, monophonic as well as highly polyphonic signals with different instrumentations, simple synthetic sound mixtures as well as professionally mixed studio recordings. The ten excerpts were stretched with each of the five procedures,

Item name	**Description**
Bongo	Regular beat played on bongos.
CastanetsViolin	Solo violin overlayed with castanets.
DrumSolo	A solo performed on a drum set.
Glockenspiel	Monophonic melody played on a glockenspiel.
Jazz	Synthetic polyphonic sound mixture of a trumpet, a piano, a bass and drums.
Pop	Synthetic polyphonic sound mixture of several synthesizers, a guitar and drums.
SingingVoice	Solo male singing voice.
Stepdad	Excerpt from *My Leather, My Fur, My Nails* by the band *Stepdad*.
SynthMono	Monophonic synthesizer with a very noisy and distorted sound.
SynthPoly	Sound mixture of several polyphonic synthesizers.

Table 4.1: List of audio excerpts.

once with a moderate stretching factor $\alpha = 1.2$ and a stronger stretching factor $\alpha = 1.8$ as they realistically occur in applications like audio remixing (TSM results for the more extreme stretching factors $\alpha = 0.5$ and $\alpha = 3$ can be found at [47]). This results in 100 test items in total. In the experiment the test listeners were sequentially presented with groups of five stretched items of one excerpt together with the original excerpt. They were asked to rate the five output signals on the five point *mean opinion score* scale from 1 (bad) to 5 (excellent). Overall 22 people, most of them with a background in audio signal processing, participated in the experiment. The results are given in Table 4.2.

One can see that the two algorithms HP-TSM and EL-TSM performed significantly better than the remaining three algorithms. For highly percussive excerpts like *Bongo*, *CastanetsViolin* and *DrumSolo*, HP-TSM even outperformed all other tested TSM procedures. These good results can mainly be devoted to the well-preserved transients in the modified signals. Looking for example at the TSM results of all five algorithms for the *CastanetsViolin* excerpt in Figure 4.12 this becomes obvious. For EL-TSM, PV-TSM and WSOLA (Figure 4.12c, d and f) one can observe stuttering or smearing artifacts at transient positions. NW-TSM (Figure 4.12e) is actually able to preserve the transients perfectly, but the tolerance region around the actual transients, which are copied from the original and reinserted into the modified signal, make the TSM result sound very unnatural (see Section 4.1.4.6). This shows that even when transients are identified correctly, the re-insertion into the modified signal is another problematic step causing unnatural transitions

	$\alpha = 1.2$					$\alpha = 1.8$				
	HP-TSM	EL-TSM	PV-TSM	NW-TSM	WSOLA	HP-TSM	EL-TSM	PV-TSM	NW-TSM	WSOLA
Bongo	**4.73**	4.32	1.68	2.27	1.64	**4.09**	3.50	1.36	1.59	1.41
CastanetsViolin	**4.41**	4.36	1.95	1.68	2.77	**3.91**	3.68	1.64	1.64	2.27
DrumSolo	**4.27**	3.77	3.09	1.95	1.68	2.95	**3.09**	1.77	1.32	1.32
Glockenspiel	3.18	**4.45**	2.86	1.82	2.18	3.14	**3.91**	2.64	1.55	2.00
Jazz	**4.55**	4.09	3.77	2.00	2.14	**3.68**	3.27	2.23	1.86	1.23
Pop	**4.23**	4.00	3.73	1.68	1.73	**3.59**	3.27	2.41	2.14	1.68
SingingVoice	2.59	**4.27**	3.32	1.73	3.82	2.91	**3.77**	2.91	1.45	2.32
Stepdad	**4.32**	3.82	3.64	1.82	2.14	**3.86**	3.36	2.91	1.73	1.82
SynthMono	2.82	**4.05**	2.32	1.95	2.82	2.45	**3.09**	2.50	1.95	2.27
SynthPoly	3.41	**4.14**	2.55	2.68	3.50	3.09	**3.91**	2.05	2.91	2.64
∅	3.85	4.13	2.89	1.96	2.44	3.37	3.49	2.24	1.81	1.90

Table 4.2: Mean opinion scores for all test items.

and also explains the bad performance of NW-TSM in the listening experiment. In contrast, HP-TSM (Figure 4.12b) succeeds in keeping the transients rather crisp while also preserving the harmonic component around the transients.

Also for complex sound mixtures with strong percussive components like *Jazz*, *Pop* and *Stepdad* HP-TSM scored best. This shows, that the harmonic-percussive separation actually performs well, also on very complex musical signals. Noticeable are the poorer performances of HP-TSM for the excerpts *SynthMono* and *SingingVoice*. For *SynthMono*, a closer investigation of HP-TSM's output signals reveals that the modified harmonic components suffers from phasiness, stemming from the harmonic component being stretched with PV-TSM (see Section 4.1.4.5). The strong presence of this artifact can be made responsible for the lower scores. Therefore, by reducing phasiness artifacts in the TSM results of the harmonic components one may further improve our proposed procedure. For *SingingVoice*, it shows that for this excerpt the separation into a harmonic and a percussive component is problematic. Due to the spectral characteristics of human voice, which are neither of clearly harmonic nor clearly percussive nature, the portions of the spectrogram which are assigned to the percussive component still cover large amounts of harmonic content. Therefore, the assumption that the percussive component is mainly noise-like is violated and, when modifying this component with OLA, the introduced phase jump artifacts are too strong to be masked by the modified harmonic component. This also explains why PV-TSM scores better than HP-TSM for this excerpt. However, HP-TSM performed significantly better than PV-TSM on average and is comparable in quality to the proprietary and highly

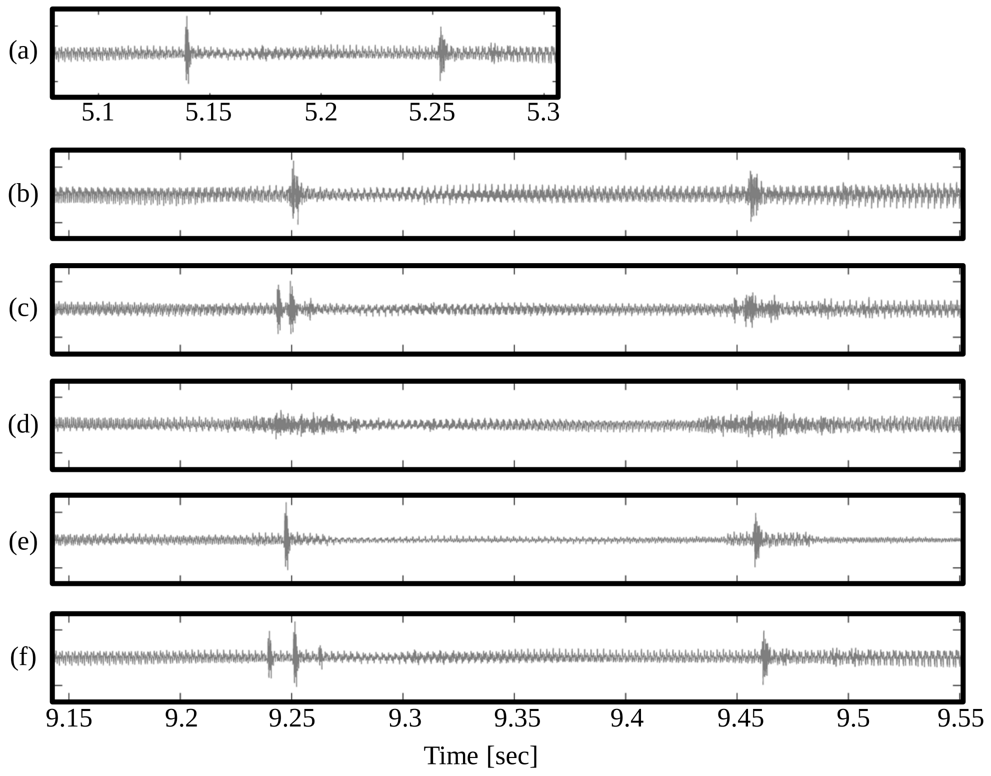


Figure 4.12: (a): Excerpt of the *CastanetsViolin* Item. **(b)-(f):** Output signals for a stretching factor $\alpha = 1.8$ of **(b):** HP-TSM. **(c):** EL-TSM. **(d):** PV-TSM. **(e):** NW-TSM. **(f):** WSOLA.

optimized EL-TSM on the tested dataset. Considering the simplicity of our method, this is quite remarkable. Our contribution shows that a robust and perceptually transparent preservation of transients is essential for a "good" TSM result. Furthermore, it shows that to achieve such a preservation, an explicit detection of transients is not necessary.

Concerning the running time, the harmonic-percussive separation can be implemented efficiently and the additional computation time is independent of the stretching factor α. For example, for stretching one minute of input audio material with stretching factors of $\alpha = 1.2$ and $\alpha = 1.8$, our MATLAB implementation of HP-TSM needed four seconds and five seconds, respectively.

4.3 TSM Based on Cascading (Cascade-TSM)

4.3.1 Overview

As mentioned in Section 4.1.4.6, Laroche and Dolsen outline a version of PV-TSM in [128] that is capable of minimizing phasiness artifacts and transient smearing for the special case that the stretching factor is an integer, i.e., $\alpha \in \mathbb{N}$ (Int-PV-TSM). However, when implementing this procedure, one needs to account for several pitfalls not discussed in the original paper. Furthermore, constraining α to be an integer obviously sets some limitations on the usefulness of the procedure.

In this section, we propose an extension to Int-PV-TSM that allows for arbitrary stretching factors $\alpha \in \mathbb{Q}^+$ by cascading it with the previously introduced HP-TSM. We first review the mathematical foundations of Int-PV-TSM in Section 4.3.2, discuss implementation issues in Section 4.3.3, and show how to cascade the procedure with HP-TSM in Section 4.3.4.

4.3.2 PV-TSM for Integer Stretching Factors (Int-PV-TSM)

For the case that the stretching factor is an integer $\alpha \in \mathbb{N}$, Laroche and Dolson state in [128] that instead of computing the modified phases φ^{Mod} by phase propagation, one can also compute them by

$$\varphi^{\text{Mod}}(m,k) = \alpha \cdot \varphi(m,k) \ . \tag{4.31}$$

Note that this implies that, for $\alpha \in \mathbb{N}$, it is not necessary to derive instantaneous frequencies. In order to show this, let us recall the phase propagation formula from Equation (4.28):

$$\varphi^{\text{Mod}}(m+1,k) = \varphi^{\text{Mod}}(m,k) + F^{\text{IF}}_{\text{coef}}(m,k)\frac{H_{\text{s}}}{F_{\text{s}}} \ . \tag{4.32}$$

We can rewrite this recursive definition by

$$\varphi^{\text{Mod}}(m,k) = \varphi^{\text{Mod}}(0,k) + \sum_{n=0}^{m-1}\left[F^{\text{IF}}_{\text{coef}}(n,k)\frac{H_{\text{s}}}{F_{\text{s}}}\right] \ . \tag{4.33}$$

Note that this formula is in closed form as long as $\varphi^{\text{Mod}}(0,k)$ is defined as some fixed value. Plugging in the definition of $F^{\text{IF}}_{\text{coef}}$ from Equation (4.26) yields

$$\begin{aligned}
\varphi^{\text{Mod}}(m,k) &= \varphi^{\text{Mod}}(0,k) + \qquad\qquad\qquad\qquad (4.34)\\
&\sum_{n=0}^{m-1}\left[\left(F_{\text{coef}}(k) + \Psi\left(\varphi(n+1,k) - \left(\varphi(n,k) + F_{\text{coef}}(k)\frac{H_{\text{a}}}{F_{\text{s}}}\right)\right)\frac{F_{\text{s}}}{H_{\text{a}}}\right)\frac{H_{\text{s}}}{F_{\text{s}}}\right]\\
&= \varphi^{\text{Mod}}(0,k) + \qquad\qquad\qquad\qquad (4.35)\\
&\sum_{n=0}^{m-1}\left[F_{\text{coef}}(k)\frac{H_{\text{s}}}{F_{\text{s}}} + \Psi\left(\varphi(n+1,k) - \left(\varphi(n,k) + F_{\text{coef}}(k)\frac{H_{\text{a}}}{F_{\text{s}}}\right)\right)\frac{H_{\text{s}}}{H_{\text{a}}}\right] \ .
\end{aligned}$$

With $\alpha = H_\mathrm{s}/H_\mathrm{a}$ we get

$$\begin{aligned}
\varphi^\mathrm{Mod}(m,k) &= \varphi^\mathrm{Mod}(0,k) + \\
&\sum_{n=0}^{m-1}\left[F_\mathrm{coef}(k)\frac{H_\mathrm{s}}{F_\mathrm{s}} + \Psi\left(\varphi(n+1,k) - \left(\varphi(n,k) + F_\mathrm{coef}(k)\frac{H_\mathrm{a}}{F_\mathrm{s}}\right)\right)\alpha\right] && (4.36)\\
&= \varphi^\mathrm{Mod}(0,k) + \\
&\sum_{n=0}^{m-1}\left[F_\mathrm{coef}(k)\frac{H_\mathrm{s}}{F_\mathrm{s}} + \Psi\left(\varphi(n+1,k) - \varphi(n,k) - F_\mathrm{coef}(k)\frac{H_\mathrm{a}}{F_\mathrm{s}}\right)\alpha\right]\ . && (4.37)
\end{aligned}$$

Using the definition of the principal argument function from Equation (4.22) yields

$$\begin{aligned}
\varphi^\mathrm{Mod}(m,k) &= \varphi^\mathrm{Mod}(0,k) + \\
&\sum_{n=0}^{m-1}\left[F_\mathrm{coef}(k)\frac{H_\mathrm{s}}{F_\mathrm{s}} + \left(\varphi(n+1,k) - \varphi(n,k) - F_\mathrm{coef}(k)\frac{H_\mathrm{a}}{F_\mathrm{s}} + \gamma_k^n\right)\alpha\right] && (4.38)\\
&= \varphi^\mathrm{Mod}(0,k) + \\
&\sum_{n=0}^{m-1}\left[F_\mathrm{coef}(k)\frac{H_\mathrm{s}}{F_\mathrm{s}} - F_\mathrm{coef}(k)\frac{\alpha\cdot H_\mathrm{a}}{F_\mathrm{s}} + \left(\varphi(n+1,k) - \varphi(n,k) + \gamma_k^n\right)\alpha\right]\ , && (4.39)
\end{aligned}$$

for a suitable $\gamma_k^n \in \mathbb{Z}$. Since $H_\mathrm{s} = \alpha \cdot H_\mathrm{a}$ the equation reduces to

$$\begin{aligned}
\varphi^\mathrm{Mod}(m,k) &= \varphi^\mathrm{Mod}(0,k) + \sum_{n=0}^{m-1}\left[\left(\varphi(n+1,k) - \varphi(n,k) + \gamma_k^n\right)\alpha\right] && (4.40)\\
&= \varphi^\mathrm{Mod}(0,k) + \alpha\sum_{n=0}^{m-1}\left[\varphi(n+1,k) - \varphi(n,k) + \gamma_k^n\right] && (4.41)\\
&= \varphi^\mathrm{Mod}(0,k) + \alpha\sum_{n=0}^{m-1}\left[\varphi(n+1,k) - \varphi(n,k)\right] + \alpha\sum_{n=0}^{m-1}\gamma_k^n && (4.42)\\
&= \varphi^\mathrm{Mod}(0,k) + \alpha\left[\varphi(m,k) - \varphi(0,k)\right] + \alpha\sum_{n=0}^{m-1}\gamma_k^n\ . && (4.43)
\end{aligned}$$

Note that in case $\alpha \in \mathbb{N}$, the sum in (4.43) has always an integer value since $\gamma_k^n \in \mathbb{Z}$. It is therefore irrelevant for the phase value $\varphi^\mathrm{Mod}(m,k)$ and can be dropped. When setting $\varphi^\mathrm{Mod}(0,k) = \alpha\cdot\varphi(0,k)$ we get

$$\begin{aligned}
\varphi^\mathrm{Mod}(m,k) &= \alpha\cdot\varphi(0,k) + \alpha\left[\varphi(m,k) - \varphi(0,k)\right] && (4.44)\\
&= \alpha\cdot\varphi(m,k) && (4.45)
\end{aligned}$$

which concludes the proof of Equation (4.31).

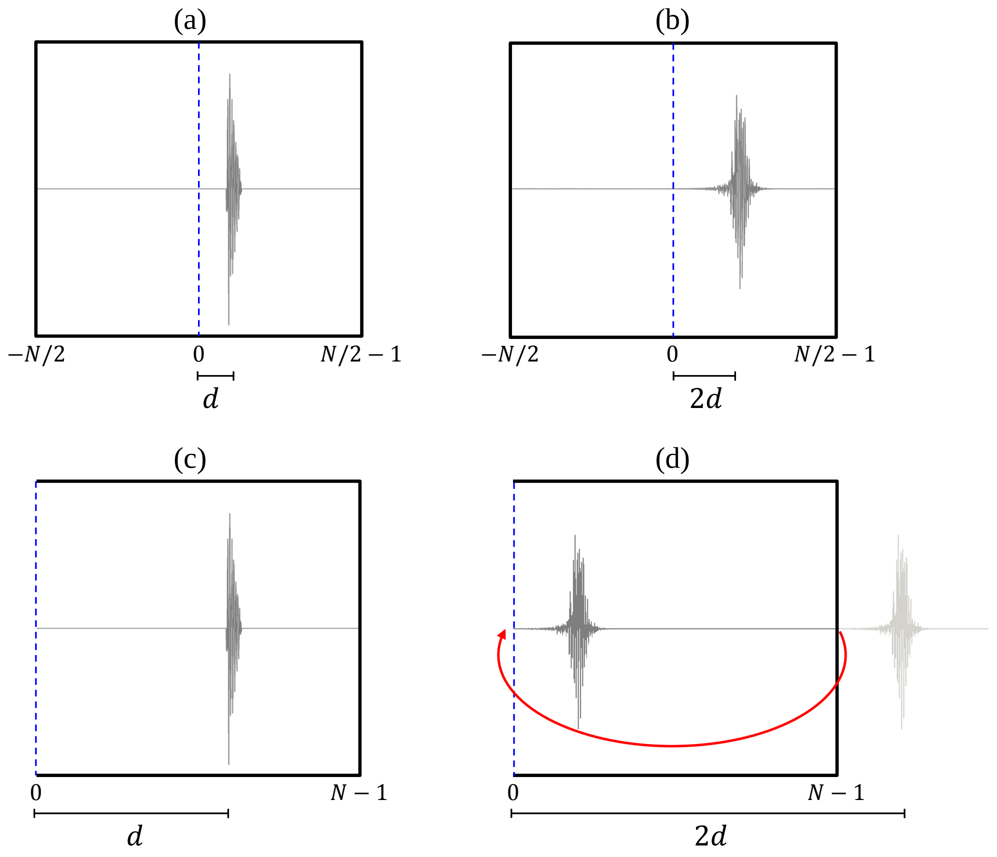


Figure 4.13: Int-PV-TSM for a single frame x_m. **(a):** Original frame x_m. Phases are measured at the frame's center (dashed blue line). **(b):** x_m^{Mod} for $\alpha = 2$. **(c):** Original frame x_m. Phases are measured at the frame's left border (dashed blue line). **(d):** x_m^{Mod} for $\alpha = 2$. The transient is cyclically shifted.

Int-PV-TSM has the advantage that the process of computing instantaneous frequencies by using the phase vocoder can be skipped. This not only simplifies the phase update but also guarantees that the modified phases are correct—unlike as in PV-TSM where errors in the instantaneous frequency estimation propagate through all modified phases. It shows that the absence of phase errors also reduces the effects of transient smearing and phasieness artifacts. However, when implemented naively the output signals of Int-PV-TSM still suffer from artifacts, which we discuss next.

4.3.3 Implementation Issues

Multiplying the phases of a frame x_m with a factor $\alpha \in \mathbb{N}$ as defined by Equation (4.31) has an effect on the localization of sound events within the frame. Loosely speaking, the operation scales the position of sound events by α with respect to the point in the frame where the phase was measured. This is visualized in Figure 4.13. In Figure 4.13a we see a frame x_m containing a transient that is located at a distance of d samples from the frame's center—the point where the phases are measured when applying the STFT as defined in Equation (4.12) [4]. Updating the frame's phases by Equation (4.31) with a stretching factor $\alpha = 2$ results in the frame x_m^{Mod} shown in Figure 4.13b. Here, the transient's position was scaled by the factor α, relative to the frame's center. In the context of Int-PV-TSM, this relocation is actually desirable. Considering the whole original input signal x, the transient is initially located at the position $H_{\mathrm{a}} \cdot m + d$ by Equation (4.1). When reconstructing the stretched output signal by Equation (4.2), the modified transient is located at

$$\begin{aligned} H_{\mathrm{s}} \cdot m + \alpha \cdot d &= \alpha \cdot H_{\mathrm{a}} \cdot m + \alpha \cdot d && (4.46) \\ &= \alpha(H_{\mathrm{a}} \cdot m + d) && (4.47) \end{aligned}$$

in the output signal y and is therefore correctly relocated to α times its original position.

When implementing Int-PV-TSM, the situation is usually slightly different. In programming languages such as MATLAB [140], audio signals are typically represented as vectors of finite length. An analysis frame x_m may therefore be represented by a vector `xm` of length N such that

$$\mathtt{xm}(r) = x_m(r - N/2) \ , \tag{4.48}$$

for $r \in [0 : N-1]$. Computing the frame's Fourier transform can be done by multiplying `xm` with the $(N \times N)$ *discrete Fourier transform* matrix

$$\mathrm{DFT}_N(r,k) = \exp(-2\pi i k r/N) \ , \tag{4.49}$$

for $r, k \in [0 : N-1]$, see also [146, Chapter 2.4]. The resulting vector of complex values represents the frame's Fourier coefficients that also encode the phase information. However, since `xm`'s indices run in the interval $r \in [0 : N-1]$ (and not in $[N/2 : N/2-1]$ as in Equation (4.12)) the phases are now measured with respect to the frames, beginning, not it's center. This situation is visualized in Figure 4.13c that shows the same frame as in Figure 4.13a, but with the index range $[0 : N-1]$. The distance d is therefore given with respect to the frame's left border. When updating the frame's phases according to Equation (4.31) with $\alpha = 2$, the transient is relocated to the position $\alpha \cdot d$, see Figure 4.13d. However, the position $\alpha \cdot d$ is located outside of the frame and the transient is therefore cyclically shifted. When reconstructing the output signal y according to Equation (4.2), the transient is therefore not located correctly. In Int-PV-TSM this may cause artifacts that are similar to transient doubling (see Section 4.1.3.2).

[4]A time-frequency bin's phase $\varphi(m,k)$ represents the phase of the sinusoidal component $X(m,k) \cdot \exp(2\pi i k r/N)$ at $r = 0$. Since frames are centered around zero by Equation (4.1), phase values obtained by the STFT as defined in Equation (4.12) are measured with respect to the frame's center.

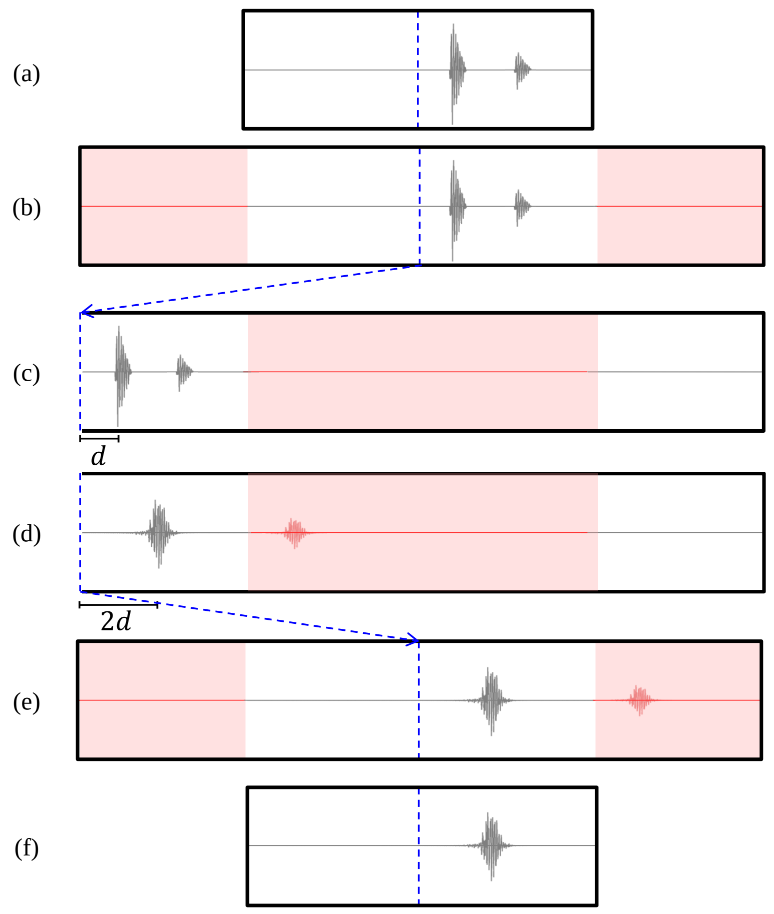


Figure 4.14: Strategy to prevent incorrect relocations of sound events in implementations of Int-PV-TSM. **(a):** Frame x_m. **(b):** Zero padded frame. **(c):** Circularly shifted frame. **(d):** Phase-updated frame. **(e):** Second circular shift. **(f):** Frame with removed zero pad.

To avoid this kind of incorrect relocations of sound events in implementations of Int-PV-TSM, it is both necessary to measure the phase at the frame's center as well as to prevent cyclic shifts of sound events. A strategy to approach both tasks is visualized in Figure 4.14. Figure 4.14a shows a frame x_m that contains two transient sound events. In a first step, the vector representing the frame is symmetrically padded with $\alpha \cdot N/2$ zeros (marked red in Figure 4.14b). The idea behind this operation is that any sound event in the original frame that would be cyclically shifted by the phase update is instead moved into the zero-padded region. Since the largest distance a sound event can have from the frame's center is $N/2$ samples, the region must have a size of at least $\alpha \cdot N/2$ in order to prevent sound events from cyclically reentering the original frame. In a second step the padded frame is circularly shifted by half its length. This operation, also known as *fftshift*, moves the frame's center to its border (Figure 4.14c). When updating the frame's phases according to Equation (4.31), sound events are now relocated with respect to the original frame's center (indicated by the blue dashed arrows in Figure 4.14). In Figure 4.14d one can

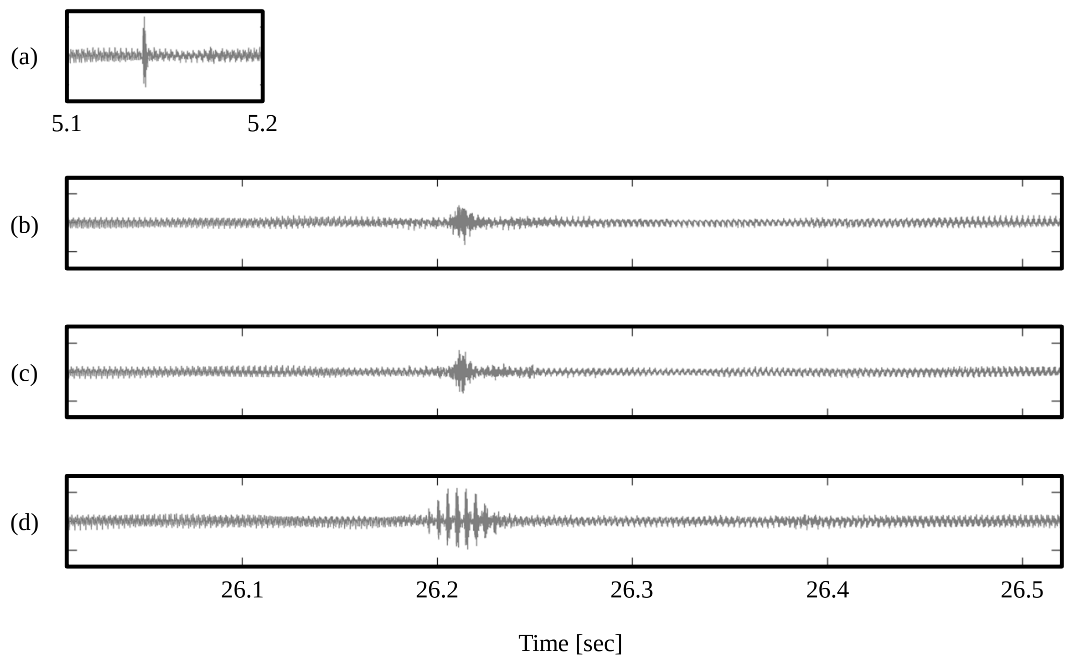


Figure 4.15: Example of an audio signal stretched with Cascade-TSM by a factor of $\alpha = 5.1$. **(a):** Input signal x. **(b):** Output signal of Cascade-TSM for the cascade coarse→fine. **(c):** Output signal of Cascade-TSM for the cascade fine→coarse. **(d):** Output signal of HP-TSM.

also see that the second transient is now located in the zero padded region. Finally, the modified frame is again circularly shifted (Figure 4.14e) and the zero pad is removed (Figure 4.14f). In the resulting frame, the first transient is relocated correctly while the second one is prevented from cyclically reentering the frame. Therefore, this strategy allows to overcome the issues discussed before. Note that the second transient, although not being included in this frame any longer, will still be contained in the output signal of Int-PV-TSM since it is also contained in other frames where it will not be removed by the described procedure.

4.3.4 Cascade-TSM

In order to stretch input signals by arbitrary stretching factors $\alpha \in \mathbb{Q}^+$, we cascade HP-TSM and Int-PV-TSM. This cascaded procedure constitutes a novel TSM approach that we call Cascade-TSM. Given an input signal x and a stretching factor α, the core idea is to sequentially stretch x by a factor of

$$\alpha^{(\text{coarse})} = \max\left(1, \text{round}(\alpha)\right) = \max\left(1, \lfloor \alpha + 0.5 \rfloor\right) \ , \tag{4.50}$$

using Int-PV-TSM and by

$$\alpha^{(\text{fine})} = \frac{\alpha}{\alpha^{(\text{coarse})}} \ , \tag{4.51}$$

using HP-TSM. When cascading the two procedures, x is stretched by $\alpha^{(\text{coarse})} \cdot \alpha^{(\text{fine})} = \alpha$. Note that by the definitions of $\alpha^{(\text{coarse})}$ and $\alpha^{(\text{fine})}$, Cascade-TSM reduces to HP-TSM for $\alpha < 1.5$.

Cascade-TSM yields output signals with high quality, even for rather extreme stretching factors ($\alpha > 3$). Figure 4.15a shows an excerpt of our running example—a tone played on a violin with a single click of castanets at the excerpt's center. This input signal should be stretched by a factor of $\alpha = 5.1$ using Cascade-TSM. To this end, it is first modified with $\alpha^{(\text{coarse})} = 5$ using Int-PV-TSM and afterwards stretched a second time by $\alpha^{(\text{fine})} = 1.02$ using HP-TSM. The output of this cascade is shown in Figure 4.15b. We can see that the transient, although being slightly prolonged, is still present. When listening to the signal, it shows that the transient actually has an acceptable quality. Note that instead of first applying Int-PV-TSM for the coarse stretching and HP-TSM for the fine adjustment afterwards (coarse→fine), it is also possible to first apply HP-TSM and Int-PV-TSM in the second step (fine→coarse). The output signal of this cascade is shown in Figure 4.15c. When comparing the two cascade's output signals, both visually and acoustically, they are rather similar. However, applying the cascade fine→coarse is, in this example, about three times faster than applying the cascade coarse→fine in terms of computation time. The reason for this is that HP-TSM is computationally more demanding than Int-PV-TSM, mainly due to the harmonic percussive separation. In the cascade fine→coarse, HP-TSM is applied to a short input signal while being applied to a long input signal in the cascade coarse→fine. This explains the difference in runtime. As a final comparison, Figure 4.15d shows the output signal when using only HP-TSM to stretch the signal. Here we see that OLA introduces significant stuttering into the output signal for such an extreme stretching factor and is therefore is not capable of preserving the transient properly. This shows that Cascade-TSM is better suited for modifications with large stretching factors than HP-TSM.

4.4 Applications of TSM

4.4.1 Music Synchronization—Non-linear TSM

In scenarios like *automated soundtrack generation* [147] or *automated DJing* [19, 110], it is often necessary to synchronize two or more music recordings by aligning musically related beat positions in time. However, music recordings do not necessarily have a constant tempo. In this case, stretching or compressing the recordings by a constant factor α is insufficient to align their beat positions. Instead, the recordings'time-scales need to be modified in a non-linear fashion. The goal of non-linear TSM is to modify an audio signal according to a strictly monotonously increasing *time-stretch function* $\tau : \mathbb{R} \to \mathbb{R}$, which defines a mapping between points in time (given in seconds) of the input and output signals. Figure 4.16 shows an example of such a function. The first part, shown in red, has a slope greater than one. As a consequence, the red

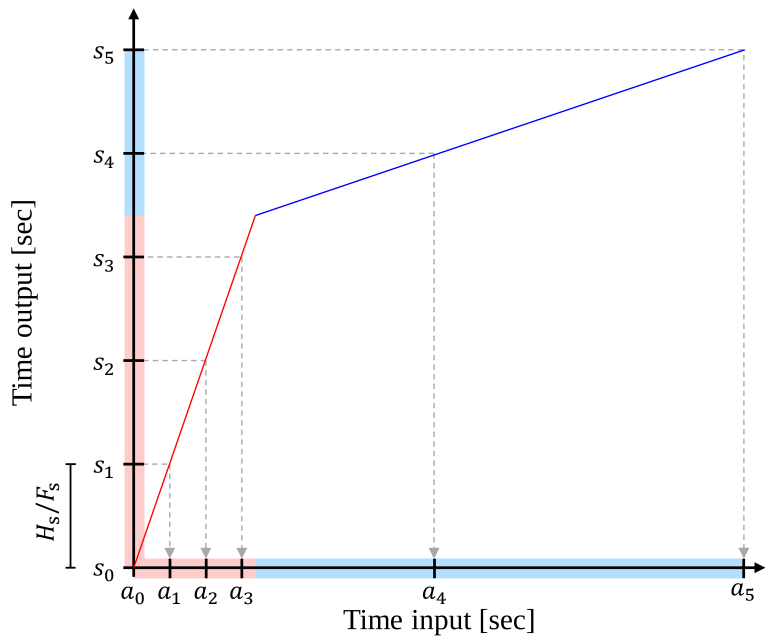


Figure 4.16: Example of a non-linear time-stretch function τ.

region in the input signal is mapped to a larger region in the output, resulting in a stretch. The slope of the function's second part, shown in blue, is smaller than one, leading to a compression of this region in the output. One possibility to realize this kind of non-linear modifications is to define the positions of the analysis frames according to the time-stretch function τ, instead of a constant analysis hopsize H_a. The process of deriving the analysis frames' positions is presented in Figure 4.16. Given τ, we first fix a synthesis hopsize H_s as we did for linear TSM (see Section 4.1). From this, we derive the *synthesis instances* $s_m \in \mathbb{R}$ (given in seconds), which are the positions of the synthesis frames in the output signal:

$$s_m = \frac{m \cdot H_\mathrm{s}}{F_\mathrm{s}} \; . \tag{4.52}$$

By inverting τ, we compute the *analysis instances* $a_m \in \mathbb{R}$ (given in seconds):

$$a_m = \tau^{-1}(s_m) \; . \tag{4.53}$$

When using analysis frames indicated by the analysis instances for TSM with a procedure of our choice, the resulting output signal is modified according to the time-stretch function τ. To this end, all of the previously discussed TSM procedures (except for Int-PV-TSM and Cascade-TSM) can also be used for non-linear TSM.

A very convenient way of defining a time-stretch function is by introducing a set of *anchor points*. An anchor point is a pair of time positions where the first entry specifies a time position in the

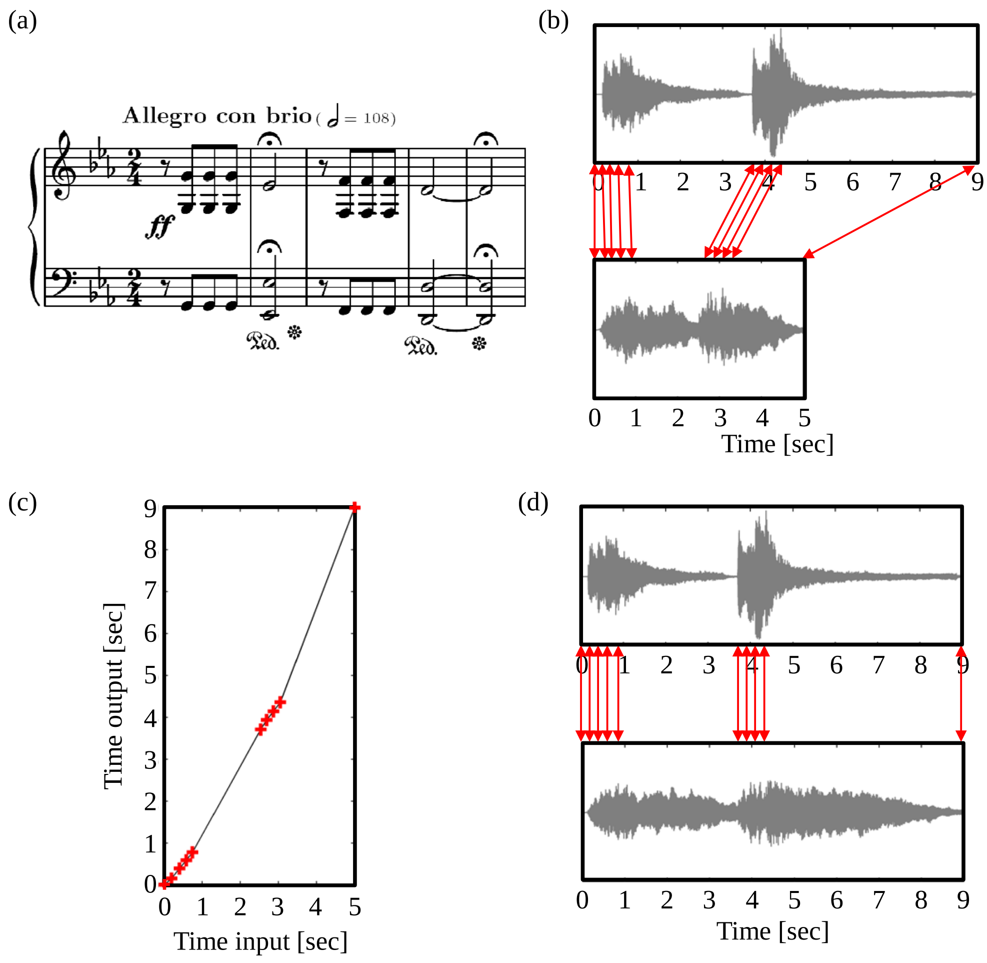


Figure 4.17: **(a)**: Score of the first five measures of Beethoven's Symphony No. 5. **(b)**: Waveforms of two performances. Corresponding onset positions are indicated by red arrows. **(c)**: Set of anchor points (indicated in red) and inferred time-stretch function τ. **(d)**: Onset-synchronized waveforms of the two performances.

input signal and the second entry is a time position in the output signal. The actual time-stretch function τ is then obtained by a linear interpolation between the anchor points. Figure 4.17 shows a real-world example of a non-linear modification. In Figure 4.17b, we see the waveforms of two recorded performances of the first five measures of Beethoven's Symphony No. 5. The corresponding time positions of the note onsets are indicated by red arrows. Obviously, the first recording is longer than the second. However, the performances' tempi do not differ by a constant factor. While the eighth notes in the first and third measure are played at almost the same tempo in both performances, the durations of the half notes (with fermata) in measures two and five are significantly longer in the first recording. The mapping between the note onsets of the two performances is therefore non-linear. In Figure 4.17c, we define eight anchor points that map the onset positions of the second performance to the onset positions of the first performance (plus two additional anchor points aligning the recordings' start times and end times, respectively). Based

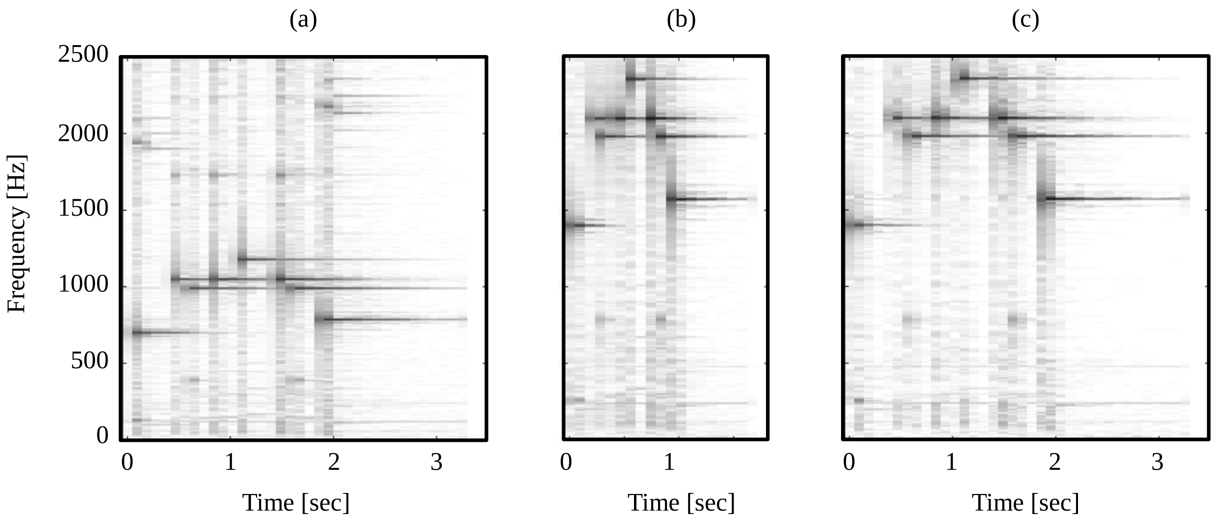


Figure 4.18: Pitch-shifting via resampling and TSM. **(a)**: Spectrogram of an input audio signal. **(b)**: Spectrogram of the resampled signal. **(c)**: Spectrogram after TSM application.

on these anchor points and the derived time-stretch function τ, we then apply the TSM procedure of our choice to the second performance in order to obtain a version that is onset-synchronized with the first performance (Figure 4.17d).

4.4.2 Pitch-Shifting

Pitch-shifting is the task of changing an audio recording's pitch without altering its length—it can be seen as the dual problem to TSM. While there are specialized pitch-shifting procedures [91, 186], it is also possible to approach the problem by combining TSM with resampling. The core observation is that resampling a given signal and playing it back at the original sampling rate changes the length and the pitch of the signal at the same time[5]. To pitch-shift a given signal, it is therefore first resampled and afterwards time-scale modified in order to compensate for the change in length. More precisely, an audio signal, sampled at a rate of $F_\mathrm{s}^{(\mathrm{in})}$, is first resampled to have a new sampling rate of $F_\mathrm{s}^{(\mathrm{out})}$. When playing back the signal at its original sampling rate $F_\mathrm{s}^{(\mathrm{in})}$, this operation changes the signal's length by a factor of $F_\mathrm{s}^{(\mathrm{out})}/F_\mathrm{s}^{(\mathrm{in})}$ and scales its frequencies by the term's inverse. For example, musically speaking, a factor of $F_\mathrm{s}^{(\mathrm{out})}/F_\mathrm{s}^{(\mathrm{in})} = \frac{1}{2}$ increases the pitch content by one octave. To compensate for the change in length, the signal needs to be stretched by a factor of $\alpha = F_\mathrm{s}^{(\mathrm{in})}/F_\mathrm{s}^{(\mathrm{out})}$, using a TSM procedure.

To demonstrate this, we show an example in Figure 4.18, where the goal is to apply a pitch-shift of one octave to the input audio signal. The original signal has a sampling rate of $F_\mathrm{s}^{(\mathrm{in})} = 44100$ Hz (Figure 4.18a). To achieve the desired pitch-shift, the signal is resampled to $F_\mathrm{s}^{(\mathrm{out})} = 22050$ Hz

[5]The same effect can be simulated with vinyl records by changing the rotation speed of the record player.

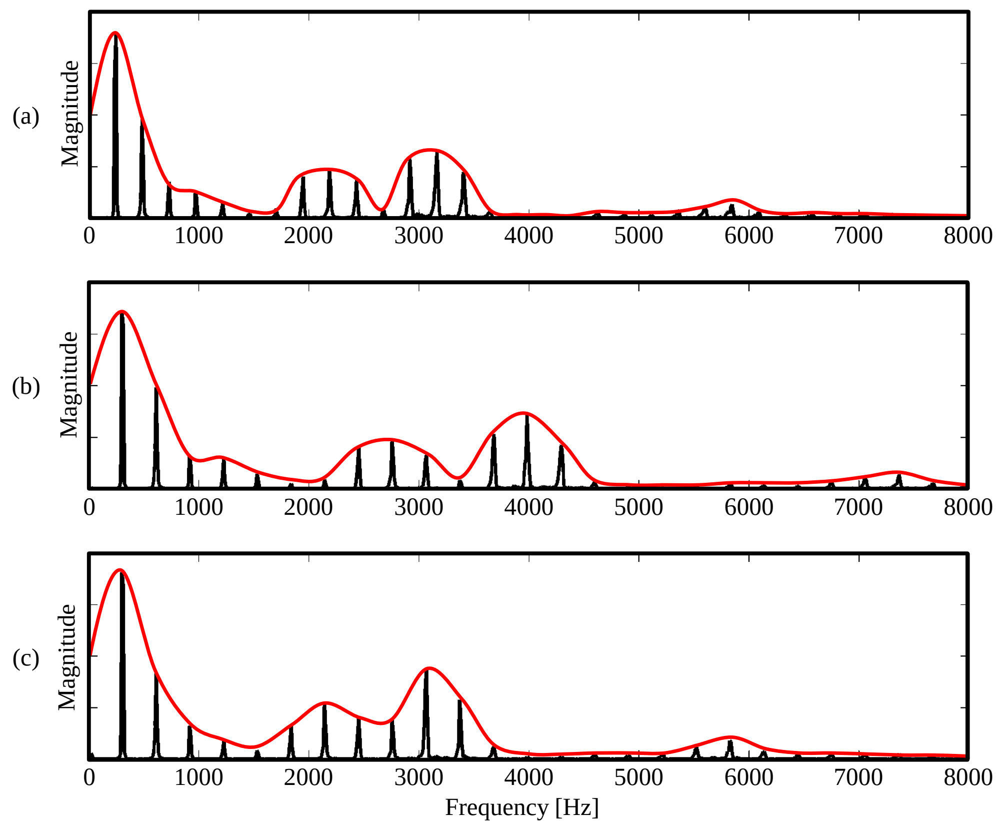


Figure 4.19: Frequency spectra for a fixed frame m of different versions of a singing voice recording. The spectral envelopes are marked in red. **(a)**: Original spectrum $X(m)$ with spectral envelope Γ_m. **(b)**: Pitch-shifted spectrum $X^{\mathrm{Shift}}(m)$ with spectral envelope $\Gamma_m^{\mathrm{Shift}}$. **(c)**: Pitch-shifted spectrum $X^{\mathrm{Mod}}(m)$ with adapted spectral envelope.

(Figure 4.18b). One can see that the resampling changed the pitch of the signal as well as its length when interpreting the signal as still being sampled at $F_{\mathrm{s}}^{(\mathrm{in})} = 44100$ Hz. While the change in pitch is desired, the change in length needs to be compensated for. Thus, we stretch the signal by a factor of $\alpha = 44100 \text{ Hz}/22050 \text{ Hz} = 2$ to its original length (Figure 4.18c).

The quality of the pitch-shifting result crucially depends on various factors. First, artifacts that are produced by the applied TSM procedure are also audible in the pitch-shifted signal. However, even when using a high-quality TSM procedure, the pitch-shifted signals generated by the method described above often sound unnatural. For example, when pitch-shifting singing voice upwards by several semitones, the modified voice has an artificial sound, sometimes referred to as the *chipmunk* effect [226]. Here, the problem is that the *spectral envelope*, which is the rough shape of a frequency spectrum, is of central importance for the timbre of a sound. In Figure 4.19a, we see a frame's frequency spectrum from a singing voice recording. Due to the harmonic nature of the singing voice, the spectrum shows a comb-like pattern where the energy is distributed at multiples of a certain frequency—in this example roughly 250 Hz. The peaks in the pattern are called the *harmonics* of the singing voice. The magnitudes of the harmonics follow a certain shape which is specified by the spectral envelope (marked in red). Peaks in the spectral envelope are known as *formants.* In the example from Figure 4.19a, we see four formants at around 200 Hz, 2200 Hz, 3100 Hz, and 5900 Hz. The frequency positions of these formants are closely related to the singing voice's timbre. In Figure 4.19b, we see the spectrum of the same frame after being pitch-shifted by four semitones with the previously described pitch-shifting procedure. Due to the resampling, the harmonics' positions are scaled. The spectral envelope is therefore scaled as well, relocating the positions of the formants. This leads to a (usually undesired) change in timbre of the singing voice.

One strategy to compensate for this change is outlined in [28]. Let X and X^{Shift} be the STFTs of a given input signal x and its pitch-shifted version x^{Shift}, respectively. Fixing a frame index m, let $X(m)$ and $X^{\mathrm{Shift}}(m)$ be the frequency spectra of the m^{th} frames in X and X^{Shift}. Furthermore, let $\Gamma_m : [0 : N-1] \rightarrow \mathbb{R}$ and $\Gamma_m^{\mathrm{Shift}} : [0 : N-1] \rightarrow \mathbb{R}$ denote the spectral envelopes of $X(m)$ and $X^{\mathrm{Shift}}(m)$, respectively. Our goal is to compute modified spectra $X^{\mathrm{Mod}}(m)$ that have the frequency content of $X^{\mathrm{Shift}}(m)$ but the original spectral envelopes Γ_m. To this end, we normalize the magnitudes of $X^{\mathrm{Shift}}(m)$ with respect to its spectral envelope $\Gamma_m^{\mathrm{Shift}}$ and then scale them by the original envelope Γ_m:

$$X^{\mathrm{Mod}}(m,k) = X^{\mathrm{Shift}}(m,k) \cdot \frac{\Gamma_m(k)}{\Gamma_m^{\mathrm{Shift}}(k)} \, . \tag{4.54}$$

In Figure 4.19c we see the frame's spectrum from Figure 4.19b after the envelope adaption, leading to a signal that sounds more natural.

There exist several approaches to estimate spectral envelopes, many of them either based on *linear predictive coding* (LPC) [139] or on the *cepstrum* (the inverse Fourier transform of the logarithmic magnitude spectrum) [178, 177]. Generally, the task of spectral envelope estimation is highly complex and error-prone. In envelope or formant-preserving pitch-shifting methods, it is often necessary to manually specify parameters (e.g., the pitch range of the sources in the recording) to make the envelope estimation more robust.

4.4.3 Websources

Various free implementations of TSM procedures are available in different programming languages. MATLAB implementations of OLA, WSOLA, PV-TSM, and TSM based on HPS, as well as additional test and demo material can be found in the *TSM Toolbox* [40, 39]. A further MATLAB implementation of PV-TSM can be found at [57]. PV-TSM implemented in Python is included in *LibROSA* [136]. Furthermore, there are open source C/C++ audio processing libraries that offer TSM functionalities. For example, the *Rubber Band Library* [12] includes a transient preserving PV-TSM and the *SoundTouch Audio Processing Library* [164] offers a WSOLA-like

TSM procedure. Finally, there also exist proprietary commercial implementations such as the *élastique* algorithm by *zplane* [231].

4.5 Conclusions and Further Notes

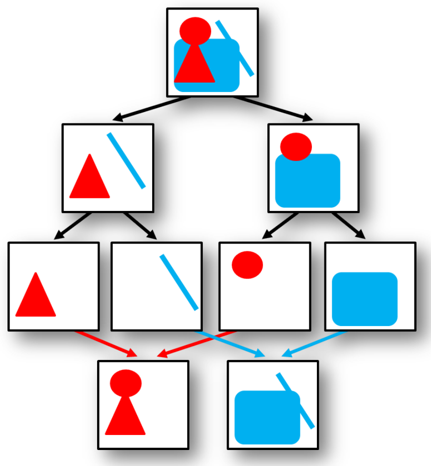
In this chapter, we reviewed time-scale modification of music signals, presented fundamental principles, and discussed various well-known time and frequency-domain TSM procedures. Additionally, we pointed out more involved procedures proposed in the literature, which are—to varying extents—based on the fundamental approaches we reviewed. Beyond discussing technical details, a main motivation was to give illustrative explanations in a tutorial-like style. We also aimed to foster a deep understanding of the strengths and weaknesses of various TSM approaches by discussing typical artifacts and the importance of parameter choices.

As a main contribution of this thesis, we proposed two novel TSM approaches that are both inspired by the idea of combining TSM strategies. The first method is based on separating an input signal into a harmonic and a percussive component, which are then modified using two different fundamental TSM procedures. This approach has the advantage that no explicit transient detection is necessary to preserve transients in the recombined output signal. The second method builds upon a TSM approach that yields output signals of high quality, but only for integer stretching factors. We proposed to cascade this approach with our TSM procedure based on harmonic-percussive separation in order to be able to modify input signals by arbitrary stretching factors with high quality.

In future research, we would like to further explore the concept of combining fundamental TSM strategies in order to overcome recurring artifacts like timbre changes in output signals. An interesting idea is to decompose a given audio signal not only into a harmonic and a percussive component, but into a harmonic, a percussive, and a residual component as discussed in Chapter 3. As stated in Section 3.4, the residual component often captures the "texture" of a sound. This kind of decomposition might therefore be helpful to better preserve the input signal's timbre in a TSM procedure's output signal.

Chapter 5

Cascaded Music Signal Decomposition for Singing Voice Extraction



In this chapter, we deal with *singing voice extraction*, the task of separating the singing voice from accompanying instruments in a music recording. We hereby closely follow our original contribution presented in [41].

In recent years, a lot of effort has been put into the development of algorithms for extracting singing voice from music recordings. This interest emerged from both scientific curiosity for better understanding the characteristics of human singing [116] as well as the commercial need

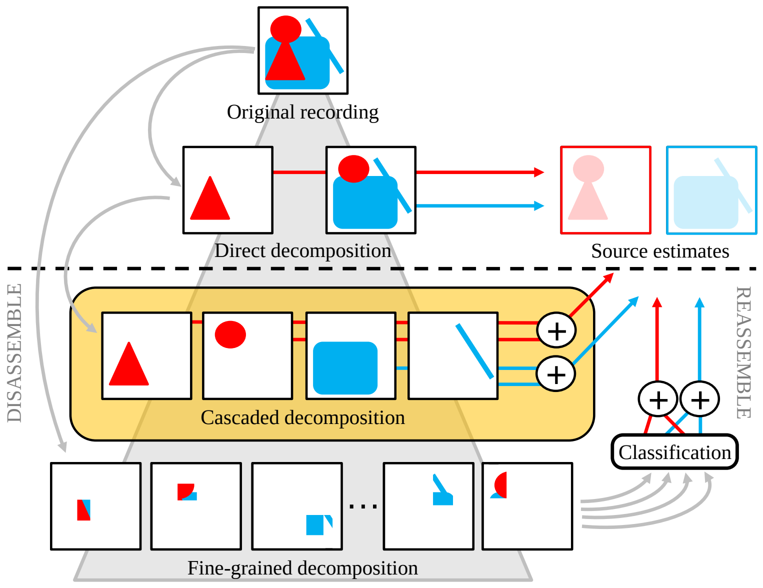


Figure 5.1: Schematic overview of different decomposition strategies. Colors encode sources, shapes depict characteristics.

for such techniques in applications such as music remixing, remastering, and production [223]. While the extraction of a specific musical voice from a complex music recording is a difficult problem in general, the extraction of the singing voice is particularly challenging due to the wide range of different sounds the human voice is capable of producing. For example, singing voice can include stable harmonic sounds, transient sounds resulting from plosive or fricative phonemes in the sung lyrics [124], noise-like breathing sounds, vibrato sounds, and so on.

There exists a large variety of algorithmic approaches to singing voice extraction. A strict classification of these approaches into specific categories is difficult as many techniques overlap or combine different ideas. However, to motivate our novel approach for singing voice extraction described in this chapter, we first take a simplified view on existing procedures and argue that many of them tend to follow either one of two basic strategies, see Figure 5.1. Approaches employing a *direct decomposition* strategy aim to decompose a given music recording directly into one component that contains the singing voice and one that contains the accompaniment. These methods are usually based on some observation about a specific characteristic of singing voice. Examples for such characteristics are the clear and strong harmonic structure of singing voice [222], its spectral sparseness [104], the high variance of singing voice in contrast to the repeating structure of accompanying music [174, 68, 138], the presence of vibrato and glissando in singing voice [207, 113], or the occurrence of specific spectral patterns [137]. Rather than explicitly extracting the singing voice, these decomposition procedures are designed to extract the specific characteristic from a given recording. As intended, this usually goes along with extracting large portions of the singing voice. However, while the resulting decompositions have an explicit semantic meaning, the procedures are often not designed to also flexibly incorporate knowledge about additional characteristics.

Approaches which follow a *disassemble and reassemble* strategy first decompose the given music recording into a large set of fine-grained components. Afterwards, all components are classified to belong to either the singing voice or the accompaniment and reassembled accordingly. Common techniques to perform this kind of decomposition are non-negative matrix factorization (NMF) and related formulations [16, 52, 160, 199, 215] or time-frequency decompositions [103]. Although the fine-grained decomposition yields a high degree of flexibility when reassembling the sources, the correct classification of the components constitutes a challenging problem. Depending on the chosen decomposition technique, the components may not even have a semantic interpretation anymore. The classification can either be done in an unsupervised fashion [16, 103], in a supervised way [92, 160, 215], or it can be derived from the decomposition process itself [16, 52, 199].

Combining ideas from both strategies, we propose in this chapter a novel approach for singing voice extraction. Inspired by the disassemble and reassemble strategy, a given music signal is first split into a set of components. However, contrary to other procedures following this methodology, we decompose the recording on a coarser granularity level by cascading different direct decomposition procedures, see Figure 5.1. This yields several advantages. On the one hand, the resulting *mid-level components* have an explicit semantic meaning, inherited directly from the sequence of applied decomposition procedures, which allows for assigning each mid-level component to either the singing voice or the accompaniment. On the other hand, the cascaded decomposition is flexible enough to account for multiple characteristics of the singing voice and the accompaniment. In the end, one mid-level component can, for example, hold sounds stemming from plosive and fricative phonemes in the singing voice while another mid-level component holds the harmonic portion of the accompaniment. Because of the explicit interpretation of all mid-level components, their classification can be done on the semantic level based on the characteristics which are assumed to be captured by the components. Estimates of the singing voice and the accompaniment are then reassembled by adding up the respective components.

The remainder of this chapter is structured as follows. In Section 5.1, we first exemplarily discuss sounds as they typical occur in recordings of singing voice and accompanying instruments and show in what kind of structures they manifest in spectrograms. A review of the proposed procedure is then given in Section 5.2, see also Figure 5.4. Our objective and subjective evaluation is described in Section 5.3 and we conclude this chapter in Section 5.4.

5.1 Characteristics of Singing Voice and Accompaniment

In this section, we motivate the choice of decomposition techniques in the cascade discussed in Section 5.2 by exemplarily investigating a spectrogram of a singing voice recording and a spectrogram of a recording of accompanying instruments. Although the investigated example, of course, does not generalize to all music recordings, it can give a feeling of what kinds of different

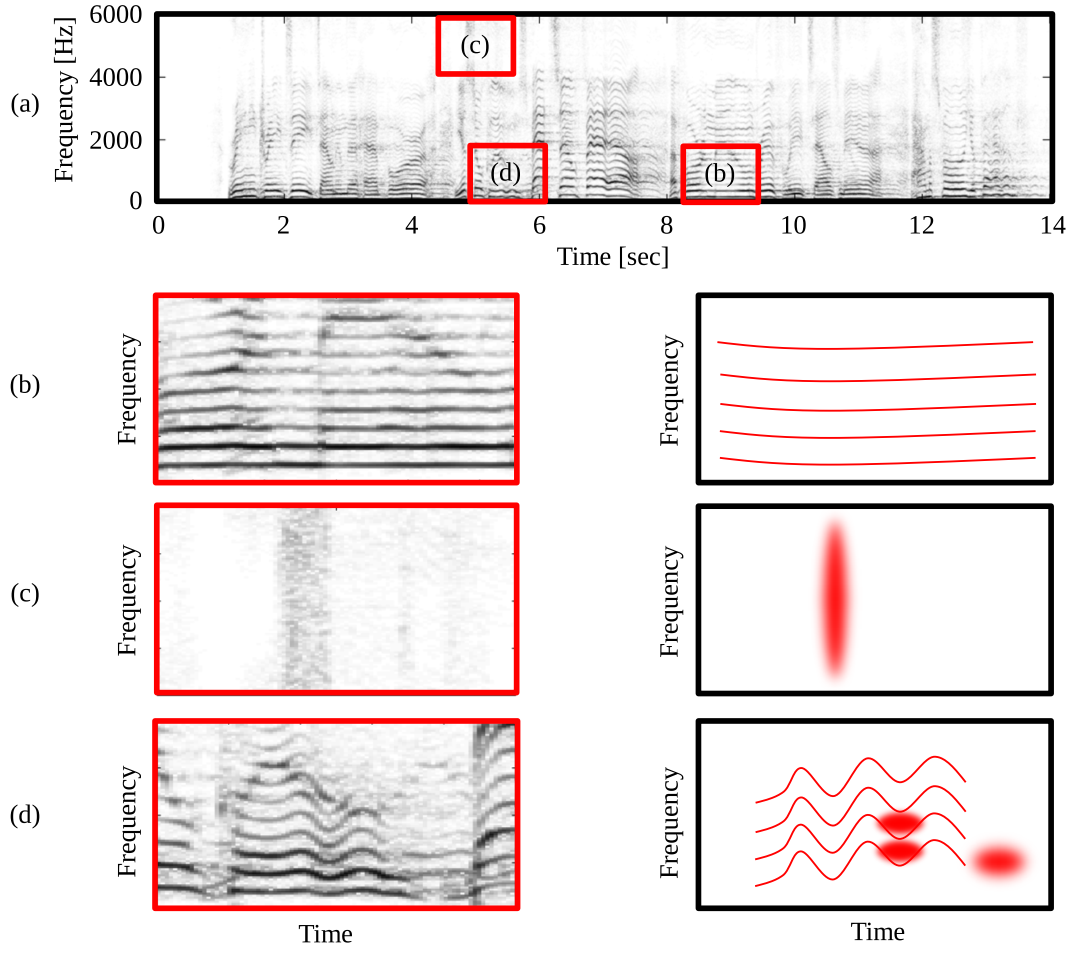


Figure 5.2: Real and depicted spectrograms of the singing voice from Bearlin. **(a):** Full spectrogram. **(b):** Tone with stable pitch. **(c):** Sound stemming from plosive/fricative phonemes in the sung lyrics. **(d):** Vibrato singing, strong formant, and breathing sound.

sounds can be typically expected in singing voice and accompaniment. Furthermore, it illustrates the spectral structures these sounds are usually associated with.

As mentioned above, the human singing voice has a wide range of different sounds that it can produce. In Figure 5.2a, we see the spectrogram of the singing voice from the music recording Bearlin (a pop recording with singing voice, accompanied by drums, piano, guitar, and bass), taken from the SiSEC dataset [224, 4] (separate tracks for the singing voice and the accompaniment are available for all recordings in this dataset). The left part of Figure 5.2b shows a magnified excerpt of the spectrogram. In this excerpt, we see a clear harmonic energy pattern that is typical for singing voice. Here, the singer sings a long tone with a stable pitch, resulting in a pattern of parallel horizontal lines. In the right part of Figure 5.2b, we see a pictrogram of this kind of structure. In the remainder of this chapter, we will often use depicted spectrograms to sketch the ideas behind our proposed decomposition cascade.

A second kind of structure that is frequently occurring in spectrograms of singing voice can be seen in Figure 5.2c. The shown vertical structures stem from plosive or fricative phonemes in the sung lyrics. These sounds are noise-like and spread over a large frequency range. Looking at Figure 5.2a, note that although one can identify various instances of these vertical structures, their exact spectral appearance differs and may depend on the respective phoneme. For example, compare the narrow vertical structure at around second 2.5 resulting from the plosive in the sung word “down” with the rather broad structure shortly after second 12, stemming from the fricative in “must”.

Figure 5.2d shows a third spectral pattern that is typical for singing voice. In contrast to Figure 5.2b, the harmonics shown in this excerpt of the singing voice’s spectrogram do not run strictly horizontally, but in a downwards, upwards, or wave-like pattern. These kinds of structures occur when a singer uses vibrato or glissando to make his singing more expressive. Furthermore, one can also see diffuse regions of energy. These regions may stem from both strong formants in the singing voice (energy in-between the harmonics) as well as from breathing sounds.

In Figure 5.3 we investigate spectral structures as they typically occur in the spectrograms of musical accompaniments. Again, as an illustrative example, we revert to the music recording Bearlin. In Figure 5.3a we see the full spectrogram of the accompaniment. Even on this coarse level, one can already notice that the structures present in this spectrogram differ from those in the spectrogram of the singing voice in Figure 5.2a. One noticeable difference is that the spectrogram of the accompaniment shows no vibrato or glissando structures. Figure 5.3b highlights a region of the full spectrogram where we see a stable harmonic energy distribution that, in this case, mainly originates from the piano. While the presence of vibrato and glissando is rather common in singing voice, this is often not the case for accompanying instruments. Many instruments, such as the piano or the accordion, even do not have a direct possibility to vary the pitch of played notes in a continuous way.

Besides the horizontal structures stemming from the stable harmonic sounds, we can also see clear vertical structures throughout the spectrogram shown in Figure 5.3a. As already discussed in Chapter 3, these kind of structures often originate from percussive sounds. In the context of musical accompaniment, these are usually produced by the drums (especially snare drum, toms, and hi-hat), but may also stem from strong onsets of other instruments such as piano. In contrast to vertical structures in Figure 5.2a, stemming from the plosive or fricative phonemes sung by the singer, the vertical structures originating from the accompanying instruments show less variation. Closely looking at Figure 5.3a shows that there are mainly two types of vertical structures that are present at multiple instances. The one kind, for example visible at around second 12.5, is a bit broader, while the second kind, that is also visible in Figure 5.3c, is thinner

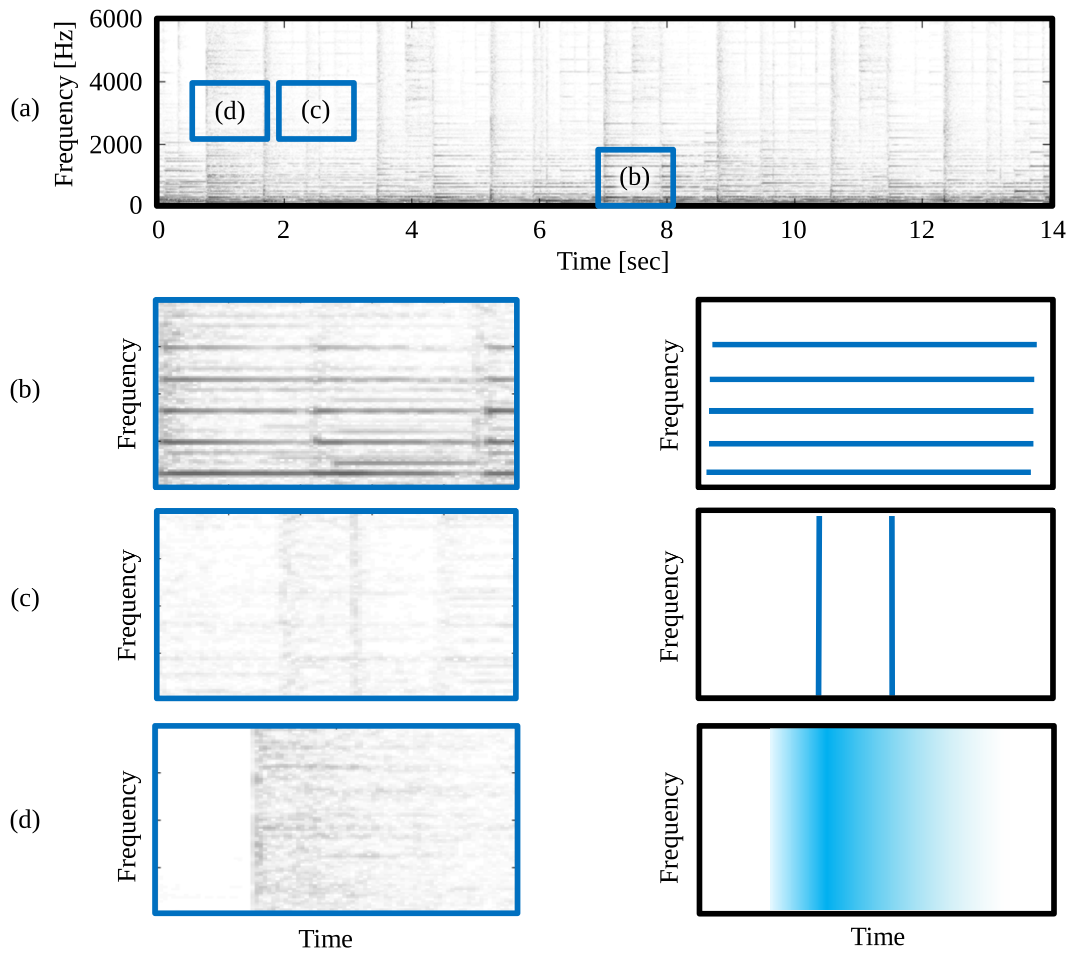


Figure 5.3: Real and depicted spectrograms of the accompaniment from Bearlin. **(a):** Full spectrogram. **(b):** Piano and bass. **(c):** Hi-hat hit. **(d):** Decaying sound of cymbal.

and less strong. Listening to the recording reveals that the broader structure corresponds to hits of the snare drum while the weaker structures originate from the hi-hat.

A third kind of spectral structure that is frequently occuring in spectrograms of musical accompaniments is shown in Figure 5.3d. Here, we see energy that is distributed rather evenly over the spectrum, slowly decaying over time. In the shown excerpt, this pattern stems from a cymbal that produces a noise-like sound. Other sources that may produce this kind of diffuse spectral structure are distorted guitars or percussion instruments like shakers. Finally, reverberation in the recording may cause reverb tails of percussive sounds, such as snare drum hits, that have a similar spectral appearance.

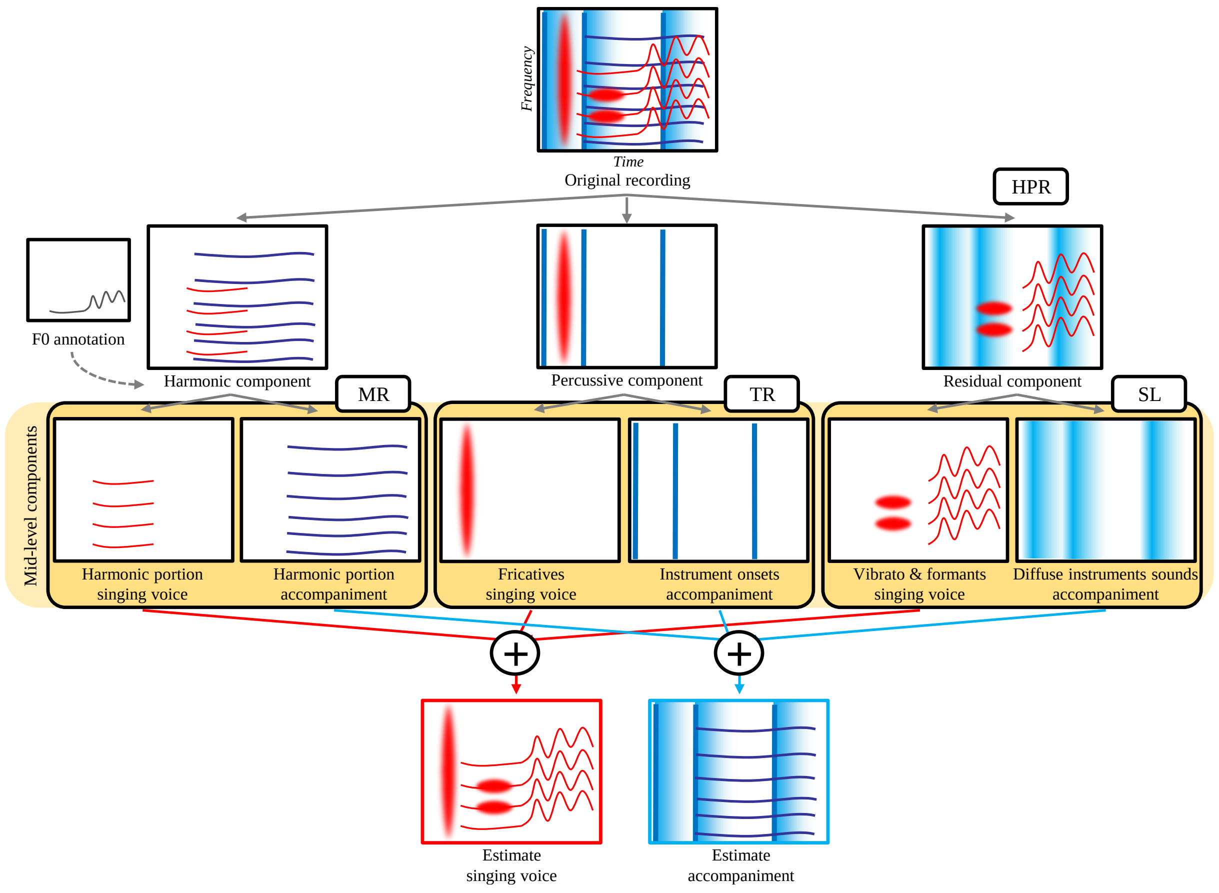


Figure 5.4: Overview of our disassemble and reassemble approach. Audio material is visualized as stylized spectrograms. Spectral portions of the singing voice are depicted in red, spectral portions of the accompaniment in blue.

5.2 Cascaded Audio Decomposition

In this section we briefly review the decomposition techniques which we use in order to disassemble a given music recording into mid-level components. As shown in Figure 5.4, we start by applying the *harmonic-percussive-residual* (HPR) decomposition procedure that we introduced in Chapter 3. The resulting three components are further processed with a fundamental frequency informed *melody-residual* (MR) decomposition procedure, a *transient-residual* (TR) decomposition procedure, and a *sparse-low rank* (SL) decomposition procedure, respectively. Finally, the resulting components are reassembled to form the estimates of the singing voice and the accompaniment.

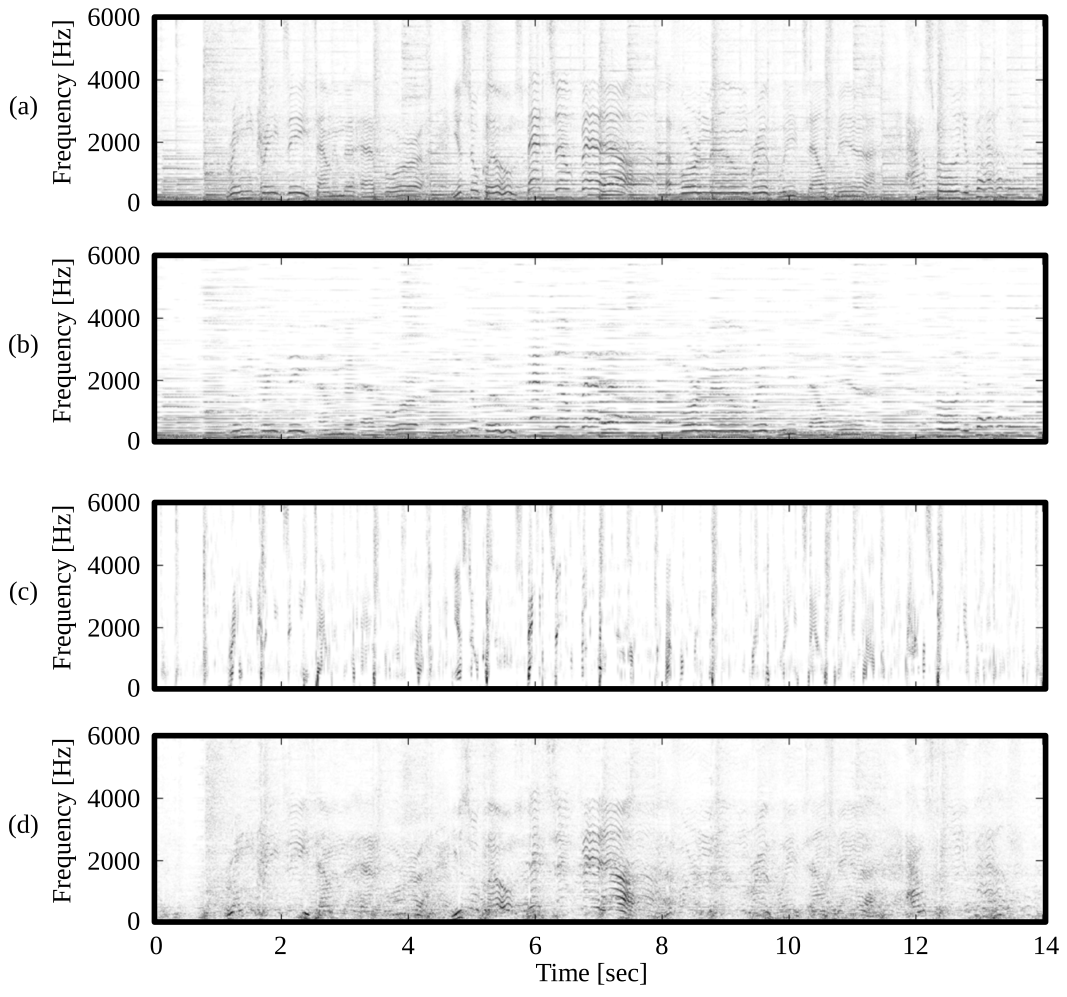


Figure 5.5: Harmonic-percussive-residual decomposition of BEARLIN. **(a):** Full spectrogram. **(b):** Spectrogram of harmonic component. **(c):** Spectrogram of percussive component. **(d):** Spectrogram of residual component.

5.2.1 Harmonic-Percussive-Residual Decomposition (HPR)

This procedure, which we carefully introduced in Chapter 3, decomposes an audio signal into a harmonic component that corresponds to horizontal spectral structures, a percussive component that corresponds to vertical spectral structures, and a residual component that captures sounds whose spectral structure is neither clearly horizontal nor vertical. Applied to a music recording with singing voice, the harmonic component usually contains most of the tonal portion of the singing voice as well as of the accompaniment. The percussive component holds sounds stemming from plosive or fricative phonemes in the sung lyrics, drum hits, or pronounced instrument onsets. In the residual component, one can often hear strong vibrato passages, breathing sounds, and sounds resulting from strong formants in the singing voice as well as noise-like instrument sounds as for example the decaying sound of a snare drum or an open hi-hat.

As an example, Figure 5.5a shows the spectrogram of the music recording Bearlin that we already used as an example in Section 5.1. Figures 5.5b-d show the spectrograms of the harmonic, the percussive, and the residual components, respectively. Here, one can for example see that the transients produced by the snare and hi-hat hits, as well as the plosive and fricative phoneme sounds of the singing voice are contained in the percussive component, see Figures 5.5c. Furthermore, one can also observe that the strong vibrato passage of the singing voice between second six and eight is captured by the residual component, see Figures 5.5d. For a detailed description and further information about the HPR decomposition method, we refer to Chapter 3 as well as to [46].

5.2.2 Melody-Residual Decomposition (MR)

This procedure, initially proposed in [222] as a singing voice extraction technique on its own, is based on the observation that singing voice usually has a clear and strong harmonic structure. Similar as in [73], given the spectrogram of a music recording and the fundamental frequency track of the sung melody (either manually annotated or automatically estimated), the singing voice is extracted from a spectrogram of the music recording by considering all time-frequency instances that correspond to the fundamental frequency track or one of its harmonics. This can be done by constructing a binary mask from the fundamental frequency track and applying it to the recording's spectrogram. This estimate, which may still contain portions of spectrally overlapping sources, is then refined using a technique known as *binary weighted non-negative matrix factorization*, see [222]. By subtracting the resulting melody component from the original recording, the accompaniment can be estimated as well. Applied to the harmonic component of the previous decomposition step, the resulting mid-level components hold the harmonic portion of the singing voice and the harmonic portion of the accompaniment, respectively.

Figure 5.6 shows the melody-residual decomposition applied to our running example Bearlin. In our proposed cascade, the melody-residual decomposition is applied to the harmonic component of the previous HPR decomposition step, which is shown in Figure 5.6a. In Figure 5.6b, we see the binary mask derived from the manually annotated fundamental frequency track of the singing voice. Finally, Figures 5.6c+d show the resulting spectrograms of the mid-level components holding the harmonic portion of the singing voice as well as the harmonic portion of the accompaniment, respectively.

5.2.3 Transient-Residual Decomposition (TR)

Initially designed for the extraction of transient noise from speech signals in [208], the core observation of this decomposition procedure is that transients produced by a specific instrument, like for example a drum, usually occur many times in a given recording. In a spectral

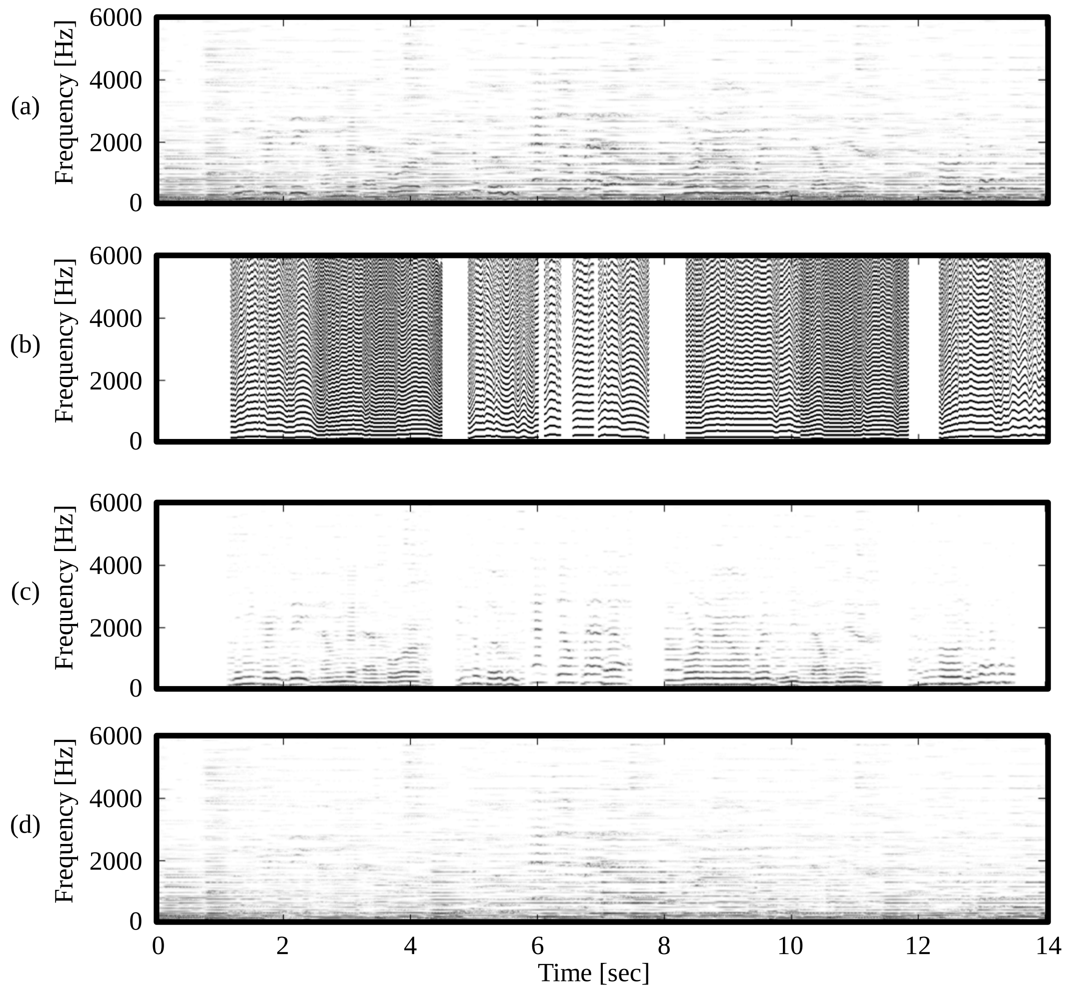


Figure 5.6: Melody-residual decomposition of the harmonic component of Bearlin. **(a):** Spectrogram of the harmonic component from the HPR decomposition. **(b):** Binary mask derived from the fundamental frequency track of the singing voice. **(c):** Spectrogram of melody mid-level component. **(d):** Spectrogram of residual mid-level component.

representation, these transients are similar to each other while the spectral structure of speech is usually more diverse. In Section 5.1 we have seen that this observation also holds for the percussive components of music recordings to a certain degree. This motivates us to apply the transient-residual decomposition proposed in [208] to this component. The procedure works as follows. Given a spectrogram representation of an audio recording, spectral frames of similar structure (having a similar spectral envelope) are identified as transient candidates, and the spectrum of a prototype transient is computed by averaging over all of these candidates. This prototype is then subtracted from the spectrogram at the identified transient positions, yielding the spectrogram of the residual component. The transient component is then computed by subtracting the residual component from the original recording. Since a music recording usually contains different kinds of frequently occurring transients, as for example the ones produced by

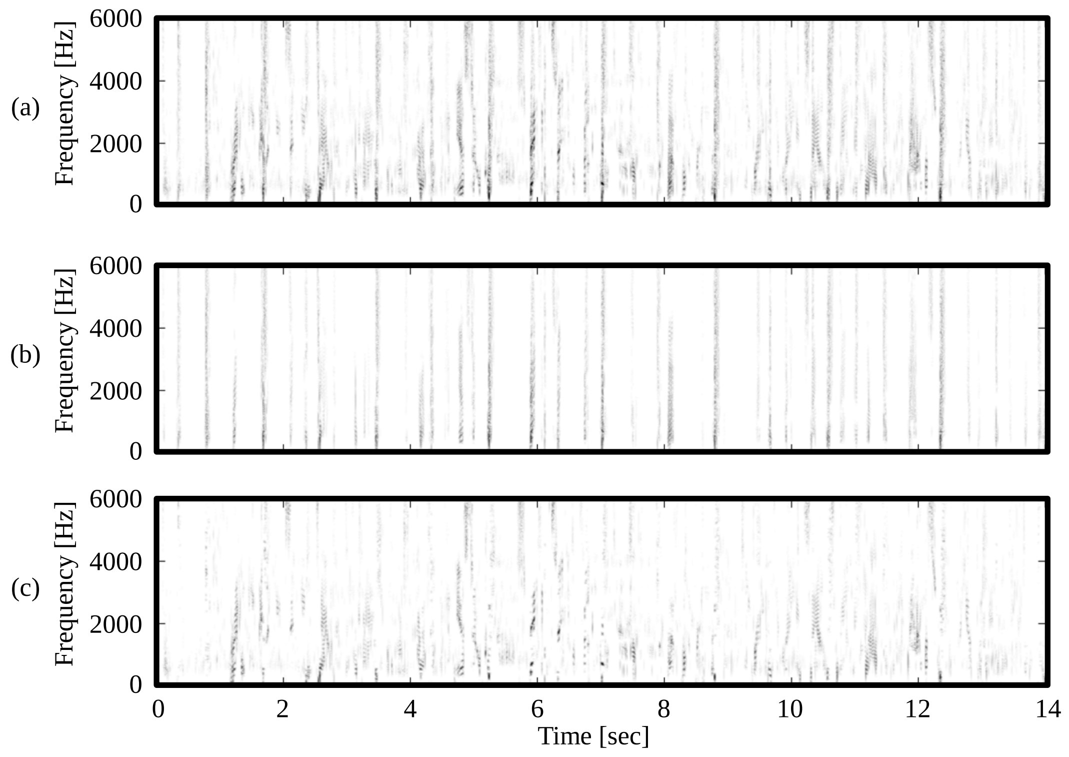


Figure 5.7: Transient-residual decomposition of the percussive component of BEARLIN. **(a):** Spectrogram of the percussive component from the HPR decomposition. **(b):** Spectrogram of transient mid-level component. **(c):** Spectrogram of residual mid-level component.

the bass drum, the snare, and the hi-hat, we apply this technique to the percussive component of the HPR decomposition in an iterative fashion. We thus decompose it into one mid-level component that typically holds the instrument transients as well as a second one typically holding plosive and fricative phonemes of the singing voice as well as other singing voice residues.

In Figure 5.7a we see the spectrogram of the percussive component from the HPR decomposition of BEARLIN, as well as the spectrograms of the transient and residual mid-level components in Figures 5.7b+c. When comparing the transient mid-level component in Figure 5.7b with the spectrogram of the accompaniment in Figure 5.3a, we can see that most of the transients occurring in the accompaniment are now captured in the transient mid-level component. Furthermore, also most of the spectral structures stemming from plosive and fricative phonemes in the singing voice as shown in Figure 5.2c are now captured in the residual mid-level component of the transient-residual decomposition.

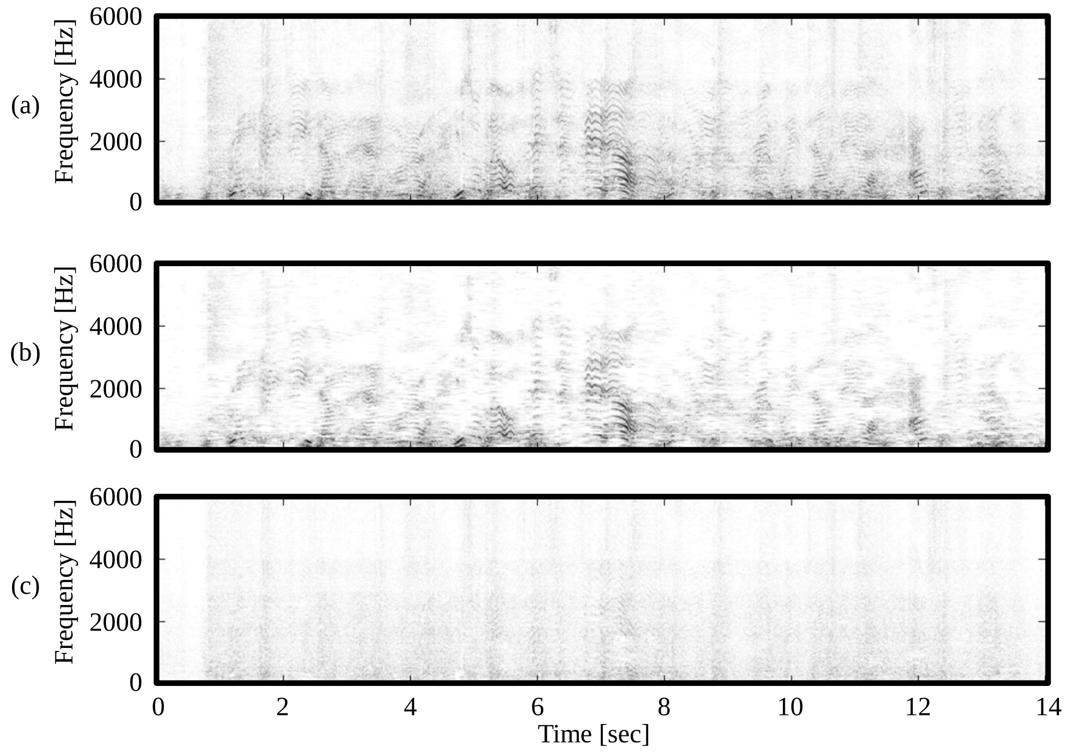


Figure 5.8: Sparse-low rank decomposition of the residual component of Bearlin. **(a):** Spectrogram of the residual component from the HPR decomposition. **(b):** Spectrogram of sparse mid-level component. **(c):** Spectrogram of low rank mid-level component.

5.2.4 Sparse-Low Rank Decomposition (SL)

This decomposition method, which has already been used in the context of singing voice extraction for example in [104, 109], is based on *robust principle component analysis* (RPCA) [15]. This technique splits a given real-valued matrix into the sum of two real-valued matrices, one being sparse and the other having a low rank, by solving an optimization problem. Here, *sparse* means that the sparse matrix should have a small ℓ_1-norm (the sum over all its entries should be minimal) and *low-rank* means that the low rank matrix should have a small *nuclear norm* (the sum of its singular values should be minimal). In [104], RPCA is applied to the magnitude spectrogram of a given recording. This results in a matrix representing the sparse structures in the given magnitude spectrogram, as well as a matrix representing its low-rank spectral structures. These two matrices are then used to derive binary masks for the original spectrogram, similar as in Equation (3.7). In our cascade, we apply this procedure to the spectrogram of the HPR decomposition's residual component. It shows that in this component, the contained formant and vibrato sounds tend to have a sparse spectral structure. The diffuse instrument sounds usually occur many times in a spectrally similar way and can be represented by a spectrogram having a low rank. Therefore, the sparse-low rank decomposition technique is well-suited to split the residual component of the first decomposition stage into a mid-level component that contains formant and vibrato sounds of the singing voice as well as a second mid-level component that holds diffuse instrument sounds.

In Figure 5.8 we again revert to our running example Bearlin. Figure 5.8a shows the spectrogram of the residual component from the HPR decomposition. The spectrograms of the components resulting from applying the sparse-low rank decomposition procedure to this residual component are shown in Figures 5.8b+c. One can see that, although the sparse mid-level component captures many structures that clearly belong to the singing voice (for example the vibrato structures between seconds six and eight), it also contains some noise-like regions of energy that stem from the accompaniment (for example directly at the beginning, stemming from a hit of the crash cymbal). Similarly, one can still see some vibrato structures in the the low rank component (again between seconds six and eight). However, the desired sounds are captured in the respective components to a large extend.

5.2.5 Reassembling of Singing Voice and Accompaniment

In a last step, the mid-level components resulting from our decomposition cascade are used to reassemble the singing voice as well as the accompaniment, see Figure 5.4. The estimate of the singing voice is computed by adding up the melody mid-level component from the melody-residual decomposition, the residual mid-level component from the transient-residual decomposition, and the sparse mid-level component from the sparse-low rank decomposition. Similarly, the estimate of the accompaniment is reassembled from the remaining mid-level components. Figures 5.9a+b show the spectrograms of the two estimates. When visually comparing these with the original track's spectrograms shown in Figures 5.9c+d, one can see that for our running example Bearlin the estimates come rather close to the original sources. In the next section, we evaluate the quality of our proposed decomposition cascade systematically.

5.3 Evaluation

We evaluate our proposed procedure in three ways. First, we compare the performance of our approach with state-of-the-art singing voice extraction methods on a standard dataset by means of objective source separation evaluation measures. Then, we discuss the results of a subjective listening experiment, which shows that the singing voice estimates of our procedure have a high perceptual quality. Finally, we also provide an accompanying website for this paper where one can find all audio files used in the objective and subjective evaluation as well as many further audio examples.

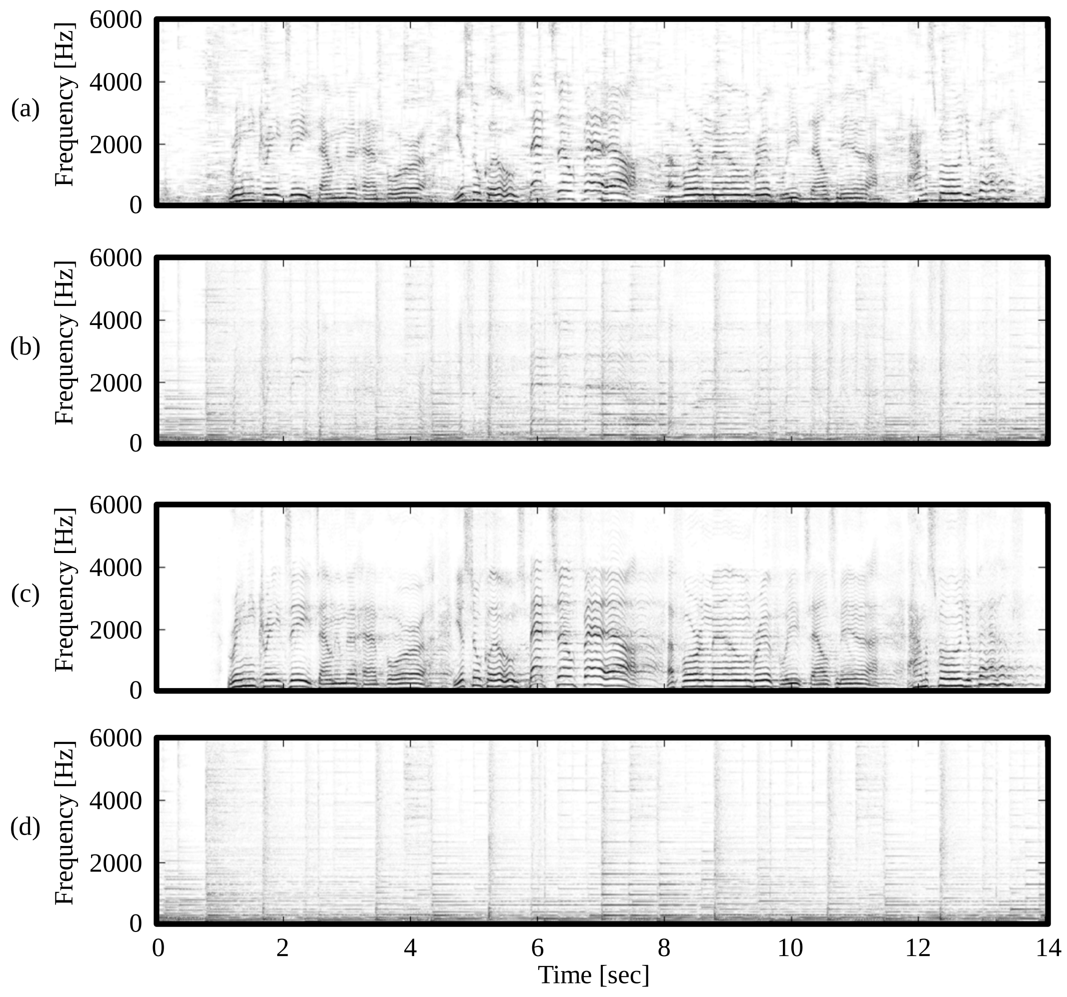


Figure 5.9: Reassembling of the mid-level components. **(a):** Spectrogram of the reassembled components that constitute the singing voice estimate. **(b):** Spectrogram of the reassembled components that constitute the estimate of the accompaniment. **(c):** Spectrogram of the original singing voice. **(d):** Spectrogram of the original accompaniment.

5.3.1 Objective Evaluation

To evaluate our proposed method in an objective way and to compare it to other procedures, we applied it to the well-known dataset of the *Signal Separation Evaluation Campaign* (SiSEC) [224, 4]. This dataset consists of five pop music multitrack recordings. For algorithms that participated in previous rounds of the campaign, separation results along with objective evaluation measures are available online at [224]. The reported evaluation measures were computed using the *Perceptual Evaluation methods for Audio Source Separation* toolkit (PEASS) [61] and consist of the Signal to Distortion Ratio (SDR), the source Image to Spatial distortion Ratio (ISR), the Signal to Interference Ratio (SIR), the Signal to Artifacts Ratio (SAR), the Overall Perceptual Score (OPS), the Target-related Perceptual Score (TPS), the Interference-related

	IBM*	CD-I	CD-B	SL	REPET*	VUIMM*
SDR [dB]	7.9	4.9	3.7	-0.9	4.1	5.6
ISR [dB]	14.9	7.8	6.1	7.2	7.7	7.9
SIR [dB]	11.4	4.6	2.7	-0.8	5.1	9.4
SAR [dB]	14.1	15.2	14.3	13.5	13.1	13.1
OPS	37.9	34.1	32.2	28.2	34.0	31.5
TPS	66.4	53.0	47.6	38.7	56.5	42.8
IPS	74.1	45.2	45.2	54.7	52.8	63.0
APS	30.3	51.0	48.1	37.8	49.2	37.8

Table 5.1: Average PEASS measures for singing voice estimates on the SiSEC dataset. Results marked with (*) were reported on the website [224]. Higher numbers indicate better results.

Perceptual Score (IPS), and the Artifacts-related Perceptual Score (APS). To examine the influence of the fundamental frequency track that is needed for the MR decomposition step (see Section 5.2.2), we applied our cascaded decomposition procedure to all recordings in the dataset twice: Once "informed" (CD-I) with a fundamental frequency track manually annotated using our *F0-Annotation Tool* presented in Appendix B.1, and once "blind" (CD-B) with a track automatically extracted using the MELODIA vamp plug-in [183]. Table 5.1 shows the computed evaluation measures for our singing voice estimates together with those of several state-of-the-art singing voice extraction algorithms. The measures for the oracle ideal binary mask (IBM), the REpeating Pattern Extraction Technique (REPET) [174], as well as the Voiced+Unvoiced Instantaneous Mixture Model technique (VUIMM) [52] were taken directly from [224]. The results for the Sparse-Low rank decomposition (SL) were computed by ourselves. REPET and SL are representatives of direct decomposition approaches, while VUIMM employs a disassemble and reassemble strategy. The first observation is that our proposed procedure yields evaluation measures in the same order of magnitude as REPET and VUIMM. The IBM can be seen as an upper limit for separation quality when using binary masking. The performance of SL which is also part of our decomposition cascade, falls slightly behind. This indicates that our proposed approach can actually improve on the separation performance of its individual decomposition procedures. Finally, we can observe that using a manually annotated fundamental frequency track goes along with slight improvements of the objective evaluation measures.

5.3.2 Listening Experiment

In order to analyze the subjective quality of the singing voice extracted by our procedure, we conducted a listening experiment. To be able to compare objective and subjective evaluation results, we considered the same procedures and the same dataset as in the objective evaluation.

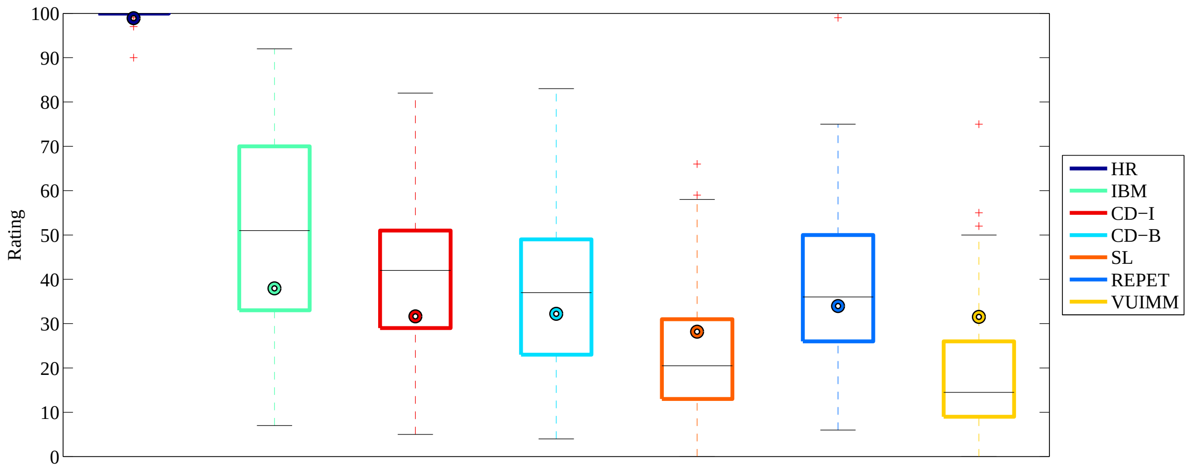


Figure 5.10: Average results of the performed listening experiment. Boxes indicate the interquartile range, black bars the median, and discs the objective OPS measures.

For each of the five recordings, the mixture of all sources as well as the separate singing voice, which constituted the *reference*, was given to the test participants. Their task was then to rate the *overall quality* of the different singing voice estimates with respect to the reference on a scale from 0 (poor) to 100 (excellent). The estimates were presented in a blind test along with a *hidden reference* (HR). Overall, 24 persons participated in the experiment from which six were excluded from the final evaluation during post screening since they were not able to detect the hidden reference reliably. The results averaged over the five recordings are visualized as boxplots in Figure 5.10 and Figure 5.11 shows the results for the five recordings separately. In each plot, one finds the respective OPS measure indicated by a colored disk. This objective measure, also having a value range from 0 to 100, is designed to predict the overall quality rating of the test participants. However, one can see from Figure 5.10 that IBM, CD-I, CD-B, and REPET tend to be underrated by the objective measure, while SL and VUIMM tend to be overrated. This indicates that the objective evaluation measures only vaguely correspond to human perception and that listening experiments are still necessary to obtain reliable measurements.

Looking at the detailed evaluation results in Figure 5.11, the subjective evaluation gives various insights. First, one can observe that VUIMM was rated rather low for all five recordings. Listening to the respective estimates reveals that here, although the accompaniment is usually suppressed well, the singing voice often has an unnatural, synthetic sound. This demonstrates that the reassembling of the singing voice from fine-grained components is a very difficult task. Also SL falls behind the remaining procedures. It shows that the direct decomposition of a recording into a sparse and a low rank component leaves the sparse singing voice estimate with a lot of musical noise which is reflected in the rating. In comparison to these two procedures, the singing voice estimates of CD-I, CD-B, and REPET are perceived to have a clearly better

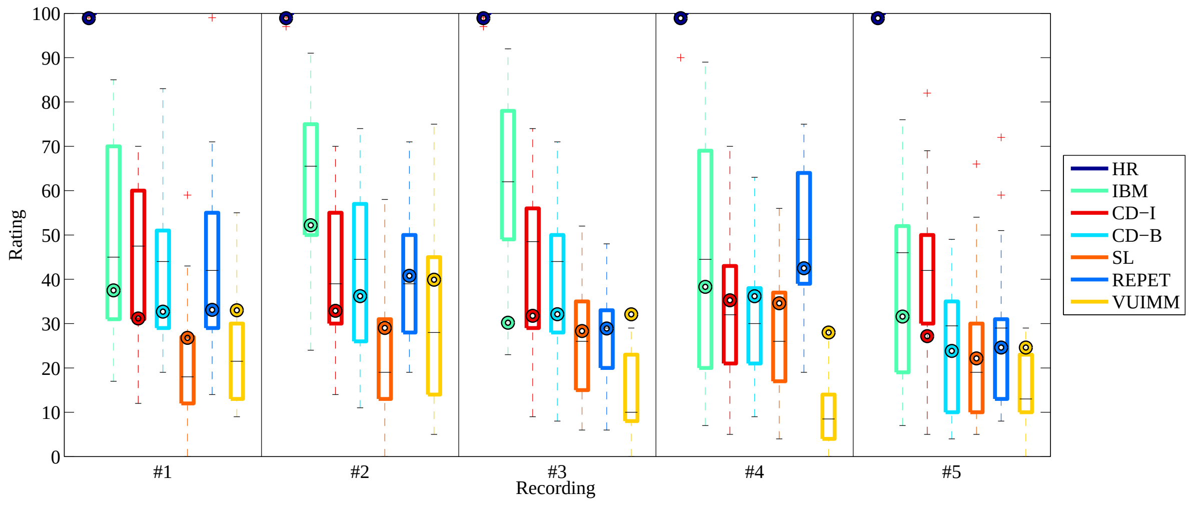


Figure 5.11: Results of the performed listening experiment. Boxes indicate the interquartile range, black bars the median, and discs the objective OPS measures. Recordings: Bearlin (#1), Tamy (#2), Another Dreamer (#3), Fort Minor (#4), Ultimate NZ Tour (#5).

quality for recordings #1 and #2. For recording #3, CD-I and CD-B demonstrate the benefit of not focusing on a single characteristic of singing voice. Here, REPET's assumption of repeating patterns in the accompaniment is not satisfied which leads to many accompaniment residues in the singing voice estimate and therefore to lower ratings. CD-I and CD-B do not rely on a specific musical structure and therefore receive good ratings for this recording as well. For recording #4, rapped lyrics over a looped beat, REPET excels all other approaches since its assumption of a repeating accompaniment is met perfectly. The decomposition cascade of CD-I and CD-B was however not optimized to capture the spoken "non-singing voice" in rap music. In particular the MR decomposition fails and yields meaningless decomposition results what explains the lower ratings. Looking at the results for recording #5, the first observation is that here even IBM receives rather low ratings. This indicates that the extraction of the singing voice from recording #5 can be considered difficult. However, while CD-I and CD-B were rated similarly for recordings #1 to #4, CD-I performs much better than all other procedures on this recording, even being close to IBM. It turns out that for this recording the blindly estimated fundamental frequency track has an octave error. Therefore, in the MR decomposition step only every second harmonic of the singing voice is extracted which leads to a thin sound of the singing voice estimate of CD-B. However, octave errors can be corrected easily by manual inspection. This example shows that it is possible to stabilize the performance of our procedure, even for difficult recordings, with very little user interaction.

5.3.3 Accompanying Website

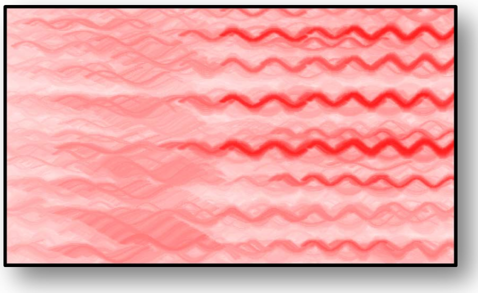
The objective and subjective evaluation showed that our proposed method yields good estimates of the singing voice for the pop music recordings of the SiSEC dataset. One of the advantages of this method lies in its flexibility in the sense that, contrary to approaches like REPET, it does not make strong assumptions about the accompanying music material. To demonstrate that our procedure works on a wide range of musical styles, including genres like classical opera music, romantic piano music with singing, or even metal, we prepared an accompanying website for this paper at [38]. On this website, one can find many illustrative audio examples of decomposition results of our procedure along with the results of the intermediate decomposition steps.

5.4 Conclusions and Further Notes

In this chapter, we have shown how different direct decomposition techniques can be cascaded to disassemble a given music recording into a set of semantically interpretable mid-level components. These components can be easily reassembled to yield estimates of the singing voice and the accompaniment in the recording. Objective and subjective evaluation on a standard dataset suggest that this approach yields singing voice estimates which are comparable to state-of-the-art methods. Furthermore, to demonstrate that our procedure works for a large variety of musical styles and genres going beyond the tested dataset, we also provide an accompanying website with additional audio material. In future work, one may investigate how further decomposition procedures (e.g. *center channel extraction* methods [214]), can be incorporated into our proposed cascade to further improve the quality of the extracted singing voice.

Chapter 6

Informed Music Signal Parametrization



In this chapter, we focus on estimating musically meaningful parameters from music recordings. First, in Section 6.1, we introduce a framework that allows for parameterizing a music recording's time-frequency representation according to the note events specified in the piece's musical score. This parametrization can then be used in a source separation scenario where the individual sound sources correspond to the note events. The resulting note-wise audio events constitute elementary building blocks of the recording that allow for score-based music editing. Additionally, the source separation result can give insights into the quality of the parametrization. This section is based on our original contribution in [37].

In Section 6.2, we then focus on vibrato, a musical effect that is commonly used by musicians to make their performance more expressive. Vibrato, which is the periodic oscillation in a musical voice's pitch, can be parameterized by its *rate* (the modulation frequency given in Hertz)

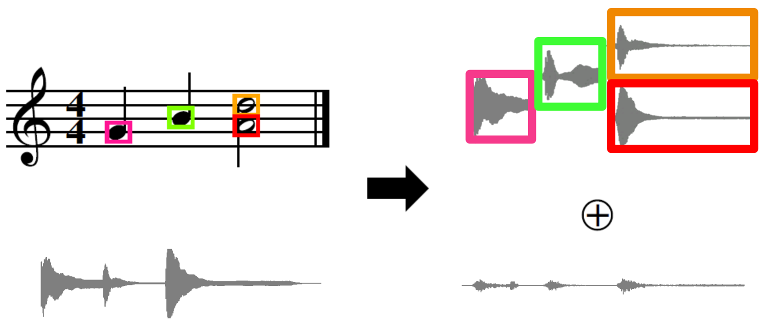


Figure 6.1: Score-informed decomposition of a given music signal into note-wise audio events and a residual signal.

and its *extent* (the modulation's amplitude given in cents). In a music recording's spectrogram representation, vibrato is evident in the form of characteristic wave-like spectro-temporal patterns. By locally comparing a given music recording's spectrogram with a set of predefined vibrato templates that resemble the spectro-temporal vibrato patterns, we aim to detect and parameterize vibrato in the recording. This section is based on our work from [35].

6.1 Score-Informed Parametrization of Note Events

In recent years, the task of separating a mixture of superimposed sound sources into its constituent components has been a central research topic in the field of digital signal processing. In speech, for example, these components could be the individual conversations of simultaneously speaking persons ("Cocktail party scenario", see [18]). In the context of music, the sources might correspond to the main melody, a bassline, a drum track or another instrument track [75, 80, 149, 181]. To guide the source separation process in such a scenario, it has become a common strategy to provide the algorithm with additional information. Such information can, for example, be given in the form of user-specified annotations [53, 130, 198], or by a musical score. In *score-informed* procedures the explicit timing, pitch and instrument information encoded by the score is utilized to guide and support the source separation processes.

Most current score-informed approaches are designed for extracting individual instruments as specified by the score, see [94, 111, 227]. In this section, we go beyond this scenario by introducing a framework for decomposing a music recording into elementary building blocks or sound events. More precisely, a musical score can be considered as a composition of elementary events given by the individual musical notes. These notes have some explicit musical meaning (in terms of pitch, onset time, and duration) and are directly intelligible by a human. The core idea of this contribution is to decompose a given music recording x into a set of I note-wise audio events x_i, $i \in [0 : I - 1]$, where each audio event is directly associated with a note in the musical score,

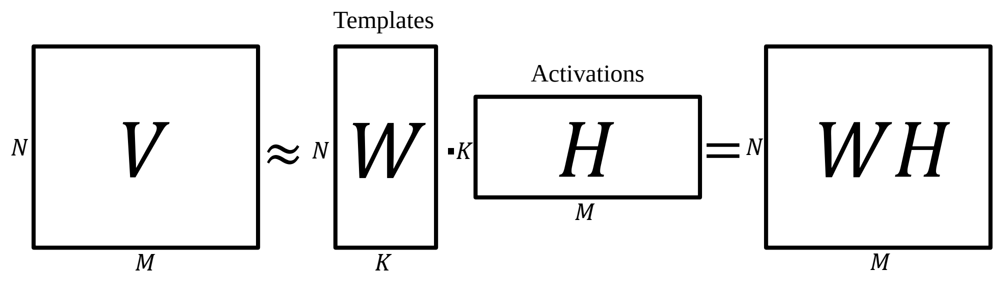


Figure 6.2: Basic concept of non-negative matrix factorization.

see Figure 6.1. Based on this decomposition, we introduce an intuitive interface that allows a user to directly access the audio recording in a note-wise fashion, which opens up explicit ways of editing and manipulating audio material. Such an interface also provides novel possibilities to better understand the quality achieved by the underlying source separation algorithm. For example, subtracting all note-wise audio events from the original recording yields a *residual signal* x_{res} which can be interpreted as the part of the recording that was not captured by the source separation process (for example because it was not reflected by the given musical score):

$$x = \left(\sum_{i=0}^{I-1} x_i \right) + x_{\mathrm{res}} \ . \tag{6.1}$$

Analyzing this residual can then reveal parts in the original recording where the source separation algorithm typically fails or where data inconsistencies occur.

The remainder of this section is structured as follows. In Section 6.1.1 we summarize a recent score-informed source separation approach used in our experiments that is based on *non-negative matrix factorization* (NMF). Next, in Section 6.1.2, we show how to derive the note-wise decomposition of the audio recording and discuss some manipulation strategies. Finally, we present our prototype user interface for intuitive score-based audio editing (Section 6.1.3) and source separation analysis (Section 6.1.4).

6.1.1 Constrained NMF-based Source Separation

In the last years, techniques based on non-negative matrix factorization (NMF) have been applied to decompose a music signal's magnitude spectrogram into a set of template (column) vectors and activation (row) vectors [197]. To better control this factorization, additional score information has been used to constrain NMF and to yield a musically more meaningful decomposition [93]. In this section, we summarize the score-informed procedure as introduced in [63].

Given a matrix $V \in \mathbb{R}_{\geq 0}^{N \times M}$,$N, M \in \mathbb{N}$, the goal of classical NMF is to derive two matrices $W \in \mathbb{R}_{\geq 0}^{N \times K}$ and $H \in \mathbb{R}_{\geq 0}^{K \times M}$, $K \in \mathbb{N}$ such that some distance measure between V and WH

is minimized [129, 146], see Figure 6.2. The matrices W and H are typically computed by iteratively applying multiplicative update rules to initializations $W^{(0)}$ and $H^{(0)}$:

$$W_{nk}^{(\ell+1)} = W_{nk}^{(\ell)} \frac{\sum_m H_{km}^{(\ell)} V_{nm}/(W^{(\ell)}H^{(\ell)})_{nm}}{\sum_m H_{km}^{(\ell)}} \,, \tag{6.2}$$

$$H_{km}^{(\ell+1)} = H_{km}^{(\ell)} \frac{\sum_n W_{nk}^{(\ell)} V_{nm}/(W^{(\ell)}H^{(\ell)})_{nm}}{\sum_n W_{nk}} \,, \tag{6.3}$$

$n \in [0 : N-1]$, $m \in [0 : M-1]$, $k \in [0 : K-1]$, $\ell \in \mathbb{N}$. In [129], the update rules (6.2) and (6.3) were shown to be non-increasing under the Kullback-Leibler divergence

$$(V||WH) = \sum_{nm} V_{nm} \log \frac{V_{nm}}{(WH)_{nm}} - V_{nm} + (WH)_{nm} \,. \tag{6.4}$$

The update process is typically terminated after a fixed number of iterations $L \in \mathbb{N}$ or when the distance measure falls below a given threshold. For a more detailed explanation and an exemplary derivation of update rules we refer to [146, Section 8.3].

In the context of source separation, given an audio recording x with spectrogram X, the goal is to factor the magnitude spectrogram[1] $V = |X|$ into a matrix W of *template vectors* (every column corresponding to the prototype spectrum of a certain tone) and a matrix H of *activations* (every row encoding when and how loud a corresponding tone is played). In standard NMF the matrices W and H are typically randomly initialized before applying the update rules (6.2) and (6.3). However, the resulting factors are often not musically meaningful in this case as discussed for example in [63].

To overcome this issue, [63] proposed a score-informed approach, where the score information is used to initialize both matrices W and H to guide the NMF update process in a musically meaningful direction. More precisely, having the score of the audio recording at hand in form of a MIDI file, high-resolution synchronization techniques are used to temporally align the MIDI events with the audio recording [64, 114]. Each of the I note events of the synchronized MIDI file yields information about the pitch, the onset time and the duration of a corresponding audio event that should occur in the recording according to the score. For each occurring pitch in the MIDI file, a harmonic template (column of the matrix W), which encodes the rough harmonic structure of the pitch, is initialized. This template is defined to have non-zero entries at frequency bins that are related to the fundamental frequency and the overtones of the given pitch and zero entries otherwise. Similarly, the activation matrix H is initialized by the specified onset times and note durations obtained from the synchronization procedure. For later use, we link the initialization of H with the corresponding synchronized MIDI events as follows. A

[1]Note that here we model the magnitude spectrogram V to be a matrix rather than a function with two arguments as in Section 2.2.

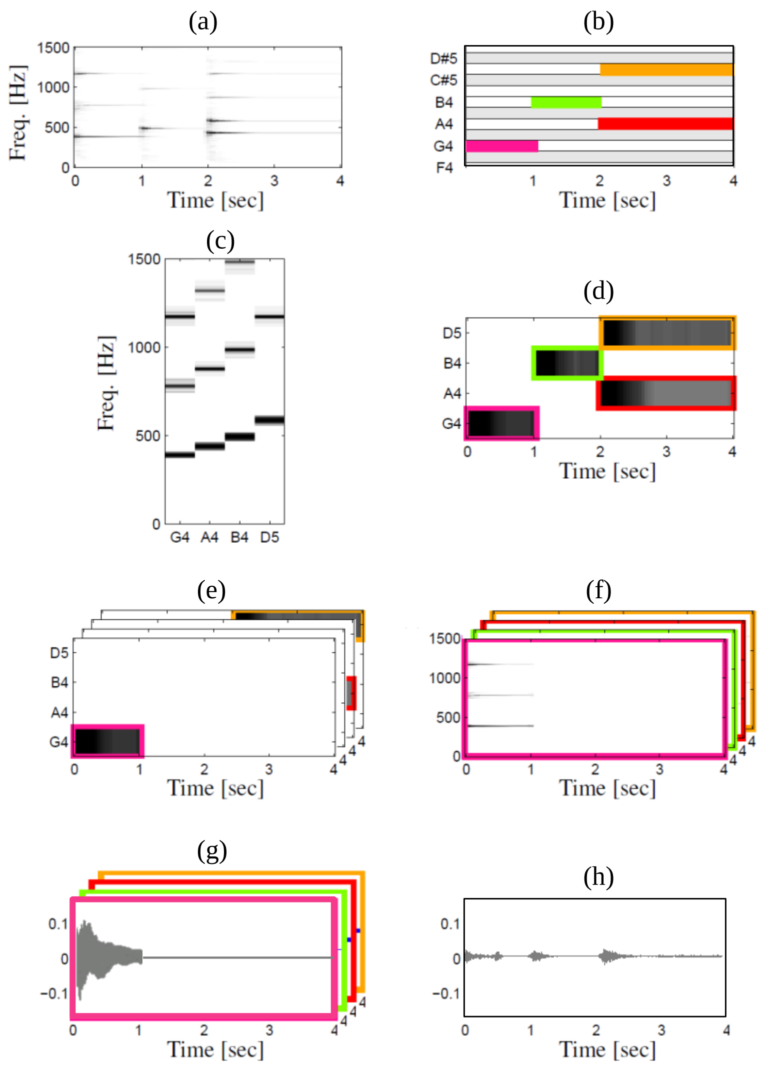


Figure 6.3: **(a)** Magnitude spectrogram V. **(b)** Synchronized MIDI note events. **(c)** Template matrix W learned by NMF. **(d)** Activation matrix H learned by NMF. **(e)** Note-wise activation matrices H_m. **(f)** Note-wise spectrograms X_m. **(g)** Note-wise audio events x_m. **(h)** Residual signal r.

binary constraint matrix $C_i \in \mathbb{R}^{K \times M}$ is constructed for each $i \in [0 : I-1]$, where C_i is one at entries that correspond to the pitch and temporal position of the i^{th} MIDI event and zero otherwise. Each C_i therefore constitutes a link between specific entries in H and a MIDI note event. The union (OR-sum) of all C_i is then used as initialization of H. At this point, the crucial observation is that the multiplicative NMF update rules can only change the non-zero entries. Therefore, applying NMF to the initialized matrices W and H yields a decomposition, where the relations expressed by the C_i between MIDI note events and entries in the activation matrix H are preserved, see Figure 6.3a-d. The result after the NMF-learning procedure can be seen as a refinement of the initially constrained harmonic template and activation matrices.

6.1.2 Note-Based Audio Decomposition

Let W and H denote the template and activation matrices after applying the NMF learning procedure. We now use the note-wise constraints given by the matrices C_i to derive the note-wise audio events x_i. To this end, we first compute a note-wise activation matrix

$$H_i = H \odot C_i \ , \tag{6.5}$$

where the operator $\odot$ denotes point-wise multiplication. Afterwards, we derive a spectral mask

$$M_i = (WH_i) \oslash (\sum_{i=0}^{I-1} WH_i + \epsilon) \ , \tag{6.6}$$

where $\oslash$ is understood as point-wise division and ϵ is a small positive constant to avoid a potential division by zero. The mask M_i can be interpreted as a weighting matrix that reflects the contribution of the i^{th} note event to the original spectrogram X. Finally, we compute the note-wise spectrogram

$$X_i = X \odot M_i \ , \tag{6.7}$$

and apply the inverse short-time Fourier transform to obtain the audio event x_i, see Figure 6.3e-h. The audio events x_i represent a decomposition of the original signal x (the music recording) according to the note events specified by the given musical score. Obviously, this decomposition becomes problematic in the case that the recorded performance deviates from the musical score. More generally, synchronization inaccuracies, i. e. deviations in the alignment of the MIDI events and their expected realization in the music recording, may lead to local errors in the decomposition. Furthermore, simplifying model assumptions (such as the assumption that the partials' relative energy distribution is independent of the loudness), deviations in the expected tuning, or additional sound components caused by resonance or reverberation may cause artifacts in the decomposition. Therefore, we also compute a residual signal

$$x_{\text{res}} = x - \sum_{i=0}^{I-1} x_i \ . \tag{6.8}$$

The signal x_{res} holds a lot of valuable information since it does not only give a deeper insight into the source separation process, but it may also reveal inconsistencies between the musical score and the audio recording. Therefore, it is a natural idea to analyze x_{res} in more detail. In the next section we present an interface that supports such an analysis by enabling the user to study the decomposition and the residual signal more thoroughly.

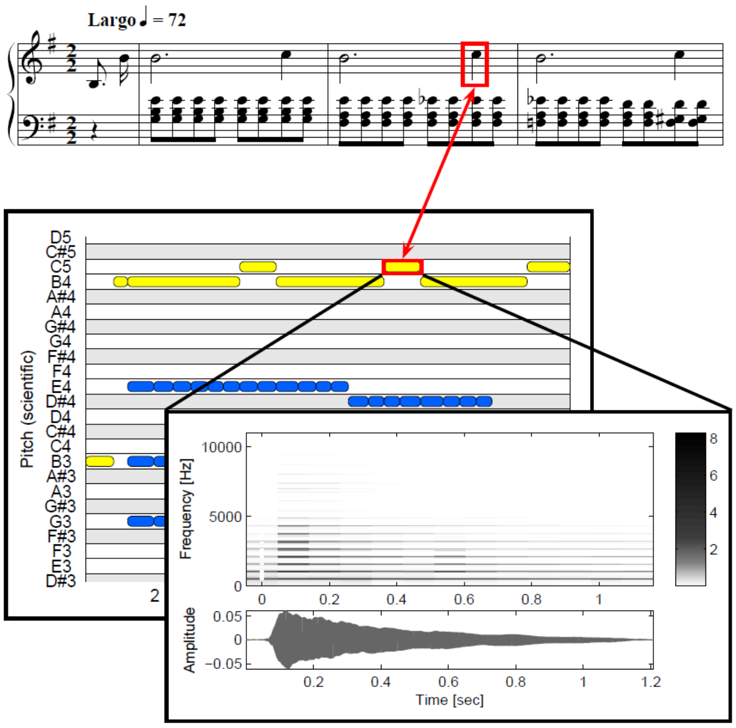


Figure 6.4: Top: Score of the first three measures of Op. 28 No. 4 by Frédéric Chopin. **Bottom:** Our user interface showing the corresponding part in an audio recording of the piece. Each note-wise audio event can be accessed separately.

6.1.3 Audio Editing

The decomposition of an audio recording into note-wise audio events can be utilized as a basis for various applications. To this end, we developed a user interface (see Appendix B.2) which offers a user-friendly access to such a decomposition (a demo of the proposed interface can be found at [36]). This interface provides easy-to-use possibilities for manipulating the audio recording in a musically informed manner. For example dragging a MIDI event in the piano roll representation shown in Figure 6.4 and dropping it at a different position is an intuitive way of changing the onset time (horizontal displacement) and the pitch (vertical displacement) of a note. Having the note-wise audio decomposition at hand, we are able to transfer the same manipulations to the audio recording. If the user wishes to displace a note with respect to time and pitch, the corresponding audio event is first subtracted from the original recording and a suitably time- and pitch-shifted version (using a standard pitch shifting procedure as discussed in Section 4.4.2) of the event is added afterwards. We keep track of all the applied manipulations such that it is possible to manipulate a previously edited note again. By using similar strategies it is also possible to change the duration or the volume of individual notes, to remove notes completely from the audio recording, or to add additional notes by copying and manipulating existing ones.

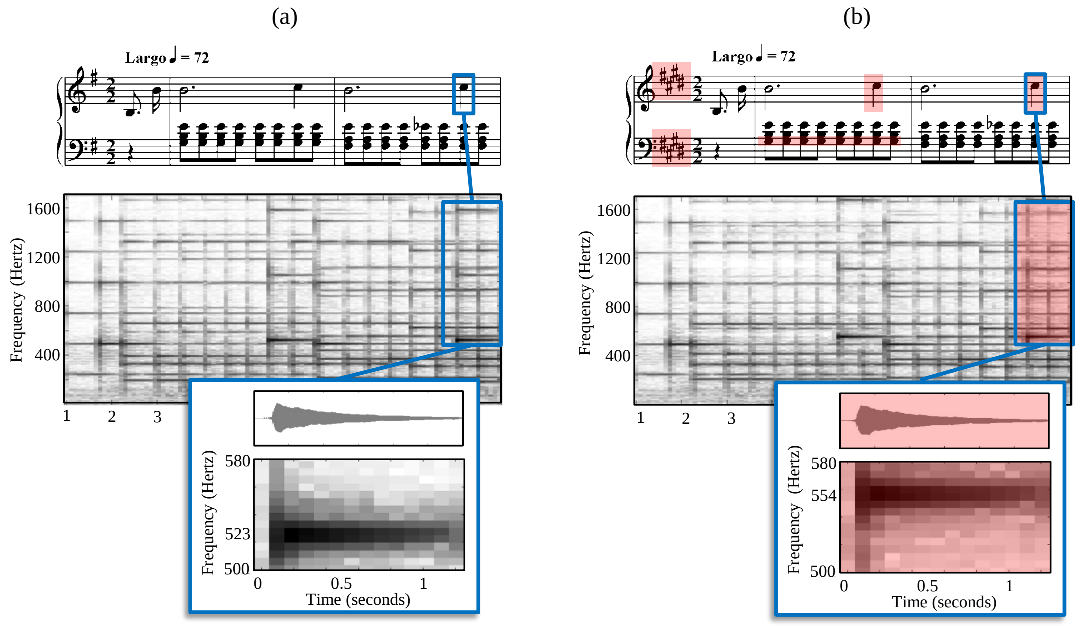


Figure 6.5: Note-wise audio editing. This figure is inspired by [65]. **(a):** Score of measures 1 and 2 of Op. 28 No. 4 by Frédéric Chopin. The spectrogram of a recording as well as a single note event are shown on the bottom. **(b):** The edited recording. The marked notes were modified to change the recording's key. Modifications made in the score as well as the resulting changes in the spectrogram and the audio event are highlighted in red.

Figure 6.5 illustrates the process of a musically meaningful modification that was done using our interface. Figure 6.5a shows the score of the first two measures of Chopin's Op. 28 No. 4 in E minor for piano. Below the score, we see the spectrogram of a recording of that piece. Applying our proposed note-wise audio decomposition procedure allows us to manipulate the individual notes. This makes it very easy to change the key of the piece from E minor to E major by pitch shifting the note events associated with the notes marked in red in Figure 6.5b upwards by one semitone. The resulting recording sounds as if the pianist played the piece not in the minor, but in the major key.

6.1.4 Source Separation Analysis

Analyzing the decomposition of a music recording, and especially the residual signal x_{res} offers novel possibilities to investigate the behavior of the underlying source separation algorithm. Positions in the original audio recording where x_{res} shows high energy indicate passages where the source separation procedure could not assign all of the recording's energy to the note-wise audio events. To analyze such positions, our interface has been equipped with a plugin that plots the color-coded short-time energy of x_{res} in the background of the standard visualization, see Figure 6.6. This way, one can directly observe temporal relations between bursts of energy in x_{res} and the synchronized MIDI note events.

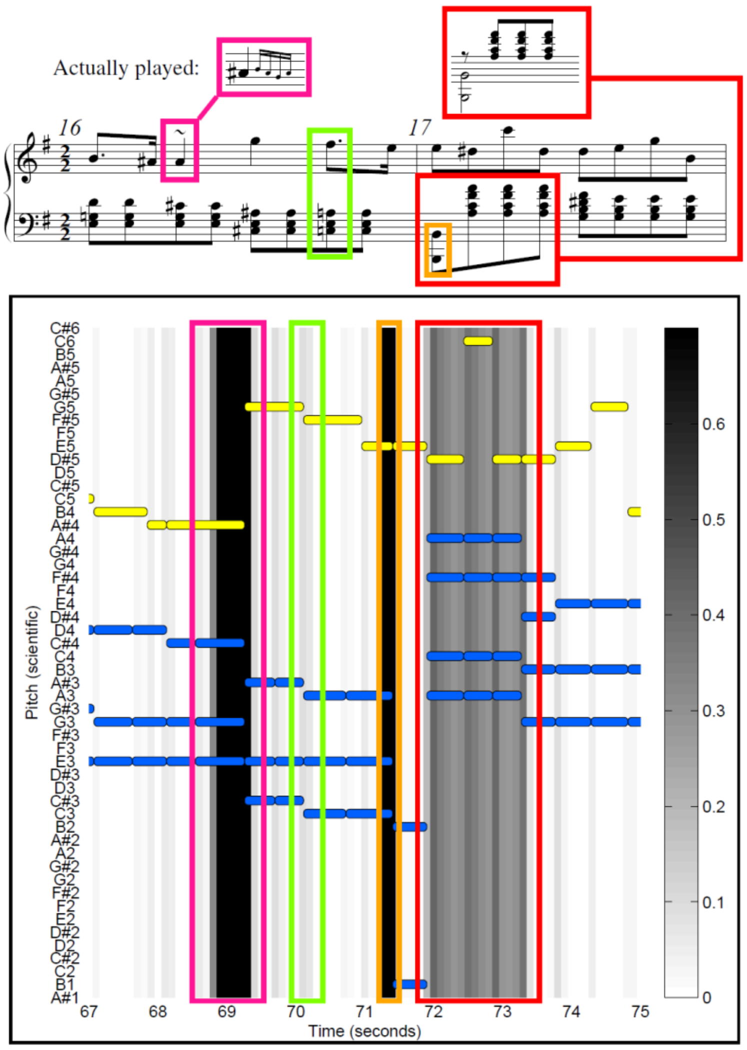


Figure 6.6: Top: Score of measures 16 and 17 of Op. 28 No. 4 by Frédéric Chopin. **Bottom:** Our user interface showing the corresponding part in an audio recording of the piece. The short-time energy of the residual signal r is visualized in a color-coded format in the background.

As an illustrative example how this tool can be used, we consider a short excerpt of Chopin's Prelude Op. 28 No. 4 as shown in Figure 6.6. A musical score does often not completely describe what is actually played by the performing musician. An example for this are ornamental notes which are not reflected directly by the score (see, e.g., the pink boxes in Figure 6.6). Such deviations typically lead to local misalignments between the MIDI events and the audio recording. Even worse, additionally played notes that are not contained in the MIDI file may neither have an appropriate template vector in W, nor entries in the activation matrix H. It is therefore impossible for the score-informed source separation procedure to properly capture these notes. As Figure 6.6 shows, the residual x_{res} can reveal such inconsistencies between the notated score and the performance.

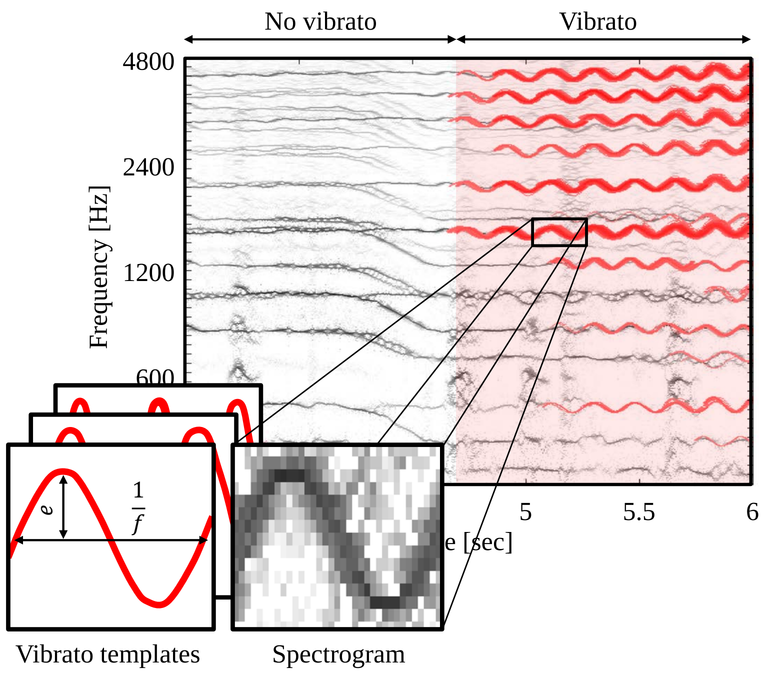


Figure 6.7: Template-based vibrato analysis. A matching vibrato template lets us infer the rate f and extent e of vibrato present in the music signal.

Local energy peaks in the residual are commonly aligned with note onsets (see the green and orange boxes in Figure 6.6). While smaller peaks, like shown in the lower green box, commonly emerge from oversimplifications in the musical model of the source separation algorithm (the derived template vectors in W can often not describe the sound of an onset accurately), more massive bursts of energy often arise from slightly misaligned MIDI events (e.g., the two bass notes marked in orange are played slightly earlier than they are encoded in our synchronized MIDI file).

Another aspect that can not be captured appropriately by the used source-separation procedure are acoustical phenomena like resonance or reverberation (see the red boxes in Figure 6.6). At the beginning of measure 17, the performing musician holds the pedal of the piano and all played notes are therefore sustained until the pedal is released again in the middle of the same measure. Furthermore, the pressed pedal allows all strings of the piano to resonate with the actually played notes, thus creating an even more complex sound mixture. This information is not reflected in the audio decomposition and a large amount of energy migrates to the residual in the source separation process.

6.2 Template-Informed Parametrization of Vibrato

The human voice and other instruments often reveal characteristic spectro-temporal patterns that are the result of specific articulation techniques. For example, vibrato is a musical effect that is frequently used by musicians to make their performance more expressive. Although a clear definition of vibrato does not exist [204], it can broadly be described as a musical voice's "periodic oscillation in pitch" [191]. It is commonly parameterized by its *rate* (the modulation frequency given in Hertz) and its *extent* (the modulation's amplitude given in cents[2]). These parameters have been studied extensively from musicological and psychological perspectives, often in a cumbersome process of manually annotating spectral representations of monophonic music signals, see for example [204, 195, 101, 169, 210].

To approach the topic from a computational perspective, the signal processing community has put considerable research efforts into developing automated vibrato analysis methods for monophonic, as well as for more complex music signals with multiple sound sources. While some applications implicitly exploit spectro-temporal characteristics of vibrato to approach higher-level tasks such as harmonic-percussive decomposition [163], singing voice detection [132], or singing voice separation [207], there also exist methods for explicitly detecting and parameterizing vibrato components in a given music signal. A common approach is to perform the vibrato analysis in two consecutive steps. In the first step, a fundamental frequency trajectory (F0-trajectory) is estimated for the musical voice that is most likely to exhibit vibrato. This trajectory is then analyzed in the second step to detect and parameterize periodic modulation patterns, see for example [96, 179, 161, 176, 183, 228]. However, computing F0-trajectories for complex signals with multiple instruments is a highly non-trivial and error-prone task by itself [184]. Therefore, a trajectory estimated in the first step may not appropriately reflect the relevant modulation patterns. This in turn renders the vibrato detection and parametrization in the second step problematic, if not impossible.

To avoid the error-prone F0-estimation step, in this work we propose a novel approach for automatically analyzing vibrato components in complex music signals. Our core idea is to detect spectro-temporal vibrato patterns directly in a music signal's spectrogram by locally comparing this representation with a set of predefined vibrato templates[3] that reflect different vibrato rates and extents. The measured similarity yields a novel mid-level feature representation—a *vibrato salience spectrogram*—in which spectro-temporal vibrato patterns are enhanced while other structures are suppressed. Figure 6.7 illustrates this idea, showing three different vibrato templates as well as a spectrogram representation of a choir with a lead singer who starts to sing

[2]A *cent* is a logarithmic frequency unit. A musical semitone is subdivided into 100 cents.

[3]Note that this approach is conceptually similar to the Hough transform [78], a mathematical tool known from image processing for the detection of parameterized shapes in binary images. However, the Hough transform is known to be very sensitive to noise and therefore not suitable for detecting vibrato patterns in spectrograms that are commonly rather noisy.

with strong vibrato in the excerpt's second half. Time-frequency bins where one of the templates is locally similar to the spectrogram, thus yielding a high vibrato salience, are indicated in red. As we can see, these time-frequency bins temporally coincide with the annotated vibrato passage at the top of Figure 6.7. Additionally, a high vibrato salience does not only indicate the presence of vibrato in the music signal, but also reveals the vibrato's rate and extent encoded in the similarity maximizing template.

The remainder of this section is structured as follows. First, we describe our proposed template-based vibrato analysis approach in detail. We discuss relevant spectrogram representations (Section 6.2.1) and describe how the vibrato templates are modeled (Section 6.2.2). Both our choice of spectrogram representation and the vibrato template's design are motivated by the correlation-like similarity measure that we use to locally compare the templates with the spectrogram. We then introduce the derivation of the vibrato salience spectrogram (Section 6.2.3) and comment on our approach's computational complexity (Section 6.2.4). Afterwards, we present our experimental results. In Section 6.2.5, we quantitatively evaluate our proposed approach in the context of a vibrato detection task. Then, in Section 6.2.6 we demonstrate the method's potential for automatically analyzing vibrato rate and extent. Finally, in Section 6.2.7, we indicate open challenges and potential solutions.

6.2.1 Spectral Representation

Given a discrete music signal x, we first compute the short-time Fourier transform X of x as described in Section 2.2. Note that w.l.o.g., we assume that $X : \mathbb{Z} \times \mathbb{Z} \to \mathbb{C}$ (instead of $X : \mathbb{Z} \times [0 : N-1] \to \mathbb{C}$) at this point. Figure 6.8a shows an excerpt of our example signal's magnitude spectrogram $|X|$ where one can clearly see wave-like vibrato patterns in the lead singer's fundamental frequency and its overtones. However, due to the STFT's linear frequency sampling, the vibrato patterns' amplitudes increase with higher overtones.

In the context of our template-based analysis it is desirable that vibrato patterns stemming from the same frequency modulated tone have the same amplitude that directly reflects the vibrato's extent. We therefore compute a log-frequency spectrogram from the STFT X, using a phase vocoder-based reassignment approach as discussed in Section 2.4. In this representation, which can be seen in Figure 6.8b, frequency bands are spaced logarithmically and have a constant logarithmic bandwidth specified in cents. This ensures the desired property in this spectrogram representation.

In a last step, we normalize the spectrogram in order to achieve two goals. First, we aim to make the representation independent of the signal's volume such that we can also detect vibrato in quiet signal passages. Second, when locally comparing our vibrato templates with the representation, the resulting similarity measure should yield values in a fixed range. A method

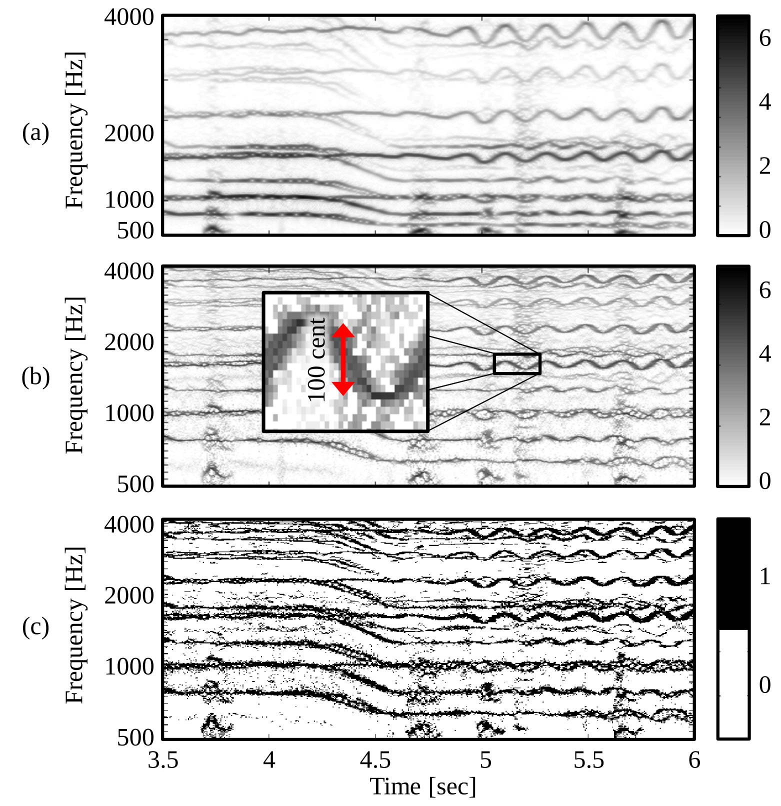


Figure 6.8: Spectrogram representations of the input signal x. **(a):** Magnitude spectrogram. **(b):** Log-frequency spectrogram. **(c):** Binarized log-frequency spectrogram Y.

that showed to be simple and effective to achieve both goals is spectrogram binarization, where we set the ten percent highest values of each frame in the log-frequency spectrogram to one and all remaining values to zero. This yields a *binarized log-frequency spectrogram* $Y : \mathbb{Z} \times \mathbb{Z} \to \{0, 1\}$, see Figure 6.8c. In our experiments, we choose parameters such that Y has a time resolution of roughly 150 frames per second and a frequency resolution of ten bands per semitone.

6.2.2 Vibrato Templates

Next, we introduce a set $\mathcal{T}$ of templates that reflect spectro-temporal vibrato patterns as expected in Y. Let us model such a template $T \in \mathcal{T}$ for vibrato having a rate of f Hertz, an extent of e cents, and a duration of at least ℓ seconds. When locally comparing the template T with Y, one should obtain high similarity values when T is aligned with a matching spectro-temporal vibrato pattern in Y and low values otherwise. The idea is therefore to have a positive portion in T that reflects the spectro-temporal vibrato pattern as well as a negative portion that prevents the template from correlating well with regions in Y that are homogeneously equal to one.

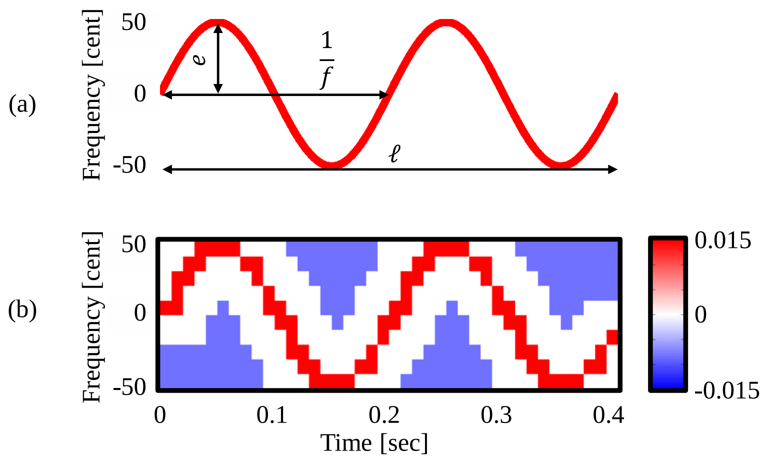


Figure 6.9: Generation of a vibrato template T with a vibrato rate $f = 5$ Hertz, extent $e = 50$ cent, and a duration of $\ell = 0.4$ seconds. **(a):** Sinusoidal vibrato trajectory s. **(b):** Vibrato template T.

Assuming a sinusoidal vibrato, we can describe the vibrato's trajectory (up to phase) by

$$s(t) = e\ \sin(2\pi f t)\ , \tag{6.9}$$

$t \in [0, \ell]$. Figure 6.9a shows such a trajectory for $f = 5$ Hertz, $e = 50$ cent, and $\ell = 0.4$ seconds. The trajectory is then discretized such that its time- and frequency resolution matches the binarized log-frequency spectrogram. Time-frequency bins that are close to s are assigned with positive values, while bins having a certain distance from s get negative values. To allow for some tolerance of the width of vibrato patterns in Y, the remaining time-frequency bins are defined to be zero. Finally, positive and negative entries in T are normalized to sum up to one and minus one, respectively, see Figure 6.9b.

6.2.3 Vibrato Salience

In order to locate and parameterize vibrato structures in the binarized log-frequency spectrogram Y, we aim to compute a *vibrato salience spectrogram* S—a kind of mid-level feature representation—in which vibrato structures are enhanced while other kinds of structures are suppressed. To this end, we define the vibrato salience spectrogram S_T for a single vibrato template $T : [0 : A-1] \times [0 : B-1] \to \mathbb{R}$, $A, B \in \mathbb{N}$. The computation process is illustrated in Figure 6.10a. Let $\mathcal{I}$ be the set of all index pairs $(a, b) \in [0 : A-1] \times [0 : B-1]$ such that $T(a, b)$ is positive (the indices of all red entries in Figure 6.9b). Furthermore, let

$$Y^{(\mu,\kappa)}(m, k) = Y(m-\mu, k-\kappa)\ , \tag{6.10}$$

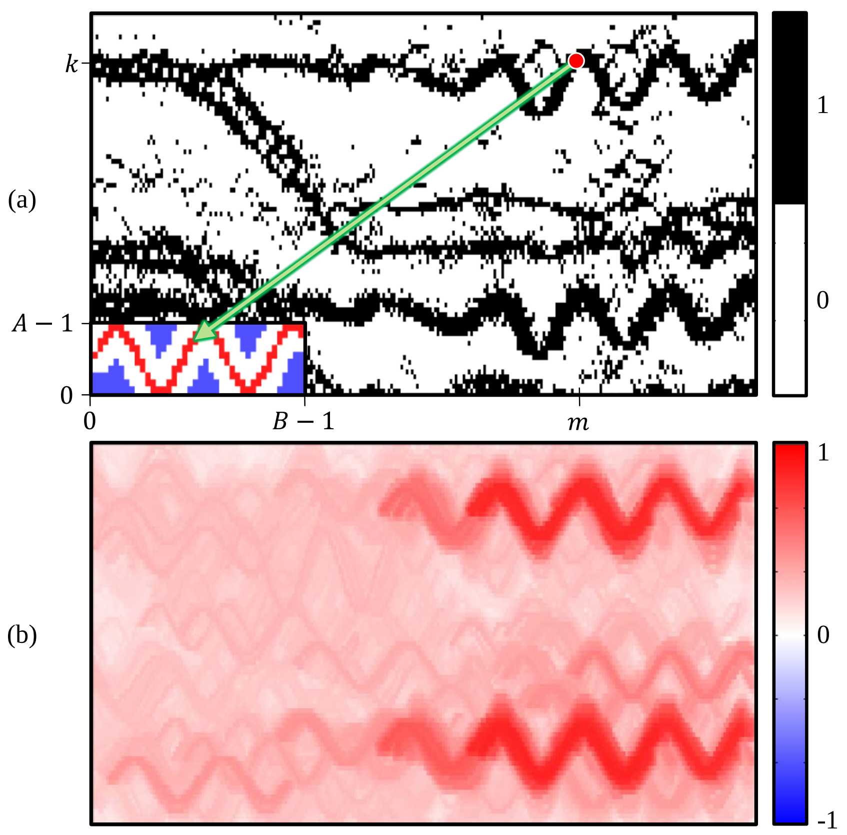


Figure 6.10: Vibrato salience spectrogram computation. **(a):** Process to compute S_T. The similarity-maximizing shift (μ, κ) that maps (m, k) onto an index pair in $\mathcal{I}$ is indicated by a green arrow. **(b):** Vibrato salience spectrogram S.

$\mu, \kappa \in \mathbb{Z}$, denote a version of Y that is shifted by μ and κ indices in time- and frequency direction, respectively. Intuitively, the vibrato salience $S_T(m, k)$ should be high if $Y(m, k)$ is part of a spectro-temporal vibrato pattern as reflected by T. To this end, we verify if there is a shift (μ, κ) that aligns $Y(m, k)$ (red dot in Figure 6.10a) with one of the positive entries in the vibrato template T such that T and $Y^{(\mu,\kappa)}$ are similar (the optimal shift for our example in Figure 6.10a is indicated by a green arrow). To compute $S_T(m, k)$, we therefore maximize the correlation-like similarity measure

$$c(T, Y) = \sum_{a=0}^{A-1} \sum_{b=0}^{B-1} T(a, b) Y(a, b) \tag{6.11}$$

over all shifts (μ, κ) that map (m, k) onto one of the index pairs in $\mathcal{I}$:

$$S_T(m, k) = \max_{\{(\mu,\kappa):(m,k)-(\mu,\kappa)\in\mathcal{I}\}} c(T, Y^{(\mu,\kappa)}) \,. \tag{6.12}$$

The full vibrato salience spectrogram can then be computed by maximizing over all vibrato templates $T \in \mathcal{T}$:

$$S(m,k) = \max_{T \in \mathcal{T}} S_T(m,k)\ . \tag{6.13}$$

Figure 6.10b shows the vibrato salience spectrogram S resulting from the binarized log-frequency spectrogram Y shown in Figure 6.10a. Note that by the vibrato template's design and $Y(m,k) \in \{0,1\}$, one obtains $S(m,k) \in [-1,1]$ for all $m,k \in \mathbb{Z}$. While the vibrato structures present in Y are also clearly visible in S, the horizontal structures as well as the glissando at the excerpt's beginning do not correlate well with the vibrato templates. They are therefore, as intended, suppressed in S.

6.2.4 Computational Complexity

The vibrato salience spectrogram's derivation as defined in the previous section is a computationally expensive process. When implemented naively, it is necessary to use a quadruply nested loop to iterate over all combinations of time-frequency bins (m,k) in Y, vibrato templates $T \in \mathcal{T}$, index shifts $\{(\mu,\kappa) : (m,k) - (\mu,\kappa) \in \mathcal{I}\}$, and index pairs (a,b) in T. However, note that many computations are redundant and that it is therefore possible to optimize the calculation process, for example by exploiting two-dimensional convolutions. Furthermore, one can speed up the derivation by considering only a limited frequency range in Y as well as by applying further heuristics such as only taking into account vibrato salience values above a threshold $\tau \in [-1,1]$. Although still being computationally demanding, the derivation therefore becomes feasible enough to be used in practice. For example, deriving S for a music signal with a duration of 60 seconds takes our MATLAB implementation roughly 40 seconds on a standard computer.

6.2.5 Evaluation: Vibrato Detection

In a first experiment, we considered the task of temporally identifying vibrato passages in a music signal. We therefore compiled a dataset of nine items (see Table 6.1), which are excerpts of music signals from the "Mixing Secrets" multitrack dataset [193]. Each item consists of a monophonic vocal signal x_{voc} and a polyphonic accompaniment signal x_{acc}. Annotations of vibrato passages in the vocal signals were created manually to serve as ground truth for the subsequent evaluation (none of the accompaniment signals x_{acc} has vibrato). To vary the difficulty of the vibrato detection task, we created three different mixes for each of the items—one were x_{voc} and x_{acc} were mixed without modification (-0 dB), one were x_{voc} was attenuated by -5 dB prior to mixing the signals, and a third mix with x_{voc} being attenuated by -10 dB.

To construct an automated vibrato detection procedure based on our proposed template-based analysis approach, we first computed vibrato salience spectrograms S for all of the resulting 27

			-0 dB		-5 dB		-10 dB		
	L_x	L_{vib}	`TB-A`	`F0-M`	`TB-A`	`F0-M`	`TB-A`	`F0-M`	`BL`
Sound On Sound Demo—Mystery	9.79	1.78	0.83	0.93	0.84	0.86	0.73	0.30	0.31
Giselle—You	5.12	2.99	0.91	0.94	0.91	0.88	0.86	0.53	0.73
Leaf—Full	5.36	1.64	0.84	0.86	0.74	0.29	0.82	0.00	0.46
Phre The Eon—Everybody is Falling Apart	2.47	0.47	0.98	0.97	0.96	0.97	0.95	0.00	0.32
Secretariat—Borderline	7.69	1.98	0.79	0.69	0.73	0.76	0.79	0.00	0.41
Sunshine Garcia Band—For I Am The Moon	12.54	3.36	0.63	0.73	0.67	0.62	0.74	0.44	0.42
Angela Thomas Wade—Milk Cow Blues	4.50	2.10	0.44	0.82	0.32	0.63	0.32	0.00	0.63
Triviul—Dorothy	5.22	0.85	0.77	0.88	0.73	0.85	0.65	0.00	0.28
Funny Valentines—Sleigh Ride	7.18	0.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00
∅	6.65	1.69	0.80	0.87	0.77	0.76	0.76	0.25	0.39

Table 6.1: Quantitative evaluation (F-measure), comparing our proposed template-based detection approach `TB-A`, F0-based vibrato detection `F0-M` (manual vibrato selection in F0-trajectories), and a baseline `BL`. Lengths of the signals (L_x) and accumulated lengths of ground truth vibrato passages (L_{vib}) are given in seconds.

mix signals. Since only high vibrato salience values in S are likely to indicate the presence of spectro-temporal vibrato patterns, we then chose a threshold $\tau \in [-1, 1]$. Time instances where the maximal vibrato salience in a frame exceeded τ were then labeled as having vibrato while all other time instances were labeled as having no vibrato. For this experiment we used a set $\mathcal{T}$ of 30 templates, reflecting vibrato rates from five to seven Hertz in steps of 0.5 Hertz, as well as extents from 50 to 100 cents in steps of 10 cents. These parameters were chosen particularly to detect the vibrato in singing voice as these are typical vibrato rates and extents for human singing, see [170, 169]. All templates had a length corresponding to $\ell = 0.4$ seconds. The threshold τ was experimentally set to $\tau = 0.55$, yielding good vibrato detection results for all items in the dataset.

One of this experiment's main objectives was to compare our template-based method's performance with F0-based strategies as discussed initially. To emulate such an approach, we used MELODIA [183]—a state-of-the-art algorithm for estimating F0-trajectories of predominant musical voices in complex music signals—to estimate trajectories for all mix signals. Instead of automatically analyzing the extracted trajectories in a second step, we then manually inspected them for passages that reflect vibrato. This was done to obtain an upper bound on the performance an automated procedure could achieve in this second step when detecting vibrato solely based on the estimated F0-trajectory.

We then computed precision (P), recall (Re), and F-measure (F) for the detection results of our automated template-based procedure (`TB-A`), for the procedure based on the manually inspected F0-trajectory (`F0-M`), as well as for a baseline approach that simply labels every time instance as having vibrato (BL):

$$\mathrm{P} = \frac{\mathrm{TP} + \epsilon}{\mathrm{TP} + \mathrm{FP} + \epsilon},\ \mathrm{R} = \frac{\mathrm{TP} + \epsilon}{\mathrm{TP} + \mathrm{FN} + \epsilon},\ \mathrm{F} = \frac{2\mathrm{PR}}{\mathrm{P} + \mathrm{R}}\ . \tag{6.14}$$

Here, TP is the number of true positives, FP the number of false positives, FN the number of false negatives, and $\epsilon > 0 \in \mathbb{R}$ is some small number to prevent division by zero. Note that all music signals and annotations used in the experiment can be found at this paper's accompanying website [34].

The evaluation's results are summarized in Table 6.1 which shows for each item its name, the music signal's length, the accumulated duration of vibrato in this signal, as well as the F-measures of `TB-A` and `F0-M` for the three different mixes (-0 dB, -5 dB, and -10 dB). The F-measure for the baseline `BL` is indicated in the last column and the table's last row indicates mean values. Here we can observe a clear trend. For mixes where x_{voc} was not attenuated (-0 dB), both `TB-A` and `F0-M` yield average F-measures (F = 0.80 and F = 0.87) clearly above the baseline `BL` (F = 0.39). For this mixing condition, `F0-M` outperforms our template-based approach. However, recall that `F0-M` constitutes an upper bound on the performance of F0-based vibrato detection approaches. Automating the vibrato detection step may therefore result in lower scores.

For mixes where x_{voc} was attenuated by -5 dB, the average F-measure of `TB-A` only slightly decreases to F = 0.77, while the performance of `F0-M` drops to F = 0.76. This tendency becomes even more extreme when considering vocal signals attenuated by -10 dB where `TB-A`'s performance stays almost constant (F = 0.76) while `F0-M`'s average F-measure goes down to F = 0.25, many of the individual items scoring F-measures of zero.

The reason for this trend becomes obvious when investigating individual items. Figure 6.11 depicts the vibrato detection results of both `TB-A` and `F0-M` in all mixing conditions for the item *Leaf—Full*. In the condition -0 dB, the results of `TB-A` (Figure 6.11a) and `F0-M` (Figure 6.11b) coincide well with the ground truth (Figure 6.11c), leading to high F-measures (F = 0.84 and F = 0.86). Here, our template-based analysis approach detects most of the spectro-temporal vibrato patterns in the signal's spectrogram (time-frequency bins where the vibrato salience exceeds the threshold τ are indicated in red in Figure 6.11a). `F0-M` also achieves a good result since the F0-trajectory extracted by MELODIA (indicated in blue in Figure 6.11b) captures the singing voice's fundamental frequency well in this mix. However, this changes when attenuating the vocal signal by -5 dB. While `TB-A` still identifies many vibrato patterns, therefore detecting the vibrato present in the mix (F = 0.74), the F0-estimation becomes problematic and MELODIA retrieves only a small segment of the singing voice's F0-trajectory correctly, leading to a poor vibrato detection (F = 0.29). When attenuating x_{voc} by -10 dB, the F0-trajectory's estimation fails completely (F = 0.00) since MELODIA's assumption of a predominant melodic voice is

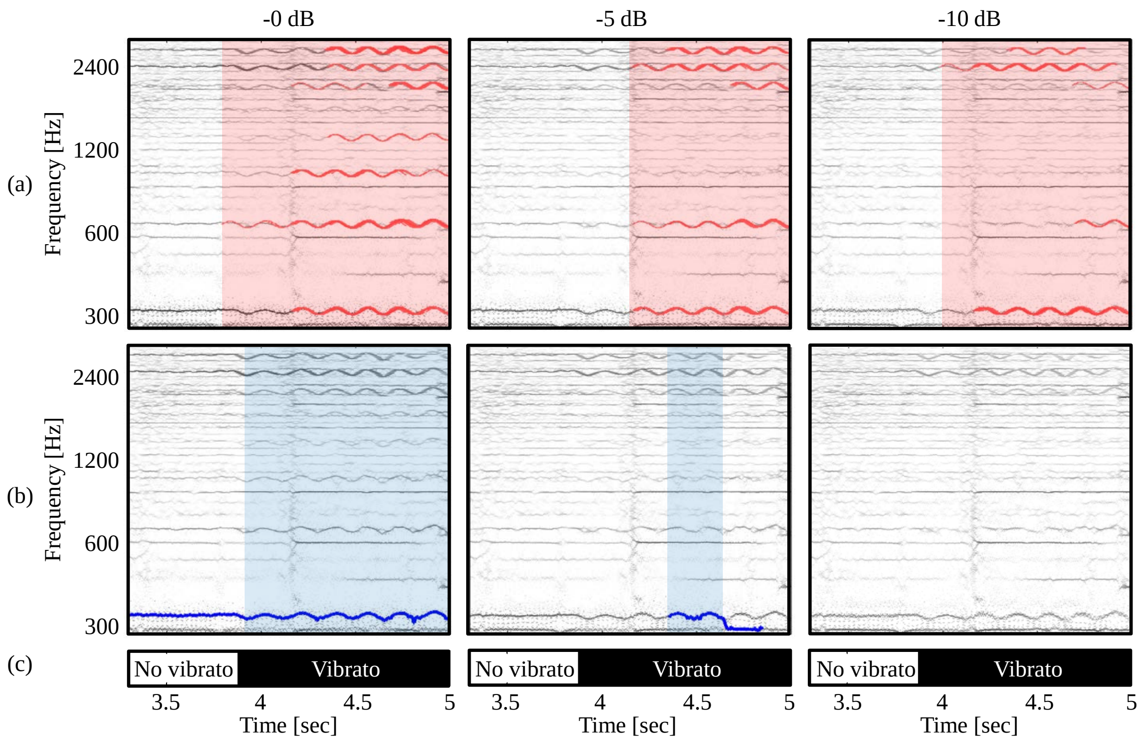


Figure 6.11: Comparison of `TB-A` and `F0-M` for the item *Leaf—Full*. **(a):** `TB-A`. Automatically derived vibrato passages are indicated in red. **(b):** `F0-M`. Manually annotated vibrato passages in the trajectory are indicated in blue. **(c):** Ground truth annotation.

violated. On the other hand, our proposed detection procedure is capable of detecting the vibrato in the mix.

As a final remark, note that our proposed approach also succeeds to recognize that the item *Funny Valentines—Sleigh Ride* does not contain any vibrato at all.

6.2.6 Evaluation: Vibrato Analysis

As we have seen in the previous section, the vibrato salience spectrogram S can be used to determine *when* vibrato is present in a music signal. Additionally, when computing S, we also implicitly obtain information about the vibrato's parameters. The rate and extent of vibrato present in the music signal are encoded by the similarity-maximizing vibrato templates T in Equation (6.13). In Figure 6.12a, we see the log-frequency spectrogram of a mixture of piano music (no vibrato) and three consecutive artificial vibrato tones. The tones have vibrato rates of seven, five, and ten Hertz and extents of 40, 200, and 70 cents, respectively. Time-frequency bins where the vibrato salience exceeds τ are indicated in red. Note that for this experiment we used a much larger template set $\mathcal{T}$, consisting of 285 templates that reflected vibrato rates from four to eleven Hertz in steps of 0.5 Hertz, as well as extents from 30 to 210 cents in steps of

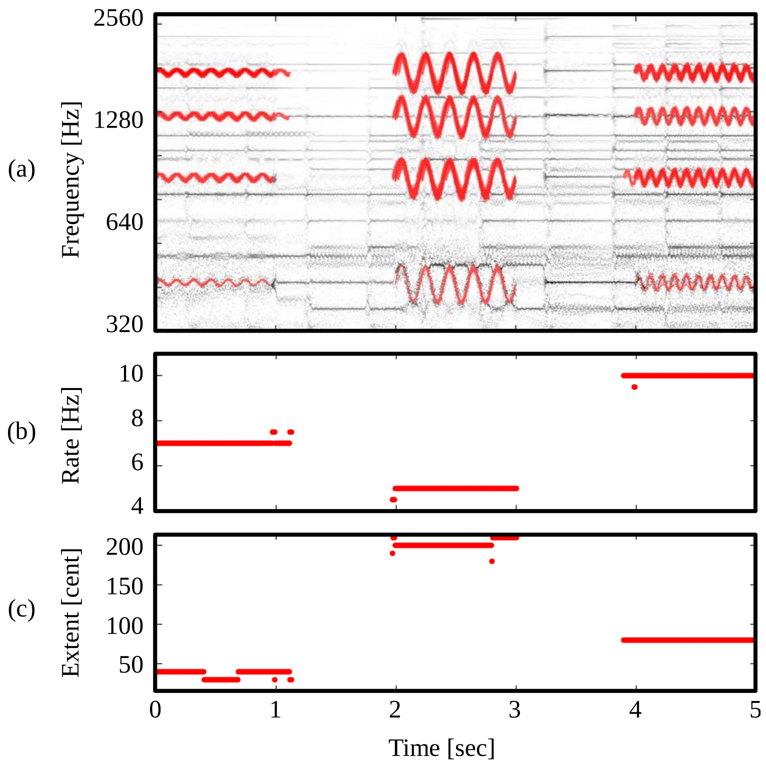


Figure 6.12: Vibrato rate and extent analysis. **(a):** Log-frequency spectrogram. Time-frequency bins (m, k) with $S(m, k) > \tau$ are indicated in red. **(b)/(c):** Vibrato rate and extent of the template T with the highest vibrato salience per frame.

10 cents. Figures 6.12a/b indicate the vibrato rate and extent of the vibrato template T that maximized the vibrato salience per frame. The two plots correctly reflect the tones' vibrato rates and extents, while showing only a few outliers. Note that values in the plots are quantized since our approach can only give estimates for rates and extents as they are reflected by one of the templates in $\mathcal{T}$. This kind of vibrato analysis could be helpful in scenarios like informed instrument identification when it is known that different instruments in a music signal perform with different vibrato rates or extents.

6.2.7 Challenges

In general, our proposed procedure yields useful analysis results for the music examples discussed in the previous sections. We now want to discuss a few difficult examples.

One potential source for incorrect analysis results are false positives as visualized in Figure 6.13a, which shows a log-frequency spectrogram excerpt of *Sunshine Garcia Band—For I Am The Moon* from our dataset. In this excerpt, one of our vibrato templates T is similar enough (with respect to our similarity measure) to a non-vibrato spectro-temporal pattern to yield vibrato salience

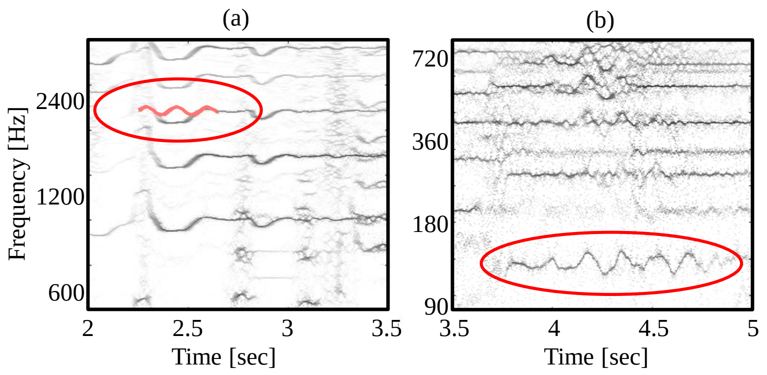


Figure 6.13: Error sources for our template-based vibrato analysis. **(a):** Spurious template matches. **(b):** Vibrato does not have a sinusoidal form.

values above the threshold τ. This could cause incorrect vibrato detection results or meaningless vibrato parametrizations. However, we experienced such spurious template matches to often occur in an isolated fashion. Here, one could exploit additional cues such as multiple template matches at the same time instance due to overtone structures of instruments to reinforce the vibrato analysis' results.

The opposite situation is visualized in Figure 6.13b. It shows a log-frequency spectrogram excerpt of "Gute Nacht", a song from Schubert's "Winterreise" for piano and tenor. In this excerpt, the singer sings a long note with strong vibrato. However, although there is a template reflecting an appropriate vibrato rate and extent in our template set $\mathcal{T}$, the vibrato is not detected by our procedure. This is the case since by our vibrato template's design—as described in Section 6.2.2—we generally assumed vibrato to have a sinusoidal spectro-temporal structure. This assumption is violated in the shown vibrato pattern. However, our approach is conceptually not limited to sinusoidal vibrato templates and one could further improve the templates' design in order to also capture these kind of vibrato patterns.

6.3 Conclusions and Further Notes

In Section 6.1, we presented a framework that allows for decomposing a given audio recording into score-based, and therefore musically meaningful audio events. Furthermore, we showed how this decomposition can be used for audio editing and analysis purposes. As discussed in Section 6.1.4, the residual signal x_{res} provides valuable information about deviations between a performance and a corresponding score as well as about misalignments of the MIDI events with the audio recording. As for future work, building a classifier that can automatically distinguish between different kinds of error sources may not only be beneficial for the source separation procedure itself, but could also aid in performance analysis applications or in improving alignment techniques. A further challenge is to investigate to which extend our decomposition may serve as an instance of object-oriented sound representation with applications to parametric audio coding and interactive remixing, see [95]. Finally, it showed that the perceptual quality of the individual audio events can be improved by post-processing them individually [26] which is especially interesting in music editing scenarios.

In Section 6.2, we presented a novel approach for analyzing vibrato in complex music signals. By locally comparing a signal's spectrogram with a set of predefined vibrato templates, we derived a vibrato salience spectrogram—a kind of mid-level feature representation—in order to locate and parameterize spectro-temporal vibrato patterns. Our approach has the advantage that the analysis does not rely on the estimation of a (possibly erroneous) F0-trajectory. Experiments indicated that our proposed procedure allows for a more robust vibrato detection than F0-based approaches, in particular for complex music signals.

In future work we would like to further explore the use of vibrato templates in various application scenarios. For example, deriving spectral masks from the vibrato salience spectrogram S could open up novel ways of decomposing a music signal into vibrato and non-vibrato components. Furthermore, we believe that the use of vibrato templates could be beneficial for tasks like F0-tracking [201, 183] or performance analysis [2].

Finally, we would like to extend the experiments performed in Section 6.2.5. Instead of manually annotating the F0-trajectories estimated by MELODIA, one possibility to construct an automated F0-based vibrato analysis procedure would be to exploit the *Tempogram Toolbox* [89]. This MATLAB toolbox—originally developed for tempo analysis of music signals—can be used to perform a Fourier-like analysis of a one-dimensional signal by locally comparing it with a set of sinusoidal prototype signals [88]. In case this signal is a novelty curve as introduced in Section 2.5, this yields a representation called a *Fourier-based tempogram*, which is a time-tempo plot of the given signal. In the context of F0-based vibrato analysis, the exact same method could be used to derive a time-rate representation from an F0-trajectory.

We can easily compute a similar representation with our proposed template-based vibrato analysis approach. For each frame in the vibrato salience spectrogram S, we can count the number of similarity-maximizing templates for a fixed vibrato rate, considering only salience values above a threshold τ. Doing this for all rates as reflected by our templates yields a histogram of vibrato rates for each frame. The sequence of histograms constitutes the desired time-rate representation. Figure 6.14 shows the results of a first experiment in that direction, where we compare to the automated F0-based time-rate representation derived by using the Tempogram Toolbox with the histogram representation derived from S. In Figure 6.14a, we see the log-frequency spectrogram of the item *Sound On Sound Demo—Mystery* from our dataset (considering the mix with no attenuation of the singing voice). Figures 6.14b/c show the F0-trajectory of the excerpt as estimated by MELODIA, as well as the vibrato salience Spectrogram S. From these

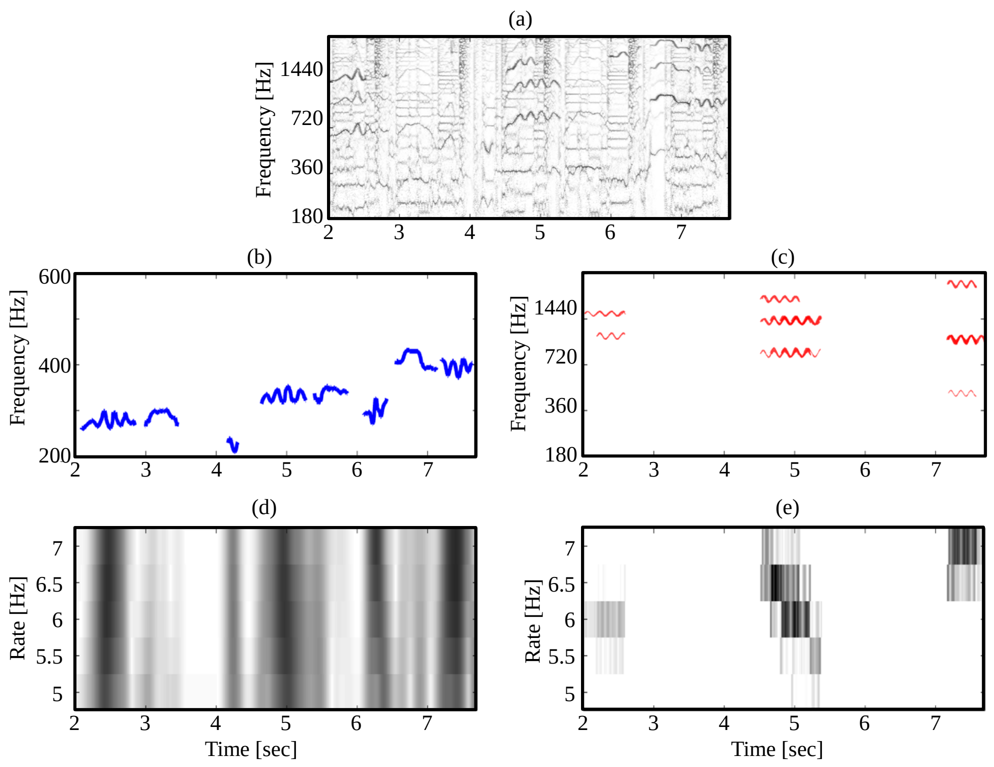
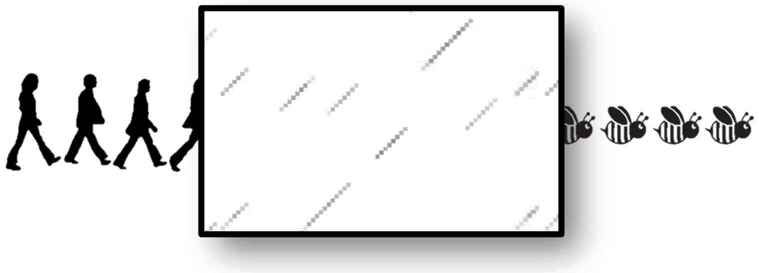


Figure 6.14: Automated F0-based vibrato analysis based on the *Tempogram Toolbox*. **(a):** Log-frequency spectrogram of *Sound On Sound Demo—Mystery*. **(b):** F0-trajectory estimated by MELODIA. **(c):** Vibrato salience spectrogram. Only values above a threshold τ are shown. **(d):** "Tempogram" derived from the F0-trajectory. **(e):** Histogram of the similarity-maximizing vibrato templates' rates.

representations, we derive a "tempogram" based on the F0-trajectory, as well as the histogram of vibrato rates from S as explained above (Figures 6.14d/e). One can see that the two time-rate representations capture similar information. Although the rate resolution in the tempogram representation is rather blurred, dark passages roughly correspond to vibrato passages in the trajectory—as it is also the case with the histogram representation. However, the tempogram representation seems to be much more sensitive to short frequency modulations in the singing voice that are usually not considered to be vibrato. For example, see the short trajectory segment in Figure 6.14b briefly after second four that causes a clear indication in the tempogram representation in Figure 6.14d. A closer investigation of this general approach is left as future work.

Chapter 7

NMF-Inspired Audio Mosaicing for Music Signals



This chapter deals with the task of *audio mosaicing* and is mainly based on our work presented in [50].

Using the sounds in a recording of buzzing bees to recreate a recording of the song “Let it be” by the Beatles is a typical example of an audio mosaic. In this example, the recording of the bees serves as *source*, while the Beatles recording is called the *target*. Ultimately, one should be able to identify the target recording when listening to the mosaic, but at the same time perceive the timbre of the source sounds. Therefore, the audio mosaic of “Let it be” with the bee recording could give the impression of bees being musicians, buzzing the song’s tune.

Audio mosaicing is an interesting audio effect which has found its way into both artistic work as well as academic research. Artists like John Oswald used thousands of manually selected source

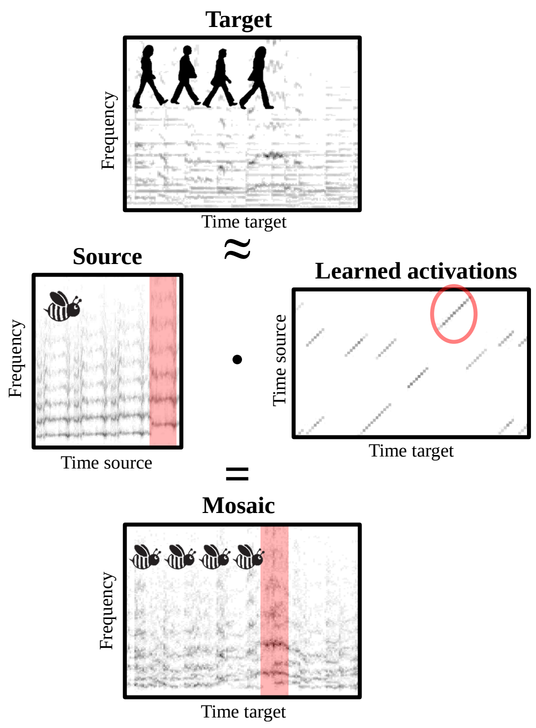


Figure 7.1: Schematic overview of our proposed audio mosaicing method. The sparse diagonal structures in the activation matrix are important in order to preserve the timbre of the source in the mosaic.

audio snippets to create new musical compositions[1] and real-time audio mosaicing has been used by musicians as an instrument in live performances [212, 23]. Over the years, many different systems for audio mosaicing were proposed [189, 20, 7, 185, 120, 112, 24, 203]. The core idea of most automated systems is to split the source into short audio segments, which are suitably concatenated afterwards to match spectral and temporal characteristics of the target [190, 203].

In this chapter, we propose a novel way to create audio mosaics. Our idea is to learn an *activation matrix* that, when multiplied with the spectrogram of the source recording, approximates the spectrogram of the target recording (see Figure 7.1). The source spectrogram hereby serves as a *template matrix* which is fixed throughout the learning process. This way, as opposed to many previous automated mosaicing approaches, a frame of the target can be re-synthesized as the superposition of several spectral frames of the source, thus allowing "polyphony" of the source sounds.

As a first contribution, we propose an audio mosaicing procedure which is inspired by well-known algorithms for *non-negative matrix factorization* (NMF) [129]. Keeping the template matrix fixed (the source's magnitude spectrogram), this basic procedure learns an activation matrix by iteratively applying a standard NMF update rule to a randomly initialized matrix. Experiments show that in case the source recording offers an appropriate amount of different sounds, this procedure can closely approximate the spectrogram of the target recording. However, the source's timbre is often barely recognizable in the resulting mosaics. The reason is that the procedure

[1]Especially on his album *Plexure* [159].

recreates every target frame independently, thus destroying temporal characteristics of the source in the final audio mosaic. Furthermore, the method can superimpose an arbitrary number of spectral frames from the source to construct a good numerical approximation of a single target frame. A superposition of a large number of source sounds may however result in a timbre that is no longer similar to the actual timbre of the source. Therefore, an exact approximation of the target's spectrogram cannot be our procedure's sole goal.

As our main technical contribution, we therefore propose an extended set of update rules that supports the development of sparse diagonal structures in the activation matrix during the learning process (see the activation matrix in Figure 7.1). Rather than single frames, diagonal structures activate whole frame sequences in their original order. This preserves the source's temporal characteristics in the resulting mosaic. Furthermore, the extended set of update rules also limits the number of simultaneous activations, making the learned activation matrix sparse and reducing the problem of too many source sounds being audible simultaneously. This way, we trade some approximation quality for a better preservation of the source's timbre.

The idea of activating sequences of frames is inspired by methods like *non-negative matrix factor deconvolution* (NMFD) and related formulations [197, 200], where template sequences of frames from a dictionary are activated by single activation values. However, our approach is conceptually different. Instead of changing the NMF problem formulation, our approach stays in the standard NMF setting, supporting the activation of whole frame sequences directly in the activation matrix with additional update rules. Besides being computationally very efficient and easy to implement, this also has the advantage that we do not need to choose a maximal length of the sequences as in NMFD. Similarly, the sparseness constraint imposed by our procedure is not enforced by penalty terms in the problem formulation (as for example in [56, 102, 115, 221]), but also by additional update rules.

The remainder of this chapter is structured as follows. In Section 7.1 we introduce the basic concept of using NMF-inspired update rules for the task of audio mosaicing. In Section 7.2 we present the extended set of update rules that supports the development of sparse diagonal structures in a learned activation matrix. The effects of these update rules on the audio mosaics are discussed and demonstrated in Section 7.3.

7.1 Basic NMF-Inspired Audio Mosaicing

As discussed in Section 6.1, non-negative matrix factorization (NMF) has been applied very successfully in a large variety of music processing tasks and beyond. For the sake of convenience, we state the basic concept again at this point. Given a non-negative matrix $V \in \mathbb{R}_{\geq 0}^{N \times M}$, the goal of NMF is to decompose this matrix into two factors $W \in \mathbb{R}_{\geq 0}^{N \times K}$ and $H \in \mathbb{R}_{\geq 0}^{K \times M}$, where

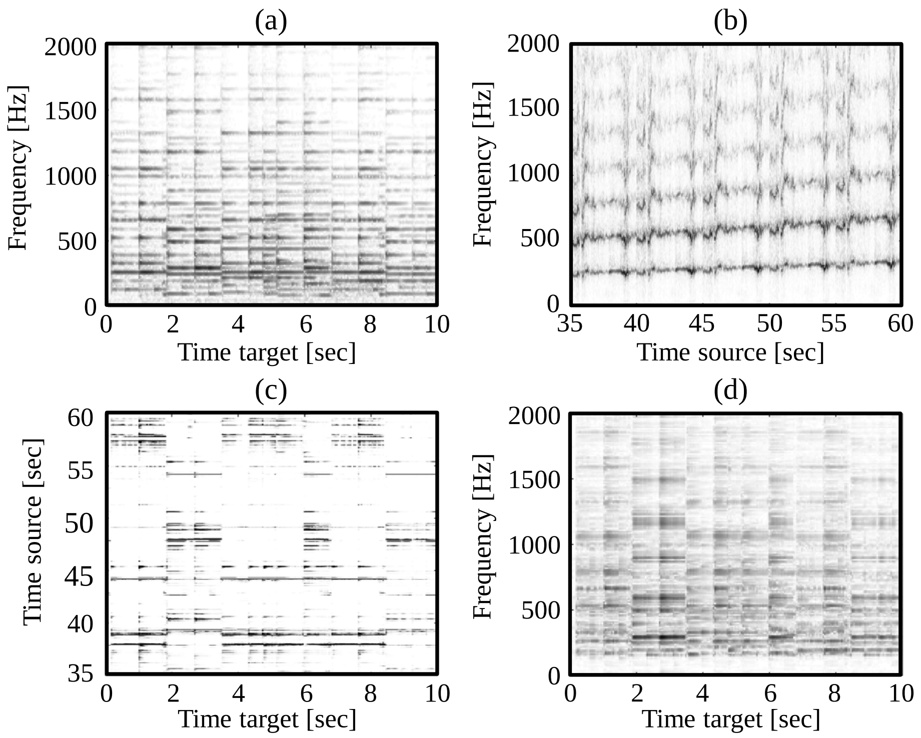


Figure 7.2: Basic NMF-inspired audio mosaicing. **(a):** Magnitude spectrogram of "Let it be" V (target). **(b):** Magnitude spectrogram of a recording of bees W (source). **(c):** Activation matrix H. **(d):** The product WH (mosaic).

$N, M, K \in \mathbb{N}$. The distance between the product WH and the matrix V is minimized with respect to some distance measure, for example the Kullback-Leibler divergence

$$(V||WH) = \sum_{nm} V_{nm} \log \frac{V_{nm}}{(WH)_{nm}} - V_{nm} + (WH)_{nm}. \tag{7.1}$$

In the context of music processing, the matrix V is usually a magnitude spectrogram of a music recording, the matrix W is interpreted as a set of spectral templates, and the matrix H constitutes an activation matrix. Non-zero values in a row of H activate the associated template in W at the respective time instance. The two factors W and H are usually learned by iteratively applying multiplicative update rules to two suitably initialized matrices [129].

Fixing the template matrix W to be the magnitude spectrogram of the source recording, the basic idea of our proposed audio mosaicing approach is to learn only the activation matrix H. More precisely, we proceed as follows. Given the target recording x_{tar} and the source recording x_{src}, we first compute the complex valued spectrograms X_{tar} and X_{src} by applying the short-time Fourier transform (STFT) to both recordings. Afterwards, we set $V := |X_{\text{tar}}|$, $W := |X_{\text{src}}|$, and randomly initialize $H^{(1)} \in (0, 1]^{K \times M}$. Fixing a number of iterations L, we then iteratively update

H with

$$H_{km}^{(\ell+1)} = H_{km}^{(\ell)} \frac{\sum_n W_{nk} V_{nm} / (WH^{(\ell)})_{nm}}{\sum_n W_{nk}} \,, \tag{7.2}$$

for $k \in [1:K]$, $m \in [1:M]$, and the iteration index $\ell \in [1:L-1]$. Finally, we set $H := H^{(L)}$. The learned activation matrix H is then multiplied with the complex valued X_{src}, yielding the complex valued spectrogram of the audio mosaic $X_{\mathrm{mos}} := X_{\mathrm{src}}H$. To compute the audio mosaic x_{mos}, we apply an "inverse" STFT to the spectrogram X_{mos} which also adjusts the phases such that artifacts from phase discontinuities are reduced [84].

Figure 7.2 shows this basic procedure applied to our running example. In Figure 7.2a we see an excerpt of the magnitude spectrogram of the song "Let it be". Our goal is to create an audio mosaic of this song, using the recording of buzzing bees, which can be seen in Figure 7.2b. To increase the range of different pitches occurring in our source, we used a pitch-shifting algorithm [40] to create differently pitched versions of the bee recording and concatenated them. Figure 7.2c shows an excerpt of the activation matrix H, derived by applying the basic procedure described above. A first observation about H is the predominance of horizontal activation structures. These patterns correspond to single spectral frames in the source which are activated repeatedly to mimic the stable spectral structures in the target. Although the resulting mosaic, shown in Figure 7.2d, closely resembles these spectral structures, one can hear a "stuttering" effect when listening to the reconstructed audio recording. This stuttering originates from the same frame of the source being repeated over and over again. In Section 7.2.1, we aim to prevent the learning process from activating the same frame in fast repetition with an additional update rule.

A second observation is that the matrix H usually activates many source frames simultaneously. The learning process can thus closely approximate the spectral shapes of the target frames. However, in the context of audio mosaicing, this has several drawbacks. Since H is multiplied with the complex spectrogram X_{src}, phase cancellation artifacts may arise when superimposing many complex spectral frames. This way, especially low pitched sounds tend to cancel each other out and are not audible in the final audio mosaic. Furthermore, since a sound's timbre is also closely related to the energy distribution in its frequency spectrum, adapting the spectral shapes may change the timbre of the source. An update rule which sets a limit on the maximal number of simultaneous activations is presented in Section 7.2.2.

A third problem connected with the activation matrix shown in Figure 7.2c is the loss of temporal characteristics of the source. The typical "buzzing sound" of the bees, which results from pitch modulations (see Figure 7.2b), is lost in the mosaic (see Figure 7.2d). This is the case since the spectral frames of the source are activated independently of their order in the source spectrogram. To preserve some temporal characteristics, the update rule presented in Section 7.2.3 supports the development of diagonal structures in the activation matrix.

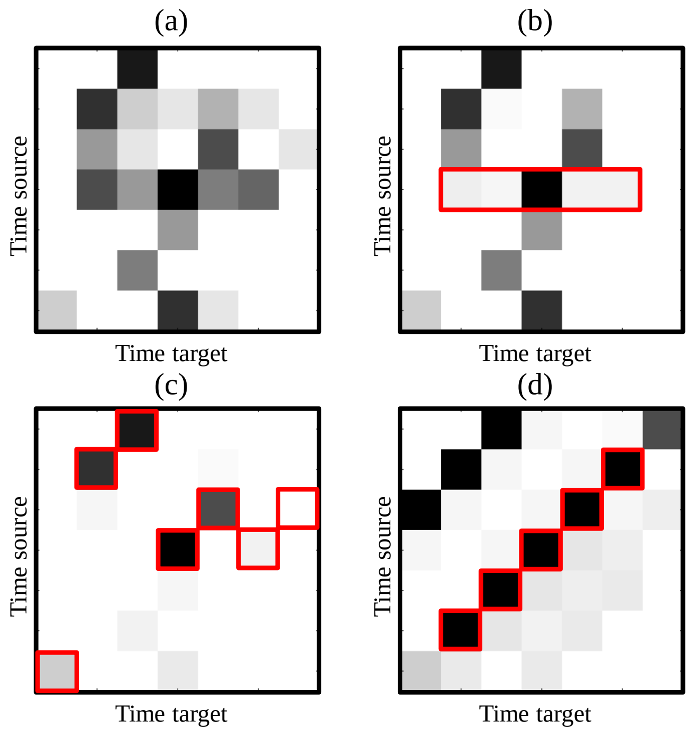


Figure 7.3: (a): Activation matrix $H^{(\ell)}$. **(b):** Repetition restricted activation matrix $R^{(\ell)}$. The horizontal neighborhood is indicated in red. **(c):** Polyphony restricted activation matrix $P^{(\ell)}$. For each column, the highest value is indicated in red. **(d):** Continuity enhancing activation matrix $C^{(\ell)}$. The diagonal kernel is indicated in red.

7.2 Learning Sparse Diagonal Activations

The core idea to overcome the issues of the basic NMF-inspired audio mosaicing procedure is to impose specific constraints on the learned activation matrices by adapting the iterative update process. As discussed in the previous section, we identified three main problems of the mosaics generated by the basic procedure, all related to properties of the the derived activation matrices. First, horizontal activation patterns cause stuttering artifacts in the mosaics. Second, too many simultaneous activations lead to phase cancellations and overfitting of the spectral shapes. Third, the source's temporal characteristics are destroyed by activating source frames independently of each other. We therefore introduce additional update rules to approach these issues, see also Figure 7.3.

7.2.1 Avoiding Repeated Activations

To avoid activating the same spectral frame of the source in subsequent time-instances, the idea is to only keep the highest activations in a horizontal neighborhood of the matrix H, suppressing the remaining values. However, we do not want to interfere too much with the actual learning process in the first few update iterations. The amount of suppression applied to the smaller values is therefore dependent on the iteration index ℓ. Given the activation matrix $H^{(\ell)}$, the size of a horizontal neighborhood r, and the number of iterations L, we compute a *repetition restricted* activation matrix $R^{(\ell)}$ by

$$R^{(\ell)}_{km} = \begin{cases} H^{(\ell)}_{km} & \text{if } H^{(\ell)}_{km} = \mu^{r,(\ell)}_{km} \\ H^{(\ell)}_{km}(1 - \frac{(\ell+1)}{L}) & \text{otherwise} \end{cases} , \tag{7.3}$$

with $\ell \in [1 : L-1]$ and $\mu^{r,(\ell)}_{km}$ being the maximum value of $H^{(\ell)}$ in a horizontal neighborhood

$$\mu^{r,(\ell)}_{km} = \max(H^{(\ell)}_{k(m-r)}, \ldots, H^{(\ell)}_{k(m+r)}) . \tag{7.4}$$

Note that the suppression of smaller values becomes strict in the last update iteration for $\ell = L-1$. Intuitively, the parameter r defines the minimal horizontal distance (and therefore the minimal time interval) between two activations of the same source frame. Figure 7.3b shows the repetition restricted activation matrix $R^{(\ell)}$ derived from the toy example activation matrix shown in Figure 7.3a, using $r = 2$, $\ell = 8$, and $L = 10$. As opposed to $H^{(\ell)}$, there are no two dominant values next to each other in $R^{(\ell)}$.

7.2.2 Restricting the Number of Simultaneous Activations

Next, we address the problem of too many simultaneous activations. Setting a limit $p \in \mathbb{N}$ on the number of activations in one column of the activation matrix, we compute a *polyphony restricted* activation matrix $P^{(\ell)}$ in a similar manner as $R^{(\ell)}$ by

$$P^{(\ell)}_{km} = \begin{cases} R^{(\ell)}_{km} & \text{if } k \in \Omega^{p,(\ell)}_{m} \\ R^{(\ell)}_{km}(1 - \frac{(\ell+1)}{L}) & \text{otherwise} \end{cases} , \tag{7.5}$$

where $\Omega^{p,(\ell)}_{m}$ contains the indices of the p highest values in the m^{th} column of $R^{(\ell)}$. The parameter p can be directly interpreted as the desired degree of polyphony in the mosaic. For example, setting $p = 1$ results in a mosaic where the source sounds are not heavily superimposed but mainly concatenated to mimic the most dominant features of the target. In Figure 7.3c, we see the polyphony restricted activation matrix $P^{(\ell)}$ derived from $R^{(\ell)}$, using $p = 1$. One can see that in $P^{(\ell)}$ there is (at most) one single dominant value left in every column.

7.2.3 Supporting Time-Continuous Activations

To support the development of diagonal structures that activate successive frames of the source, we now compute a *continuity enhancing* activation matrix $C^{(\ell)}$. The idea here is to convolve the matrix P with a diagonal kernel. Choosing $c \in \mathbb{N}$, which defines the length of the kernel, we compute

$$C^{(\ell)}_{km} = \sum_{i=-c}^{c} P^{(\ell)}_{(k+i)(m+i)} \, . \tag{7.6}$$

Intuitively, the length $2c+1$ of the kernel defines the minimal number of source frames that we would like to successively activate. Figure 7.3d shows the matrix $C^{(\ell)}$ for our toy example, computed with $c = 2$. Note that in $C^{(\ell)}$ the number of simultaneous dominant activations may locally exceed the limit which was imposed in the computation of the polyphony restricted activation matrix $P^{(\ell)}$. In practice, this is however not a problem and even desirable since this way, the diagonal structures can overlap with each other to some degree. Therefore, the corresponding audio segments of the source are overlapped in the final mosaic as well, leading to smooth transitions between them.

7.2.4 Adapting the Activations to Fit the Target

Finally, we perform the standard NMF update step to let the mosaic adapt to the target again. Similarly to Equation (7.2), we compute the activation matrix for the next iteration by

$$H^{(\ell+1)}_{km} = C^{(\ell)}_{km} \frac{\sum_n W_{nk} V_{nm} / (WC^{(\ell)})_{nm}}{\sum_n W_{nk}} \, . \tag{7.7}$$

In summary, a single update step of the activation matrix H is computed by applying Equations (7.3), (7.5), (7.6), and (7.7) sequentially.

Note that in one update iteration, the three intermediate update rules (7.3), (7.5), and (7.6) are insensitive to the target and therefore may increase the distance measure of Equation (7.1). However, as already discussed above we are not interested in minimizing this measure, but trade some approximation accuracy for a better preservation of the source's timbre. In practice, our procedure usually yields an activation matrix that, when multiplied with the source spectrogram, approximates the target spectrogram to a sufficient degree, while preserving the source's timbre in the mosaic much better than the basic procedure described in Section 7.1.

Figure 7.4 shows an excerpt of the activation matrix H of our running example "Let it be" for several iteration indices ℓ. Here, we set the repetition restriction parameter to $r = 3$, the limit of simultaneous activations to $p = 10$, the kernel parameter to $c = 3$ (resulting in a diagonal kernel of length 7), and the number of update iterations to $L = 10$. Figure 7.4a shows the

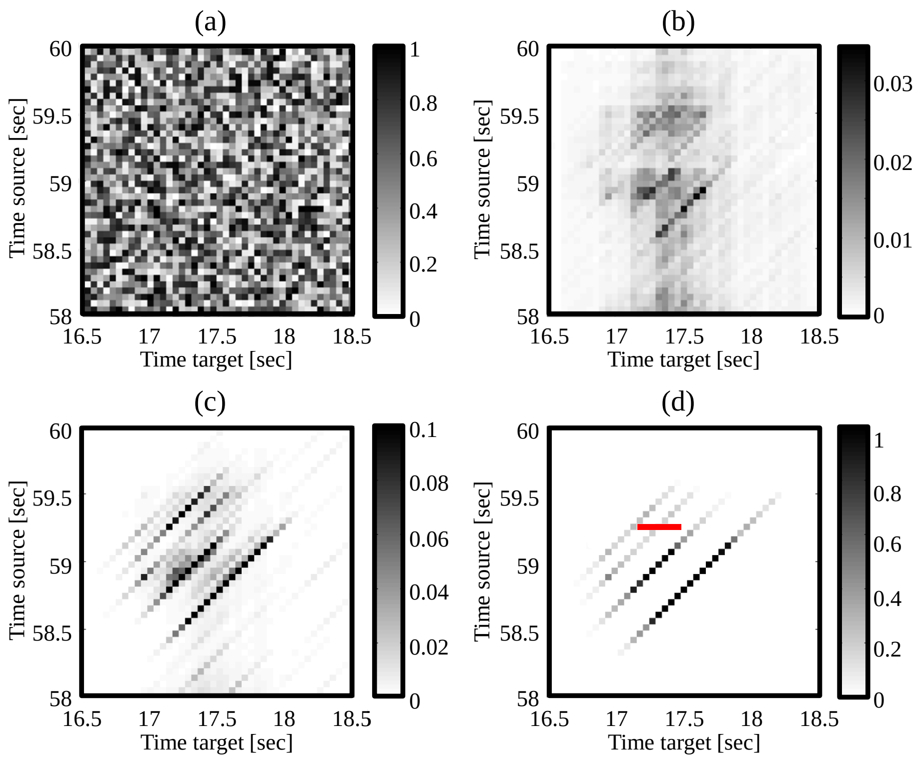


Figure 7.4: The activation matrix H for the mosaic of "Let it bee" with a recording of bees in different states. **(a):** $H^{(1)}$. **(b):** $H^{(3)}$. **(c):** $H^{(6)}$. **(d):** $H^{(10)}$. The repetition restricting neighborhood is indicated in red.

random initialization of the activation matrix $H^{(1)}$. After two iterations, one can already notice diagonal patterns in $H^{(3)}$, see Figure 7.4b. Figure 7.4c shows the activations after another three update iterations. The diagonal patterns in $H^{(6)}$ are even more prominent and one can observe that separate diagonal structures start to emerge, leaving regions of lower values inbetween them. In Figure 7.4d, the activation matrix $H^{(10)}$ is shown. In this final activation matrix, four clear diagonal structures have emerged. The remaining activations are outside the visible range. Looking at the two upper diagonals, one can see that although they seem to be rather close together, they obey the repetition restricting horizontal neighborhood indicated in red. Furthermore, it is noteworthy that the length of the diagonals greatly exceeds the length of the diagonal kernel. For example, while we used a diagonal kernel of length 7, the lowest diagonal has a length of 25 non-zero activations, corresponding to an audio segment in the source of roughly one second. This means that the procedure uses a whole one-second patch of source audio material to recreate the target between second 17 and 18.

Figure 7.5 exemplarily shows the progression of the Kullback-Leibler divergence (7.1) of the target's and the different mosaics' magnitude spectrograms over the course of $L = 20$ update iterations for our running example. The divergence values for the different mosaics resulting from the respective activation matrices are indicated by different symbols, see the legend of

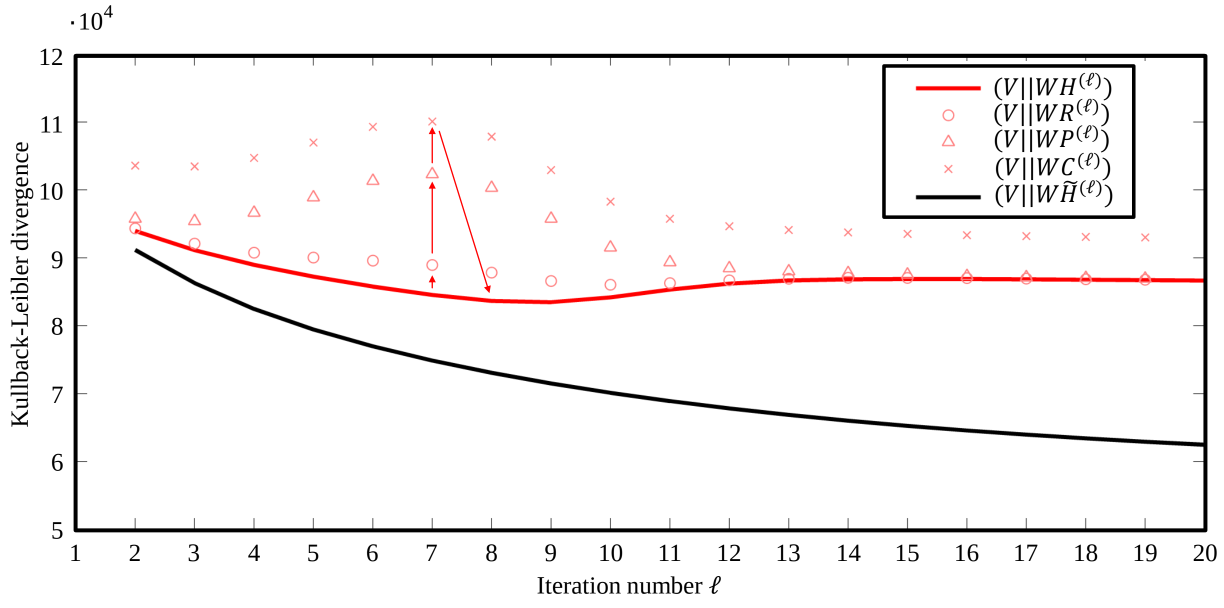


Figure 7.5: Progression of the Kullback-Leibler divergence of the target and mosaic magnitude spectrograms for different activation matrices over the course of $L = 20$ update iterations. The application sequence of update rules is indicated once with red arrows.

Figure 7.5. Note that in Figure 7.5 the activation matrix of the basic NMF-inspired mosaicing approach (see Section 7.1) is denoted by $\tilde{H}^{(\ell)}$ in order to distinguish it from the activation matrix resulting from applying the extended set of update rules H. Since $H^{(1)}$ and $\tilde{H}^{(1)}$ are initialized randomly, the divergence values for $\ell = 1$ are very large for all activation matrices and outside of the plot's visible range. In the subsequent update iterations of the basic NMF-inspires procedure (solid black line for $\ell > 1$) the divergence values are steadily decreased, as expected. The repeated application of our proposed extended set of update rules has a different effect on the divergence values. After one full update iteration (solid red line for $\ell = 2$), the mosaic's magnitude spectrogram is still rather similar to the one of the basic procedure in terms of its Kullback-Leibler divergence to the target's magnitude spectrogram. The repetition restriction as well as the restriction of the polyphony only have a mild effect on the divergence (the values of $(V||WP^{(2)})$ and $(V||WR^{(2)})$ are only slightly higher than $(V||WH^{(2)})$). This is the case, since the suppression of activation values by the update rules (7.3) and (7.5) is rather small for the first update iterations. The update rule (7.6) is independent of the update iteration ℓ and the diagonal convolution therefore leads to a larger increases of the the divergence measure. As intended, the application of the update rule (7.7) then leads to an adaption of the mosaic to the target and therefore decreases the divergence measure again. Over the course of iterations, the repetition and polyphony restriction constraints become more and more strict. The application of the respective update rules therefore increases the divergence measure by larger amounts. While the divergence measure for $H^{(\ell)}$ is decreasing over the course of the first few iterations,

it starts to increase again at around iteration 9. Interestingly, the divergence stabilizes around iteration 14. The applications of the repetition restricting, as well as the polyphony restricting update rule do not influence the divergence measure any more, although their constraints become more and more strict. This indicates, that the iterative update procedure led to a state in which the repetition- as well as the polyphony restriction constraints were satisfied, even before the update iterations were fully completed. In this state, only the continuity enhancing update rule (7.6) influences the divergence measure. However, its effect is always compensated for by the standard NMF-update. As expected, the overall Kullback-Leibler divergence of the target and the mosaic resulting from using the extended set of update rules is larger than the divergence for the mosaic resulting from the basic NMF-inspired approach (see Figure 7.5 for $\ell = 20$). As discussed before, this is acceptable since the extended set of update rules allows for a better preservation of the source's timbre in the mosaic as we will see in the next section.

7.3 Experiments and Examples

In this section, we both visually and acoustically demonstrate the effectiveness of our proposed method. As discussed in previous sections, the main drawbacks of the basic audio mosaicing approach described in Section 7.1 were both the loss of temporal characteristics and spectral shapes of the source sounds in the resulting audio mosaics. The idea was to approach these problems by supporting the development of sparse diagonal structures in the activation matrix with an extended set of update rules. In the following, we exemplify how these structures can preserve the source's desired characteristics in the audio mosaic.

7.3.1 Preserving the Source's Temporal Characteristics

In Figure 7.6, we once again revert to our running example. Here, spectrogram excerpts of the target recording "Let it be" as well as the source recording of buzzing bees are shown in Figures 7.6a and 7.6b, respectively. The spectrogram of the target recording exhibits sounds with very stable pitches, resulting from the solo piano at the song's beginning. In contrast, the buzzing of the bees leads to rather strong amplitude modulations that are characteristic for the sound. Figure 7.6c shows an excerpt of the activation matrix H as derived by the basic NMF-inspired audio mosaicing procedure. In this excerpt of H, only two different spectral frames of the source are activated repeatedly by the procedure to mimic the stable pitch of the piano sound. The resulting spectrogram of the audio mosaic, shown in Figure 7.6e, approximates the target's spectrogram quite precisely. However, the characteristic pitch modulations of the buzzing bee sound are lost almost completely. Looking at Figure 7.6d, one can see the activation matrix H derived by our proposed procedure based on the extended set of update rules. The

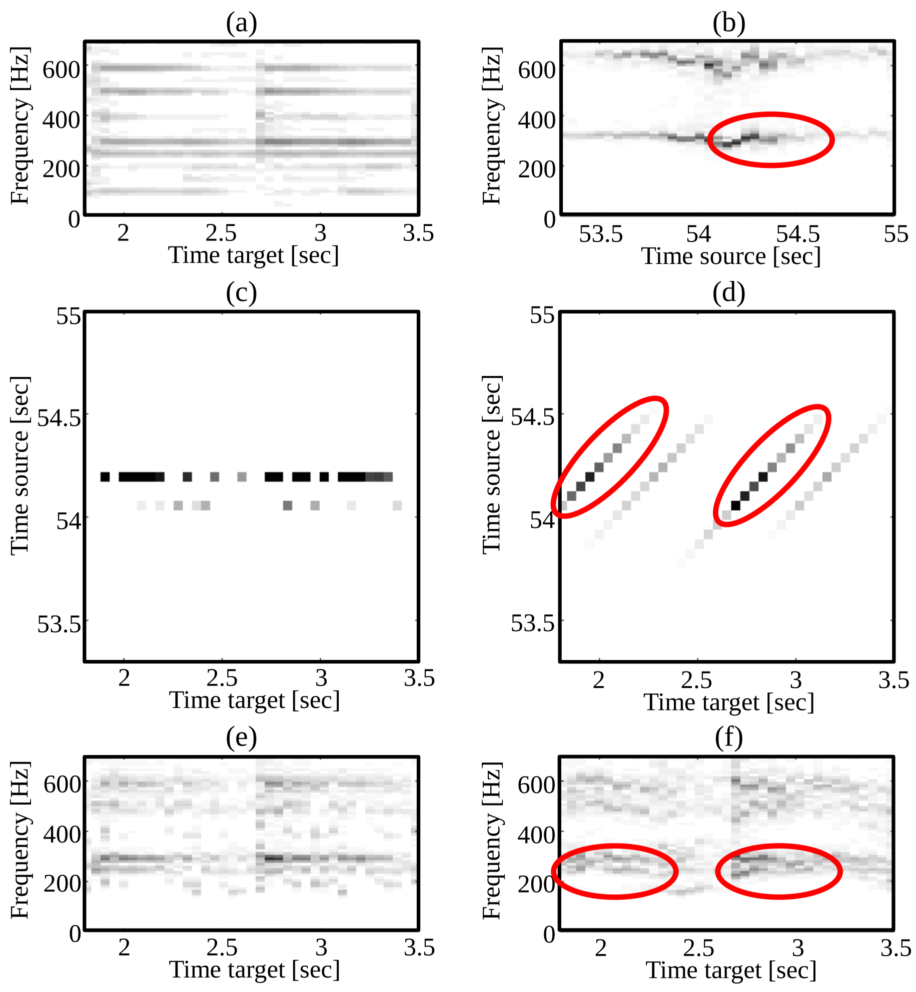


Figure 7.6: The effect of diagonal activation patterns. **(a):** Spectrogram of the target recording "Let it be". **(b):** Spectrogram of the source recording of buzzing bees. **(c):** Activation matrix H derived with the basic approach. **(d):** Activation matrix H derived with the extended set of update rules. **(e):** Spectrogram of the audio mosaic resulting from the basic approach. **(f):** Spectrogram of the audio mosaic resulting from the extended procedure.

diagonal patterns shown activate segments of the source that have a duration of roughly half a second. As can be seen by comparing the regions marked in red in the source (Figure 7.6b) and the mosaic spectrogram (Figure 7.6f), the temporal structures of these segments are preserved in the mosaic. While the mosaic computed with the extended set of update rules exhibits a lot of pitch modulations, which reflect the preserved timbre of the buzzing bee sound, the tonal content as well as rhythmic structures of the target are still maintained. For example, the two strong partials of the target recording at around 270 Hz and 300 Hz in Figure 7.6a are also visible in the audio mosaic in Figure 7.6f, only this time pitch modulated. Similarly, the onset in the target at second 2.6 is present in the mosaic as well.

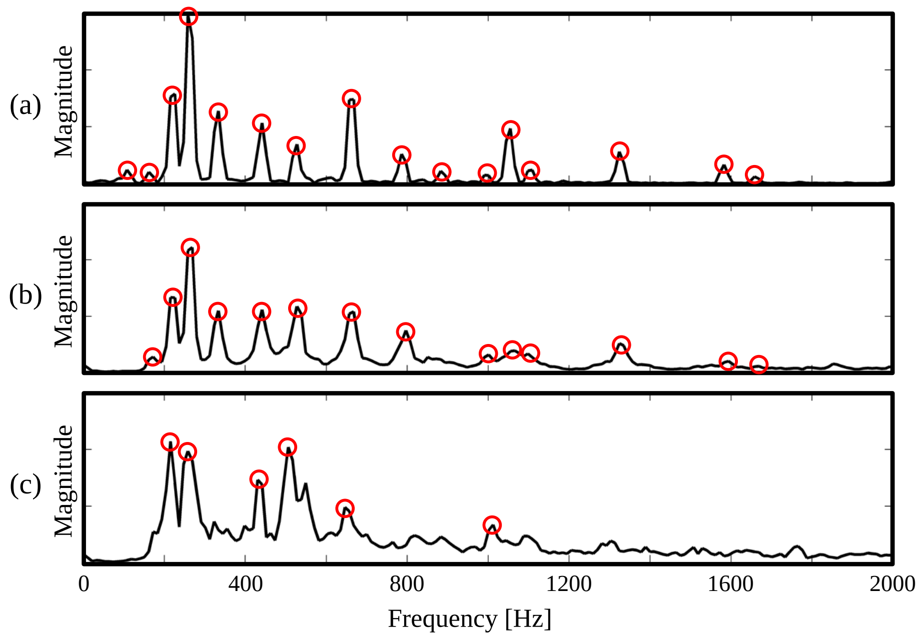


Figure 7.7: Comparison of spectral shapes. **(a):** A single spectral frame of the target recording ("Let it be"). Harmonics are indicated by red circles. **(b):** The spectral frame of the mosaic computed with the basic procedure at the same temporal position. Harmonics which are present in both the original frame as well as in the mosaic are indicated by red circles. **(c):** The spectral frame of the mosaic computed by using the extended set of update rules.

7.3.2 Preserving the Source's Spectral Shapes

In Figure 7.7, we investigate typical spectral shapes of the target as well as the mosaic for our running example. Figure 7.7a shows the spectral frame of the target's spectrogram at second 4.6 as a frequency-magnitude plot. One can see the harmonic structure with several clear partials in this frame, resulting from the piano sound in the target. The corresponding spectral frame of the mosaic computed by the basic procedure shown in Figure 7.7b shows a very similar spectral structure. Most of the harmonics visible in the target are also present in this frame (indicated by the red circles) and even the relations between peak heights are often preserved. In contrast, the spectral frame of the mosaic computed with the extended set of update rules only roughly corresponds to the spectral shape of the target frame, see Figure 7.7c. However, some of the dominant peaks in the target frame are still present in the mosaic, leading to a sound that captures only the dominant tonal characteristics of the target. The noisy timbre of the buzzing bees, visible by the increased noise level in the frame, is therefore preserved.

Target's name	**Description of the target**	**Source's name**	**Description of the source**
LetItBe	An excerpt of the song "Let it be" by the Beatles (piano & singing).	Bees	Recording of a buzzing swarm of bees.
GuteNacht	An excerpt of "Gute Nacht" by Franz Schubert which is part of the romantic *Winterreise* song cycle, taken from [152].	Wind	Recording of howling wind.
FunkJazz	An excerpt from a jazz piece performed by the band "Music Delta" (saxophone, synthesizer, bass, and drums), taken from [9].	Whales	Recording of whale songs and whale sounds.
Stepdad	Excerpt from the song "My leather, my fur, my nails" by the pop band Stepdad (synthesizers, drums, and singing).	Chainsaw	Recording of a chainsaw's sawing and engine sounds.
Freischütz	Excerpt from the opera "Der Freischütz" by Carl Maria von Weber (full orchestra, applause at the end).	AirRaid	Recording of an air raid siren.
Vermont	An excerpt of the song "Vermont" by the band "The Districts" (singing, guitar, bass, and drums), taken from [9].	RaceCars	Recording of engine sounds of starting race cars.

Table 7.1: List of target and source recordings used in our experiments.

7.3.3 Audio Examples

In order to also give an auditory demonstration of our method, we set up an accompanying website for this chapter at [49]. On this website, one finds the target recordings as well as source recordings listed in Table 7.1. To ensure that each source recording offers an adequate pitch range, we computed several pitch-shifted versions of it (using a pitch-shifting algorithm from [40]) and concatenated them. For each pair of target and source, we then generated an audio mosaic using both the basic mosaicing procedure described in Section 7.1 as well as the procedure based on the extended set of update rules proposed in Section 7.2. For these experiments, we used music recordings sampled at 22050 Hz, an STFT frame length of 2048 samples and a hop size of 1024 samples to compute the spectrograms. In order to derive the activation matrices for both procedures, we performed $L = 20$ iterations of the respective update steps. For the extended set of update rules, we set the repetition restriction parameter to $r = 3$, the limit of simultaneous activations to $p = 10$, and the kernel parameter to $c = 3$. To reconstruct time-domain signals from the derived complex valued mosaic spectrograms, we finally performed 20 iterations of the STFT inversion procedure proposed in [84].

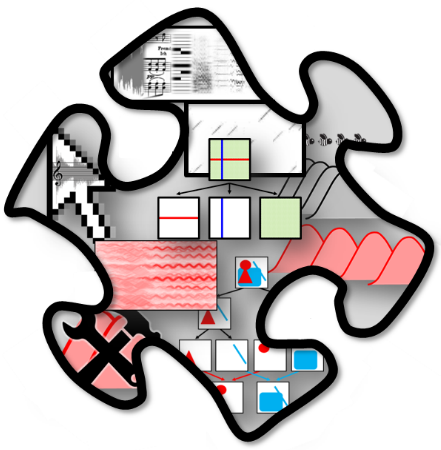
7.4 Conclusions and Further Notes

In this chapter, we presented a novel approach for automatically generating an audio mosaic of a target recording using the sounds from a source recording. The core idea of this NMF-inspired procedure was to learn an activation matrix that, when multiplied with the spectrogram of the source recording, yields the spectrogram of the mosaic recording. As our main technical contribution, we proposed an extended set of update rules that supports the development of sparse diagonal structures in the activation matrix during the learning process. Our experiments showed that these diagonal activation structures correspond to the activation of whole sequences of spectral frames and help to preserve timbral characteristics of the source in the mosaic.

In future work we want to investigate if our proposed procedure can also be applied in scenarios beyond audio mosaicing. One possibility is to examine whether supporting the development of diagonal structures in the activation matrix can also be beneficial when learning not only the activation matrix, but also the template matrix. Such an NMF procedure could be applied for learning and identifying repeating patterns in feature sequences, similar to [225] who used techniques based on NMFD for this task. In this context, we hope that our approach may yield a simpler implementation as well as more flexibility since the maximal length of sequences does not need to be fixed.

Chapter 8

Summary and Future Work



In this thesis, we explored the idea of processing music signals using audio decomposition techniques. Looking back, we saw various recurring key concepts which we now summarize.

A core principle which we applied in various settings throughout this thesis was to use decomposition techniques, such as our harmonic-percussive-residual decomposition procedure (Chapter 3), to split music signals into a set of mid-level components with an explicit interpretation. We were the able to use this information about the components as prior knowledge in subsequent processing tasks. For example, we cascaded various audio decomposition techniques in order to disassemble a music recording of singing voice and accompanying instruments into a set of semantically interpretable signal components (Chapter 5). These components could then easily be reassembled to form estimates of the isolated singing voice and accompaniment, since the components' explicit interpretation made it obvious whether a sound component was part of the singing voice or the accompaniment. We also harnessed the explicit interpretation of mid-level components to design our TSM procedures (Chapter 4) where we could apply specialized modification methods to a signal's harmonic and percussive sound components.

A guiding strategy in this thesis was to avoid explicit decisions at early stages in processing pipelines—instead, we aimed to handle problems rather implicitly. For example, our proposed TSM procedures (Chapter 4) avoided explicit transient detection in a given input signal—in contrast with other state-of-the-art TSM methods. Since when applying a harmonic-percussive decomposition technique, the signal's transients were typically contained in the percussive component, we could implicitly preserve them by modifying this component with OLA (see Section 4.1.2). Another example is our proposed singing voice extraction method (Chapter 5). In the context of this application, an important observation was that artifacts audible in the individual component signals were often perceptually masked in the reassembled source estimates—thus rendering attempts to reduce computational artifacts in the individual component signals unnecessary. Finally, our template-based vibrato parametrization approach (Section 6.2) followed the same strategy by avoiding the explicit extraction of an F0-trajectory prior to the actual vibrato analysis. For this application, however, it turned out that the robustness gained by avoiding the F0-estimation comes at the cost of efficiency. In future work, we therefore want to investigate whether combining ideas from our template-based approach with explicit F0-estimation could yield a trade-off between robustness and efficiency.

Throughout our work, it became evident that the very process of decomposing audio signals often gave us deeper insights into the music processing tasks at hand. For example, investigating our note-wise audio decomposition method's residual signal (Section 6.1) not only revealed shortcomings of our musical model (such as assuming that a small set of template vectors can sufficiently describe all onsets occurring in a music recording), but also enabled us to infer passages in a recording where a musician deviated from the given score—a valuable information that could be used in applications far beyond this decomposition scenario. A second example is the audio mosaicing procedure (Chapter 7) where applying the basic NMF-inspired method lead to perceptually unsatisfactory audio mosaics. However, investigating the learned activation matrices revealed that the reason for these poor mosaics were dense, horizontal activation structures. Due to this insight we could develop an extended set of update rules that supports the development of sparse diagonal activation structures. While these update rules greatly helped to improve the quality of audio mosaics, it remains as future work to investigate this concept's relationship to established decomposition techniques such as non-negative matrix factor deconvolution (NMFD) [197, 200] and to establish a theoretical foundation for them.

Throughout this thesis, we often exploited explicit musical knowledge in order to hand-craft solutions for specific problems. This informed approach had the advantage that we were often able to achieve good results without needing large amounts of training data. In contrast, a current trend in music processing is to use *deep learning*—a well established machine learning paradigm [108]—in order to learn relevant processing steps directly from large amounts of audio data. Deep learning has been successfully applied in a large variety of signal processing and music information retrieval scenarios such as singing voice extraction [105], downbeat tracking [51], chord recognition [196], singing voice activity detection [131], or tempo estimation [10]. An interesting research direction to address in future work is therefore to apply deep learning techniques in order to decompose music recordings into mid-level components as considered in this thesis.

Appendix A

TSM Toolbox

In the context of our publication [40] we released the TSM toolbox, which is freely available at the website [39] under a GNU-GPL license. This self-contained toolbox serves various purposes. First, it delivers basic tools to work in the field of TSM. The toolbox includes well-documented reference implementations for some of the most important classical TSM procedures within a unified framework. This not only allows users and researchers to get a better feeling for TSM results by experimenting with the algorithms, but also gives insights into implementation details and potential pitfalls. Second, to give an example of how those fundamental procedures can be combined to improve TSM results, the toolbox also supplies the code of our recently proposed TSM approach based on harmonic-percussive separation (see Section 4.2), as well as the code of the harmonic-percussive separation procedure itself. Third, the toolbox provides a MATLAB wrapper function for a commercial, proprietary, and widely used TSM algorithm. Because of its 'state-of-the-art' character, this is particularly interesting when conducting listening experiments which are the most common method for judging the perceptual quality of TSM results. Finally, the toolbox provides additional code for various example applications. Such

Filename	Main parameters	Optional parameters	Description
`wsolaTSM.m`	x, α	synHop$\hat{=}H_\mathrm{s}$, win$\hat{=}w$, tolerance$\hat{=}\Delta_\mathrm{max}$	Application of OLA & WSOLA.
`pvTSM.m`	x, α	synHop$\hat{=}H_\mathrm{s}$, win$\hat{=}w$, phaseLocking	Application of the phase vocoder (with or without identity phase locking).
`hpTSM.m`	x, α	hpsFilLenHarm$\hat{=}\ell_\mathrm{h}$, hpsFilLenPerc$\hat{=}\ell_\mathrm{p}$, pvSynHop, pvWin, olaSynHop, olaWin	Application of TSM based on HPSS.
`elastiqueTSM.m`	x, α	–	MATLAB wrapper for the *élastique* algorithm.
`win.m`	ℓ, β	–	Generates a $\sin^\beta$ window function of length ℓ.
`stft.m`	x	anaHop, win	Short-time Fourier transform of x.
`istft.m`	spec	synHop, win	Inversion of a short-time Fourier transform, see [84].
`hpSep.m`	x	filLenHarm$\hat{=}\ell_\mathrm{h}$, filLenPerc$\hat{=}\ell_\mathrm{p}$	Harmonic-percussive source separation.
`pitchShiftViaTSM.m`	x, n	algTSM	Pitch-shifting the signal x by n cents.
`visualizeWav.m`	x	fsAudio, timeRange	Visualization of TSM results.
`visualizeSpec.m`	spec	fAxis, tAxis, logComp	Visualization of a short-time Fourier transform.
`visualizeAP.m`	anchorpoints	fsAudio	Visualization of a set of anchorpoints.

Table A.1: Overview of the main MATLAB functions contained in the TSM toolbox [39] and the most important parameters.

applications include the automated generation of interfaces for comparing TSM results, the non-linear synchronization of audio recordings, and the pitch-shifting of audio signals. Although there already exist MATLAB implementations of individual TSM algorithms (for example [83, 57]), we believe that supplying an entire collection of different TSM approaches along with example applications within a unifying framework can be highly beneficial for both researchers as well as educators in the field of audio processing.

A.1 Code Example

The TSM procedures discussed in Chapter 4 (except for Int-PV-TSM and CascadeTSM) form the core of our toolbox. Table A.1 gives an overview of the main MATLAB functions along with the most important parameters. Note that there are many more parameters and additional functions not discussed here. However, for all parameters there are default settings such that none of the parameters need to be specified by the user.

To demonstrate how the TSM procedures contained in the TSM toolbox can be applied, we now discuss the code example shown in Table A.2, which is also contained in the toolbox as script `demoTSMtoolbox.m`. Our example starts in lines 1-4 with specifying an audio signal as well

```
 1 filename = 'CastanetsViolin.wav';
 2 alpha = 1.8;
 3
 4 [x,sr] = wavread(filename);
 5
 6 paramOLA.tolerance = 0;
 7 paramOLA.synHop = 128;
 8 len = 256; beta = 2;
 9 paramOLA.win = win(len,beta);
10 yOLA = wsolaTSM(x,alpha,paramOLA);
11
12 paramWSOLA.tolerance = 512;
13 paramWSOLA.synHop = 512;
14 len = 1024; beta = 2;
15 paramWSOLA.win = win(len,beta);
16 yWSOLA = wsolaTSM(x,alpha,paramWSOLA);
17
18 paramPV.phaseLocking = 0;
19 paramPV.synHop = 512;
20 len = 2048; beta = 1;
21 paramPV.win = win(len,beta);
22 yPV = pvTSM(x,alpha,paramPV);
23
24 paramPVpl.phaseLocking = 1;
25 paramPVpl.synHop = 512;
26 len = 2048; beta = 1;
27 paramPVpl.win = win(len,beta);
28 yPVpl = pvTSM(x,alpha,paramPVpl);
29
30 paramHP.hpsFilLenHarm = 10;
31 paramHP.hpsFilLenPerc = 10;
32 paramHP.pvSynHop = 512;
33 len = 2048; beta = 1;
34 paramHP.pvWin = win(len,beta);
35 paramHP.olaSynHop = 128;
36 len = 256; beta = 2;
37 paramHP.olaWin = win(len,beta);
38 yHP = hpTSM(x,alpha,paramHP);
39
40 % To execute elastique, you will need
41 % an access id from http://www.sonicapi.com.
42 % Furthermore, you need to download 'curl'
43 % from http://curl.haxx.se/download.html.
44 % yELAST = elastiqueTSM(x,alpha);
45
46 paramVis.timeRange = [5.1 5.3];
47 visualizeWav(x,paramVis);
48 paramVis.timeRange = [5.1 5.3] * alpha;
49 visualizeWav(yOLA,paramVis);
50 wavwrite(yOLA,sr,'Output_OLA.wav');
```

Table A.2: Code example for computing TSM results of various TSM algorithms, generating the visualizations, and writing the TSM results to the hard disk.

as a stretching factor α. Furthermore, the audio signal is loaded from the hard disk using the MATLAB function `wavread` and stored in the variable `x` while its sampling rate is stored in `sr`.

The first TSM procedure which is applied to the loaded signal is OLA in lines 6-10. Since OLA is a special case of WSOLA, this is done by calling the `wsolaTSM.m` function with a specialized set of parameters. In line 6, the analysis frame position tolerance $\Delta_{\max}$ of WSOLA is set to 0, turning WSOLA into OLA. Afterwards, the synthesis hopsize H_s is set to 128 samples in line 7. In lines 8 and 9, a $\sin^\beta$-window of length $N = 256$ samples and $\beta = 2$ (effectively a Hann window as defined in Equation (4.3)) is generated by calling `win.m`. The size of the generated window specifies at the same time the size of the analysis and synthesis frames. Together with the synthesis hopsize of 128 samples, this means that in the output of the TSM procedure the synthesis frames will have a half-overlap of 128 samples. Finally the actual TSM procedure is applied to the input signal x with the stretching factor α and the specified set of parameters in line 10. The resulting waveform is stored in the variable `yOLA`.

Next, in lines 12-16, the WSOLA algorithm is applied. We first set the analysis frame position tolerance $\Delta_{\max}$ to 512 in line 12. Since WSOLA works optimally for medium sized frames which are half-overlapped, we set the synthesis hopsize H_s to 512 in line 13 and chose a $\sin^\beta$-window of length $N = 1024$ samples and $\beta = 2$ in lines 14 and 15. Finally, the function `wsolaTSM.m` is called in line 16.

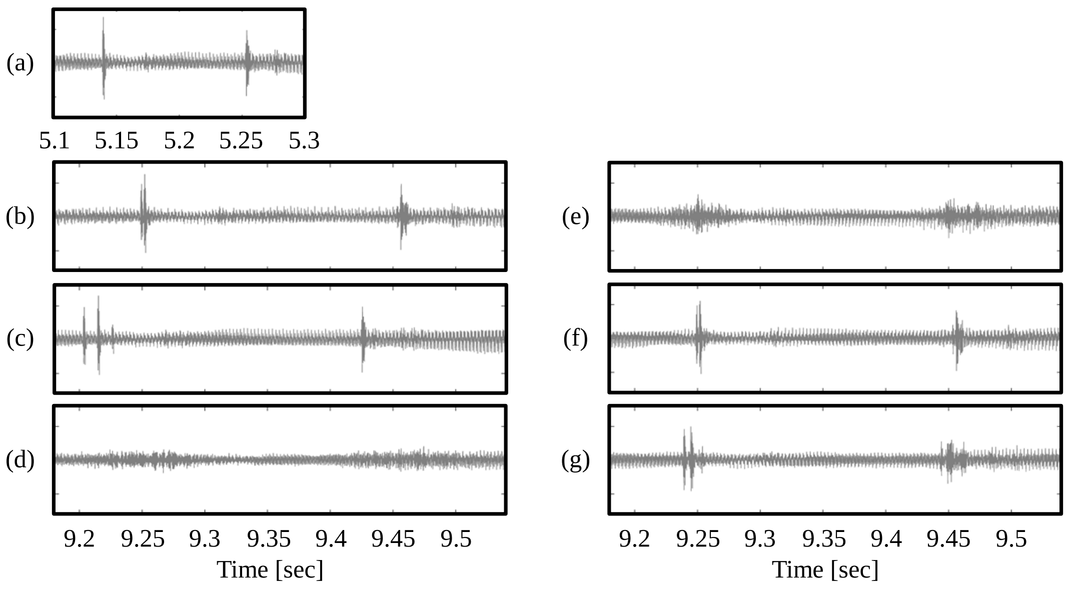
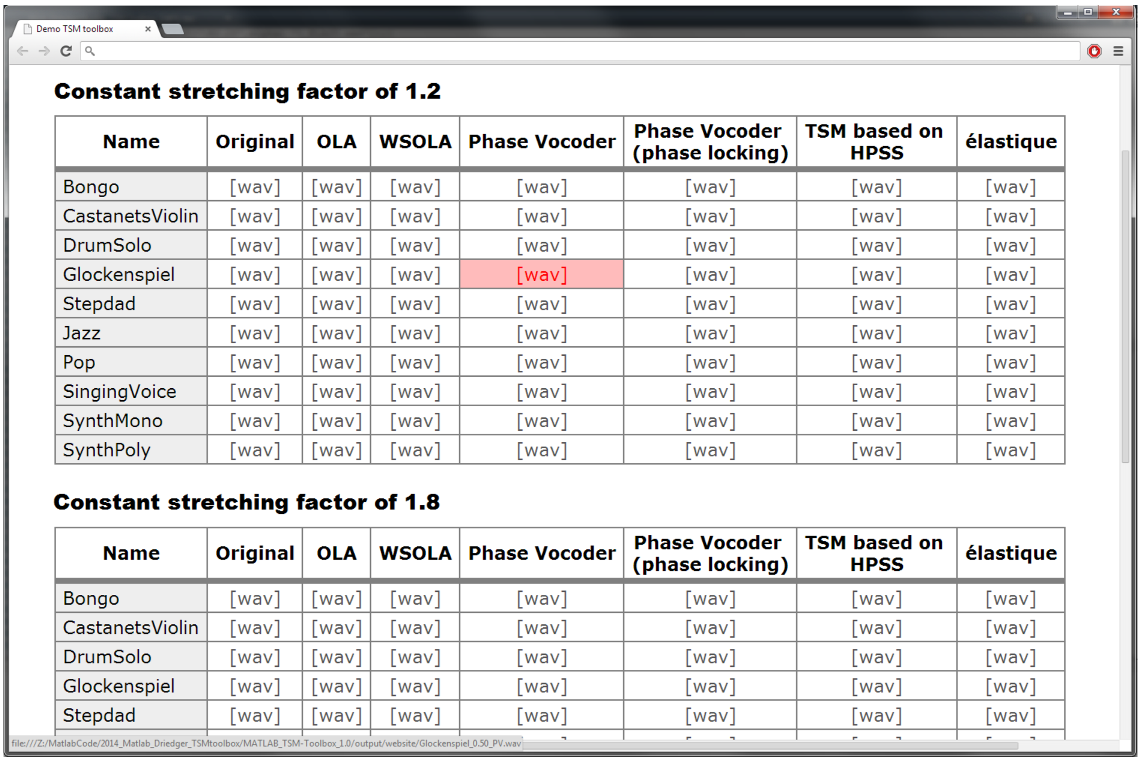


Figure A.1: TSM results of different algorithms for an audio recording of a violin and castanets. **(a)**: Original waveform. **(b)**: OLA. **(c)**: WSOLA. **(d)**: PV-TSM. **(e)**: PV-TSM with identity phase locking. **(f)**: TSM based on harmonic-percussive separation. **(g)**: TSM based on the commercial *élastique* algorithm.

In lines 18-22 the standard phase vocoder PV-TSM is applied by a call of `pvTSM.m`. To this end, we first specify that no phase locking should be applied (line 18). Being a frequency-domain TSM algorithm, the phase vocoder is dependent on a high frequency resolution of the used Fourier transform and therefore on a large frame size. Furthermore, also a large overlap of the synthesis frames is beneficial for the quality of the output signal as well as a sin-window function. We therefore set the synthesis hopsize H_s to 512 (line 19) and chose a $\sin^\beta$-window of length $N = 2048$ samples and $\beta = 1$ (lines 20 and 21), resulting in a 75% frame overlap. The actual function call is then executed in line 22. For the application of the phase vocoder with identity phase locking in lines 24-28, the only difference is the `phaseLocking` parameter set to one (line 24).

The TSM algorithm based on harmonic-percussive separation, which is applied in lines 30-38, is a combination of multiple techniques. First, we set the length of the median filters ℓ_h and ℓ_p used in the separation procedure both to 10 (lines 30 and 31). Then, the synthesis hopsizes and windows, which are used in the two TSM algorithms OLA and phase vocoder with identity phase locking, are set separately in lines 32-37. In line 38 the algorithm is then executed by a call of `hpTSM.m`.

The last TSM algorithm is the MATLAB wrapper for *élastique* [231]. Since this function requires a *sonicAPI* access id as well as the additional tool `curl`, the function call in line 44 is commented out by default. However, when supplying the additional sources the algorithm can be applied by



Constant stretching factor of 1.2

Name	Original	OLA	WSOLA	Phase Vocoder	Phase Vocoder (phase locking)	TSM based on HPSS	élastique
Bongo	[wav]	[wav]	[wav]	[wav]	[wav]	[wav]	[wav]
CastanetsViolin	[wav]	[wav]	[wav]	[wav]	[wav]	[wav]	[wav]
DrumSolo	[wav]	[wav]	[wav]	[wav]	[wav]	[wav]	[wav]
Glockenspiel	[wav]	[wav]	[wav]	[wav]	[wav]	[wav]	[wav]
Stepdad	[wav]	[wav]	[wav]	[wav]	[wav]	[wav]	[wav]
Jazz	[wav]	[wav]	[wav]	[wav]	[wav]	[wav]	[wav]
Pop	[wav]	[wav]	[wav]	[wav]	[wav]	[wav]	[wav]
SingingVoice	[wav]	[wav]	[wav]	[wav]	[wav]	[wav]	[wav]
SynthMono	[wav]	[wav]	[wav]	[wav]	[wav]	[wav]	[wav]
SynthPoly	[wav]	[wav]	[wav]	[wav]	[wav]	[wav]	[wav]

Constant stretching factor of 1.8

Name	Original	OLA	WSOLA	Phase Vocoder	Phase Vocoder (phase locking)	TSM based on HPSS	élastique
Bongo	[wav]	[wav]	[wav]	[wav]	[wav]	[wav]	[wav]
CastanetsViolin	[wav]	[wav]	[wav]	[wav]	[wav]	[wav]	[wav]
DrumSolo	[wav]	[wav]	[wav]	[wav]	[wav]	[wav]	[wav]
Glockenspiel	[wav]	[wav]	[wav]	[wav]	[wav]	[wav]	[wav]
Stepdad	[wav]	[wav]	[wav]	[wav]	[wav]	[wav]	[wav]

file:///Z:/MatlabCode/2014_Matlab_Driedger_TSMtoolbox/MATLAB_TSM-Toolbox_1.0/output/website/Glockenspiel_0.50_PV.wav

Figure A.2: Screenshot of the interface generated using the function `demoGenerateTSMwebsite.m` of the TSM toolbox.

a call to `elastiqueTSM.m`. Since *élastique* is a proprietary procedure it is not possible to tweak the algorithm with additional parameters.

In lines 46 and 47, the visualization of the original input signal takes place. First, the segment of the input audio signal to be visualized is set to the section of the waveform between second 5.1 and 5.3 (line 46). Afterwards the visualization function `visualizeWav.m` is applied to `x` in line 47. To visualize the corresponding stretched audio segment, the segments boundaries are just multiplied with the stretching factor α in line 48. Afterwards, the visualization function is called again exemplarily for OLA's TSM result in line 49. Finally, in line 50, the TSM result of OLA is also written to the hard disk using the MATLAB function `wavwrite`. The visualizations generated by this demo script can be seen in Figure A.1.

A.2 Additional Functionality and Content

In addition to pitch-shifting functionality (`demoPitchShift.m`) and the possibility to perform non-linear TSM (`demoNonlinearTSM.m`) as discussed in Sections 4.4.2 and 4.4.1, respectively, the TSM toolbox also allows to automatically generate interfaces that can be used in the context of

Item name	**Description**
Bongo	Regular beat played on bongos.
CastanetsViolin	Solo violin overlayed with castanets.
DrumSolo	A solo performed on a drum set.
Glockenspiel	Monophonic melody played on a glockenspiel.
Jazz	Synthetic polyphonic sound mixture of a trumpet, a piano, a bass and drums.
Pop	Synthetic polyphonic sound mixture of several synthesizers, a guitar and drums.
SingingVoice	Solo male singing voice.
Stepdad	Excerpt from *My Leather, My Fur, My Nails* by the band *Stepdad*.
SynthMono	Monophonic synthesizer with a very noisy and distorted sound.
SynthPoly	Sound mixture of several polyphonic synthesizers.

Table A.3: List of audio items included in the TSM toolbox.

listening experiments. Listening experiments are the most common way of comparing the quality of different TSM algorithms. To this end, one usually generates time-stretched versions of several audio items using different TSM procedures and stretching factors. This results in large amounts of audio data. To be able to compare the generated TSM results, interfaces which allow a user to order and access the audio signals in a convenient way are of great help. With the script `demoGenerateTSMwebsite.m`, the toolbox provides the code for generating such a HTML-based interface automatically (see FigureA.2). The toolbox also includes the set of audio items listed in Table A.3, which has been already used for evaluation purposes in the context of TSM in [48, 47].

Appendix B

User Interfaces for Music Signal Analysis and Editing

In this appendix, we present prototypes of MATLAB user interfaces that we developed in the context of this thesis. These interfaces serve as demonstrators for some of our proposed techniques. In Appendix B.1, we present a tool for the interactive estimation of fundamental frequency trajectories that we originally proposed in [43]. Then, in Appendix B.2, we discuss the interface for score-based music editing based on our publication [37]. Finally, in Appendix B.3, we present an interface that focuses on the modification of singing voice in a music recording, based on our singing voice extraction procedure proposed in [41].

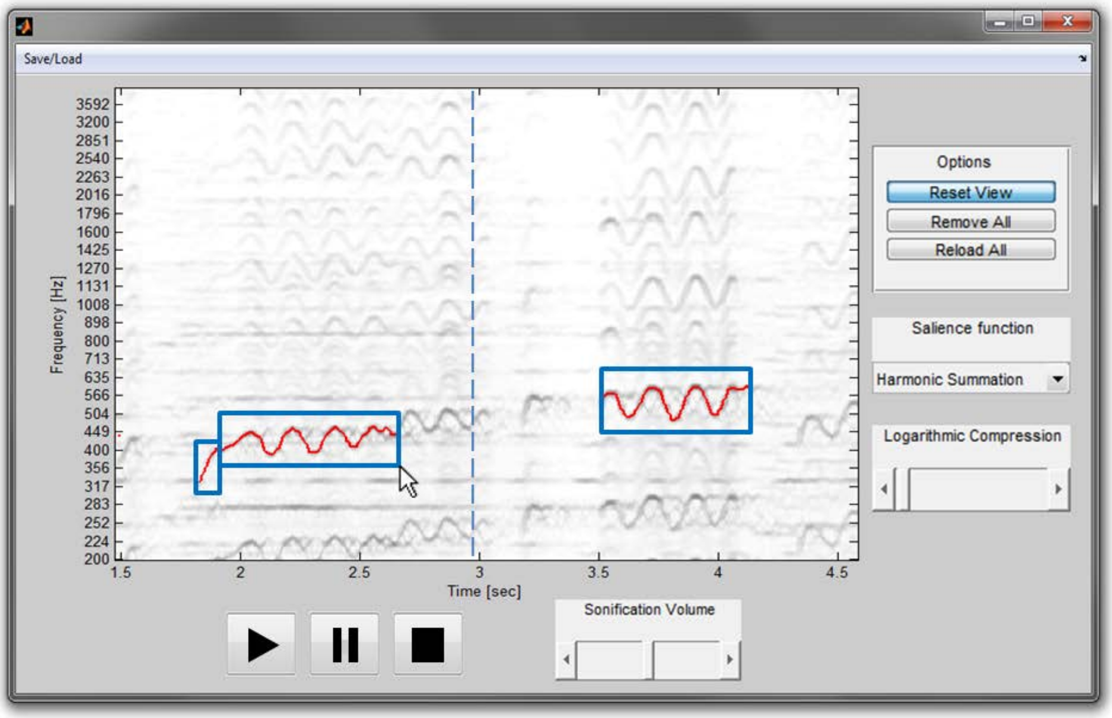


Figure B.1: Screenshot of the *F0-Annotation Tool*.

B.1 F0-Annotation Tool

As noted in Section 2.4.2, the automated estimation of melody- or predominant F0-trajectories from complex music recordings is a difficult task that is far from being solved, see also [184, 183]. To this end, when the goal is to generate trajectories of high quality, it is helpful to integrate a user in the estimation process. In [43] we proposed a user interface that, based on the salience spectrogram introduced in Section 2.4.2, allows a user to estimate a melody trajectory interactively. Figure B.1 shows a screenshot of our interface. In addition to standard audio player functionalities (see the buttons for starting, pausing, and stopping the playback of the loaded music recording at the bottom of the interface), the salience spectrogram is our interface's central element. When playing back the music recording, the respective time position is indicated by a vertical dashed playback bar in the salience representation. This way, salient structures in the visual representation can be directly compared to the auditory cues in the recording. Now, a user can indicate the rough location of partial melody trajectories in the salience spectrogram by drawing *constraint regions*—the blue rectangles in the screenshot. Our tool then automatically uses techniques based on *dynamic time warping* (DTW) [173, 145, 146] to find a plausible trajectory through the specified region of the salience spectrogram—visualized by the red trajectory inside of the constraint regions. Now, when playing back the recording again, the estimated trajectory is sonified using a sinusoidal synthesizer. This way, the user can check the correctness of the current trajectory by listening. Previously drawn constraint regions can then be edited to correct errors in the melody trajectory. To account for extremely fine-grained corrections, the user can even use a *draw* option to draw the trajectory free hand. The user can also influence the salience spectrogram itself—for example by applying a logarithmic compression—to visually enhance interesting structures and therefore to simplify the tracking process. Finally, it is possible to save the current state of the interactive melody estimation at any given time to either use the trajectory as it currently is or to resume the interactive estimation at a later stage.

There exist various other software programs—both commercial and academic—that can be used for the interactive computation of melody trajectories. *Melodyne* is a product by the company *Celemony* for the decomposition of music recordings into note-like audio events (so-called *Blobs*) [17]. In the decomposition process, the software computes a melody trajectory for each of those blobs. The decomposition itself can be influenced by changing the parameters that are used for the signal analysis or by providing prior information about the musical piece such as its key. This also influences the derived melody trajectories. However, these trajectories serve only visualization purposes and cannot be exported.

Other examples of tools for the interactive derivation of melody trajectories are *Tony* by Mauch et al. [141] or the interface by Pant et al. [162]. After having analyzed a given music recording, these programs offer a choice of different melody trajectories. A user can then select the trajectory that matches the recording best in his or her opinion. This approach is very time-efficient since the set of possible melody trajectories is rather limited and it is therefore not necessary for the user to verify the correctness of the trajectory on a fine-grained level. However, especially when dealing with complex music recordings that are highly polyphonic, it may happen that none of the offered melody trajectories appropriately reflects the recording's actual melody. Using our proposed tool is in general more time-consuming since a user needs to estimate a melody trajectory from scratch by manually defining constraint regions. Especially for complex music recordings it is very likely that the number of constraint regions that are necessary to yield an appropriate melody trajectory is high. On the other hand, this approach allows a user to generate melody trajectories of high quality, even for polyphonic recordings.

Another software program that is very popular in academia is *Praat* [11]. This tool was particularly developed for the phonetic analysis of speech and also offers the possibility to estimate F0-trajectories. A user can influence the estimation process, for example by specifying the trajectory's expected frequency range. However, Praat does not seem to be suitable for the analysis of complex music recordings as the underlying F0 estimation procedure is particularly designed for monophonic recordings (this task is commonly termed as *monopitch estimation*, see [97, 98]).

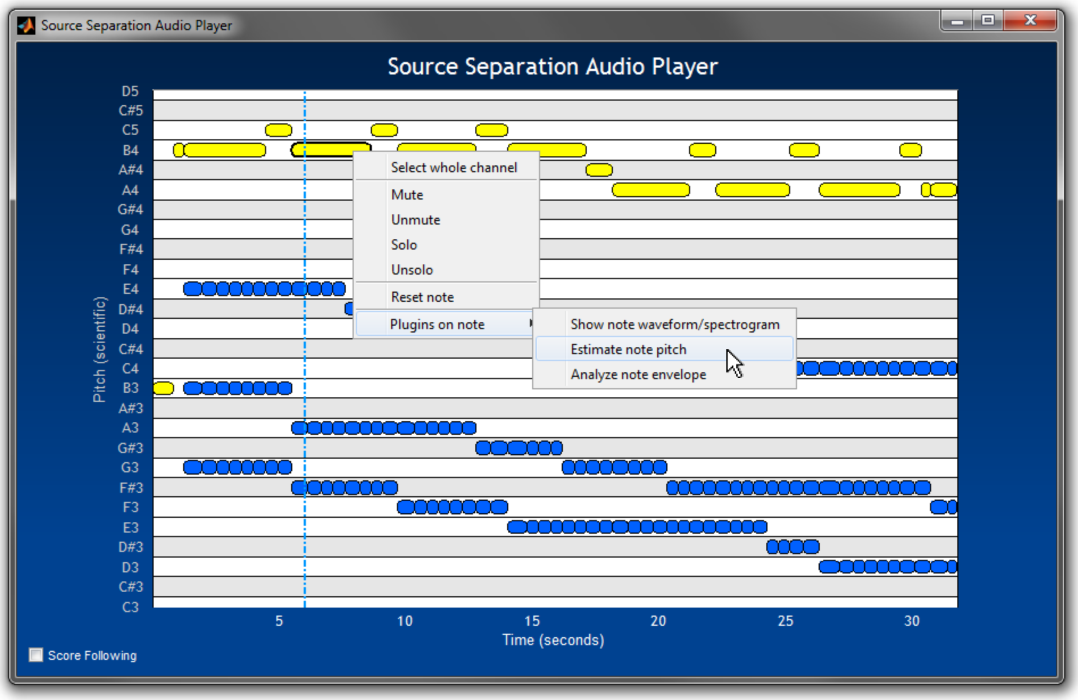


Figure B.2: Screenshot of the *Source Separation Audio Player*.

B.2 Source Separation Audio Player

In the course of our work for [37], we developed a MATLAB user interface which offers a user-friendly access to note-wise audio decompositions as introduced in Section 6.1. Figure B.2 shows a screenshot of this interface. Besides standard audio player functionalities, it is equipped with a set of additional tools.

In its current state, our interface provides a piano roll representation of the musical score. While playing back the audio recording, the synchronized MIDI events are displayed and allow a user to directly access and manipulate every single corresponding note-wise audio event. By dragging a note object in the piano roll representation and dropping it at a different position, the note event's onset time (horizontal displacement) as well as its pitch (vertical displacement) can be manipulated in an intuitive way. Furthermore, it is also possible to change the duration or the volume level of note events, to remove them completely from the audio recording, or to add additional events by copying and manipulating existing ones. This allows for a music editing workflow that is similar to *Melodyne* [17], a commercial music editing tool by *Celemony* that we already discussed in Appendix B.1. However, unlike the *blobs* in Melodyne that only roughly correspond to individual notes, the note events displayed by our interface exactly represent the notes as they occur in the musical score (reflected by the MIDI).

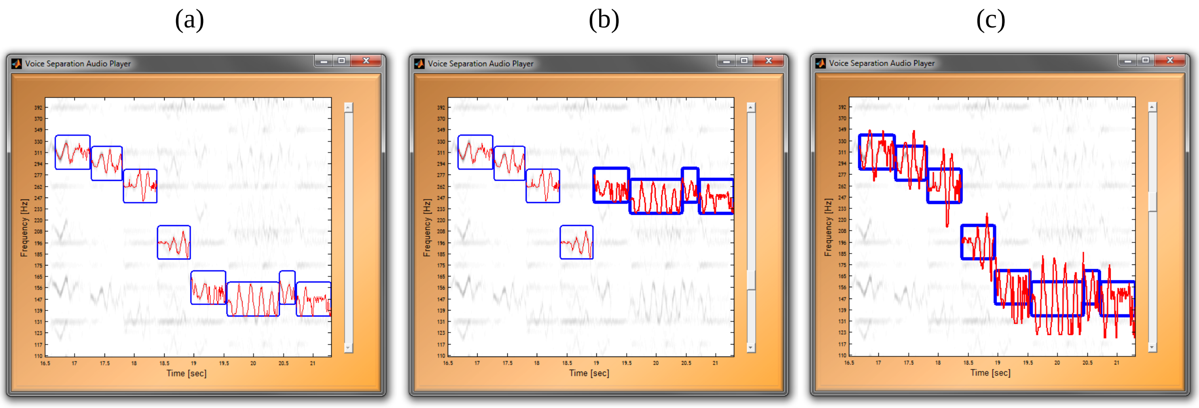


Figure B.3: Screenshots of the *Voice Separation Audio Player*. **(a):** The standard interface. **(a):** Pitch-shifted singing voice. **(a):** Singing voice with amplified vibrato.

Additionally, the interface also provides a set of plugins that can be used to analyze and visualize the note-wise audio events, see Figure B.2. To this end it is, for example, possible to analyze the residual signal x_{res}, as defined in Equation (6.8), directly in the interface. A demo of the proposed interface can be found at [36].

B.3 Voice Separation Audio Player

In this section, we present the *Voice Separation Audio Player*, an interface that combines several techniques discussed in this thesis. Similar as the Source Separation Audio Player presented in Appendix B.2, this interface offers standard audio player functionalities to start, pause, and stop the playback of a loaded music recording. Additionally, it focuses on providing tools that allow a user to edit the singing voice in the music recording independently of the musical accompaniment.

As in the F0-Annotation Tool (Appendix B.1), the interface's main element is a salience spectrogram representation (see Section 2.4.2) of the loaded music signal. Given a MIDI file that was synchronized with the recording, the MIDI events related to the singing voice are used to derive constraint regions in the salience representation (blue boxes in Figure B.3a). The singing voice's F0-trajectory is then automatically derived inside of these constraint regions, using dynamic time warping techniques [173, 145, 146] (red trajectories in Figure B.3a). We then apply our singing voice extraction method that we presented in Chapter 5 to separate the singing voice from the musical accompaniment, utilizing the automatically derived F0-trajectory. The singing voice signal is finally temporally segmented according to the MIDI note events. Each of the blue rectangles in the interface's visualization as shown in Figure B.3a therefore represents a singing voice audio event that corresponds to the respective MIDI note event.

This representation allows a user to manipulate and edit the singing voice in the recording in an intuitive way. For example, marking several note events and moving them vertically indicates that the user would like to change these notes' pitches, see Figure B.3b. We then use the formant-preserving pitch shifting method as discussed in Section 4.4.2 to apply the indicated pitch modification to the singing voice audio events. The formant preservation significantly improves the quality of the resulting edited music recording since it allows to avoid the chipmunk effect [226].

Another tool provided by our interface allows a user to manipulate vibrato in the singing voice. After having selected individual notes, the slider on the right side of the interface can be used to either amplify or decrease the extent of vibrato present in the singing voice, see Figure B.3c. Again, the desired modifications are then applied to the respective singing voice audio events by utilizing pitch shifting techniques.

Bibliography

[1] Toshihiko Abe, Takao Kobayashi, and Satoshi Imai. Robust pitch estimation with harmonics enhancement in noisy environments based on instantaneous frequency. In *Proceedings of the Fourth International Conference on Spoken Language Processing ICSLP*, volume 2, pages 1277–1280, Philadelphia, Pennsylvania, USA, October 1996.

[2] Jakob Abeßer, Hanna M. Lukashevich, and Gerald Schuller. Feature-based extraction of plucking and expression styles of the electric bass guitar. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 2290–2293, Dallas, Texas, USA, March 2010.

[3] Jakob Abeßer, Martin Pfleiderer, Klaus Frieler, and Wolf-Georg Zaddach. Score-informed tracking and contextual analysis of fundamental frequency contours in trumpet and saxophone jazz solos. In *Proceedings of the 17th International Conference on Digital Audio Effects (DAFx-14)*, pages 181–186, Erlangen, Germany, 2014.

[4] Shoko Araki, Francesco Nesta, Emmanuel Vincent, Zbynek Koldovský, Guido Nolte, Andreas Ziehe, and Alexis Benichoux. The 2011 signal separation evaluation campaign (SiSEC2011): Audio source separation. In *LVA/ICA*, pages 414–422, 2012.

[5] Mark A. Bartsch and Gregory H. Wakefield. Audio thumbnailing of popular music using chroma-based representations. *IEEE Transactions on Multimedia*, 7(1):96–104, 2005.

[6] Juan Pablo Bello, Laurent Daudet, Samer Abdallah, Chris Duxbury, Mike Davies, and Mark Sandler. A tutorial on onset detection in music signals. *IEEE Transactions on Speech and Audio Processing*, 13(5):1035–1047, 2005.

[7] Gilberto Bernardes. *Composing Music by Selection: Content-Based Algorithmic-Assisted Audio Composition.* PhD thesis, Faculty of Engineering, University of Porto, 2014.

[8] Josef Bigun and Gösta H. Granlund. Optimal orientation detection of linear symmetry. In *Proceedings of the IEEE First International Conference on Computer Vision*, pages 433—438, London, UK, 1987.

[9] 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, October 2014.

[10] Sebastian Böck, Florian Krebs, and Gerhard Widmer. Accurate tempo estimation based on recurrent neural networks and resonating comb filters. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, pages 625–631, Malaga, Spain, 2015.

[11] Paul Boersma. Praat, a system for doing phonetics by computer. *Glot International*, 5(9/10):341–345, 2001.

[12] Breakfast Quay. Rubber band library. `http://breakfastquay.com/rubberband/`. Web resource, last consulted in November 2015.

[13] Judith C. Brown and Miller S. Puckette. An efficient algorithm for the calculation of a constant Q transform. *Journal of the Acoustic Society of America (JASA)*, 92:2698–2701, 1992.

[14] Francisco J. Cañadas-Quesada, Pedro Vera-Candeas, Nicolás Ruiz-Reyes, Julio J. Carabias-Orti, and Pablo Cabañas Molero. Percussive/harmonic sound separation by non-negative matrix factorization with smoothness/sparseness constraints. *EURASIP Journal on Audio, Speech and Music Processing*, 26, 2014.

[15] Emmanuel J. Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *Journal of the ACM*, 58(3):11:1–11:37, June 2011.

[16] Michael A. Casey and Alex Westner. Separation of mixed audio sources by independent subspace analysis. In *Proceedings of the International Computer Music Conference*, pages 154–161, Berlin, Germany, 2000.

[17] Celemony. Melodyne. `www.celemony.com/de/melodyne/what-is-melodyne`. Web resource, last consulted in January 2016.

[18] Collin E. Cherry. Some experiments on the recognition of speech, with one and with two ears. *Journal of the Acoustic Society of America (JASA)*, 24:975–979, 1953.

[19] Dave Cliff. Hang the DJ: Automatic sequencing and seamless mixing of dance-music tracks. Technical report, HP Laboratories Bristol, 2000.

[20] Graham Coleman, Esteban Maestre, and Jordi Bonada. Augmenting sound mosaicing with descriptor-driven transformation. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, Graz, Austria, 2010.

[21] Nick Collins. A comparison of sound onset detection algorithms with emphasis on psychoacoustically motivated detection functions. In *AES Convention 118*, Barcelona, Spain, 2005.

[22] Nick Collins. Using a pitch detector for onset detection. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 100–106, London, UK, 2005.

[23] Josep M. Comajuncosas, Alex Barrachina, John O'Connell, and Enric Guaus. Nuvolet: 3D gesture-driven collaborative audio mosaicing. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 252–255, Oslo, Norway, 2011.

[24] Edward Costello, Victor Lazzarini, and Joseph Timoney. A streaming audio mosaicing vocoder implementation. In *Proceedings of the 16th International Conference on Digital Audio Effects (DAFx)*, Maynooth, Ireland, September 2013.

[25] Norberto Degara, Antonio Pena, and Soledad Torres-Guijarro. A comparison of score-level fusion rules for onset detection in music signals. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 117–122, Kobe, Japan, 2009.

[26] Christian Dittmar, Jonathan Driedger, and Meinard Müller. A separate and restore approach to score-informed music decomposition. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, New Paltz, NY, USA, October 2015.

[27] Simon Dixon. Onset detection revisited. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, pages 133–137, Montreal, Quebec, Canada, 2006.

[28] Mark Dolson. The phase vocoder: A tutorial. *Computer Music Journal*, 10(4):14–27, 1986.

[29] David Dorran. *Audio Time-Scale Modification.* PhD thesis, School of Control Systems and Electrical Engineering, Dublin Institute of Technology, 2005.

[30] David Dorran, Eugene Coyle, and Robert Lawlor. Audio time-scale modification using a hybrid time-frequency domain approach. In *Proceedings of the Workshop on Applications of Signal Processing (WASPAA)*, New Paltz, New York, USA, 2005.

[31] David Dorran, Robert Lawlor, and Eugene Coyle. A hybrid time-frequency domain approach to audio time-scale modification. *Journal of the Audio Engineering Society*, 54(1/2):21–31, 2006.

[32] J. Stephen Downie. Music information retrieval. *Annual Review of Information Science and Technology (Chapter 7)*, 37:295–340, 2003.

[33] Karin Dressler. Sinusoidal extraction using an efficient implementation of a multi-resolution FFT. *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, pages 18–20, 2006.

[34] Jonathan Driedger, Stefan Balke, Sebastian Ewert, and Meinard Müller. Accompanying website: Towards template-based vibrato analysis in complex music signals. `http://www.audiolabs-erlangen.de/resources/MIR/2016-ISMIR-Vibrato/`.

[35] Jonathan Driedger, Stefan Balke, Sebastian Ewert, and Meinard Müller. Towards template-based vibrato analysis in complex music signals, *submitted to ISMIR 2016.* NY, USA.

[36] Jonathan Driedger, Harald Grohganz, Thomas Prätzlich, Sebastian Ewert, and Meinard Müller. Accompanying website: Score-informed audio decomposition and applications. `www.audiolabs-erlangen.de/resources/2013-ACMMM-AudioDecomp/`.

[37] 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.

[38] Jonathan Driedger and Meinard Müller. Accompanying website: Extracting singing voice from music recordings by cascading audio decomposition techniques. `http://www.audiolabs-erlangen.de/resources/MIR/2015-ICASSP-SVECD`.

[39] Jonathan Driedger and Meinard Müller. TSM toolbox. `http://www.audiolabs-erlangen.de/resources/MIR/TSMtoolbox/`.

[40] 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.

[41] Jonathan Driedger and Meinard Müller. Extracting singing voice from music recordings by cascading audio decomposition techniques. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 126–130, Brisbane, Australia, 2015.

[42] Jonathan Driedger and Meinard Müller. Harmonisch-Perkussiv-Rest Zerlegung von Musiksignalen. In *Proceedings of the Deutsche Jahrestagung für Akustik (DAGA)*, pages 1421–1424, Nuremberg, Germany, 2015.

[43] Jonathan Driedger and Meinard Müller. Verfahren zur Schätzung der Grundfrequenzverläufe von Melodiestimmen in mehrstimmigen Musikaufnahmen. In Wolfgang Auhagen, Claudia Bullerjahn, and Richard von Georgi, editors, *Musikpsychologie – Anwendungsorientierte Forschung*, volume 25 of *Jahrbuch Musikpsychologie*, pages 55–71. Hogrefe-Verlag, 2015.

[44] Jonathan Driedger and Meinard Müller. A review on time-scale modification of music signals. *Applied Sciences*, 6(2):57–82, February 2016.

[45] Jonathan Driedger, Meinard Müller, and Sascha Disch. Accompanying website: Extending harmonic-percussive separation of audio signals. `http://www.audiolabs-erlangen.de/resources/2014-ISMIR-ExtHPSep/`.

[46] Jonathan Driedger, Meinard Müller, and Sascha Disch. Extending harmonic-percussive separation of audio signals. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 611–616, Taipei, Taiwan, October 2014.

[47] Jonathan Driedger, Meinard Müller, and Sebastian Ewert. Accompanying website: Improving time-scale modification of music signals using harmonic-percussive separation. `http://www.audiolabs-erlangen.de/resources/2014-SPL-HPTSM/`. Web resource, last consulted in March 2014.

[48] Jonathan Driedger, Meinard Müller, and Sebastian Ewert. Improving time-scale modification of music signals using harmonic-percussive separation. *IEEE Signal Processing Letters*, 21(1):105–109, 2014.

[49] Jonathan Driedger, Thomas Prätzlich, and Meinard Müller. Accompanying website: Let it bee – towards NMF-inspired audio mosaicing. `http://www.audiolabs-erlangen.de/resources/MIR/2015-ISMIR-LetItBee/`.

[50] 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, Malaga, Spain, 2015.

[51] Simon Durand, Juan Pablo Bello, Bertrand David, and Gaël Richard. Downbeat tracking with multiple features and deep neural networks. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 409–413, Brisbane, Australia, April 2015.

[52] Jean-Louis Durrieu, Bertrand David, and Gaël Richard. A musically motivated mid-level representation for pitch estimation and musical audio source separation. *IEEE Journal of Selected Topics in Signal Processing*, 5(6):1180–1191, 2011.

[53] Jean-Louis Durrieu and Jean-Philippe Thiran. Musical audio source separation based on user-selected F0 track. In *Proceedings of the International Conference on Latent Variable Analysis and Signal Separation (LVA/ICA)*, pages 438–445, 2012.

[54] Chris Duxbury, Mike Davies, and Mark Sandler. Separation of transient information in audio using multiresolution analysis techniques. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, Limerick, Ireland, 2001.

[55] Chris Duxbury, Mike Davies, and Mark Sandler. Improved time-scaling of musical audio using phase locking at transients. In *Proceedings of the Audio Engineering Society (AES) Convention*, Munich, Germany, May 2002. Preprint 5530.

[56] Julian Eggert and Edgar Körner. Sparse coding and NMF. In *Proceedings of the IEEE International Joint Conference on Neural Networks*, volume 4, pages 2529–2533, July 2004.

[57] Daniel P. W. Ellis. A phase vocoder in Matlab. `http://www.ee.columbia.edu/~dpwe/resources/matlab/pvoc/`, 2002. Web resource, last consulted in February 2014.

[58] Daniel P.W. Ellis. Chroma feature analysis and synthesis. `www.ee.columbia.edu/~dpwe/resources/matlab/chroma-ansyn/`. Web resource, last consulted in January 2016.

[59] Daniel P.W. Ellis. Spectrograms: Constant-q (log-frequency) and conventional (linear). `www.ee.columbia.edu/ln/rosa/matlab/sgram/`. Web resource, last consulted in January 2016.

[60] Daniel P.W. Ellis and Graham E. Poliner. Identifying 'cover songs' with chroma features and dynamic programming beat tracking. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 4, Honolulu, Hawaii, USA, 2007.

[61] 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.

[62] Antti J. Eronen and Anssi P. Klapuri. Music tempo estimation with k-NN regression. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(1):50–57, 2010.

[63] 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, March 2012.

[64] 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.

[65] 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.

[66] Florian Eyben, Sebastian Böck, Björn Schuller, and Alex Graves. Universal onset detection with bidirectional long short-term memory neural networks. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 589–594, Utrecht, The Netherlands, 2010.

[67] Derry FitzGerald. Harmonic/percussive separation using median filtering. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, pages 246–253, Graz, Austria, 2010.

[68] Derry FitzGerald. Vocal separation using nearest neighbours and median filtering. In *Irish Signals and Systems Conference (ISSC 2012)*, pages 1–5, June 2012.

[69] J. L. Flanagan and R. M. Golden. Phase vocoder. *Bell System Technical Journal*, 45:1493–1509, 1966.

[70] Jonathan Foote. Automatic audio segmentation using a measure of audio novelty. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, pages 452–455, New York, NY, USA, 2000.

[71] Richard Füg, Andreas Niedermeier, Jonathan Driedger, Sascha Disch, and Meinard Müller. Accompanying website: Harmonic-percussive-residual sound separation using the structure tensor on spectrograms. `www.audiolabs-erlangen.de/resources/MIR/2016-ICASSP-HPRST/`.

[72] Richard Füg, Andreas Niedermeier, Jonathan Driedger, Sascha Disch, and Meinard Müller. Harmonic-percussive-residual sound separation using the structure tensor on spectrograms. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Shanghai, China, 2016.

[73] Hiromasa Fujihara, Masataka Goto, Jun Ogata, Kazunori Komatani, Tetsuya Ogata, and Hiroshi G. Okuno. Automatic synchronization between lyrics and music cd recordings based on viterbi alignment of segregated vocal signals. In *IEEE International Symposium on Multimedia (ISM)*, pages 257–264, Los Alamitos, California, USA, 2006.

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

[75] Olivier Gillet and Gaël Richard. Transcription and separation of drum signals from polyphonic music. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(3):529–540, 2008.

[76] Aggelos Gkiokas, Vassilios Katsouros, George Carayannis, and Themos Stafylakis. Music tempo estimation and beat tracking by applying source separation and metrical relations. In *Proceedings*

of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), pages 421–424, 2012.

[77] Aggelos Gkiokas, Vassilis Papavassiliou, Vassilis Katsouros, and George Carayannis. Deploying nonlinear image filters to spectrograms for harmonic/percussive separation. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, York, UK, September 2012.

[78] K. Glossop, P. J. G. Lisboa, P. C. Russell, A. Siddans, and G. R. Jones. An implementation of the hough transformation for the identification and labelling of fixed period sinusoidal curves. *Computer Vision and Image Understanding*, 74(1):96–100, 1999.

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

[80] Masataka Goto. A robust predominant-F0 estimation method for real-time detection of melody and bass lines in CD recordings. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 2, pages 757–760, 2000.

[81] Masataka Goto. An audio-based real-time beat tracking system for music with or without drum-sounds. *Journal of New Music Research*, 30(2):159–171, 2001.

[82] Masataka Goto. A real-time music-scene-description system: Predominant-F0 estimation for detecting melody and bass lines in real-world audio signals. *Speech Communication (ISCA Journal)*, 43(4):311–329, 2004.

[83] Amalia De Götzen, Nicola Bernardini, and Daniel Arfib. Traditional (?) implementations of a phase vocoder: the tricks of the trade. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, Verona, Italy, December 2000.

[84] Daniel W. Griffin and Jae S. Lim. Signal estimation from modified short-time Fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(2):236–243, 1984.

[85] Shahaf Grofit and Yizhar Lavner. Time-scale modification of audio signals using enhanced WSOLA with management of transients. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(1):106–115, 2008.

[86] Peter Grosche. *Signal Processing Methods for Beat Tracking, Music Segmentation, and Audio Retrieval*. PhD thesis, Saarland University and MPI Informatik, 2012.

[87] Peter Grosche and Meinard Müller. A mid-level representation for capturing dominant tempo and pulse information in music recordings. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 189–194, Kobe, Japan, 2009.

[88] Peter Grosche and Meinard Müller. Extracting predominant local pulse information from music recordings. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(6):1688–1701, 2011.

[89] Peter Grosche and Meinard Müller. Tempogram toolbox: Matlab implementations for tempo and pulse analysis of music recordings. In *Late-Breaking News of the International Society for Music Information Retrieval Conference (ISMIR)*, Miami, Florida, USA, 2011.

[90] Peter Grosche, Meinard Müller, and Frank Kurth. Cyclic tempogram – a mid-level tempo representation for music signals. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 5522 – 5525, Dallas, Texas, USA, 2010.

[91] Azadeh Haghparast, Henri Penttinen, and Vesa Välimäki. Real-time pitch-shifting of musical signals by a time-varying factor using normalized filtered correlation time-scale modification. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, pages 7–14, Bordeaux, France, September 2007.

[92] Marko Helén and Tuomas Virtanen. Separation of drums from polyphonic music using nonnegative matrix factorization and support vector machine. In *Proceedings of the European Signal Processing Conference (EUSIPCO)*, 2005.

[93] Romain Hennequin, Roland Badeau, and Bertrand David. Time-dependent parametric and harmonic templates in non-negative matrix factorization. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, pages 246–253, Graz, Austria, 2010.

[94] Romain Hennequin, Bertrand David, and Roland Badeau. Score informed audio source separation using a parametric model of non-negative spectrogram. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 45–48, Prague, Czech Republic, 2011.

[95] Jürgen Herre and Leon Terentiv. Parametric coding of audio objects: Technology, performance and opportunities. In *Proceedings of the Audio Engineering Society Conference (AES)*, Ilmenau, Germany, 2011.

[96] Perfecto Herrera and Jordi Bonada. Vibrato extraction and parameterization in the spectral modeling synthesis framework. In *Digital Audio Effects Workshop (DAFX98)*, Barcelona, Spain, November 1998.

[97] Wolfgang Hess. *Pitch Determination of Speech Signals*. Springer-Verlag, Berlin, 1983.

[98] Wolfgang Hess. Pitch and voicing determination. In Sadaoki Furui and M. M. Sohndi, editors, *Advances in Speech Signal Processing*, pages 3–48. Marcel Dekker, New York, 1992.

[99] 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)*, Philadelphia, USA, 2008.

[100] André Holzapfel, Yannis Stylianou, Ali C. Gedik, and Bariş Bozkurt. Three dimensions of pitched instrument onset detection. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(6):1517–1527, 2010.

[101] Yoshiyuki Horii. Acoustic analysis of vocal vibrato: A theoretical interpretation of data. *Journal of Voice*, 3(1):36–43, 1989.

[102] Patrik O. Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, 5:1457–1469, 2004.

[103] Chao-Ling Hsu and Jyh-Shing Roger Jang. On the improvement of singing voice separation for monaural recordings using the MIR-1K dataset. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(2):310–319, February 2010.

[104] Po-Sen Huang, Scott Deeann Chen, Paris Smaragdis, and Mark Hasegawa-Johnson. Singing-voice separation from monaural recordings using robust principal component analysis. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2012.

[105] Po-Sen Huang, Minje Kim, Mark Hasegawa-Johnson, and Paris Smaragdis. Singing-voice separation from monaural recordings using deep recurrent neural networks. In *Proceedings of the 15th International Society for Music Information Retrieval Conference ISMIR*, pages 477–482, Taipei, Taiwan, October 2014.

[106] Barbara Burke Hubbard. *The world according to wavelets.* AK Peters, Wellesley, Massachusetts, 1996.

[107] David Miles Huber. *The MIDI manual.* Focal Press, 1999.

[108] Eric J. Humphrey, Juan Pablo Bello, and Yann LeCun. Feature learning and deep architectures: new directions for music informatics. *Journal of Intelligent Information Systems*, 41(3):461–481, 2013.

[109] Yukara Ikemiya, Kazuyoshi Yoshii, and Katsutoshi Itoyama. Singing voice analysis and editing based on mutually dependent f0 estimation and source separation. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 574—578, Brisbane, Australia, 2015.

[110] Hiromi Ishizaki, Keiichiro Hoashi, and Yasuhiro Takishima. Full-automatic DJ mixing system with optimal tempo adjustment based on measurement function of user discomfort. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 135–140, Kobe, Japan, 2009.

[111] Katsutoshi Itoyama, Masataka Goto, Kazunori Komatani, Tetsuya Ogata, and Hiroshi G. Okuno. Instrument equalizer for query-by-example retrieval: Improving sound source separation based on integrated harmonic and inharmonic models. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 133–138, Philadelphia, USA, 2008.

[112] Jordi Janer and Maarten de Boer. Extending voice-driven synthesis to audio mosaicing. In *5th Sound and Music Computing Conference*, Berlin, Germany, July 2008.

[113] Il-Young Jeong and Kyogu Lee. Vocal separation from monaural music using temporal/spectral continuity and sparsity constraints. *IEEE Signal Processing Letters*, 21(10):1197–1200, October 2014.

[114] 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.

[115] Jingu Kim and Haesun Park. Toward faster nonnegative matrix factorization: A new algorithm and comparisons. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 353–362, Pisa, IT, 2008.

[116] Youngmoo Edmund Kim. Singing voice analysis, synthesis, and modeling. In David Havelock, Sonoko Kuwano, and Michael Vorländer, editors, *Handbook of Signal Processing in Acoustics*, pages 359–374. Springer New York, 2008.

[117] Anssi P. Klapuri. Multiple fundamental frequency estimation by summing harmonic amplitudes. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 216–221, 2006.

[118] Anssi P. Klapuri. Multipitch analysis of polyphonic music and speech signals using an auditory model. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(2):255–266, 2008.

[119] Hans Knutsson. Representing local structure using tensors. In *6th Scandinavian Conference on Image Analysis*, pages 244–251, Oulu, Finland, 1989.

[120] Ryoho Kobayashi. Sound clustering synthesis using spectral data. In *Proceedings of the International Computer Music Conference (ICMC)*, Singapore, 2003.

[121] Sebastian Kraft, Martin Holters, Adrian von dem Knesebeck, and Udo Zölzer. Improved PVSOLA time stretching and pitch shifting for polpyhonic audio. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, York, September 2012.

[122] Carol L. Krumhansl. *Cognitive foundations of musical pitch.* Oxford University Press, 1990.

[123] Alexandre Lacoste and Douglas Eck. A supervised classification algorithm for note onset detection. *EURASIP Journal on Applied Signal Processing*, pages 153–165, 2007.

[124] Peter Ladefoged and Ian Maddieson. *The Sounds of the World's Languages.* Phonological Theory. Wiley, 1996.

[125] Mathieu Lagrange and Sylvaim Marchand. Estimating the instantaneous frequency of sinusoidal components using phase-based methods. *Journal of the Audio Engineering Society*, 55(5):385–399, 2007.

[126] Jean Laroche. Autocorrelation method for high-quality time/pitch-scaling. In *IEEE Workshop Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 131–134, Mohonk, NY, 1993.

[127] Jean Laroche and Mark Dolson. Phase-vocoder: about this phasiness business. In *IEEE ASSP Workshop on Applications of Signal Processing to Audio and Acoustics, 1997*, October 1997.

[128] Jean Laroche and Mark Dolson. Improved phase vocoder time-scale modification of audio. *IEEE Transactions on Speech and Audio Processing*, 7(3):323–332, 1999.

[129] Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In *Proceedings of the Neural Information Processing Systems (NIPS)*, pages 556–562, Denver, CO, USA, 2000.

[130] Augustin Lefevre, Francis Bach, and Cédric Févotte. Semi-supervised NMF with time-frequency annotations for single-channel source separation. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 115–120, Porto, Portugal, 2012.

[131] Bernhard Lehner, Gerhard Widmer, and Sebastian Böck. A low-latency, real-time-capable singing voice detection method with lstm recurrent neural networks. In *Proceedings of the European Signal Processing Conference (EUSIPCO)*, pages 21–25, Nice, France, 2015.

[132] 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.

[133] Scott N. Levine and Julius O. Smith III. A sines+transients+noise audio representation for data compression and time/pitch scale modications. In *Proceedings of the Audio Engineering Society (AES) Convention*, 1998.

[134] librosa development team. Librosa—chroma filters. `bmcfee.github.io/librosa/generated/librosa.filters.chroma.html`, 2015. Web resource, last consulted in January 2016.

[135] librosa development team. Librosa—onset detection. `bmcfee.github.io/librosa/onset.html`, 2015. Web resource, last consulted in January 2016.

[136] librosa development team. Librosa—time stretching. `https://bmcfee.github.io/librosa/generated/librosa.effects.time_stretch.html`, 2015. Web resource, last consulted in November 2015.

[137] Antoine Liutkus, Derry FitzGerald, Zafar Rafii, Bryan Pardo, and Laurent Daudet. Kernel additive models for source separation. *IEEE Transactions on Signal Processing*, 62(16):4298–4310, 2014.

[138] Antoine Liutkus, Zafar Rafii, Roland Badeau, Bryan Pardo, and Gaël Richard. Adaptive filtering for music/voice separation exploiting the repeating musical structure. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 53–56, 2012.

[139] John D. Markel and Augustine H. Gray. *Linear Prediction of Speech.* Springer Verlag, Secaucus, NJ, USA, 1976.

[140] MATLAB. High-performance numeric computation and visualization software. The MathWorks Inc., `http://www.mathworks.com`, 2013.

[141] Matthias Mauch, Chris Cannam, and Gyorgy Fazekas. Efficient computer-aided pitch track and note estimation for scientific applications, April 2014. extended abstract accepted at SEMPRE 2014 conference.

[142] Alexis Moinet and Thierry Dutoit. PVSOLA: A phase vocoder with synchronized overlapp-add. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, pages 269–275, Paris, September 2011.

[143] Alexis Moinet, Thierry Dutoit, and Philippe Latour. Audio time-scaling for slow motion sports videos. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, Maynooth, Ireland, September 2013.

[144] Eric Moulines and Francis Charpentier. Pitch-synchronous waveform processing techniques for text-to-speech synthesis using diphones. *Speech Communication*, 9(5-6):453–467, December 1990.

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

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

[147] Meinard Müller and Jonathan Driedger. Data-driven sound track generation. In Meinard Müller, Masataka Goto, and Markus Schedl, editors, *Multimodal Music Processing*, volume 3 of *Dagstuhl Follow-Ups*, pages 175–194. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2012.

[148] Meinard Müller, Jonathan Driedger, and Sebastian Ewert. Notentext-informierte Quellentrennung für Musiksignale. In *Proceedings of 43th GI Jahrestagung*, pages 2928–2942, Koblenz, Germany, 2013.

[149] Meinard Müller, Daniel P. W. Ellis, Anssi Klapuri, and Gaël Richard. Signal processing for music analysis. *IEEE Journal on Selected Topics in Signal Processing*, 5(6):1088–1110, 2011.

[150] Meinard Müller and Sebastian Ewert. Towards timbre-invariant audio features for harmony-based music. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(3):649–662, 2010.

[151] 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.

[152] Meinard Müller, Verena Konz, Wolfgang Bogler, and Vlora Arifi-Müller. Saarland music data (SMD). In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR): Late Breaking session*, 2011.

[153] 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.

[154] 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.

[155] Frederik Nagel and Andreas Walther. A novel transient handling scheme for time stretching algorithms. In *Proceedings of the Audio Engineering Society (AES) Convention*, pages 185–192, New York, NY, USA, 2009.

[156] Nobutaka Ono, Kenichi Miyamoto, Hirokazu Kameoka, and Shigeki Sagayama. A real-time equalizer of harmonic and percussive components in music signals. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 139–144, Philadelphia, Pennsylvania, USA, 2008.

[157] Nobutaka Ono, Kenichi Miyamoto, Jonathan LeRoux, Hirokazu Kameoka, and Shigeki Sagayama. Separation of a monaural audio signal into harmonic/percussive components by complementary diffusion on spectrogram. In *European Signal Processing Conference (EUSIPCO)*, pages 240–244, Lausanne, Switzerland, 2008.

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

[159] John Oswald. Plexure. CD, 1993. `http://www.allmusic.com/album/plexure-mw0000621108`.

[160] Alexey Ozerov, Ngoc Q. K. Duong, and Louis Chevallier. Weighted nonnegative tensor factorization: on monotonicity of multiplicative update rules and application to user-guided audio source separation. Technical report, Technicolor R & I, October 2013.

[161] Hee-Suk Pang. On the use of the maximum likelihood estimation for analysis of vibrato tones. *Applied Acoustics*, 65(1):101–107, 2004.

[162] Sachin Pant, Vishweshwara Rao, and Preeti Rao. A melody detection user interface for polyphonic music. In *National Conference on Communications (NCC)*, pages 1–5, January 2010.

[163] Jeongsoo Park and Kyogu Lee. Harmonic-percussive source separation using harmonicity and sparsity constraints. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 148–154, Málaga, Spain, 2015.

[164] Olli Parviainen. Soundtouch audio processing library. `http://www.surina.net/soundtouch/`. Web resource, last consulted in November 2015.

[165] Geoffroy Peeters. Template-based estimation of time-varying tempo. *EURASIP Journal on Advances in Signal Processing*, 2007.

[166] Alexey Petrovsky, Elias Azarov, and Alexander Petrovsky. Hybrid signal decomposition based on instantaneous harmonic parameters and perceptually motivated wavelet packets for scalable audio coding. *Signal Processing*, 91(6):1489–1504, 2011.

[167] Graham E. Poliner, Daniel P.W. Ellis, Andreas F. Ehmann, Emilia Gómez, Sebastian Streich, and Beesuan Ong. Melody transcription from music audio: Approaches and evaluation. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(4):1247–1256, 2007.

[168] M. R. Portnoff. Implementation of the digital phase vocoder using the fast Fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 24(3):243–248, 1976.

[169] Eric Prame. Measurements of the vibrato rate of ten singers. *The Journal of the Acoustical Society of America (JASA)*, 96(4):1979–1984, 1994.

[170] Eric Prame. Vibrato extent and intonation in professional western lyric singing. *The Journal of the Acoustical Society of America (JASA)*, 102(1):616–621, 1997.

[171] 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)*, Shanghai, China, 2016.

[172] John G. Proakis and Dimitris G. Manolakis. *Digital Signal Processing.* Prentice Hall, 1996.

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

[174] Zafar Rafii and Bryan Pardo. Repeating pattern extraction technique (REPET): A simple method for music/voice separation. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(1):71–82, 2013.

[175] Emmanuel Ravelli, Gaël Richard, and Laurent Daudet. Union of MDCT bases for audio coding. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(8):1361–1372, November 2008.

[176] Lise Regnier and Geoffroy Peeters. Singing voice detection in music tracks using direct voice vibrato detection. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 1685–1688, Taipei, Taiwan, 2009.

[177] Axel Röbel and Xavier Rodet. Efficient spectral envelope estimation and its application to pitch shifting and envelope preservation. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, pages 344–349, Madrid, Spain, September 2005.

[178] Axel Röbel and Xavier Rodet. Real time signal transposition with envelope preservation in the phase vocoder. In *Proceedings of the International Computer Music Conference (ICMC)*, pages 672–675, Barcelona, Spain, September 2005.

[179] S. Rossignol, P. Depalle, J. Soumagne, X. Rodet, and J.-L. Collette. Vibrato: detection, estimation, extraction, modification. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, Trondheim, Norway, December 1999.

[180] Salim Roucos and Alexander M. Wilgus. High quality time-scale modification for speech. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 10, pages 493–496, April 1985.

[181] Matti Ryynänen and Anssi P. Klapuri. Automatic transcription of melody, bass line, and chords in polyphonic music. *Computer Music Journal*, 32(3):72–86, 2008.

[182] Nicolas Saint-Arnaud and Kris Popat. Analysis and synthesis of sound textures. In *Computational auditory scene analysis*, pages 293–308. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1998.

[183] Justin Salamon and Emilia Gómez. Melody extraction from polyphonic music signals using pitch contour characteristics. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(6):1759–1770, 2012.

[184] Justin Salamon, Emilia Gómez, Daniel P. W. Ellis, and Gaël Richard. Melody extraction from polyphonic music signals: Approaches, applications, and challenges. *IEEE Signal Processing Magazine*, 31(2):118–134, 2014.

[185] Norbert Schnell, Marco Antonio Suárez Cifuentes, and Jean-Philippe Lambert. First steps in relaxed real-time typo-morphological audio analysis/synthesis. In *Sound and Music Computing*, Barcelona, Spain, 2010.

[186] Christian Schörkhuber, Anssi Klapuri, and Alois Sontacchi. Audio pitch shifting using the constant-q transform. *Journal of the Audio Engineering Society*, 61(7/8):562–572, 2013.

[187] Christian Schörkhuber and Anssi P. Klapuri. Constant-Q transform toolbox for music processing. In *Sound and Music Computing Conference (SMC)*, Barcelona, Spain, 2010.

[188] Franz Schubert. Winterreise. Ein Cyclus von Liedern von Wilhelm Müller. `conquest.imslp.info/files/imglnks/usimg/9/92/IMSLP00414-Schubert_-_Winterreise.pdf`, 1827. Gesänge für eine Singstimme mit Klavierbegleitung, Edition Peters, No.20a, n.d. Plate 9023.

[189] Diemo Schwarz. A system for data-driven concatenative sound synthesis. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, Verona, Italy, july 2000.

[190] Diemo Schwarz. Concatenative sound synthesis: The early years. *Journal of New Music Reaserch*, 35(1), march 2006.

[191] Carl E. Seashore. The natural history of the vibrato. *Proceedings of the National Academy of Sciences of the United States of America*, 17(12):623–626, 1931.

[192] Eleanor Selfridge-Field, editor. *Beyond MIDI: the handbook of musical codes.* MIT Press, Cambridge, Massachusetts, USA, 1997.

[193] Mike Senior. Mixing secrets for the small studio—additional resources. `www.cambridge-mt.com/ms-mtk.htm`. Web resource, last consulted in January 2016.

[194] Xavier Serra and Julius Smith III. Spectral modeling synthesis: A sound analysis/synthesis system based on a deterministic plus stochastic decomposition. *Computer Music Journal*, 14(4):12–24, 1990.

[195] T. Shipp, R. Leanderson, and J. Sundberg. Some acoustic characteristics of vocal vibrato. *Journal of Research in Singing*, IV(1):18–25, 1980.

[196] Siddharth Sigtia, Nicolas Boulanger-Lewandowski, and Simon Dixon. Audio chord recognition with a hybrid recurrent neural network. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, pages 127–133, Malaga, Spain, 2015.

[197] Paris Smaragdis. Non-negative matrix factor deconvolution; extraction of multiple sound sources from monophonic inputs. In *Proceedings of the International Conference on Independent Component Analysis and Blind Signal Separation ICA*, pages 494–499, Grenada, Spain, 2004.

[198] Paris Smaragdis and Gautham J. Mysore. Separation by humming: User guided sound extraction from monophonic mixtures. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 69–72, New Paltz, NY, USA, 2009.

[199] Paris Smaragdis, Bhiksha Raj, and Madhusudana Shashanka. Supervised and semi-supervised separation of sounds from single-channel mixtures. In *Proceedings of the International Conference on Independent Component Analysis and Signal Separation (ICA)*, pages 414–421, London, UK, 2007.

[200] Paris Smaragdis, Bhiksha Raj, and Madhusudana Shashanka. Sparse and shift-invariant feature extraction from non-negative data. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 2069–2072, Las Vegas, Nevada, USA, 2008.

[201] Fabian-Robert Stöter, Nils Werner, Stefan Bayer, and Bernd Edler. Refining fundamental frequency estimates using time warping. In *Proceedings of EUSIPCO 2015*, Nice, France, September 2015.

[202] Dan Stowell and Mark Plumbley. Adaptive whitening for improved real-time audio onset detection. In *Proceedings of the International Computer Music Conference (ICMC)*, Copenhagen, Denmark, 2007.

[203] Bob Sturm. Adaptive concatenative sound synthesis and its application to micromontage composition. *Computer Music Journal*, 30(4):46–66, 2006.

[204] Johan Sundberg. Acoustic and psychoacoustic aspects of vocal vibrato. *STL-QPSR*, 35(2-3):45–68, 1994.

[205] Andrew Swift. An introduction to midi. `www.doc.ic.ac.uk/~nd/surprise_97/journal/vol1/aps2/`. Web resource, last consulted in January 2016.

[206] Hideyuki Tachibana, Nobutaka Ono, Hirokazu Kameoka, and Shigeki Sagayama. Harmonic/percussive sound separation based on anisotropic smoothness of spectrograms. *IEEE/ACM Transactions on Audio, Speech & Language Processing*, 22(12):2059–2073, 2014.

[207] Hideyuki Tachibana, Nobutaka Ono, and Shigeki Sagayama. Singing voice enhancement in monaural music signals based on two-stage harmonic/percussive sound separation on multiple resolution spectrograms. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(1):228–237, January 2014.

[208] Ronen Talmon, Israel Cohen, and Sharon Gannot. Transient noise reduction using nonlocal diffusion filters. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(6):1584–1599, August 2011.

[209] Balaji Thoshkahna and Ramakrishnan R. Kalpathi. A postprocessing technique for improved harmonic/percussion separation for polyphonic music. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 251–256, 2011.

[210] Renee Timmers and Peter Desain. Vibrato: Questions and answers from musicians and science. In *Proceedings of the International Conference on Music Perception and Cognition (ICMPC)*, volume 2, 2000.

[211] Chee Chuan Toh, Bingjun Zhang, and Ye Wang. Multiple-feature fusion based onset detection for solo singing voice. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 515–520, Philadelphia, PA, USA, 2008.

[212] Pierre Alexandre Tremblay and Diemo Schwarz. Surfing the waves : Live audio mosaicing of an electric bass performance as a corpus browsing interface. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 447–450, Sydney, Australia, September 2010.

[213] Yushi Ueda, Yuuki Uchiyama, Takuya Nishimoto, Nobutaka Ono, and Shigeki Sagayama. HMM-based approach for automatic chord detection using refined acoustic features. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 5518–5521, Dallas, Texas, USA, 2010.

[214] Christian Uhle and Emanuël Habets. Subband center scaling using power ratios. In *Proceedings of the AES 53rd International Conference*, London, UK, 2014.

[215] Shankar Vembu and Stephan Baumann. Separation of vocals from polyphonic audio recordings. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 337–344, London, UK, 2005.

[216] Werner Verhelst and Marc Roelands. An overlap-add technique based on waveform similarity (WSOLA) for high quality time-scale modification of speech. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Minneapolis, USA, 1993.

[217] Tony S. Verma and Teresa H.Y. Meng. An analysis/synthesis tool for transient signals that allows a flexible sines+transients+noise model for audio. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 3573–3576, Seattle, Washington, USA, May 1998.

[218] Tony S. Verma and Teresa H.Y. Meng. Time scale modification using a sines+transients+noise signal model. In *Digital Audio Effects Workshop (DAFX98)*, Barcelona, Spain, November 1998.

[219] Tony S. Verma and Teresa H.Y. Meng. Extending spectral modeling synthesis with transient modeling synthesis. *Computer Music Journal*, 24(2):47–59, July 2000.

[220] 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.

[221] Tuomas Virtanen. Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(3):1066–1074, 2007.

[222] Tuomas Virtanen, Annamaria Mesaros, and Matti Ryynänen. Combining pitch-based inference and non-negative spectrogram factorization in separating vocals from polyphonic music. In *Proceedings of the ISCA Tutorial and Research Workshop on Statistical And Perceptual Audition (SAPA)*, pages 17–22, Brisbane, Australia, 2008.

[223] Website. Audionamix, ADX Trax, commercial singing voice extraction software. `http://www.audionamix.com/`.

[224] Website. Signal separation evaluation campaign (SiSEC). `http://www.onn.nii.ac.jp/sisec13/evaluation_result/MUS/devMUS2013.htm`.

[225] Ron J. Weiss and Juan Pablo Bello. Unsupervised discovery of temporal structure in music. *IEEE Journal of Selected Topics in Signal Processing*, 5:1240–1251, 2011.

[226] Wikipedia. Alvin and the chipmunks—recording technique. `https://en.wikipedia.org/wiki/Alvin_and_the_Chipmunks\#Recording_technique`, 2015. Web resource, last consulted in December 2015.

[227] 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.

[228] Luwei Yang, Elaine Chew, and Khalid Z. Rajab. Vibrato performance style: A case study comparing erhu and violin. In *Proceedings of the International Conference on Computer Music Modeling and Retrieval (CMMR)*, 2013.

[229] Ruohua Zhou, Marco Mattavelli, and Giorgio Zoia. Music onset detection based on resonator time frequency image. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(8):1685–1695, 2008.

[230] Udo Zölzer. *DAFX: Digital Audio Effects*. John Wiley & Sons, Inc., New York, NY, USA, 2002.

[231] zplane development. élastique time stretching & pitch shifting SDKs. `http://www.zplane.de/index.php?page=description-elastique`.