# Analyzing Sample-Based Electronic Music Using Audio Processing Techniques

# Audioverarbeitungsmethoden zur Analyse Sample-basierter elektronischer Musik

Der Technischen Fakultät

der Friedrich-Alexander-Universität Erlangen-Nürnberg

zur

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

vorgelegt von

Patricio López Serrano Erickson

aus

Mexiko-Stadt

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

| | |
|---|---|
| Tag der mündlichen Prüfung: | 19. Februar 2019 |
| Vorsitzender des Promotionsorgans: | Prof. Dr.-Ing. Reinhard Lerch |
| 1. Gutachter: | Prof. Dr. Meinard Müller |
| 2. Gutachter: | Prof. Dr. Jason Hockman |

# Abstract

The advent of affordable digital sampling technology brought about great changes in music production. Experienced producers of sample-based electronic music (SBEM) are capable of understanding the intricate relationship between a sampled source and its use in a new track, harnessing its properties to shape the structure, timbre, and rhythm of their compositions. However, automated analysis of the phenomena surrounding both SBEM and the sources it uses for its samples is a challenge which involves many tasks from music information retrieval (MIR) and audio processing.

In this thesis we develop models and techniques to better understand SBEM at different levels. In particular, we offer four main technical contributions to retrieval and analysis tasks. First, we explore how timbral changes affect the spectral peak maps used in audio fingerprinting as a means to identify overlapping samples. Second, we analyze the structure of typical SBEM tracks using audio decomposition based on non-negative matrix factor deconvolution (NMFD). Third, we investigate the interaction of timbre and structure by designing a mid-level audio feature based on cascaded harmonic-residual-percussive source separation (CHRP). Fourth, we apply random forests to identify the quintessential sampling source: drum breaks. Given their prominent role in SBEM, we devote considerable attention to drum breaks. As an application to computational musicology, we formalize an algorithm for calculating local swing ratio in drum breaks and adapt an autocorrelation-based method to analyze their microrhythmic properties. Finally, we present a creative application to music production which allows combining the temporal and timbral properties of two separate drum breaks (redrumming).

Despite the massive commercial success of SBEM and the practice of sampling, they mostly remain outside the attention of formal academic studies and MIR research. In this thesis our overarching goal is to identify and formalize some of the fundamental audio processing tasks related to SBEM, proposing methods that can seed further research.

# Zusammenfassung

Die zunehmende Verbreitung erschwinglicher digitaler Sampling-Technologie wurde die Musikproduktion revolutioniert. Erfahrene Produzenten von Sample-basierter elektronischer Musik (SBEM) können die vielschichtige Beziehung zwischen der ursprünglichen Musik eines Samples und der Verwendung in einem neuen Track erkennen und die Eigenschaften des Samples nutzen, um die Struktur, Klangfarbe und den Rhythmus neuer Kompositionen zu gestalten. Der Zusammenhang von SBEM und den dort verwendeten Samples ist vielschichtig. Die automatisierte Analyse von SBEM führt somit zu unterschiedlichen Aufgabenstellungen, die im Bereich des *Music Information Retrieval* (MIR) und der Audioverarbeitung angesiedelt sind. In dieser Arbeit entwickeln wir Modelle und Techniken, um SBEM auf verschiedenen Ebenen besser zu verstehen. Insbesondere umfasst die Arbeit vier technische Beiträge zu Retrieval- und Analyseaufgaben. Zuerst untersuchen wir, wie klangliche Veränderungen die spektralen Peak-Maps beeinflussen, die beim Audio-Fingerprinting verwendet werden, um überlappende Samples zu identifizieren. Zweitens analysieren wir die Struktur typischer SBEM-*Tracks* mithilfe von Audiozerlegungstechniken, die auf sogenannter Non-Negative Matrix Factorization Deconvolution (NMFD) basieren. Drittens untersuchen wir die Wechselwirkung von Klangfarbe und Struktur, indem wir ein Mid-Level-Audio-Merkmal entwickeln, das auf kaskadierter harmonisch-residual-perkussiver Quellentrennung (CHRP) basiert. Angesichts ihrer herausragenden Rolle in SBEM widmen wir Drum Breaks, eine der meistverwendeten Sample-Quellen, besondere Aufmerksamkeit. In einem vierten Beitrag wenden wir Techniken des maschinellen Lernens (insbesondere Random Forests) an, um Drum Breaks automatisiert in Musikaufnahmen aufzuspüren. Als Anwendung in der computergestützten Musikwissenschaft formalisieren wir einen Algorithmus zur Berechnung des lokalen *Swing Ratios*. Dazu adaptieren wir eine Autokorrelations-basierte Methode zur Analyse der mikrorhythmischen Eigenschaften von Drum Breaks. Schließlich präsentieren wir eine kreative Anwendung für die Musikproduktion, bei der zeitliche und klangliche Eigenschaften zweier unterschiedlicher Drum Breaks kombiniert werden können (*Redrumming*). Trotz des überaus großen kommerziellen Erfolgs von SBEM und der Verwendung von Samples bleibt diese Musikpraxis meist außerhalb der Aufmerksamkeit akademischer Studien und MIR-Forschung. Ein übergeordnetes Ziel dieser Arbeit besteht in der Formalisierung neuartiger wissenschaftlicher Fragestellungen und der Bereitstellung von Methoden zur computergestützten Analyse von SBEM.

# Contents

# Chapter 1

# Introduction

Almost in parallel, technological advances fostered changes on two fronts. On the one hand, computation technology enabled emergent fields of study like music information retrieval (MIR) and audio processing. On the other hand, music technology—particularly affordable digital sampling—democratized production, putting music-making capabilities into the eager hands of amateur musicians with small home studios. Each of these fronts has become established in its own right. MIR and audio processing are widely recognized research fields, with conferences and publications (such as ISMIR, ICASSP, and DAFx) that are devoted to them. At the same time, sample-based electronic music (SBEM) and the practice of sampling have grown out of the genres where they were first used (such as hip-hop, jungle, and drum'n'bass), and have become central to many genres, including contemporary (as of 2018), chart-topping pop music.

Recording technology and digital sampling made possible the emergence of both MIR and SBEM. The workflows and practices related to SBEM production open up interesting analysis and retrieval problems—especially new perspectives on well-established tasks such as fingerprinting, audio decomposition, and structure analysis. However, up to now, MIR and audio processing literature have dedicated relatively little attention to SBEM.

Audio fingerprinting has typically been used to match a (potentially distorted or modified) query to a reference in a database—in other words, as a means to identify the query. Nevertheless, it can also be used for structure analysis in SBEM, by exploiting the fact that loops are exact copies of each other, up to differences caused by applying effects or overlaying samples in different combinations. This piece of knowledge—that we can expect mostly identical copies of a sample throughout a track—also motivates interesting audio decomposition and structure analysis applications.

While making tracks, SBEM producers often have to manage large sample and track collections, seeking musical material with certain characteristics that will fit into their production. To this end, musicians can use audio features to visualize important timbral and structural changes, helping to locate and retrieve

interesting material. Paired with machine learning systems that classify signals according to desirable characteristics, we can automate time-consuming tasks like sifting through sizable music collections.

Thus, our main goal in this thesis is to identify some of the essential timbral, structural, and rhythmic characteristics of SBEM and its archetypal sampling sources, putting them into the formal context of well-known MIR tasks and techniques.

## 1.1 Structure

This thesis is organized as follows. We begin with Chapter 2, introducing the fundamental aspects of SBEM—illustrating them for later use and placing them into the context of current MIR and audio processing literature.

In Chapter 3 we begin with an in-depth study of how peak-map-based fingerprinting behaves when faced with the challenges that arise in SBEM tracks/production. Audio fingerprinting is the task of recognizing a query by matching it to an existing reference in a database. A desirable characteristic of fingerprinting algorithms is that they should be capable of delivering accurate results despite modifications and distortions in the query signal. In the MIR literature, most fingerprinting algorithms are based on peak maps (PM), also known as constellation maps: a sparse spectral representation where local maxima are used to encode a signal's most distinctive characteristics. One of the main challenges with PM-based fingerprinting is source superposition (for instance, drums overlaid with a bassline)—in SBEM it is almost never the case that the fingerprint of the sum of two signals equals the sum of the component fingerprints. Thus we conduct systematic, in-depth experiments using audio samples from certain genres in electronic music, observing how PM are affected by the types of modifications and distortions that are typically applied in SBEM. Given the fact that state-of-the-art fingerprinting methods such as the one by Sonnleitner and Widmer [129] are particularly challenged by overlapping musical sources, our study is important to elucidate the causes and potential solutions to this phenomenon.

Structure analysis and segmentation are fundamental tasks in MIR; many approaches have been developed for pop and classical music. In Chapter 4 we lay the foundation for structure analysis in SBEM, exploiting the particularities that result from music production with (practically) identical copies of a sample in repetition throughout a track. We start by defining a simplified mathematical model for the prototypical form seen in SBEM: many tracks follow a layered structure where samples (or *loops*) are introduced and removed at certain time intervals. We then approach the following task: given a downmix of a SBEM track that follows the structural rules above, as well as one copy of each loop used in the track, we wish to find out at what point in the track each loop was activated. We compare three methods for solving this task. First, we establish a naïve baseline using the cosine distance of short-time spectral features. Our second approach uses peak maps as discussed in Chapter 3. The third approach relies on non-negative matrix factor deconvolution (NMFD)—a technique that we use for multiple tasks in this thesis.

In Chapter 5 we present a mid-level audio feature based on cascaded harmonic-residual-percussive (CHRP) source separation (SS). Decomposing audio mixtures into their harmonic, residual, and percussive components is a widely studied technique throughout the audio signal processing literature, often used as a preprocessing step for other tasks such as beat tracking. In contrast to the typical scenario where HRPSS is used to obtain component signals from a downmix (for instance, separating a mixture of drums and singing into two components as if recorded separately), we use the energies of separated components to visualize changes in event density (like playing a snare drum with varying speed), or major changes in instrumentation—like the transition from a full band playing into a drum solo section.

Drum breaks are perhaps the most sought-after sample source—finding drum breaks within collections of vinyl records is a very time-consuming task. In Chapter 6 we first define the concept of *drum break* in a technically sound way, clearing up the ambiguity that otherwise surrounds the term. With a dataset of over 200 tracks, we then investigate how to find drum breaks in these digital music recordings. We use a machine learning approach based on random forests (RF), which was originally designed for the binary classification task of singing voice detection (SVD).

In Chapter 7 we study the microrhythmic properties of famous drum breaks and, more specifically, their swing ratio. Swing is an expressive technique by which musicians systematically delay even-numbered notes or events at a certain metrical level, lending a *shuffly* or *groovy* feel to the music. Up to now, musicologists have mainly focused on this rhythmic practice within jazz music, but in [56], Frane manually computed local swing ratios for a corpus of 30 drum breaks from the hip-hop *canon*. In this chapter, we first formalize Frane's method for computing local swing ratio (LSR). Then, we adapt an automatic swing ratio estimation (SRE) method and apply drum source separation as a pre-processing step. We end the chapter with a large-scale study on a corpus of more than 500 drum breaks.

In Appendix A we present the `NMF Toolbox`—a collection of NMF variants and implementations which are documented through concrete applications of audio decomposition to SBEM-related tasks. The examples in this appendix illustrate both fine-grained scenarios (such as drum replacement or *redrumming*), as well as coarser tasks, like analysis and decomposition at the loop scale.

## 1.2 Publications

Substantial parts of this thesis are based on publications where I am the first and main author. More details are given in the respective thesis chapters.

[90] Patricio López-Serrano, Christian Dittmar, Jonathan Driedger, and Meinard Müller. Towards modeling and decomposing loop-based electronic music. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, pages 502–508, New York City, USA, August 2016

[93] Patricio López-Serrano, Christian Dittmar, and Meinard Müller. Mid-level audio features based on cascaded harmonic-residual-percussive separation. In *Proceedings of the Audio Engineering Society Conference on Semantic Audio (AES)*, pages 32–44, Erlangen, Germany, June 2017

[92] Patricio López-Serrano, Christian Dittmar, and Meinard Müller. Finding drum breaks in digital music recordings. In *Proceedings of the International Symposium on Computer Music Multidisciplinary Research (CMMR)*, pages 68–79, Porto, Portugal, September 2017

[89] Patricio López-Serrano, Matthew E. P. Davies, Jason Hockman, Christian Dittmar, and Meinard Müller. Break-informed audio decomposition for interactive redrumming. In *Late Breaking and Demo Session of the International Society for Music Information Retrieval Conference (ISMIR):*, 2018

The following article, which is part of Chapter 7, has been submitted for publication:

[91] Patricio López-Serrano, Christian Dittmar, Andrew V. Frane, and Meinard Müller. Estimating sixteenth note swing ratio in drum break recordings. 2018

The following additional publication is not considered in this thesis:

[42] Christian Dittmar, Patricio López-Serrano, and Meinard Müller. Unifying local and global methods for harmonic-percussive source separation. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Calgary, Canada, April 2018

## 1.3 Contributions

The following contributions are based on publications which were presented at conferences in the field of audio signal processing and MIR.

- A mathematical model for the typical structure of SBEM tracks ([90] and Section 4.3).
- A comparison of techniques for finding the activations of a given sample or loop within a SBEM track, including repurposing audio fingerprinting for structure analysis ([90] and Section 4.4).
- A mid-level audio feature based on cascaded harmonic-residual-percussive source separation that can be used to visualize regions of timbral change ([93] and Section 5.3).
- The task definition of finding drum breaks, including a technical definition of drum breaks as "percussion-only regions" for use with audio classification methods ([92] and Sections 6.1–6.2).
- A dataset of 280 tracks with annotated drum break regions ([92] and Section 6.4.1).
- A study of how well a binary classification method for singing voice detection, based on random forests, could be repurposed for finding drum breaks ([92] and Section 6.3.2).

- Formalized, as an algorithm, a method by Frane [56] for computing LSR ([91] and Section 7.2.2).
- An extensive study of the correspondence between Frane's manual findings for LSR and an automated SRE method by Dittmar et al. [45]. ([91] and Sections 7.3–7.4).
- A first approach for *redrumming* across pairs of tracks that contain drum breaks ([89] and Appendix B).
- A definition and in-depth study of *peak survival* or *peak agreement* for peak-map-based fingerprinting (Sections 3.6–3.9).
- Contributions to and documentation of the `NMF Toolbox`, which provides implementations of several NMF variants (Appendix A).

## 1.4 Acknowledgments

It is not lost on me how fortunate I have been to pursue this avenue of research—I mean, who gets to do a PhD on electronic music?

I wish to thank Consejo Nacional de Ciencia y Tecnología (CONACYT) as well as Deutscher Akademischer Austauschdienst (DAAD) for providing financial support throughout my studies.

To my advisor, Meinard Müller: Thank you for never ceasing to expect the best, but always taking the time to get there together.

I cannot thank `GroupMM` enough for their support, kindness, and friendship throughout these years: Vlora Arifi-Müller, Stefan Balke, Christian Dittmar, Jonathan Driedger, Thomas Prätzlich, Sebastian Rosenzweig, Hendrik Schreiber, Christof Weiß, Frank Zalkow, and Julia Zalkow. Thank you to Pedro Solórzano for the work we did together on fingerprinting.

Thank you to Jason Hockman for reviewing my thesis, to Matthew Davies for hosting my research visit at INESC-SMC, and to Andrew Frane for the interdisciplinary collaboration.

Thank you to my friends and colleagues from AudioLabs: Alexander Adami, Soumitro Chakrabarty, Pablo Delgado, Esther Fee Feichtner, Johannes Fischer, María Luis Valero, Konstantin Schmidt, Fabian-Robert Stöter, and Nils Werner.

Special thanks to Prof. Emanuël Habets for the feedback and yearly reviews.

Thank you to Fraunhofer IDMT-SMT for the great collaboration (and hackdays!): Jakob Abeßer, Estefanía Cano Cerón, Daniel Gärtner, Sascha Grollmisch, Anna Kruspe, Hanna Lukashevich, and Stylianos Ioannis Mimilakis.

A big thank you to the administrative team at AudioLabs: Tracy Harris, Jennifer Kohlhage, Day-See Riechmann, Eva Schmidt, Kerstin Schröppel, Carola Seibold, Stefan Turowski, and Elke Weiland.

To my family: This would be unthinkable without your love and support. Thank you so much!

To my life partner, Luisa: I can't say many people have endured two theses of mine. Thank you, with love, for filling my life with colors and showing me the tiniest things make the greatest difference.

*And to all those who were told not to touch the record...*

# Chapter 2

# Fundamental Concepts of SBEM

In this chapter we introduce the fundamental concepts and practices of SBEM that motivate the tasks and models presented throughout the rest of the thesis. DJing and production are the main creative activities in SBEM, and one cannot be understood without the other. Following the coarse chronology of SBEM, we examine how modern production principles grew out of the foundation laid by early hip-hop DJ practice.

## 2.1 Drum Breaks and Layering

We can trace the roots of SBEM to the first hip-hop DJs in the mid-to-late 1970s [118], who played funk, soul, jazz, and rock records on vinyl. Through their interaction with the audience they came to realize that dancers particularly favored *drum breaks*: percussion-only passages (often drum solos) found on these types of recordings [118]. The drum break soon became a central element of DJ practice, to the extent that DJs harnessed the technology in their setups in order to arbitrarily prolong the length of breaks: by using two copies of the same record on separate turntables, together with a mixer, DJs could *beat juggle* (or loop the break) by quickly switching between instances [118]. This fascination with drum breaks spawned a culture of looking for increasingly obscure drum breaks, whereby DJs gained respect for finding hidden gems that were previously unheard of. The act of looking for vinyl records containing drum breaks is called (crate) digging and it involves considerable effort; it is a tradition that remains alive to the present day, with DJs and producers looking for rare records in basements, flea markets, auctions, and during trips abroad. Once a record has been found, SBEM artists need to locate the break by randomly dropping the needle along the grooves. Although there are hundreds of well-known breaks that have become classic in SBEM, perhaps two of the most famous are the "Amen break" from "Amen, Brother" by The Winstons and "Funky Drummer" by James Brown. Given their importance within SBEM, we dedicate considerable attention to drum breaks in this thesis. We first touch upon the subject in Chapter 5 and then discuss it at greater length in Chapters 6 and 7, as well as Appendix A. Seminal work on breakbeats was done by

Collins, who approaches various tasks revolving around breakbeats such as algorithmic composition [24], automatic cutting methods [25], and interactive evolution of breakbeat sequences [26].

Although drum breaks may be the best-known or most recognizable source, sampling is a vast practice that goes beyond this type of material. For example, Sewell [123] developed a typology of sampling in hip hop which divides samples into categories according to their function. In [112], Ratcliffe proposes another sampling typology, offering "a system of classification that takes into account the sonic, musical and referential properties of sampled elements". Ratcliffe enumerates four high-level categories for samples: short isolated fragments, loops and phrases, larger elements, and transformed material. This provides a formal theoretical framework to discuss sampling practices in electronic music. To conclude this section, we discuss some of the most prominent MIR literature on the topic of sampling and sample identification. In [5], Van Balen et al. introduce the problem of automatic sample identification and study a first approach by using peak-map-based audio fingerprinting (which we discuss at length in Chapter 3). In [40], Dittmar et al. present a toolbox to inspect cases of suspected music plagiarism, based on non-negative matrix factorization (NMF) of both the source and suspected target spectrograms. Whitney [144] describes a method for detecting samples in hip-hop music, focusing on robustness against the modifications that are typical for that genre. Whitney's method (inspired by [40]) also relies on a joint NMF of source and target, but the activation matrices are compared using techniques from image processing, rather than correlation, as in [40]. Gururani and Lerch [70] also use joint, semi-fixed NMF, but then train a random forest classifier with high-level features obtained from the activation matrices in order to analyze sampling cases. In [72, pp. 172] and [73], Hockman proposed the problem of *breakbeat classification*, offering an approach to identify which sampled breakbeat was used in a given composition.

## 2.2 Production

SBEM production began when affordable digital samplers became available for use at small (home) studios. According to Snoman [126], "a sampler can be viewed as the digital equivalent of an analogue tape recorder but rather than recording the audio signal onto a magnetic tape, it is recorded digitally into random access memory (RAM) or onto the computer's hard disk drive". Samplers introduced the possibility of playing multiple loops at the same time, which meant that producers could now take a drum break from one source and combine it with tonal or harmonic musical material sampled from other sources [118]. This gave rise to layering in SBEM, which is a central practice that drives the timbre, rhythm, and structure of SBEM tracks.

SBEM artists often begin creating their tracks by looping a sampled drum break or predominantly rhythmic pattern (potentially including bass) for the simple reason that it provides the timekeeping for the rest of the track's elements [38]. With this foundation in place, frequently supplied by the sampler, producers can then introduce further electronic instrumentation into their compositions, such as drum machines and various types of synthesizers. Figure 2.1 shows an overview of a simplified SBEM production setup.

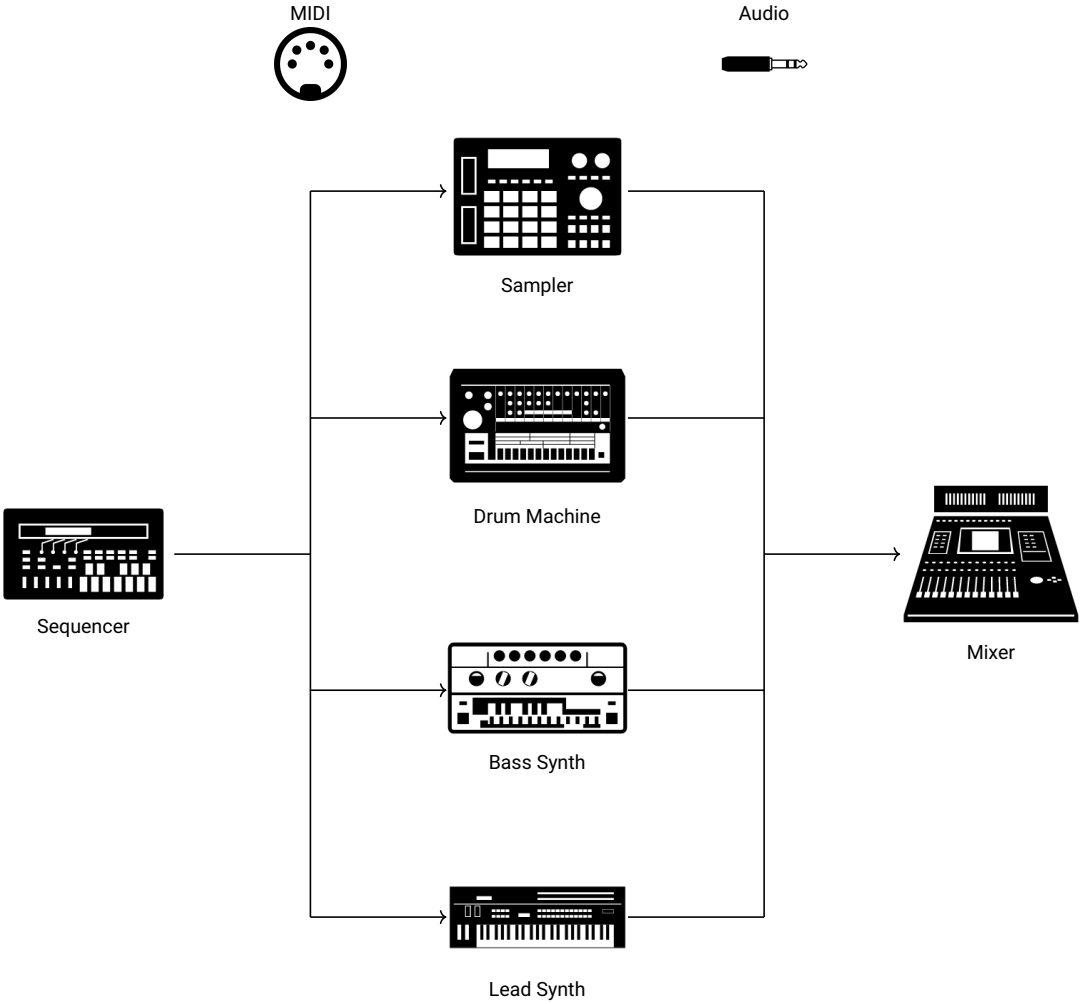



Figure 2.1: Typical hardware setup for producing SBEM.

We can think of this setup in terms of an orchestra: on the left side of Figure 2.1 we have the sequencer, which serves as a "conductor" by sending abstract musical instructions (or *messages*) to the individual instruments shown in the middle column of the figure. This is typically done using an industry-standard communication protocol called MIDI, or a more modern standard like OSC.[1] The instructions are carried out by the instruments: the sampler triggers (or mutes) a pattern stored in its internal memory, the synthesizer plays a sequence of specified musical notes, and the resulting audio signals are fed into a mixer for recording and further processing (right side of Figure 2.1). The fact that most drum breaks are only a few bars (musical measures) long, together with the limited memory available on early samplers, prompted a production style where short blocks of sound are introduced and removed at certain time intervals. This principle carried over into more modern production—as is the case with software and digital audio workstations—and its influence is present throughout SBEM. In Chapter 4 we formalize and model these basic principles that influence the structure of SBEM tracks. Some prominent genres

[1] For more information about MIDI we refer the reader to [97, pp. 13] and [126, pp. 61].

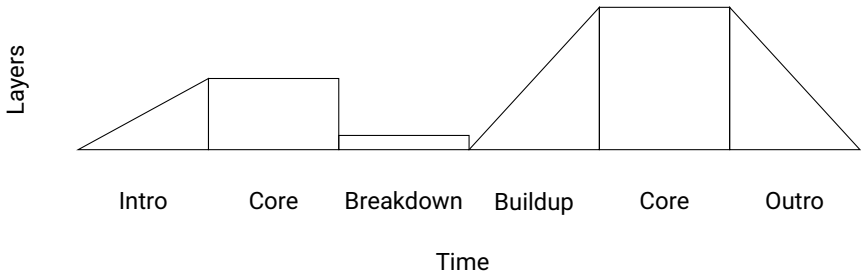



Figure 2.2: Prototypical form of electronic music tracks, taken from [14].

that emerged from DJ-oriented music (and as a result of sampling technology) are hardcore, jungle, and drum'n'bass (collectively referred to as HJDB in [72]). For an in-depth study on the history, production techniques, and computational analysis of HJDB, we refer the reader to [72].

This simple principle of layering, where elements are brought in and out of the mixture in a looping fashion, motivates a prototypical structure that is present in many types of electronic music and that Butler describes graphically in [14, pp. 222]. We have reproduced this representation in Figure 2.2, which shows a prototypical track structure: the horizontal axis represents time and the vertical axis corresponds to how many layers are active at a certain point in time. The number of active layers is strongly correlated with the perceived *energy* or *intensity* of that segment in a track, and musical *tension* is created by introducing and removing layers. During the intro (leftmost part of Figure 2.2), producers present the track's main elements, inducing familiarity in the listeners. The tension is built up by adding layers until reaching a first core. Then, the track drops to a timbrally sparse breakdown, which is the lowest point in the track, creating anticipation in the listeners for the more dominant elements to return. There is a second upward slope, this time reaching a core at a greater height, which means that certain elements are introduced which had been absent from the first core. This is the most intense part of the track, before progressively removing elements during the outro. In Section 2.3 we will see how this form also largely responds to the way DJs mix tracks in their sets.

## 2.3 DJing

The simplest and perhaps most widespread DJ setup consists of two *sources* or music-playing devices (such as vinyl turntables or professional CD players) and a mixer. In order to select music for mixing in their sets, DJs consider various musical aspects and levels. At the rhythmic level, they perform *beat matching*: making sure that two tracks in a mixture have the same tempo and their beats and downbeats are aligned. DJs can use headphones and an independent output on the mixer to test a mixture and make the necessary adjustments—while the audience continues to listen only to the main output and cannot hear this preparation. For this reason, tracks that start (and end) with a clear beat and a sparse layer of percussion are considered more *DJ-friendly*, since they are easier to beatmatch and do not oversaturate the

mixture with too many additional layers at one time [14, pp. 232]. Beyond correct beat matching, DJs may choose to also match the tonal and harmonic characteristics of the tracks they are mixing—if so, they are said to be "mixed in key" or avoiding "key clash". Finally, DJs can also use their knowledge and intuition to select tracks that have shared timbral characteristics, resulting in smoother and more pleasant transitions.

The increasing popularity of DJing and SBEM, together with the availability of DJ controllers and software for digital DJing, has motivated substantial research on understanding and automating the subtasks which can help digital DJs to quickly and easily create complex, sonically pleasing transitions, or help them find the next track to mix in from a vast library of digital music.

## 2.4 SBEM in MIR Literature

In the following we give an overview of MIR and audio processing literature related to SBEM. As a general note, Van Balen, Hein, and Brown offered a comprehensive tutorial [6][2,3] about why hip hop is interesting for the MIR community, touching upon many of the sampling-related issues discussed in this thesis. In one of the earliest works, Cliff [22] proposed an automated method capable of sequencing and seamlessly mixing dance music tracks. Along the same lines, Ishizaki et al. [75] developed a system for DJ mixing with optimal tempo adjustment and a function to estimate user discomfort. Kell and Tzanetakis [78] investigate the process of mix or playlist creation by means of timbre, key, loudness, and tempo analysis. Given an existing playlist, a recent approach by Bittner et al. [8] is capable of sequencing the tracks and adding DJ-sytle transitions between them.

On the topic of segmentation, Rocha et al. [115] put forth an algorithm to find structural boundaries and assess timbral similarity for electronic dance music (EDM), using audio data. By combining the use of audio with user-generated tags from online social platforms, Yadati et al. [146] developed a method to detect the *drop*: a moment where musical tension is released and the audience reacts by dancing vigorously. In [4], Aljanaki et al. also employ a multimodal approach to detect drops in EDM. Further work on the topic of drops can be found in [127], a study of the correlation between emotional experiences and production techniques. In [121], Seetharaman and Pardo exploit knowledge about layered composition practices in order to perform simultaneous segmentation and source separation of loop-based tracks. Smith and Goto [125] later approached source separation of loop-based tracks with non-negative tensor factorization. Focusing on the longer-range timescale of entire DJ sets, Scarfe et al. [116, 117] approach segmenting DJ mixes using self-similarity, and Glazyrin [66] details a system to segment DJ sets by using constant-Q log-spectrograms, self-similarity, and clustering. In what could be considered the converse task, Vande Veire and De Bie [139] developed a system capable of generating seamless mixes of drum'n'bass music.

[2] `https://wp.nyu.edu/musedlab/2016/08/02/why-hip-hop-is-interesting/`
[3] `https://wp.nyu.edu/ismir2016/event/tutorials/#hiphop`

As mentioned previously, DJs select and mix tracks based on rhythmic, timbral, and harmonic similarity. Faraldo et al. deal with the notion of key detection in EDM in [52], [53], and [108]. Gebhardt et al. repurpose the psychoacoustic concept of *roughness* as a further feature for harmonic compatibility in [62] and [63]. In order to enable large-scale testing of key and tempo detection algorithms for EDM, Knees et al. published two datasets (*GiantSteps Key* and *GiantSteps Tempo*) for these tasks in [80]. Bernardes et al. [7] used *GiantSteps Key* as one dataset to test their method for key detection with key mode bias. The annotations for *GiantSteps Tempo* were later refined by Schreiber and Müller in [119] by crowdsourcing the task of tapping along with musical excerpts. More recently, a dataset with 1878 works of "historic electronic music from 1950–1999" was published by Collins et al. in [28].

Regarding rhythm, Gärtner [61] presents a method for tempo detection in urban music based on NMF. Panteli et al. [104, 105] provide computational models and audio features for calculating timbral and rhythmic similarities between tracks—this can be used to organize large personal collections or help DJs in the process of track selection. In [74], Honingh et al. outline their web-based perceptual studies on timbral and rhythmic similarity in EDM. Gómez-Marín et al. [68, 67] investigate the concepts of EDM drum pattern similarity and drum spaces, based on similarity judgments. Vogl and Knees [140] designed an interface to generate rhythmic patterns in the context of EDM. In [100], Ó Nuanáin presents computational methods for generating and varying rhythmic patterns.

In [18], Casey and Slaney propose an algorithm for detecting remixes of pop songs, based on audio shingles and locality-sensitive hashing. Bryan and Wang [13] propose a system for analyzing musical influence networks at the song, artist, and genre levels. Collins [27] approaches a related question—determining musical influence of Chicago house and Detroit techno. Regarding the problem of track identification in DJ sets, Sonnleitner et al. [128, 129] presented a quad-based fingerprinting system robust to pitch-shifting and time-stretching—two very common signal modifications present in DJ sets.

Finally, we come to the topic of *mashups*, which are (often humorous) mixtures of two or more pop tracks. Creating mashups automatically requires all the aforementioned subtasks: beat tracking and matching, timbral and harmonic compatibility, and accurate structure analysis. In [33] and [34], Davies et al. detail systems to create mashups based on rhythmic and harmonic similarity (a real-time system is outlined in [36]). In [82], Lee et al. propose a system that can create mashups with multiple background and lead track segments. Finally, in [134], Tokui presents a web-based interface that allows collective mashup creation.

## 2.5 Further Literature

The following is a list of complementary resources in the greater context of SBEM.

In [10], Brewster and Broughton interview world-renowned DJs and artists related to electronic music, compiling a history across multiple genres. In [19], Chang collected interviews with hip-hop-related

personalities into a chronicle of how the genre emerged. In a further genre-specific work, James explores jungle and drum'n'bass in [76]. On the topic of funk, Davis [37] studies the origins of this genre and proposes a "framework for understanding the process of genre formation." Funk music—and especially James Brown's "Funky Drummer" are central to drum breaks and breakbeat culture. Along these lines, Danielsen [30] analyzes several pieces by James Brown and Parliament, with particular attention to structure and rhythm—and then focuses on microrhythm (*groove*) as found in "Funky Drummer" in [31]. A fundamental question in jazz research is *What makes music swing?* In [58], Friberg and Sundström measure the swing ratio produced by drummers on the ride cymbal. Prögler [110] analyzes several microrhythmic traits including swing, and how they emerge from the interaction between (jazz) musicians. In [130] Stewart investigates how American musicians influenced each other in terms of microrhythm and swing. Finally, although production techniques are mostly out of the scope of this thesis, it is important to mention *trackers* as an early example of computer programs used for producing SBEM; Collins reviews the topic of trackers in [23]. Furthermore, in [133], Taylor explores the intersection between music and technology at different times throughout history.

# Chapter 3

# Fingerprinting in Electronic Music

This chapter is the result of collaborating with and supervising the work of Pedro Solórzano, resulting in a Master's thesis [94]. Together with Meinard Müller, we proposed *peak survival* as a method to evaluate constellation map degradation. My main contributions in this chapter are the problem formalization, dataset preparation, experimental design, and verification.

In this chapter we first give a brief overview of audio signals and the Fourier transform—two fundamental concepts that are prerequisite not only for the remainder of the chapter, but also for the rest of the thesis. Section 3.1, closely following the formulations from [97, Chapter 2], deals with key topics. Then, from Sections 3.2 to 3.9, we discuss the task of audio fingerprinting in the context of SBEM.

## 3.1 Audio Signals and Fourier Analysis

Sound travels in the form of waves—compressions and rarefactions of a medium (such as air) with a certain *amplitude*. Following [97, Section 2.2.1], one way of expressing this phenomenon is as an analog signal, a function $f : \mathbb{R} \to \mathbb{R}$ which assigns an amplitude value $f(t) \in \mathbb{R}$ to each time point $t \in \mathbb{R}$. Also called a *continuous time* (CT) signal, this definition allows us to model infinitesimal variations in time and amplitude. However, when dealing with signals on a computer, we must resort to discrete-time (DT) representations, commonly achieved by digitization: a combination of *sampling* and *quantization*. We refer the reader to [97, Section 2.2.2.2] for a more detailed description of quantization, and proceed to discuss sampling. In Chapter 2 we used the term `sampling` in the artistic sense, referring to the musical practice of repurposing previously existing music recordings in new works. In the following we will discuss sampling in signal processing (see also [97, pp. 386]). According to [97, Section 2.2.2.1], the

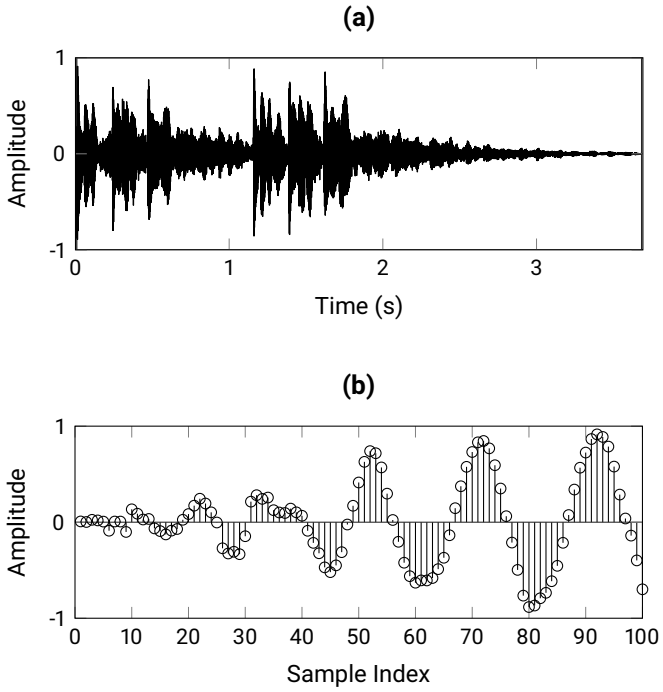



Figure 3.1: **(a)** Waveform of a synthesizer sound sampled at $F_s = 22050\,\text{Hz}$. **(b)** Detailed view of the first 100 samples from the signal in (a).

most common procedure to transform a CT signal $f : \mathbb{R} \to \mathbb{R}$ into a DT signal $x : \mathbb{Z} \to \mathbb{R}$ is equidistant sampling. If we fix a positive real number $T > 0$, then we have

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

for $n \in \mathbb{Z}$. The particular value $x(n)$ is called a *sample*, taken at time $t = n \cdot T$. The number $T$ is the sampling period, and its inverse is called the sampling rate:

$$F_s := \frac{1}{T}. \tag{3.2}$$

The sampling rate is measured in Hertz (Hz), telling us how many samples per second were extracted from the CT signal. Note that in Eq. 3.1, the signal has an infinite length (defined for $n \in \mathbb{Z}$). Again, since we are dealing with signals on a computer, we have to restrict the signal $x$ to a finite length $N \in \mathbb{N}$, considering only a finite set of samples $x(0),\ x(1), \ldots,\ x(N-1)$. For simplicity, we also set $x(n) = 0$ for $n \in \mathbb{Z} \setminus [0 : N-1]$, where $[0 : N-1] := \{0, 1, \ldots, N-1\}$. Figure 3.1a shows the waveform for a digital music signal sampled at 22050 Hz. In the signal, the musical note sequence C6-G5-C5 is played twice with a synthesizer: once at 0 s and then shortly after the one-second mark. Figure 3.1b is a zoomed-in version, showing the first 100 samples of the same signal.

Music signals are complex mixtures of sound sources and if we consider that typical sampling rates for audio are 44100 Hz or 22050 Hz, it quickly becomes apparent that processing waveforms (in the time

domain) can be a computationally costly task. This is where the Fourier transform comes into play. We can trace the history of the Fourier transform (FT) to the eighteenth century, with two main problems that motivated what would become Fourier analysis: describing the vibration of a string anchored at both ends and determining the orbits of celestial bodies [12].

In general terms, the FT can help us better understand signals by decomposing them into a *spectrum* of sinusoidal components. Thus, the FT converts time-dependent representations into frequency-dependent representations. On a related note, it is important to mention that human hearing is also spectral in nature and the FT is particularly useful for approximating our perception of sound. Following [97, Section 2.4.2], we define the *discrete* Fourier transform (DFT) of length $N \in \mathbb{N}$ for a DT signal $x$ as

$$X(k) := \sum_{n=0}^{N} x(n) \cdot \exp(-2\pi i k n/N) \tag{3.3}$$

for $k \in [0 : N-1]$. Each complex value $X(k) \in \mathbb{C}$ is called a *Fourier coefficient*, which encodes the magnitude and phase of a sinusoidal component with frequency

$$F_{\text{coef}}(k) := \frac{k \cdot F_{\text{s}}}{N}. \tag{3.4}$$

Since the DFT is invertible, if we sum these components we can obtain the original time-domain signal by

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) \cdot \exp(2\pi i k n/N), \tag{3.5}$$

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

Figure 3.2b shows the magnitude Fourier spectrum for the synthesizer signal. We can now clearly see the peaks for the notes C5 ($\sim$523 Hz), G5 ($\sim$784 Hz), and C6 ($\sim$1046 Hz)—the individual frequencies are hard to glean from the waveform in Figure 3.2a. On the other hand, only by looking at the spectrum, it is no longer clear *when* each note was played. Since the analysis function has a non-local nature, the frequency information is always averaged over the entire time domain [97, Section 2.5].

To address this shortcoming, Gabor [60] introduced the short-time Fourier transform (STFT). The STFT partitions the signal into smaller segments (also called *frames*) and applies the FT at a local scale to each one. Thus, the STFT offers frequency information that can be assigned to a given time interval. Let us define the STFT in formal terms, following [97, Section 2.5.3]. If $x$ is a DT signal, $w$ is a discrete window function of length $N$ with coefficients $w(n)$ for $n \in [0 : N-1]$, and $H \in \mathbb{N}$ is the *hop size* in samples, then

$$\mathcal{X}(m,k) := \sum_{n=0}^{N-1} x(n+mH) \cdot w(n) \cdot \exp(-2\pi i k n/N) \tag{3.6}$$

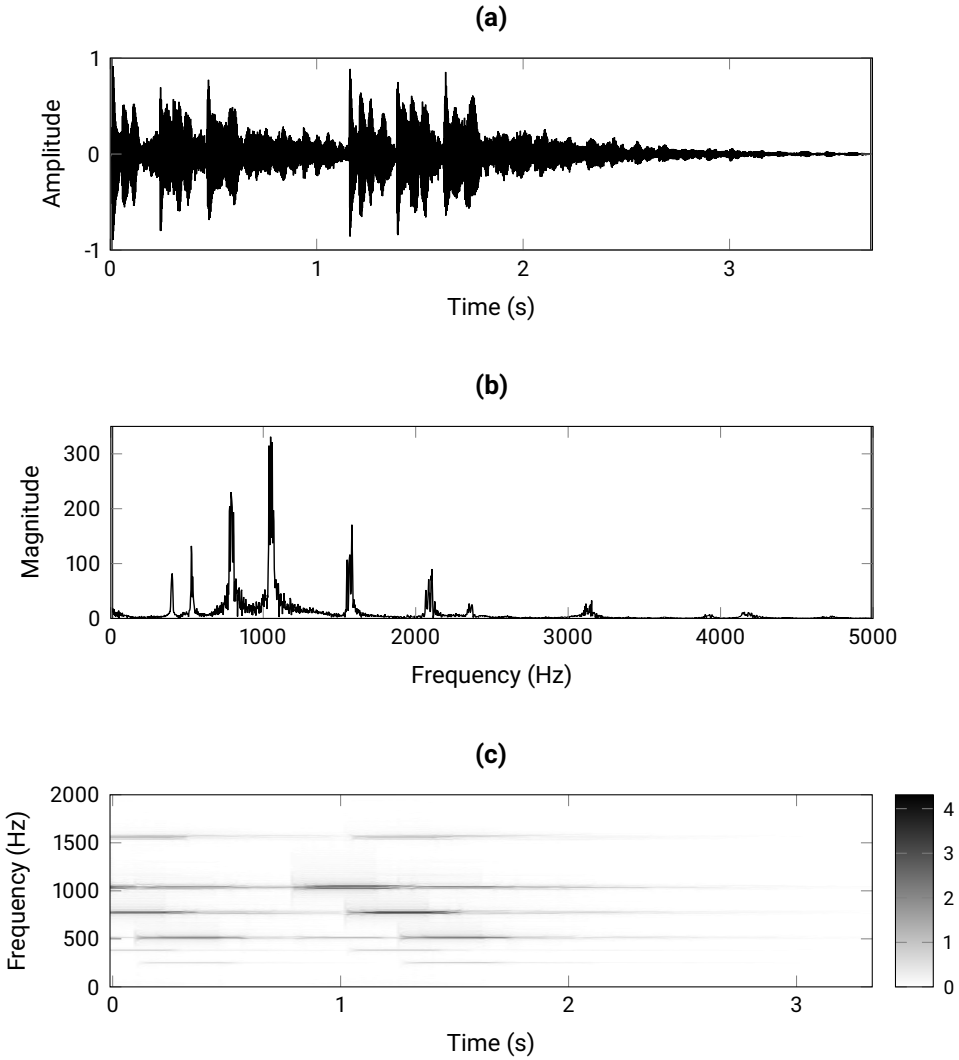



Figure 3.2: **(a)** Waveform of a synthesizer sound. **(b)** Magnitude Fourier spectrum. **(c)** Magnitude spectrogram.

is the *discrete* STFT, or STFT of $x$. In $\mathcal{X}(m,k)$, $m \in \mathbb{Z}$ is the *frame index* and $k \in [0 : N-1]$ is the frequency coefficient. The frame index $m$ corresponds to the physical time position $T_{\text{coef}}$, such that

$$T_{\text{coef}}(m) := \frac{m \cdot H}{F_{\text{s}}}. \tag{3.7}$$

Looking back at Figure 3.2c, there is indeed both time *and* frequency information for the synthesizer signal: the dark horizontal lines which start at 0 s and 1 s show the notes that were played, at their temporal location.

Now that we have defined these fundamental concepts, we will discuss a first application—audio fingerprinting—in the following sections of this chapter. Fourier analysis is of central importance to MIR and audio processing—we will encounter these topics throughout the remainder of this thesis.

## 3.2 Audio Fingerprinting

Audio fingerprinting was developed to match a recording of a fragment of music (obtained in potentially adverse conditions, subject to environmental noise, reverb, or compression codec artifacts) against a database containing an undistorted version. Fingerprinting typically relies on *constellation maps* or *peak maps*—a binary feature representation which captures local maxima and is resilient to a certain amount of acoustic interference. The original intent of fingerprinting is to answer the question *does a (potentially modified) audio query match any documents in the database?* We deviate from this objective and study constellation maps in the context of electronic music (EM), where existing recorded sounds or music excerpts (called samples) are reused, for instance, as loops (we use the terms *loop*, *pattern* and *sample* interchangeably). Additional instruments are often mixed with the sample, and several samples may be stacked on top of each other, creating *musically* modified versions of the original query. In this chapter we use two experiments to test the limits of reliable fingerprint matching at a lower level by studying the *peak agreement*—a measure of how much information is shared between the constellation maps of an unmodified musical fragment, and one modified by various means. First, we quantify the peak agreement between a loop and versions overlaid with increasing amounts of white noise. Secondly, we study the effects of progressively adding *musically coherent* loops to the query (i.e., further loops with the same tempo and genre).

## 3.3 `EM-Mini` Dataset Description

In order to study peak maps in a controlled EM scenario, we compiled a dataset of 111 loops originally included with *MAGIX Music Maker Premium*—a digital audio workstation (DAW) which focuses on loop-based EM production (we refer to it as the `EM-Mini` dataset). At the topmost level, the dataset is

Table 3.1: Overview of the `EM-Mini` dataset by genre [94].

| Genre | Number of loops | Number of timbral families | BPM | Length in bars |
|---|---|---|---|---|
| Dance | 10 | 10 | 130 | 2, 4, 8 |
| Deep House | 19 | 11 | 120 | 1, 2, 4 |
| Dubstep | 19 | 8 | 135 | 4, 8 |
| HipHop | 20 | 12 | 90 | 1, 2, 4 |
| Techno | 23 | 8 | 130 | 1, 2 |
| Trap | 20 | 8 | 75 | 1, 2, 4 |

Table 3.2: Overview of the `EM-Mini` dataset by timbral family [94].

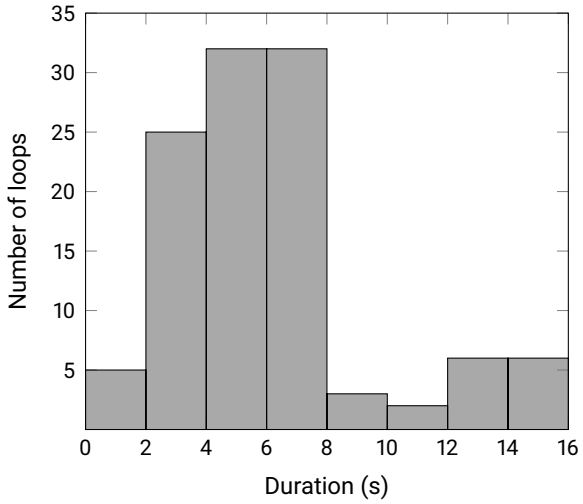
| Timbral family | Number of loops | Number of genres | BPM | Length in bars |
|---|---|---|---|---|
| Bass | 13 | 6 | 75, 90, 120, 130, 135 | 1, 2, 4 |
| Brass | 3 | 3 | 75, 90, 120 | 2, 4 |
| Drums | 22 | 6 | 75, 90, 120, 130, 135 | 2, 4, 8 |
| Fx | 11 | 6 | 75, 90, 120, 130, 135 | 2, 4 |
| Guitar | 2 | 2 | 90, 130 | 2, 8 |
| Keys | 4 | 4 | 90, 130, 135 | 2, 4 |
| Lead | 2 | 1 | 135 | 4, 8 |
| Mallet | 1 | 1 | 130 | 2 |
| Pad | 7 | 6 | 75, 90, 120, 130, 135 | 2, 4 |
| Percussion | 8 | 3 | 120, 130 | 1, 2, 4 |
| Sequence | 15 | 6 | 75, 90, 120, 130, 135 | 1, 2, 4 |
| Special | 3 | 1 | 120 | 2 |
| Strings | 3 | 2 | 90, 120 | 2, 4 |
| Synth | 7 | 5 | 75, 90, 120, 130, 135 | 1, 2, 4 |
| Vocals | 9 | 4 | 75, 90, 120, 130 | 2, 4, 8 |
| Vocal Rap | 1 | 1 | 90 | 1 |

organized by genre—loops of a given genre have the same tempo, but may have different lengths (1, 2, 4, or 8 bars). For an overview of how loop lengths are distributed within the dataset, see Figure 3.3. Within each genre, the loops are categorized by *timbral family*, a loose designation of the type of synthesizer they were made with. Tables 3.1 and 3.2 (taken from [94]) provide a summary of the dataset's composition. The timbral families are unevenly distributed among genres; for instance, *bass* and *drum* loops are present in all genres, but *lead* is only available within the *dubstep* genre. Additionally, we classified all loops into more general categories *harmonic* (H), *residual* (R) and *percussive* (P), inspired by [48]. This was done manually, according to subjective timbral criteria. The H, R, and P categories contain 42, 39, and 30 loops, respectively.

## 3.4 Fingerprinting with Peak Maps

We implement constellation maps as outlined in [90]. First, we convert the loops in the `EM-Mini` dataset to mono and a sampling frequency $F_s = 22050\,\text{Hz}$. For each individual loop, as well as each version modified through the addition of noise or further loops, we compute an STFT $\mathcal{X}$ such that $\mathcal{X}(n,k)$ refers to the $k^{\text{th}}$ Fourier coefficient for the $n^{\text{th}}$ time frame, where $k \in [0 : K-1]$ and $n \in \mathbb{Z}$. We use the following parameters: block size $N = 4096$, hop size $H = N/2$ and a Hann window.

Figure 3.3: Distribution of loop durations in the `EM-Mini` dataset.




The next step is to compute magnitude spectrograms $\mathcal{Y} = |\mathcal{X}|$ (Figure 3.5a). We then map $\mathcal{Y}$ to a logarithmically spaced frequency axis with a lower cutoff frequency of 32 Hz, an upper cutoff frequency of 8000 Hz and spectral selectivity of 36 bins per octave, resulting in $\mathcal{Y}_{\mathrm{LF}}$ (Figure 3.5b). Under these STFT settings, the 36-bin spectral selectivity does not hold in the two lowest octaves; however, their spectral peak contribution is negligible.

We now apply log-compression to $\mathcal{Y}_{\mathrm{LF}}$, obtaining $\mathcal{Y}_{\mathrm{LFC}} = \log(1 + \mathcal{Y}_{\mathrm{LF}})$ (Figure 3.5c). Then, we apply an exponential moving average to smooth the spectrograms $\mathcal{Y}_{\mathrm{LFC}}$ over time, defined as:

$$\mathcal{Y}_e(n,k) = \max\left(\alpha \cdot \mathcal{Y}_e(n-1,k) + \mathcal{Y}_{\mathrm{LFC}}(n,k) \cdot (1-\alpha), \mathcal{Y}_{\mathrm{LFC}}(n,k)\right) \tag{3.8}$$

for $n > 0$ and $k \in [0 : K-1]$. In the case where $n = 0$ (the first STFT frame), we take $\mathcal{Y}_e(0,k) = \mathcal{Y}_{\mathrm{LFC}}(0,k)$. In our implementation, we use a smoothing factor $\alpha = 0.9$. Exponential smoothing assigns exponentially decreasing weights over time—in contrast to the simple moving average, where past observations are weighted equally [145]. In Equation 3.8, the maximum operator allows us to only affect falling frequency slopes along time, but not rising slopes. This leads to a smoothing along the time axis where onsets are emphasized (Figure 3.5d).

Although our scenario is slightly different to that of audio fingerprinting and identification, both require a feature representation which captures an individual pattern's characteristics despite the superposition of further sound sources. To this end, we use spectral peak maps (or constellation maps) as described in [97]; we will first give the theoretical details and then outline our efficient implementation. The idea behind peak maps is to summarize the STFT as a sparse set of time-frequency (TF) points. Peak picking is used to identify TF bins with a magnitude greater than that of their neighbors. Let $\tau > 0$ and $\kappa > 0$ be

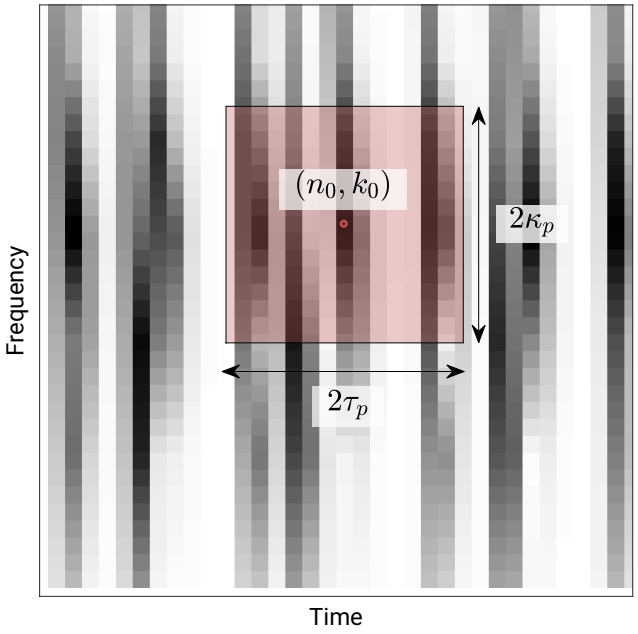



Figure 3.4: A TF bin indexed by $(n_0, k_0)$ and the corresponding neighborhood $\mathcal{N}_{\tau_p,\kappa_p}(n_0,k_0)$ that surrounds it (red square).

the neighborhood size parameters in time and frequency, respectively. We now define the neighborhood $\mathcal{N}_{\tau,\kappa}$ around a TF bin $(n_0,k_0)$ as

$$\mathcal{N}_{\tau,\kappa}(n_0,k_0) = \{(n,k) : |n-n_0| \leq \tau \wedge |k-k_0| \leq \kappa\}. \tag{3.9}$$

For this peak picking task, we set $\tau = \tau_p = 7$ and $\kappa = \kappa_p = 7$—in Section 3.6 we will define $\tau_e$ and $\kappa_e$ as tolerance parameters for evaluation purposes. Thus, a TF bin $(n_0,k_0)$ is selected as a peak if $|\mathcal{X}(n_0,k_0)| \geq |\mathcal{X}(n,k)|$ for all $(n,k) \in \mathcal{N}_{\tau_p,\kappa_p}(n_0,k_0)$ (see Figure 3.4)

In practice, we convolve $\mathcal{Y}_e$ with a $\tau_p \times \kappa_p$ maximum filter, obtaining $\mathcal{Y}_{\max}$ (Figure 3.5e). This is equivalent to:

$$\mathcal{Y}_{\max}(n_0,k_0) = \max(\mathcal{Y}_e(n,k) \mid (n,k) \in \mathcal{N}_{\tau_p,\kappa_p}(n_0,k_0)). \tag{3.10}$$

Finally, to obtain the peak map $\mathcal{C}$ (Figure 3.5f), we use the following definition:

$$\mathcal{C}(n,k) = \begin{cases} 1 & \text{if } (\mathcal{Y}_e(n,k) = \mathcal{Y}_{\max}(n,k)) \wedge (\mathcal{Y}_{\max}(n,k) > 1) \\ 0 & \text{otherwise.} \end{cases} \tag{3.11}$$

Figure 3.6 shows $\mathcal{Y}_{\text{LFC}}$ overlaid with corresponding peak maps $\mathcal{C}$ for nine representative loops from the dataset. Each row corresponds to a category H, R, or P (top to bottom). Conceptually, we are following an early approach for loop retrieval inside hip hop recordings which was presented in [137] and later refined

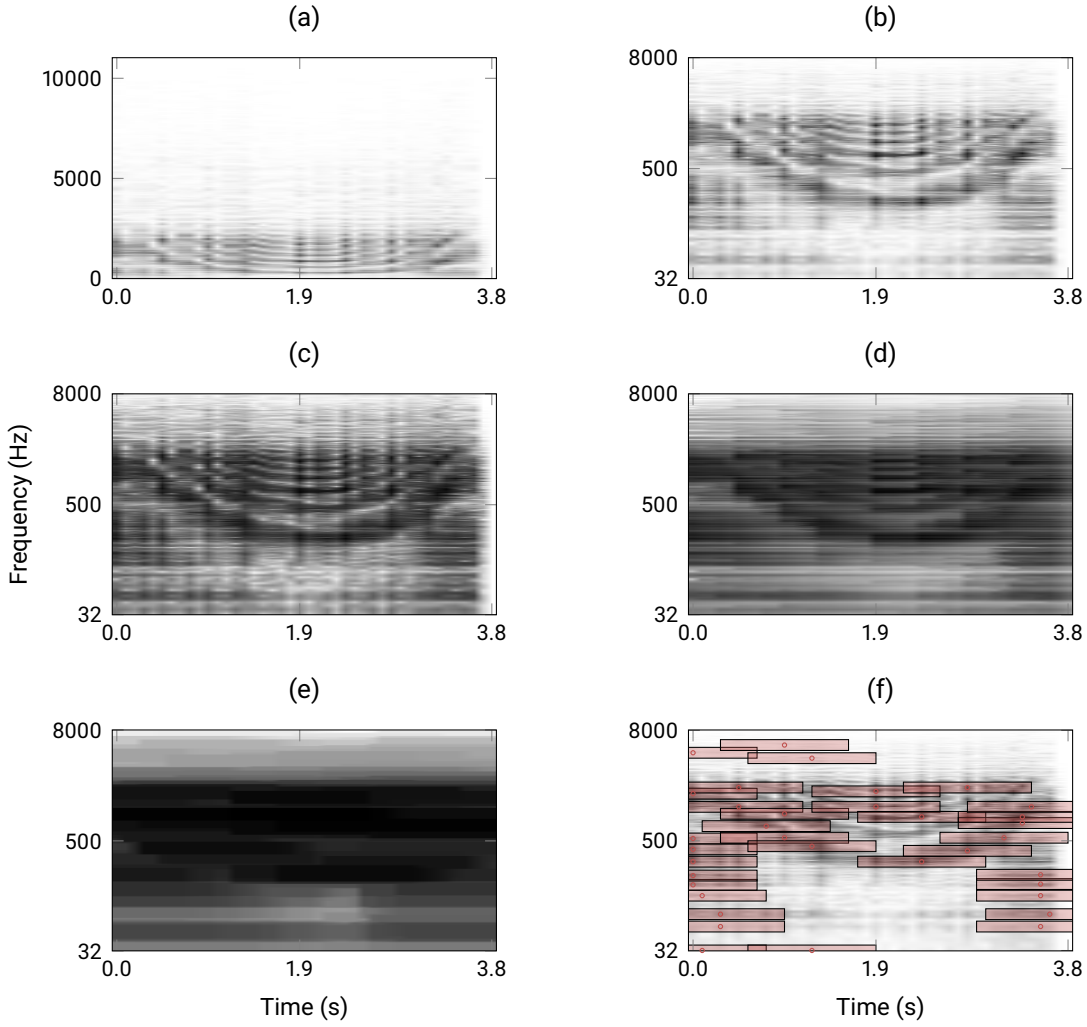



Figure 3.5: Steps to compute peak maps, illustrated with a loop from the residual (R) category of the `EM-Mini` dataset. **(a)** Magnitude spectrogram $\mathcal{Y}$ with a linear frequency axis. **(b)** Log-frequency spectrogram $\mathcal{Y}_{\mathrm{LF}}$. **(c)** Log-compressed spectrogram $\mathcal{Y}_{\mathrm{LFC}}$. **(d)** After exponential decay, $\mathcal{Y}_e$. **(e)** After max filtering, $\mathcal{Y}_{\max}$. **(f)** $\mathcal{Y}_{\mathrm{LF}}$ overlaid with the resulting peak map (local maxima shown as red circles). The red rectangles that enclose each local maximum represent the neighborhood $\mathcal{N}_{\tau,\kappa}$. Although $\tau_p = \kappa_p = 7$, the neighborhoods appear larger along the time axis, given that there are fewer STFT frames than feature dimensions, affecting the aspect ratio.

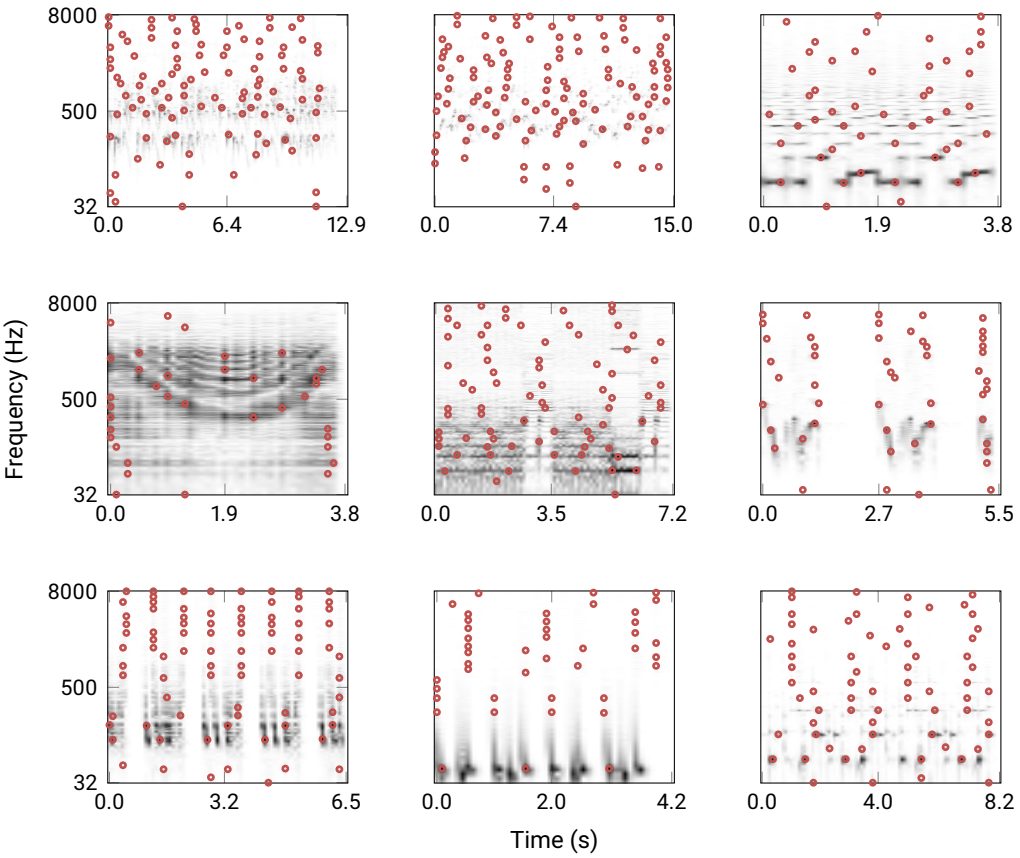



Figure 3.6: Log-frequency spectrograms (LS) overlaid with corresponding peak maps (PLS) for nine loops from the `EM-Mini` dataset. Each row contains loops from the harmonic (H), residual (R), or percussive (P) category.

| STFT Parameters | | Log-Frequency Parameters | | Peak Map Parameters | |
|---|---|---|---|---|---|
| Sampling frequency | 22 050 Hz | Lower cutoff | 32 Hz | Window dimensions | $\tau_p = 7$ |
| Block size | 4096 samples | Upper cutoff | 8000 Hz | | $\kappa_p = 7$ |
| | 64 ms | Spectral Selectivity | 36 bins per octave | 7 time frames | 0.65 s |
| $\Delta f$ | 5.39 Hz | | | 7 freq. bins | 40.38 Hz (linear) |
| Hop size | 2048 samples | | | | 232.9 cents (log) |
| $\Delta t$ | 92.9 ms | | | | |
| Window type | Hann | | | | |

Table 3.3: Overview of implementation parameters.

in [138]. Their method is based on a modification of the fingerprinting procedure originally described in [141]. For each time-frequency bin in the respective log-frequency spectrogram (LS), a rectangular analysis window is constructed. The maximum value within each window is kept (with the value 1 on the output) and all neighbors are set to 0 on the output.

In further sections we only use LS to compute peak maps, given their superior performance when compared to magnitude spectrograms (MS). For more details on this comparison, refer to [90] and [94].

## 3.5 Reference Implementation

Wang [141] did not supply a reference implementation of the Shazam algorithm, as it is industrial intellectual property. However, Ellis [50] provides `MATLAB` source code that follows the functionality outlined in [141]. It is a full-fledged implementation of the fingerprinting pipeline, which we describe in the following. Let's assume that we have a collection of audio tracks that we wish to add to a database for later matching. For each track, the algorithm first computes a peak map. To compute peak maps, the audio signal is first converted to mono and a sampling frequency $F_s = 8000$ Hz. Then, an STFT is computed with a window size of 512 samples (64 ms), a hop size of 256 samples (32 ms) and a *Hanning* window. The magnitude spectrogram is extracted and logarithmically compressed. This log-compressed spectrogram is made zero-mean (i.e., by subtracting the overall mean from each element), and then filtered with a high-pass filter. Ellis [50] comments in the source code that the high-pass filter blocks slowly varying terms and emphasizes onsets. These peak maps have a target density of 10 landmarks per second. Ellis' implementation uses a frequency neighborhood parameter $\kappa = 30$, corresponding to a half-width of 468.75 Hz. Furthermore, up to 5 peaks per STFT frame are allowed. Each peak map is summarized as a set of *hashes* or *landmarks*: a hash is an integer representation of a pair of peaks in the constellation map. These hashes are now added to the database—they allow efficient lookups in a database of considerable size. For the matching stage, the algorithm also computes peak maps and then hashes for a query excerpt. It compares the hashes stored in the database with those of the query, returning a ranked list of matching database documents.

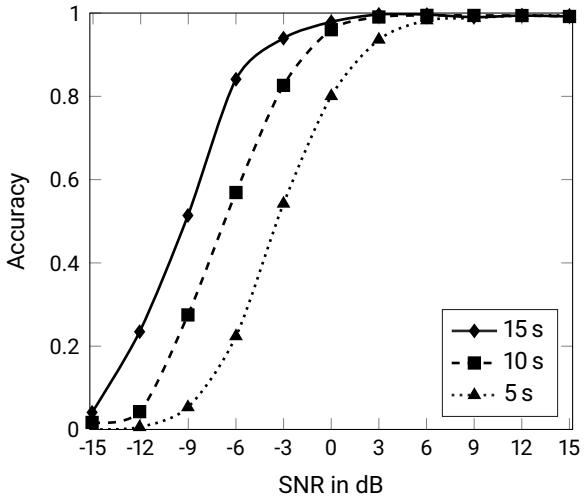



Figure 3.7: Shazam algorithm performance when matching queries recorded with pub noise at different SNR levels (horizontal axis). Each curve shows a particular query length. (Taken from [141])

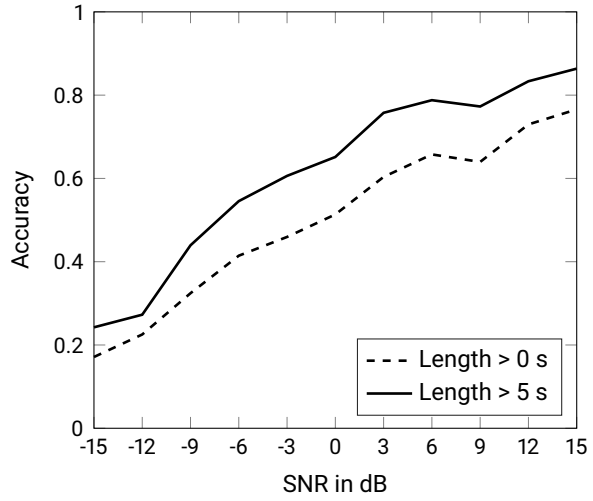



Figure 3.8: Recognition rate for queries from our `EM-Mini` dataset. The dashed line is the average across all loops in the dataset, the solid line is the average for all loops with a length greater than 5 s.

One way of evaluating a matching algorithm is through the *recognition rate*. For a number of queries—which are often distorted—the recognition rate (or accuracy) measures how often the system returns the correct database document as a top match. Figure 3.7 was taken from [141] and shows the recognition rate under additive pub noise, from $-15$ to 15 dB SNR. Each of the three curves shows results for a particular query length in seconds.

We used Ellis' implementation [50], without modifications, to compute the recognition rate with the `EM-Mini` dataset loops, modified with increasing amounts of white noise. Our results are shown in Figure 3.8: the dashed line is the average recognition rate for the entire dataset (irrespective of length) and the solid line is the average for all loops whose length is greater than five seconds.

Figures 3.7 and 3.8 are not directly comparable for a number of reasons. First, Wang's implementation [141] is unavailable and probably contains undisclosed techniques that improve recognition. Second, Wang used pub noise, and we used white noise. Third, the underlying datasets are different: our `EM-Mini` dataset

Table 3.4: Parameters for fingerprinting implementation by Ellis [50].

| Ellis' [50] STFT Parameters | |
|---|---|
| Sampling frequency | 8000 Hz |
| Block size | 512 samples |
| | 64 ms |
| $\Delta f$ | 15.6 Hz |
| Hop size | 256 samples |
| $\Delta t$ | 32 ms |
| Window type | Hanning |

contains 45 loops that are shorter than five seconds (see Figure 3.3) and is confined to electronic musical genres; Wang used 250 excerpts (from a database of 10,000 popular music pieces) whose length is at least 5 seconds. The overall lower performance in Figure 3.8 serves to highlight the difficulty of matching audio at a smaller scale (e. g., loops in EM), as opposed to longer passages in an entire pop song.

Figure 3.8 is meant to give context for fingerprinting loops in electronic music—it reflects the output when using an end-to-end reference implementation of the Shazam algorithm. In the following sections we concentrate on peak maps; we wish to better understand how they behave, given that landmark and hash computation both rely on the results of this previous stage.

## 3.6 Peak Agreement

Once we have computed the peak maps for both the original loop and the modified version, we wish to quantify how much shared information remains and how many peaks are added by the particular modification. Formally, we refer to the original loop as the *database* $\mathcal{D}$ and the modified version as the *query* $\mathcal{Q}$, conforming to the typical information retrieval concepts. Their constellation maps $\mathcal{C}_\mathcal{Q}$ and $\mathcal{C}_\mathcal{D}$ are overlaid and compared by using Boolean operators to determine the *recall*, answering the question *how many peaks survive the modification?*, as well as the *precision*, telling us how many *spurious* or additional peaks the modification has produced. Furthermore, we can use the definition of a neighborhood (Equation 3.9) to introduce a *tolerance* around the peaks in the database, leading to a potential increase in true positives. In our implementation, we have used $\tau = \tau_e = 1$ and $\kappa = \kappa_e = 1$, or one TF bin in every direction around each peak in $\mathcal{C}_\mathcal{D}$. Figure 3.9 illustrates an example of *peak agreement* between two PLS.

In Figure 3.9, the lighter, larger squares represent peaks in the original constellation map $\mathcal{C}_\mathcal{D}$, extended by a tolerance window of one TF bin in every direction. The smaller squares are peaks in the constellation map of a modified version $\mathcal{C}_\mathcal{Q}$. Four evaluation cases are surrounded by a red circle: (1) is a perfect match (true positive) between the original and modified versions – since it is centered, it would remain a true positive even without a tolerance window; (2) is a peak present in the modified version, but not in the original – thus, a false positive; (3) is a peak in the original version which remains unmatched by the modified version – a false negative; (4) is a true positive, but only because of the tolerance window. Both

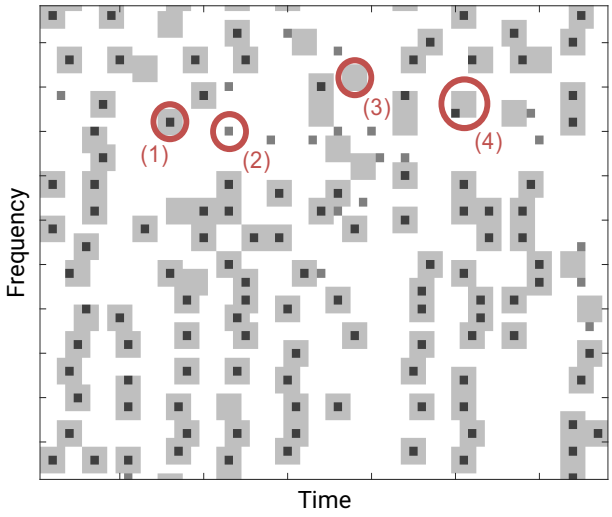



Figure 3.9: Peak agreement between an unmodified loop and one modified by adding noise. Large squares correspond to the peaks (plus tolerance) of the unmodified loop. Small squares correspond to the modified loop. Four cases are surrounded by a red circle. (1): True positive, even without tolerance. (2): False positive. (3): False negative. (4): True positive, due to tolerance.

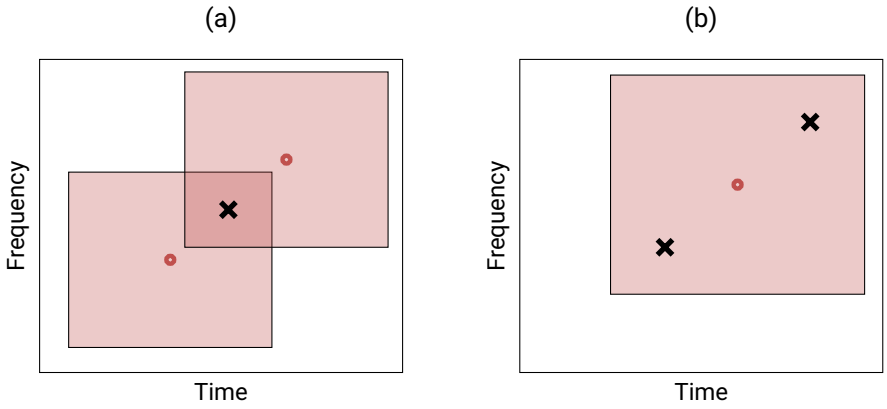



Figure 3.10: Two potentially problematic cases when using a tolerance window for evaluation. Red dots are a peak in $\mathcal{C}_{\mathcal{D}}$, surrounded by a tolerance window (red square). Black crosses are peaks in $\mathcal{C}_{\mathcal{Q}}$. **(a)** The tolerance windows for two peaks in $\mathcal{C}_{\mathcal{D}}$ overlap; they both contain a matching peak from $\mathcal{C}_{\mathcal{Q}}$. **(b)** The tolerance window for a single peak in $\mathcal{C}_{\mathcal{D}}$ contains two matching peaks from $\mathcal{C}_{\mathcal{Q}}$.

peak maps are shown with *transparency*: the smaller squares in (1) and (4) are darker because of the overlap between $\mathcal{C}_{\mathcal{D}}$ and $\mathcal{C}_{\mathcal{Q}}$; the small square in (2) and the large square in (3) are comparatively lighter because there is no overlap between the peak maps.

Using a tolerance window for evaluation can be problematic, as shown in Figure 3.10. In Figure 3.10a we show two peaks from $\mathcal{C}_{\mathcal{D}}$ (red dots), surrounded by their overlapping tolerance windows (red squares). In the overlapping region, a peak from $\mathcal{C}_{\mathcal{Q}}$ (black cross) is a potential true positive for both peaks in the database. In Figure 3.10b, two different peaks from the query can match a single peak from the database. These problems do not affect our implementation since we use peak picking windows which are considerably larger than the evaluation tolerance: recall that $\tau_p = \kappa_p = 7$ and $\tau_e = \kappa_e = 1$.

In the following sections we use peak agreement to evaluate constellation maps under various loop modifications and degradations.

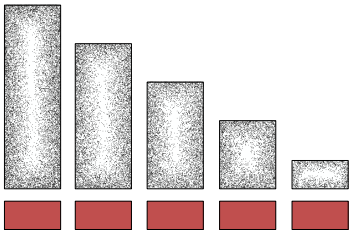


Figure 3.11: Adding white noise: the query (red rectangles, bottom row) is mixed with white noise (gray rectangles, height denotes power). Each column represents a single track.

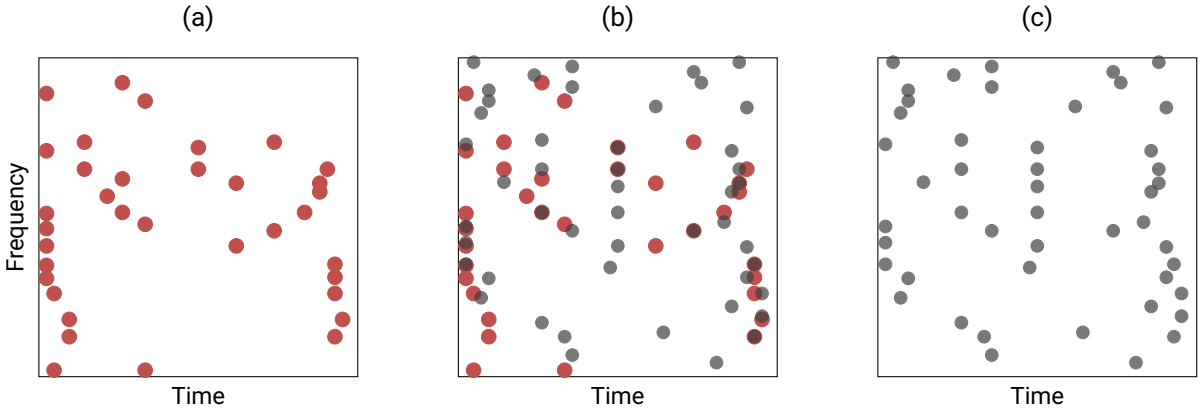



Figure 3.12: Peak map comparison when adding white noise to a loop from the *residual* category. **(a)** Peak map for unmodified loop (red dots). **(b)** Peak map for unmodified loop (red dots) overlaid with peak map for the same loop with white noise added at 0 dB SNR (grey dots). **(c)** Peak map for loop with white noise at 0 dB SNR (grey dots).

## 3.7 Adding White Noise

We illustrate the concept of these experiments in Figure 3.11, where the query (red rectangles, bottom row) is mixed with white noise (gray rectangles, height denotes power) and each column represents a single track. Figure 3.12 shows the effects of adding white noise to a loop in the residual (R) category. In Figure 3.12a we show the peak map for the unmodified loop (grey dots); Figure 3.12b shows the peak map for the unmodified loop (grey dots) overlaid with the peak map for the same loop with white noise added at 0 dB SNR (red dots); in Figure 3.12c we show the peak map for the loop with white noise added at 0 dB SNR (red dots).

Figure 3.13 shows the mean peak agreement recall and precision under modification by white noise. For each of the 111 loops in our dataset we generated 11 modified versions (corresponding to the SNR values shown along the horizontal axis). For a given SNR value, we show the mean recall and precision across the entire dataset: the solid blue curves were computed without any tolerance and the dashed red curves were computed with a tolerance of one TF bin in every direction (see also Figure 3.9).

Beyond the results for white noise across the entire dataset (Figure 3.13), we now investigate what role the timbral category (harmonic, residual, or percussive) plays in peak agreement.Figure 3.14 shows recall and precision, averaged by timbral category. They were computed without a tolerance window and contain

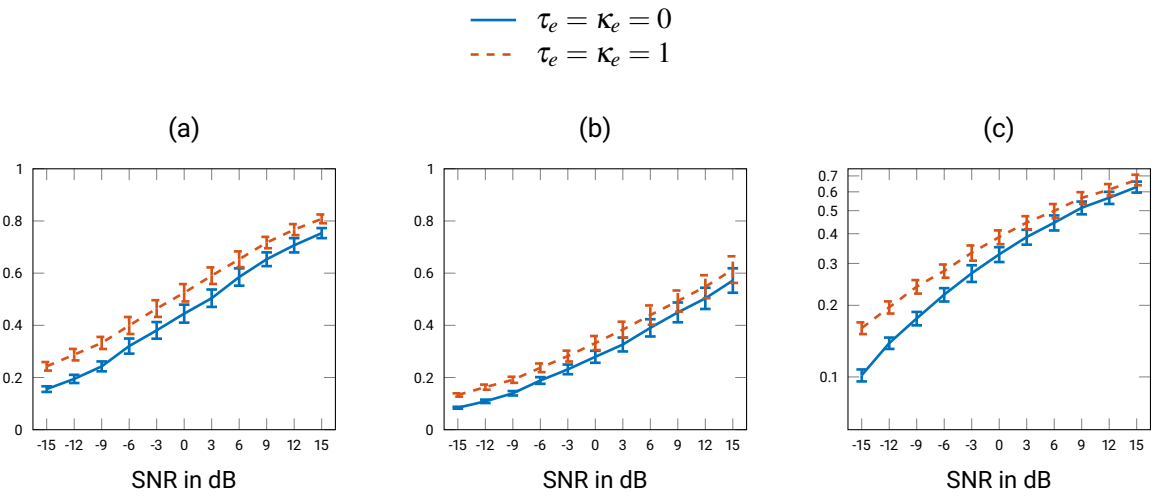



Figure 3.13: Mean peak agreement recall (a), precision (b), and F-measure (c) after adding white noise. Recall and precision have linear vertical axes, F-measure has a log-axis. Each point in the curves denotes the mean recall for a given SNR value across the whole dataset (111 loops). The solid blue curve shows results without a tolerance window; the dashed red curve was produced with a tolerance of one TF bin in every direction.

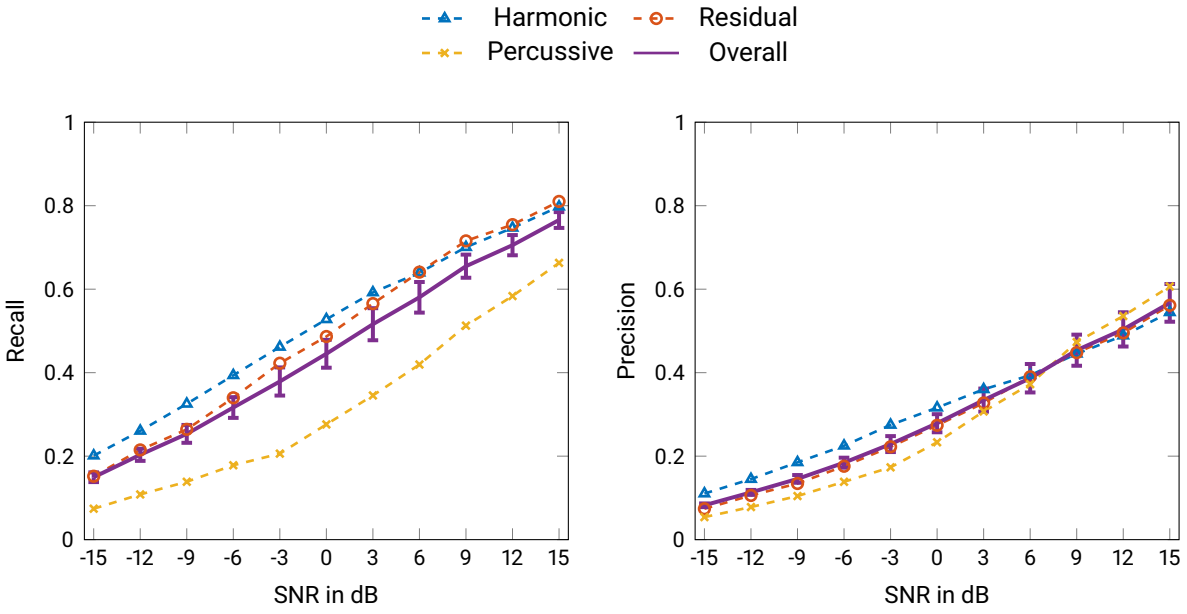



Figure 3.14: Mean peak agreement recall after adding white noise. Each curve corresponds to a timbral category (harmonic, residual, or percussive). Results were computed without a tolerance window. The dashed line is the mean across the entire dataset.

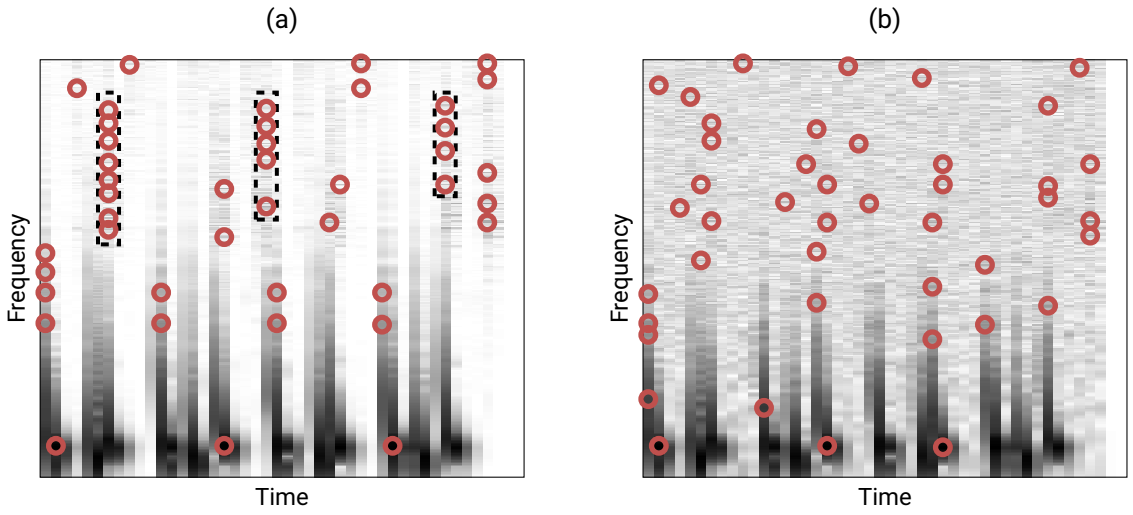
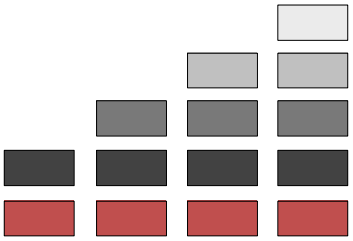



Figure 3.15: **(a)** Log-compressed log-frequency spectrogram of a percussive loop. Red circles denote peaks in its constellation map. The dashed rectangles highlight vertical peak structures typical of percussive signals. **(b)** Log-compressed log-frequency spectrogram of a percussive loop with white noise added at –3 dB SNR. Red circles denote peaks in its constellation map.

Figure 3.16: Mixture complexity: the query (red rectangles, bottom row) is mixed with further loops from the dataset. Each column represents a single track.



the dataset average as a dashed line for reference. In Figure 3.14, the recall for harmonic and residual loops is slightly better than average, whereas percussive loops are consistently below average by about 10%, suggesting that peak maps may not be optimal for this timbral category. Because of their high, wide-band onset energy, percussive loops produce peak maps with vertical structures and empty spaces between onsets. Figure 3.15 shows the log-compressed log-frequency spectrogram $\mathcal{Y}_{\mathrm{LFC}}$ of a percussive loop overlaid with dashed rectangles that enclose groups of vertically structured peaks. Since these STFT frames (or columns of $\mathcal{Y}_{\mathrm{LFC}}$) contain many local maxima, the peak picking can be thought of as a *random sampling*, or an arbitrary decision mainly related to the choice of neighborhood size. Adding white noise to a percussive loop has two prominent effects: it modifies this *random sampling* in the peak selection at the onsets and it fills the low-energy spaces between onsets with additional peaks.

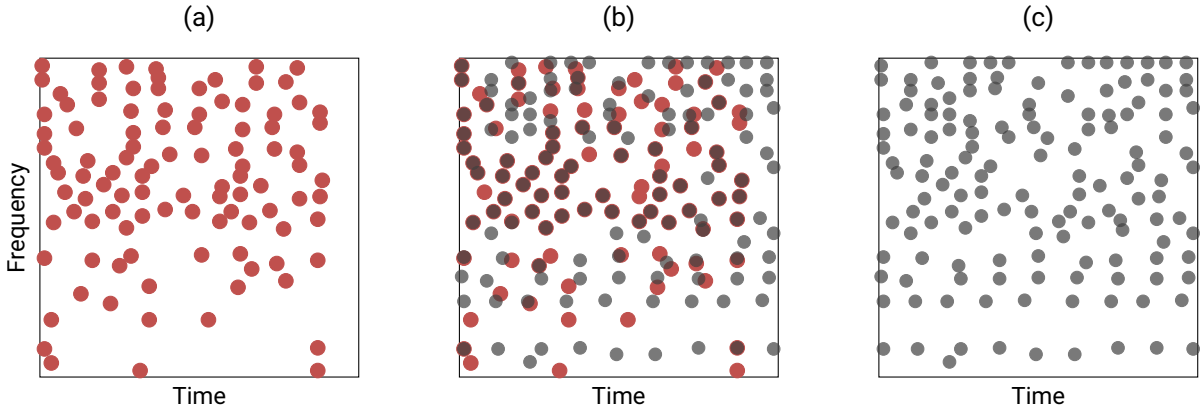



Figure 3.17: Peak map comparison for a mixture of vocals (H category) and percussions (P). **(a)** Peak map for vocals (query, red dots). **(b)** Peak map for vocals (red dots) overlaid with peak map for mixture (grey dots). **(c)** Peak map for mixture track (grey dots).

## 3.8 Multi-Loop Mixtures

For this series of experiments, we generated *tracks* or mixtures by adding loops from the dataset. We tested the peak agreement between a single, unmodified loop, and mixtures consisting of an increasing number of loops—only in cases where the original loop is contained in the complex mixture. The *track height* or *mixture complexity* is the number of loops present in the mixture—a track height of 1 is trivial, as it contains simply one of the loops. Figure 3.17 illustrates the effects of mixing a vocal track with a percussive track: Figure 3.17a shows the peak map for the unmodified vocal loop (red dots), Figure 3.17b is a peak map overlay of the vocal track (red dots) and the mixture track (grey dots), Figure 3.17c is the peak map for the mixture track (grey dots), which was made by combining the loop in Figure 3.17a with a percussive loop.

Figure 3.16 illustrates five mixtures with increasing height, from left to right. The query (red rectangles, bottom row) is always present, and is mixed with a certain number of additional loops from the dataset. Each column yields a distinct track for evaluation. A track's duration (in samples) is always fixed to the query. A loop longer than the query is trimmed to fit; a shorter loop is repeated until the query length is achieved. The following steps outline our procedure for random track generation and evaluation:

1. Choose a genre.
2. For the current genre, randomly choose ten loops—one of these loops will be fixed as the query.
3. For each track height from two to ten:
   (a) generate a mixture with the current height;
   (b) evaluate the peak agreement between the fixed query and the current mixture.

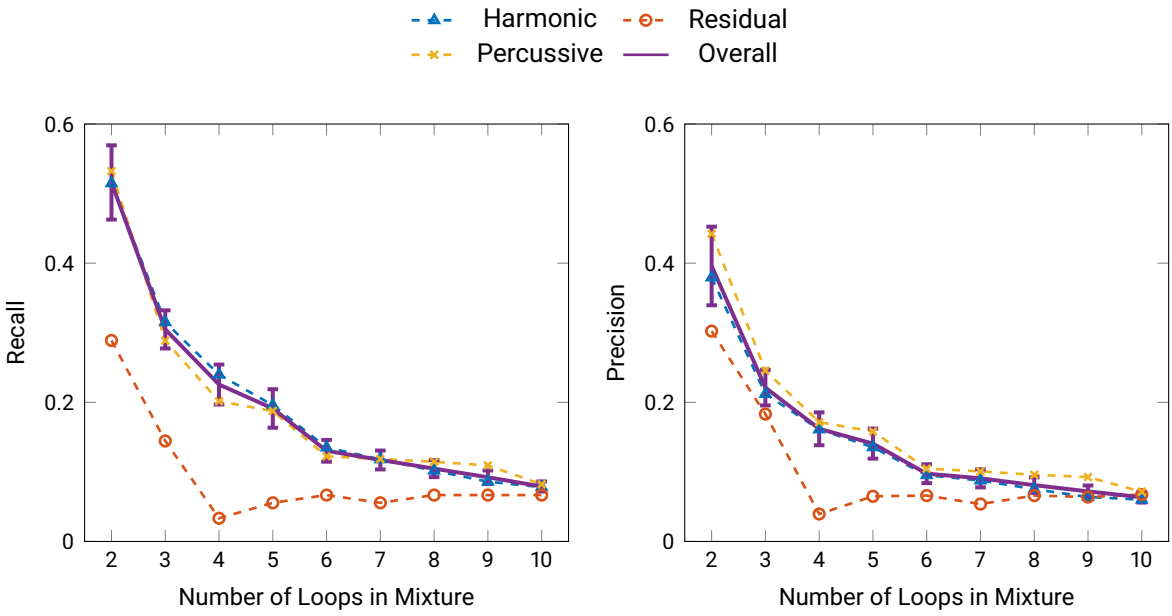



Figure 3.18: Mean peak agreement recall and precision for increasing track complexity. Each curve corresponds to a timbral category (harmonic, residual, or percussive). Results were computed without a tolerance window. The dashed line is the mean across all experiments.

We make a total of ten passes over the dataset, executing the method above for every genre. Figure 3.18 reveals a number of trends in these experiments that we discuss in the following. First, note that approximately half of the peak information is modified once a further loop is added to the mixture (recall drops to 0.5 and precision to 0.4). From that point onwards, there is a steady drop which ends slightly below 0.1. Second, harmonic and percussive loops retain close-to-average precision and recall across all mixture complexities, whereas residual loops fare worse throughout. This is explained by the fact that percussive loops induce localized vertical structures (with high energy) that are unperturbed by the addition of harmonic loops (with predominantly horizontal structures) or residual loops (with more random, sparse peak maps). Conversely, residual loops suffer the most from combinations, since their peak structure is easily disrupted by the other two types. Harmonic loops have roughly average performance; they have minimal peak overlap with the well-confined percussive loops and are only slightly modified by residual loops.

We can further support these hypotheses by analyzing the average number of peaks that a loop of a given timbral category contains. Figure 3.20 shows statistics for the average number of peaks per STFT frame that the loops in our dataset contain. The fact that percussive loops contain the most peaks explains their superior performance in the complexity experiments. Since there is only a slight difference between harmonic and residual loops, the deciding factor for peak agreement in these categories has more to do with the spatial distribution of the peaks, rather than the amount.

Figure 3.19: F-measure for increasing track complexity (log vertical axis). The solid blue curve was computed with no tolerance. The dashed red curve was computed with one TF bin tolerance in every direction.

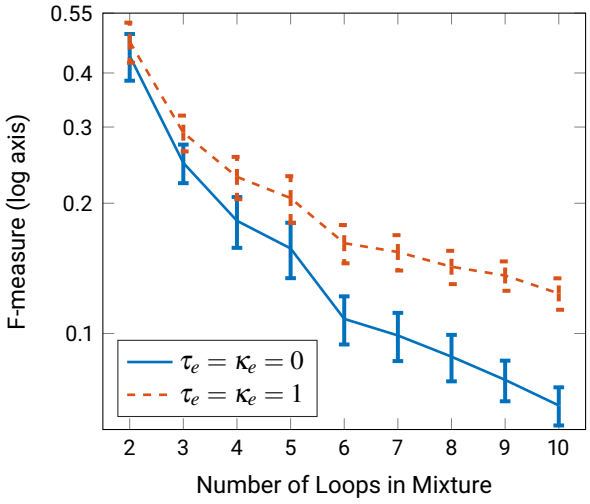
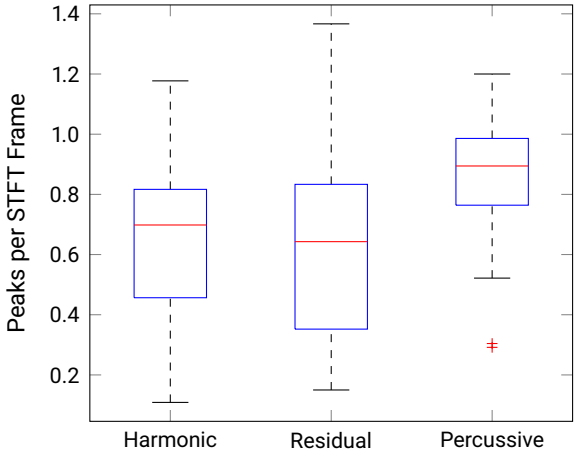



Figure 3.20: Average number of peaks per STFT frame, grouped by timbral category.




## 3.9 STFT Frame Offset

In the previous sections we studied how peak maps behave under different modifications of the signals' content. We now focus on an aspect of the algorithm which can decrease peak agreement without changing the signals themselves. Within our controlled scenario, we have computed STFTs and peak maps for isolated instances of a loop — however, in real-world applications, there is no guarantee that the query and database will be processed with identical STFT frame offsets. For instance, we might have an isolated loop as a query and an entire track containing that loop in the middle as the database, inducing unequal STFT frame offsets. For all loops in the `EM-Mini` dataset, we zero-padded the query signals (with increments of 256 samples, up to the hop size of 2048) while keeping unmodified versions as a database. Figure 3.21 illustrates how peak maps can change due to different STFT framing offsets. In Figure 3.21a we show the peak map for an unmodified query of the residual category (red dots). In Figure 3.21b, the unmodified loop (red dots) is overlaid with a version shifted by 1024 samples (one-half hop size, grey dots). The frame-shifted version is shown in Figure 3.21c (grey dots). Figure 3.22 shows the mean peak agreement

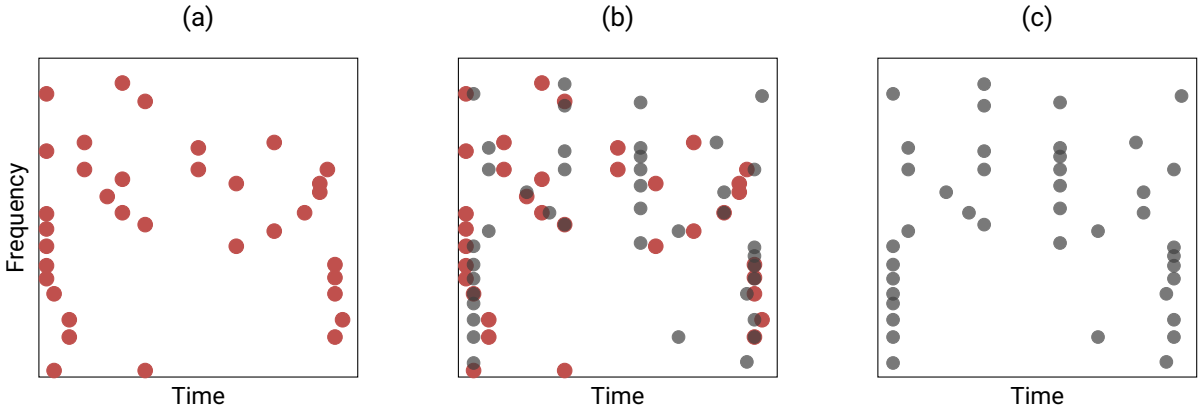



Figure 3.21: Peak map comparison when changing the frame offset for a loop from the *residual* category. **(a)** Peak map for unmodified loop (red dots). **(b)** Peak map for unmodified loop (red dots) overlaid with peak map for the same loop, shifted by 1024 samples (one-half hop size) by zero-padding (grey dots). **(c)** Peak map for shifted loop (grey dots).

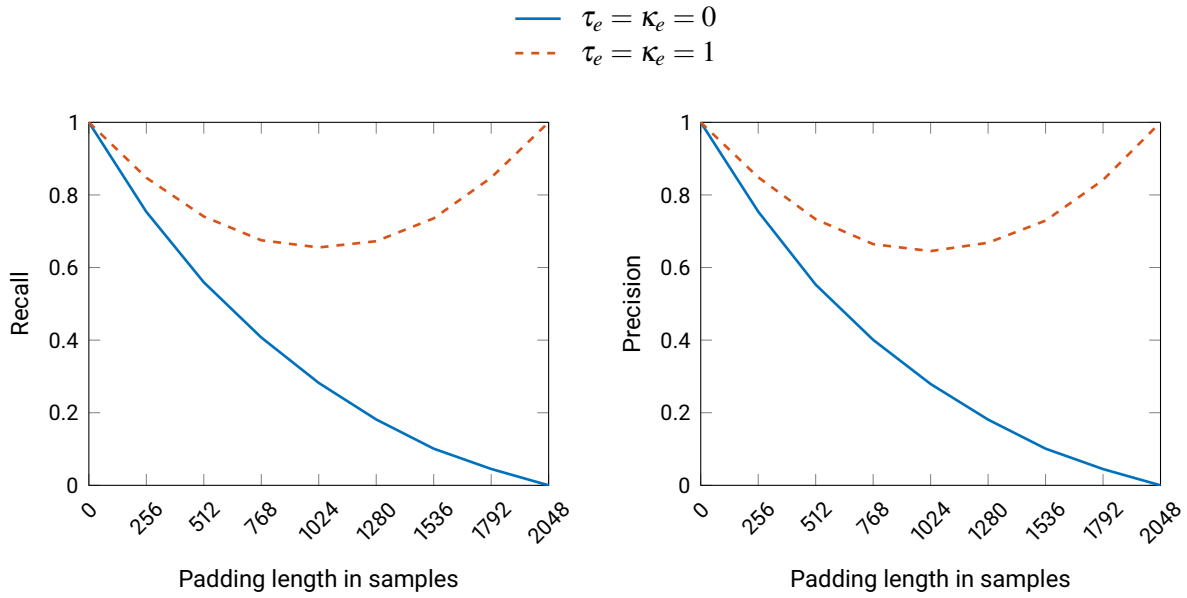



Figure 3.22: Mean peak agreement recall and precision for increasing amounts of zero-padding, up to the hop size.

precision and recall across the dataset. The solid blue curve shows results without a tolerance window and the dashed red line corresponds to one TF bin in every direction.

The results in Figure 3.22 illustrate how important it is for a fingerprinting algorithm to be robust against varying STFT frame offsets. Recall and precision with no tolerance (solid blue curves) decrease steadily, reaching a value of zero when a full hop size (2048 samples) of zero-padding samples has been added to the query. With a tolerance of one TF bin in every direction (dashed red lines), recall and precision

decrease until half the hop size is reached (1024 samples), and then increase for the second half of the experimental configurations, achieving perfect peak agreement again at 2048 samples. This tolerance setting has a maximum information loss of approximately thirty percent (or, conversely, 0.7 precision and recall). Our experiments highlight how unreliable individual peaks can be. Ellis' [50] implementation augments the original query landmarks by computing further sets of landmarks at quarter-hop, half-hop and three-quarter-hop offsets (see file `match_query.m`) and produces good results with a reduced number of landmark matches.

# Chapter 4

# Modeling SBEM

This chapter is based mainly on [90]. The core idea was born at HAMR@ISMIR 2015, with teammates Jonathan Driedger and Hendrik Schreiber. My main contributions were identifying the task, formulating the main constraints, preparing the dataset, and conducting the experiments. Christian Dittmar provided code and guidance for NMFD and the Pearson correlation evaluation. Together with Meinard Müller, we developed the mathematical formalization, identified the similarity with NMFD, and desinged the experimental setup.

As discussed in Chapters 1 and 2, SBEM is a popular family of genres which has increasingly received attention as a research subject in the field of MIR. Although this thesis focuses mainly on SBEM, in this chapter we present models and methods that work with electronic music (EM) in general, and thus we will use both terms interchangeably. A fundamental structural unit in SBEM are loops—audio fragments whose duration can span several seconds. The devices commonly used to produce EM, such as sequencers and digital audio workstations, impose a musical structure in which loops are repeatedly triggered and overlaid. This particular structure allows new perspectives on well-known MIR tasks. In this chapter we first review a prototypical production technique for EM from which we derive a simplified model. We then use our model to illustrate approaches for the following task: given a set of loops that were used to produce a track, decompose the track by finding the points in time at which each loop was activated. To this end, we repurpose established MIR techniques such as fingerprinting and non-negative matrix factor deconvolution.

This work offers three main contributions. First, we review the production process of EM and how it leads to the prototypical structure outlined previously (Section 4.2). Second, we propose a simplified formal model that captures these structural characteristics (Section 4.3). Third, we use our model to approach the EM decomposition task from two angles: first, we interpret it within a standard retrieval scenario by using

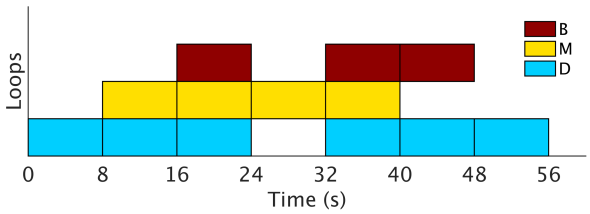



Figure 4.1: A condensed EM track built with three loop layers: drums (D), melody (M) and bass (B). Each block denotes the activation of the corresponding pattern during the time it spans.

fingerprinting and diagonal matching (Section 4.4). Our second approach is based on non-negative matrix factor deconvolution (NMFD), a technique commonly used for audio source separation (Section 4.5). We summarize our findings and discuss open issues in Section 4.6.

## 4.1 EM Composition Basics

With the arrival of affordable electronic music production technology, various loop-based genres emerged: techno, house, drum'n'bass and some forms of hip hop; this family of genres is subsumed under the umbrella term *Electronic Music* (EM). EM has garnered mainstream attention within the past two decades and has recently become a popular subject in MIR: standard tasks have been applied to EM (structure analysis [115]); new tasks have been developed (breakbeat analysis and resequencing [73, 71]); and specialized datasets have been compiled [80].

A central characteristic of EM that has not been extensively considered is its sequencer-centric composition. As noted by Collins [29], *loops* are an essential element of EM: loops are short audio fragments that are "generally associated with a single instrumental sound" [14]. Figure 4.1 illustrates a simplified EM track structure similar to that encouraged by digital audio workstations (DAWs) such as *Ableton Live* [1]. The track starts with the activation of a drum loop (blue, bottom row). After one cycle, a melody loop (yellow, middle row) is added, while the drum loop continues to play. A third layer—the bass (red, top row)—is activated in the third cycle. Over the course of the track, these loops are activated and deactivated. An important observation is that all appearances of a loop are identical; a property that can be modeled and exploited in MIR tasks. In particular, we consider the task of *decomposing* an EM track: given the set of loops that were used to produce a track and the final, *downmixed* version of the track itself, we wish to retrieve the set of timepoints at which each loop was activated.

## 4.2 Structure and Production Process

Unlike other genres, EM is often produced by starting with a single distinct musical pattern [126] (also called *loop*) and then adding and subtracting further musical material to shape the tension and listener's expectation. An EM track is built by combining layers (with potentially different lengths) in looping cyclical time—where the overall form corresponds to the multitrack layout of sequencers and digital audio

workstations (DAWs) [29]. Figure 4.1 provides a simple example of such a track (total duration 56 s), consisting of three layers or loops: drums (D), bass (B) and melody (M), each with a duration of 8 s. We will be using this track as a running example to clarify the points made throughout this chapter.

A common characteristic of EM tracks is their relative *sparseness* or low timbral complexity during the intro and outro—in other words, a single loop is active. This practice is rooted in two facts: Firstly, EM tracks are conceived not as isolated units, but rather as part of a seamless mix (performed by a DJ), where two or more tracks are overlaid together. Thus, in what could be termed *DJ-friendly* tracks [29], a single, clearly percussive element at the beginning and end facilitates the task of beat matching [14] and helps avoid unpleasantly dense transitions. We have constructed our running example following this principle: in Figure 4.1, the only active layer during the intro and outro is the drum loop (bottom row, blue).

The second reason for having a single-layer intro is that this section presents the track's main elements, making the listener aware of the sounds [14]. Once the listener has become familiar with the main musical idea expressed in the intro, more layers are progressively brought in to increase the tension (also known as a *buildup*), culminating in what Butler [14] designates as the track's *core*: the "thicker middle sections" where all loop layers are simultaneously active. This is reflected in Figure 4.1, where the melody is brought in at 8 s and the bass at 16 s, overlapping with uninterrupted drums. After the core has been reached, the majority of layers are muted or removed—once again, to create musical anticipation—in a section usually known as *break* or *breakdown* (see the region between 24–32 s in Figure 4.1, where only the melody is active). To release the musical tension, previous loops are reintroduced after the *breakdown*, (seconds 32–40, Figure 4.1) only to be gradually removed again, arriving at the outro. In the following sections we will develop a model that captures these structural characteristics and provides a foundation for analyzing EM tracks.

## 4.3 Simplified Model for EM

In Section 4.2 we illustrated the typical form of loop-based electronic music. With this in mind, our goal is to analyze an EM track's structure. More specifically, our method takes as input the set of loops or patterns that were used to produce a track, as well as the final, *downmixed* version of the track itself. From these, we wish to retrieve the set of timepoints at which each loop was activated within the track. We begin by formalizing the necessary input elements.

Let $V \in \mathbb{R}^{K \times M}$ be the feature representation of an EM track, where $K \in \mathbb{N}$ is the feature dimensionality and $M \in \mathbb{N}$ represents the number of elements or frames along the time axis. We assume that the track was constructed from a set of $R$ patterns $\mathrm{P}^r \in \mathbb{R}^{K \times T^r}$, $r \in [0 : R-1] := \{0, \ldots, R-1\}$. The parameter $T^r \in \mathbb{N}$ is the number of feature frames or observations for pattern $\mathrm{P}^r$. In practice, the patterns can have different lengths—however, without loss of generality, we define their lengths to be the same $T := T^0 = \ldots = T^{R-1}$, which could be achieved by adequately zero-padding shorter patterns until they reach the length of the

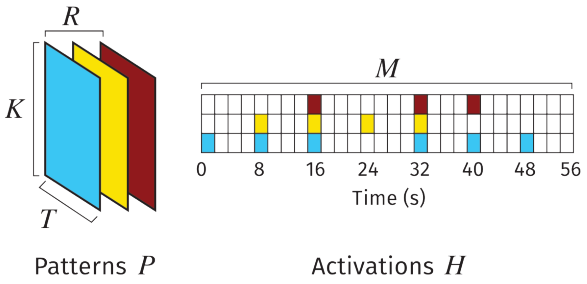



Figure 4.2: *Left*: Tensor P with three patterns (drums, bass, melody). *Right*: Activation matrix $H$ with three rows; the colored cells denote an activation of the corresponding pattern.

longest. Based on this assumption, the patterns can be grouped into a pattern tensor $\mathrm{P} \in \mathbb{R}^{K \times R \times T}$. In the case of our running example, seen in Figure 4.1, $T \triangleq 8\,\mathrm{s}$ and the number of patterns is $R = 3$. Consequently, the subdimension of the tensor which refers to a specific pattern with index $r$ is $\mathrm{P}^r := \mathrm{P}(\cdot,\ r,\ \cdot)$ (i. e., the feature matrix for either (D), (M), or (B) in our example); whereas $\mathrm{P}_t := \mathrm{P}(\cdot,\ \cdot,\ t)$ refers to frame index $t$ simultaneously in all patterns.

In order to construct the feature representation $V$ from the pattern tensor P, we require an activation matrix $H \in \mathbb{B}^{R \times M}$ with $\mathbb{B} := \{0, 1\}$, such that

$$V \triangleq \sum_{t=0}^{T-1} \mathrm{P}_t \cdot \overset{t\rightarrow}{H}, \tag{4.1}$$

where $\overset{t\rightarrow}{(\cdot)}$ denotes a frame shift operator [124]. Figure 4.2 depicts P and $H$ as constructed for our running example. The model assumes that the sum of pattern signals and their respective transformations to a feature representation are linear, which may not always be the case. The additive assumption of Eq. 4.1 implies that no time-varying and/or non-linear effects were added to the mixture (such as compression, distiortion, or filtering), which are often present in real-world EM. Aside from this, we specify a number of further constraints below.

The devices used to produce early EM imposed a series of technical constraints which we formalize here. Although many of these constraints were subsequently eliminated in more modern equipment and DAWs, they have been ingrained into the music's aesthetic and remain in use up to the present day.

*Non-overlap constraint*: A pattern is never superimposed with itself, i. e., the distance between two activations of any given pattern is always equal to or greater than the pattern's length. Patterns are loaded into a device's memory and triggered by a sequencer—usually without further activation signals before it has played through to the end. If a pattern $\mathrm{P}^r$ is activated at time $m \in [0 : M-1]$, then $H^r(m) \neq 0 \Rightarrow H^r(m+1) = \ldots = H^r(m+T-1) = 0$.

*Length constraint*: As noted by [29], multiple layers in EM are complementary, creating aggregate effects and capable of being independently inserted and removed. For this reason, we make the simplifying assumption that $T := T^0 = T^1 = \ldots = T^{R-1}$, i. e., that all patterns have the same length.

*T-grid constraint*: Motivated by the use of centralized MIDI clocks and the fixed amount of musical time available on prevalent devices such as drum machines (which typically allow programming one musical measure at a time, in 16 steps), we enforce a timing grid which restricts the possible activation points in

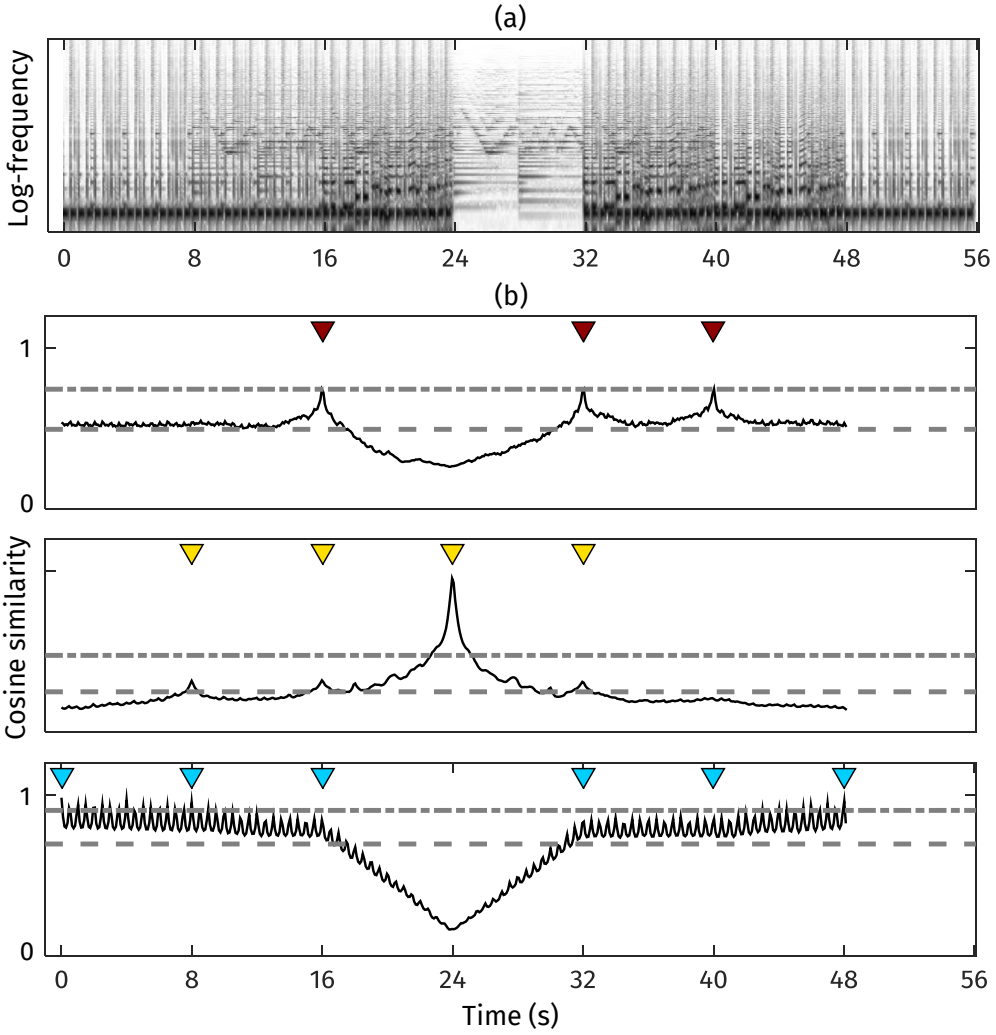



Figure 4.3: **(a)**: Log-frequency spectrogram for entire track. **(b)**: Matching curves computed using cosine similarity for drums, melody, and bass (bottom to top). The dashed line represents the curve's global mean; the dash-dotted line is the GT mean (see Section 4.4.4 for definition of *gain*). Colored triangles indicate GT loop activation positions.

$H$. In Figure 4.1, patterns are always introduced and removed at multiples of 8 s.

*Amplitude constraint*: We assume that a pattern is always activated with the same *intensity* throughout a track, and therefore each row $r$ in the activation matrix $H$ fulfills $H^r := H(r,\cdot) \in \mathbb{B}^{1\times M}$.

## 4.4 Fingerprint-Based EM Decomposition

In the running example, multiple patterns are overlaid in different configurations to form the track. If we know *a priori* which patterns are included and wish to find their respective activation positions, we

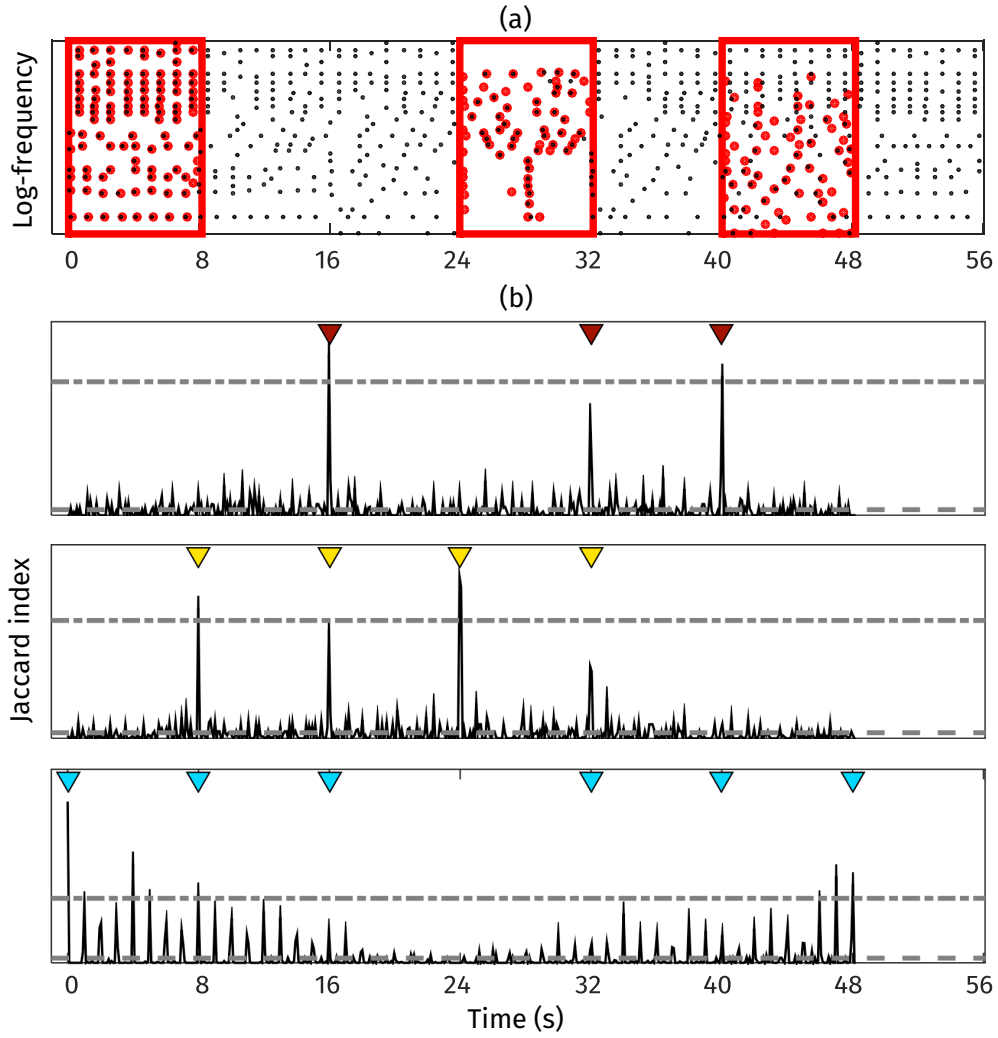



Figure 4.4: **(a)**: Log-frequency spectral peak map for the entire track (black dots) and for each query (red dots enclosed in red, from left to right: drums, melody, and bass). **(b)**: Matching curves computed with the Jaccard index and each pattern as a query for drums, melody, and bass (bottom to top).

need a technique capable of identifying an audio query within a database where further musical material is superimposed. We first examine log-frequency spectrograms and diagonal matching as a baseline approach, and continue with audio fingerprinting techniques based on spectral peaks in combination with various similarity measures. In Section 4.5 we discuss an alternative approach based on NMFD. The running example is constructed with one audio file for each pattern and a generic EM track arrangement seen in Figure 4.1. The complete track is generated in the time domain by summing the individual patterns that are active at a given point in time. All audio files have been downmixed to mono with a sampling rate $F_s = 22050\,\text{Hz}$.

### 4.4.1 Diagonal Matching

We implement the diagonal matching procedure outlined in [97, pp. 376–378] to measure the *similarity* between each query pattern $\mathrm{P}^r$ and the track feature matrix $V$. In simple terms, to test if and where the query $\mathrm{P}^r = (\mathrm{P}^r_0, \ldots, \mathrm{P}^r_{T-1})$ is contained in $V = (V_0, \ldots, V_{M-1})$, we shift the sequence $\mathrm{P}^r$ over the sequence $V$ and locally compare $\mathrm{P}^r$ with suitable subsequences of $V$. In general, let $\mathcal{F}$ be the feature space (for example, $\mathcal{F} = \mathbb{R}^K$ in the case of log-frequency spectrograms). A similarity measure $s : \mathcal{F} \times \mathcal{F} \to \mathbb{R} \cap [0,1]$ between two feature frames will yield a value of 1 if the query is identical to a certain region of the database, and 0 if there is no resemblance at all.

### 4.4.2 Baseline Procedure

For each individual pattern, as well as the entire track, we compute an STFT $\mathcal{X}$ with the following parameters: block size $N = 4096$, hop size $H = N/2$ and a Hann window. From these, magnitude spectrograms (abbreviated as MS) are computed and mapped to a logarithmically spaced frequency axis with a lower cutoff frequency of 32 Hz, an upper cutoff frequency of 8000 Hz and spectral selectivity of 36 bins per octave (abbreviated LS). Under these STFT settings, the 36-bin spectral selectivity does not hold in the two lowest octaves; however, their spectral peak contribution is negligible. Preliminary experiments have shown that this musically meaningful feature representation is beneficial for the matching procedure both in terms of efficiency and accuracy. We begin with a baseline experiment (Figure 4.3), using LS and cosine similarity:

$$\textit{Cosine} : s_{\cos}(u,v) := \frac{\langle u|v\rangle}{||u|| \cdot ||v||},\ u,v \in \mathbb{R}^K \setminus \{0\}. \tag{4.2}$$

Notice that the clearest peak is produced by the melody activation at 24 s (Figure 4.3b, middle row), which occurs without any other patterns being overlaid. The three remaining activation points for the melody have a very low gain relative to their neighboring values. The matching curve for the drums (Figure 4.3b, bottom row) displays a coarse downwards trend starting at 0 s and reaching a global minimum at 24 s (the point at which the drum pattern is not activated in our example); this trend is reversed as the drums are added again at 32 s. The internal repetitivity (or self-similarity) of the drum pattern causes the periodic peaks seen throughout the matching curve. Overall, it is evident from all three curves that the combination of LS with cosine similarity is insufficient to capture the activations when multiple patterns are superimposed—motivating our next experimental configuration which uses spectral peak maps.

### 4.4.3 Fingerprinting with Peak Maps

Although our scenario is slightly different to that of audio fingerprinting and identification, both require a feature representation which captures an individual pattern's characteristics despite the superposition of further sound sources. To this end, we use spectral peak maps as described in [97]. Conceptually, we are

following an early approach for loop retrieval inside hip hop recordings which was presented in [137] and later refined in [138]. Their method is based on a modification of the fingerprinting procedure originally described in [141].

For each time-frequency bin in the respective LS, a rectangular analysis window is constructed. The maximum value within each window is kept (with the value 1 on the output) and all neighbors are set to 0 on the output. In Figure 4.4a we show the spectral peak map for the entire track (black dots) and the query peak map for each query pattern (red dots in red rectangles). These log-frequency peak maps populate a pattern tensor $\mathrm{P} \in \mathbb{B}^{K \times R \times T}$, where $K = 286$. Thus $\mathrm{P}^r$ corresponds to the peak map for the $r^{\text{th}}$ pattern, while $V$ corresponds to the entire track.

In addition to the cosine measure defined in Eq. 4.2, we test different similarity measures $s$:

$$\textit{Jaccard} : s_{\text{Jac}}(u,v) := \frac{||u \wedge v||}{||u \vee v||},\ u,v \in \mathbb{B}^K, \tag{4.3}$$

$$\textit{Inclusion} : s_{\text{inc}}(u,v) := \frac{||u \wedge v||}{||u||},\ u,v \in \mathbb{B}^K, \tag{4.4}$$

where we set $\frac{0}{0} := 1$. The inclusion metric aims to quantify the extent to which the query is contained or *included* in the database and has a similar definition to the Jaccard index.

### 4.4.4 Evaluation

We use two measures to quantify how well the matching curves capture the pattern activations. For the first measure, termed *gain*, we compute the average of the activation values at the ground truth (GT) activation points: in Figures 4.3b and 4.4b, these locations are marked by colored triangles, corresponding to each loop in the running example; their mean value is shown as a dash-dotted line. We also compute the mean value for the entire curve (dashed line) and use the ratio between these two means in order to assess the quality of the matching curve. Ideally, the curve assumes large values at the GT activation points and small values elsewhere, resulting in a larger gain. As a second measure we take the *Pearson correlation* between a computed matching curve and its corresponding row $H^r$ in the GT activation matrix, where the activation points have a value of 1, and 0 elsewhere. Again, a high Pearson correlation reflects high matching curve quality.

We generated a set of patterns used to build prototypical EM tracks. To foster reproducible research, we produced them ourselves, avoiding potential copyright issues—they are available under a Creative Commons Attribution-ShareAlike 4.0 International license and can be obtained at the companion website[1]. We chose seven prominent EM subgenres such as *big beat*, *garage* and *drum'n'bass* (in a tempo range

[1] `https://www.audiolabs-erlangen.de/resources/MIR/2016-ISMIR-EMLoop`

Table 4.1: Results for diagonal matching experiments with magnitude spectrograms (MS), log-frequency spectrograms (LS), and log-frequency peak maps (PLS) using the cosine, inclusion and Jaccard similarity measures. Each column shows the mean and variance for peak gain and Pearson correlation.

| | *Gain* | | *Pearson* | |
|---|---|---|---|---|
| | $\mu$ | $\sigma$ | $\mu$ | $\sigma$ |
| MS/cos | 1.72 | 0.31 | 0.13 | 0.05 |
| LS/cos | 1.57 | 0.29 | 0.11 | 0.05 |
| PLS/cos | 19.46 | 10.45 | 0.52 | 0.18 |
| PLS/inc | 21.69 | 11.90 | 0.51 | 0.19 |
| PLS/Jac | 19.54 | 9.76 | 0.53 | 0.18 |

between 120–160 BPM). For each subgenre, we generated four patterns in the categories of drums, bass, melody and additional effects.

As stated by Wang [141], spectral peak maps are robust to superposition of multiple sources; a fact which becomes clear when comparing Figures 4.3b and 4.4b. In Figure 4.4b, the *peak gain* has greatly increased compared to the baseline approach with LS. In Table 4.1 we list the mean peak gain and Pearson correlation for all seven tracks, along with standard deviations for each value. The first two rows, MS/cos and LS/cos, correspond to the baseline approach—the last three rows summarize the experiments with spectral peak maps. Note that spectral peak maps have approximately ten times the peak gain of MS/LS, whereas the Pearson correlation increases by a factor of four. Figures 4.3 and 4.4 illustrate the results in Table 4.1 at an intuitive level. With MS, the spectral content shared among different types of patterns impedes distinct peaks from emerging. By discarding this irrelevant information, LS better represent the characteristics of each pattern. From the perspective of peak quality, only the self-similarity of the drum pattern continues to pose a challenge.

## 4.5 NMFD-Based EM Decomposition

By design, our model for EM is very close to the formulation of NMFD; in this section we explore the performance of NMFD and compare it with our fingerprinting methods.

### 4.5.1 Related Work

In this section, we briefly review the NMFD method that we employ for decomposing the feature representation $V$. Weiss and Bello [142] used non-negative matrix factorization (NMF) to identify repeating patterns in music. By adding sparsity constraints and shift-invariant probabilistic latent component analysis (SI-PLCA), they automatically identify the number of patterns and their lengths—applied to beat-synchronous chromagrams in popular music. Masuda et al. [96] propose a query-by-audio system based on NMF to identify the locations where a query musical phrase is present in a musical piece. Among more general techniques for investigating alleged music plagiarism, Dittmar et al. [40] proposed a

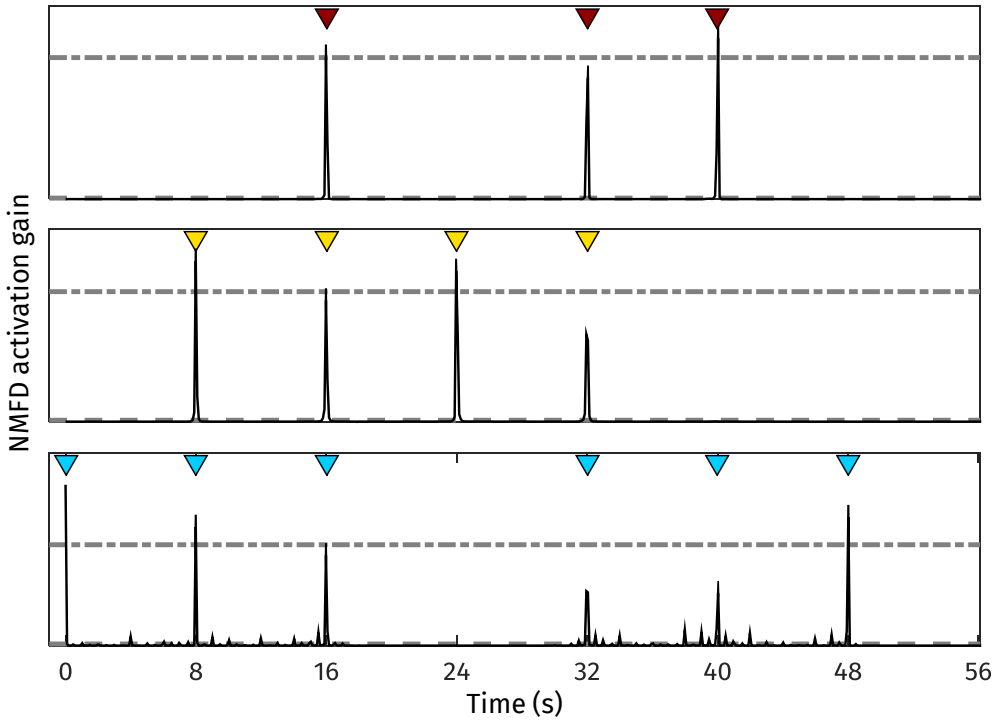



Figure 4.5: Activation curves learned by NMFD applied to the magnitude spectrogram of the running example. The dashed lines represent the mean value for the complete curve, but are very close to the $x$-axis.

method for retrieval of sampling. Their approach, based on NMF, was not supplemented with systematic evaluation, but was further investigated in [144]. Previous works [124, 87, 114, 43] successfully applied NMFD—a convolutive version of NMF—for drum transcription and separation. Hockman et al. [71, 73] specifically focused on analyzing breakbeats, i. e., drum-only loops as used in hip hop and drum'n'bass. Detecting sample occurrences throughout a track is a secondary aspect, as they address the more challenging scenario of estimating the loop resequencing [73]. All these previous works have in common that they attempt to retrieve one loop inside a song, whereas we pursue a more holistic approach that allows to deconstruct the whole track into loops.

### 4.5.2 NMFD Model

Our objective is to decompose V into component magnitude spectrograms that correspond to the distinct musical elements.[2] Conventional NMF can be used to compute a factorization $\mathrm{V} \approx \mathrm{W} \cdot \mathrm{H}$, where the columns of $\mathrm{W} \in \mathbb{R}_{\geq 0}^{K \times R}$ represent spectral basis functions (also called templates) and the rows of $\mathrm{H} \in \mathbb{R}_{\geq 0}^{R \times M}$ contain time-varying gains (also called activations). The rank $R \in \mathbb{N}$ of the approximation (i. e., number of

[2]This section was formulated together with Christian Dittmar and also appears in [44].

Table 4.2: Results for NMFD with magnitude spectrograms (MS) and log-frequency spectrograms (LS), initializing the number of templates (R) and also the loop templates (RP). Each column shows the mean and variance for peak gain and Pearson correlation.

| | *Gain* | | *Pearson* | |
|---|---|---|---|---|
| | $\mu$ | $\sigma$ | $\mu$ | $\sigma$ |
| NMFD@MS, R | 55.20 | 29.80 | 0.65 | 0.25 |
| NMFD@MS, RP | 89.62 | 39.67 | 0.88 | 0.11 |
| NMFD@LS, R | 52.39 | 35.95 | 0.64 | 0.22 |
| NMFD@LS, RP | 79.59 | 34.99 | 0.87 | 0.12 |

components) is an important but generally unknown parameter. NMFD extends NMF to the convolutive case by using two-dimensional templates so that each of the $R$ spectral bases can be interpreted as a magnitude spectrogram snippet consisting of $T \ll M$ spectral frames. The convolutive spectrogram approximation $\mathrm{V} \approx \Lambda$ is modeled as

$$\Lambda := \sum_{t=0}^{T-1} \mathrm{W}_t \cdot \overset{t\rightarrow}{\mathrm{H}}, \tag{4.5}$$

where $\overset{t\rightarrow}{(\cdot)}$ denotes a frame shift operator (see also Eq. 4.1). As before, each column in $\mathrm{W}_t \in \mathbb{R}_{\geq 0}^{K \times R}$ represents the spectral basis of a particular component, but this time we have $T$ different versions $\mathrm{W}_t$, with $t \in [0 : T-1]$ available. If we take lateral slices along the columns of $\mathrm{W}_t$, we can obtain $R$ prototype magnitude spectrograms $\mathrm{U}^r \in \mathbb{R}_{\geq 0}^{K \times T}$. NMFD typically starts with a suitable initialization (with random values or constant values) of matrices $\mathrm{W}_t^{(0)}$ and $\mathrm{H}^{(0)}$. These matrices are iteratively updated to minimize a suitable distance measure between the convolutive approximation $\Lambda$ and V. In this work, we use the update rules detailed in [124], which extend the well-known update rules for minimizing the Kullback-Leibler Divergence (KLD) [83] to the convolutive case.

### 4.5.3 Evaluation

For our experiments with NMFD we used MS and LS to conduct the procedure in two variants. For the first variant (referred to as R in Table 4.2), the only *a priori* information used is the number of patterns (or templates) $R$ and their length $T$. The templates are initialized randomly and fifty iterative updates are used to minimize the KLD. To account for the effects of random initialization, we carry out ten initialization passes per track. The results in Table 4.2 reflect the mean and standard deviation across all passes. For the second variant (RP), we supply the pattern templates themselves at initialization (i. e., $R$, $T$ and $P$ are known). We also disallow template updates and only allow activation updates. Since the templates in variant R are initialized randomly, there is no direct relationship between the learned activation curves and the corresponding ground truth curves. We deal with this permutation indeterminacy by comparing all computed activation curves with all ground truth curves and taking the results which maximize the overall score. For all configurations in Table 4.2, we observe a peak gain at least twice as high as that obtained through diagonal matching; the Pearson correlation increases by a factor of 1.2–1.7, depending on the NMFD configuration taken for comparison. Focusing on the differences among NMFD configurations,

Table 4.3: Computation times for diagonal matching with log-spectral peak maps (PLS), NMFD with magnitude spectrograms (MS), and NMFD with log-frequency spectrograms (LS). The choice of initialization R or RP for NMFD does not impact execution time.

| *Method* | *Time (s)* |
|---|---|
| PLS | 0.2 |
| NMFD@LS,(R/RP) | 2.5 |
| NMFD@MS,(R/RP) | 36.0 |

RP brings peak gain improvements by a factor slightly greater than 1.5; the Pearson correlation increases by about 1.35. The feature choice (MS or LS) does not play a significant role in result quality. Due to the amount of prior knowledge used to initialize the RP configuration, we consider it as an upper bound for less informed approaches.

## 4.6 Conclusions and Future Work

In the preceding sections we developed a better understanding of the feature representations and matching techniques that are commonly used for pattern activation discovery. In this section, we reflect on some of the limitations of our work, further research topics, and computational performance issues.

Clearly, our work only provides a baseline for further work towards more realistic scenarios. As to our model's inherent shortcomings, real-world EM tracks usually contain more than four individual patterns, which are rarely available. Moreover, activations of a given pattern are often (spectrally) different from one another due to the use of effects such as delay and reverb, filter sweeps or resequencing. Thus, we consider this study as a stepping stone towards a fully-developed pipeline for EM structure analysis and decomposition. One potential research direction would be the automatic identification of suitable pattern candidates. A repetition-based analysis technique as described in [98] could be used in conjunction with spectral peak maps to compute self-similarity matrices (SSMs) that saliently encode inclusion relationships. Furthermore, semi-informed variants of NMFD might be helpful in discovering additional patterns that are not explicitly given, where the use of rhythmic structure can serve as prior knowledge to initialize the activations. Although diagonal matching curves can be computed efficiently with a straightforward implementation, we have seen they have certain shortcomings; we wish to investigate the feasibility of using them as rough initial guesses and leaving the refinement up to NMFD. Beyond each method's capabilities, as seen in Tables 4.1 and 4.2, there is also the issue of their running time and memory requirements. For the running example, we tested our `MATLAB` implementation on a 3.2 GHz Intel Core i5 CPU with 16 GB RAM, yielding the mean execution times in Table 4.3.

From Tables 4.3 and 4.2 we can conclude that NMFD@LS offers the best balance between quality and resource intensity. NMFD@MS takes approximately 14 times longer to compute than NMFD@LS and

only produces marginally better results. Indeed, recall that the feature dimensionality $K = 286$ for LS and $K = 2049$ for MS, which explains the large difference in execution times.

As a final remark, musical structure analysis is an ill-defined problem, primarily because of ambiguity; a segmentation may be based on different principles (homogeneity, repetition, novelty) that can conflict with each other [107]. The main advantage of our method is that we avoid the philosophical issue of how a track's structure is *perceived*, and rather attempt to determine how it was *produced*—a univocal problem. It can then be argued that the listeners' perception is influenced by the cues inherent to EM's compositional style.

# Chapter 5

# Cascaded Harmonic-Residual-Percussive Features

In this chapter, based on [93], my main contribution was identifying phenomena in electronic music that could be visualized well by their timbral changes along the HRP axis. The original HRPSS method and code are by Jonathan Driedger. Christian Dittmar contributed critical insight about the discrepancies between time- and frequency-domain implementations, as well as important parameter settings. Meinard Müller provided the initial idea of a cascaded feature, and we then collaborated to develop the mathematical formalization.

Harmonic-percussive separation is an audio decomposition technique that coarsely splits music recordings into harmonic and percussive components—it can be used for karaoke, DJing, or remixing; or as a preprocessing step that can potentially facilitate further tasks like key detection (for the harmonic component) or drum transcription (for the percussive component). We propose a cascaded harmonic-residual-percussive (HRP) procedure yielding a mid-level feature representation that can help analyze musical phenomena such as percussive event density, shifts in timbral properties, and homogeneous structural segments. In this chapter, we first outline the steps to compute cascaded HRP features (CHRP) and then illustrate their capabilities by means of three examples: to visualize the percussive and noise-like properties of snare-drum playing techniques, to examine changes between harmonic and percussive timbres in electronic music, and to identify homogeneous, purely percussive passages in funk and soul recordings (also known as *breaks*).

This chapter is laid out as follows: Section 5.2 outlines previous work in (cascaded) HPSS, Section 5.3 describes our CHRP approach and how to use it to construct an energy-based feature matrix, Section 5.4

deals with concrete musical applications where the CHRP feature is particularly relevant, and Section 5.5 contains some final thoughts on our method and how it can be used and extended in future work.

## 5.1 Harmonic-Residual-Percussive Source Separation

Music signals are typically mixtures of different sound sources, such as melodic instruments, singing voice, and drums—which can be categorized as (predominantly) harmonic or percussive. Harmonic-Percussive Source Separation (HPSS) aims at splitting a downmixed music recording into a harmonic, sustained part (e. g., melodic instruments, singing voice) and a percussive, unpitched part (e. g., drums and percussion). Inspired by the Sinusoids+Transients+Noise (S+T+N) signal model [86, 32], HPSS was then extended to harmonic-residual-percussive source separation (HRPSS) in [48, 59], where the residual component captures the musical elements that cannot clearly be categorized as either harmonic or percussive. In this chapter we present a cascaded procedure to obtain a finer-grained decomposition (which goes beyond the standard components H, R, and P) and then derive mid-level features based on this cascade.

In real-world applications, even a single musical source—which can naïvely be considered as either purely harmonic or percussive—often contains elements from multiple realms: the human voice (idealized as harmonic) also contains noise-like elements in the form of sibilants and percussive elements in the form of plosives; percussive instruments (especially kicks, snares, and toms) have a percussive onset and a tonal/harmonic decay. Although HPSS is frequently used to isolate a certain component (e. g., singing voice) with minimal interference from other elements, our main goal is to develop a feature based on HRPSS which helps visualize these timbral *ambiguities*. Furthermore, we propose a *cascaded* approach to increase resolution along the HRP axis: the HRPSS procedure is used iteratively, beginning with the original signal in the first pass and then applying it repeatedly to the residual component from the previous stage. Prior works which use multiple passes of HPSS are discussed in detail further in this section.

Driedger et al. [48] introduced a *separation factor* as a parameter to control the *strictness* with which spectral elements are classified as either harmonic or percussive. Intuitively speaking, the separation factor ($\beta$) determines the width of the harmonic and percussive components: higher $\beta$ values yield narrower H and P components (and a broader R component).

Figure 5.1 illustrates, on a conceptual level, how we incorporated both cascading and varying separation factors into our procedure to obtain a seven-dimensional feature. In the first row, a comparatively high separation factor is used to compute the H, P, and R components of the input signal; note that the H and P components have a small width relative to the total HRP width. In the second row ($\beta = 3$), the RH, RR, and RP components are computed using R from the first pass—a lower separation factor allows the RH and RP components to have a greater relative width. In the third row ($\beta = 1.5$), obtained from RR, all components are distributed equally. The last row illustrates how the components from all three stages

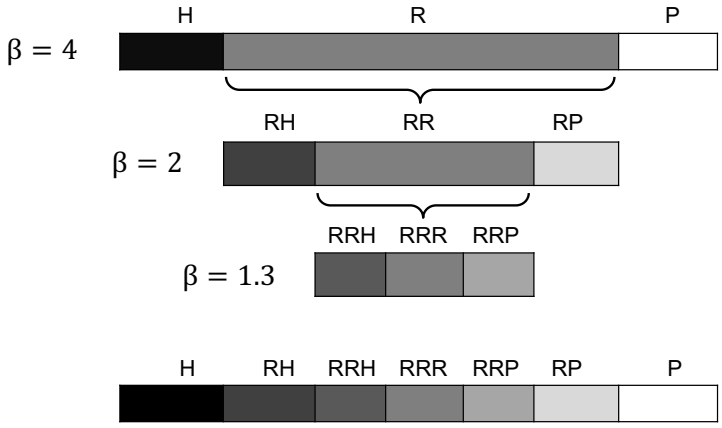



Figure 5.1: Conceptual illustration of cascaded harmonic-residual-percussive source separation. In the first row, a signal is separated into harmonic, residual, and percussive components (using $\beta = 5$ as a separation factor). In the second and third rows, the residual component from the previous pass is separated into the H, R, and P components, using decreasing separation factors. The last row shows how the cascade components can be ordered on an HRP axis.

are aggregated into a single feature. In general, the prefix of a component's name indicates how it was derived.

## 5.2 Related Work

Harmonic-percussive separation yields component signals which are of interest under various circumstances; they may be used for DJing, remixing, or karaoke, or they can simplify further tasks such as pitch tracking, key detection, or drum transcription. Using the separated percussive component can lead to a quality improvement for beat tracking [65], rhythm analysis [101] and transcription of rhythm instruments. The separated harmonic component might be more suitable for the transcription of pitched instruments and chord detection [101, 135]. Furthermore, HPSS can be used as a remixing tool by changing the level ratio between both signal components [102]. A comprehensive overview of existing methods for HPSS is given in [131].

The first line of HPSS approaches relied on factorizing a time-frequency (TF) representation of the music mixture into low-rank components and subsequently clustering them into harmonic and percussive subsets. In an early work [136], the magnitude spectrogram of the music mixture is decomposed via Independent Subspace Analysis. Spectral low-level features are computed from the components that serve to discriminate them into either harmonic or percussive via a manually optimized decision tree. In later works, factorization techniques such as Non-Negative Matrix (or Tensor) Factorization (NMF/NTF) were used together with more elaborate classification schemes, such as Support Vector Machines [64, 120, 55, 113]. It is worth noting that a HPSS procedure based on NMF that is intended to

emphasize harmonic structures in some of the components was recently proposed in [106]. An alternative approach based on phase-cues was proposed in [17]. The second line of research simplifies the HPSS task by assuming that harmonic sounds usually exhibit a horizontal structure in the magnitude spectrogram of the mixture (in time direction), while percussive sounds appear as vertical structures (in frequency direction). Ono et al. [103] proposed a method that creates harmonically or percussively enhanced spectrograms based on complementary diffusion. A similar method was introduced in [54] by Fitzgerald, where the enhanced spectrograms were derived from the mixture by applying median filtering in either time or frequency direction.

Cascaded or multi-stage HPSS has been used mainly to enhance spectrograms in tasks related to singing voice detection and melody line estimation [84]. Tachibana et al. [132] compute a two-stage HPSS to enhance melodies. In the first stage, they use a long analysis window (200 ms) to obtain harmonic and percussive components; in the second stage, a shorter analysis window (30 ms) is applied to the percussive component from the first stage.

## 5.3 Cascaded Harmonic-Residual-Percussive Features

This section is divided into three subsections. In the first subsection, we detail our general procedure to conduct HRPSS, which follows the work of [48]. In the second subsection, we describe our cascaded application of HRPSS. While other works have used the concept of cascading to better capture the properties of a specific component (e. g., the singing voice, as in [84]), our approach retains *all* components—inducing an HRP *axis* with greater resolution. In the third subsection we present our main contribution: using the results of the cascading step to assemble a mid-level, energy-based CHRP feature representation.

### 5.3.1 General HRP Procedure

Given the original audio recording $x$, let $X$ be its short-time Fourier transform (STFT) with coefficients $X(t,k)$, where $t \in [0:T-1]$ and $k \in [0:K]$, $T$ is the number of frames, $K = N/2$ is the frequency index corresponding to the Nyquist frequency, $N$ is the window (frame) size, and $h^{\mathrm{STFT}}$ is the hop length.

As observed in [101, 48, 54], if we take $Y = |X|$ to be the magnitude spectrogram of the signal, percussive elements tend to be distributed along the bins of one STFT observation in $Y$, whereas harmonic elements tend to distribute relatively stably within neighboring frequency bins of consecutive frames of $Y$. Thus, the corresponding harmonic and percussive components can be obtained by applying a median filter to $Y$ in the horizontal and vertical directions, yielding the component magnitude spectrograms $\tilde{Y}_h$ and $\tilde{Y}_p$ with enhanced percussive content [54]:

$$\begin{aligned}\tilde{Y}_h(t,k) &:= \text{median}(Y(t-\ell_h,k),\ldots,Y(t+\ell_h,k)),\\ \tilde{Y}_p(t,k) &:= \text{median}(Y(t,k-\ell_p),\ldots,Y(t,k+\ell_p)),\end{aligned}$$

for $\ell_h,\ell_p \in \mathbb{N}$ where $2\ell_h+1$ and $2\ell_p+1$ are the lengths of the median filters, respectively.

Additionally, Driedger et al. [48] introduce the *separation factor* $\beta \in \mathbb{R}$, $\beta \geq 1$, which defines how "strict" the harmonic-percussive separation is to be, or how prominently percussive or harmonic a sound must be in order to be classified as such. Thus, binary masks $M_h$, $M_p$, and $M_r$ are defined for each component:

$$\begin{aligned}M_h^\beta(t,k) &:= \big(\tilde{Y}_h(t,k)/(\tilde{Y}_p(t,k)+\varepsilon)\big) > \beta\\ M_p^\beta(t,k) &:= \big(\tilde{Y}_p(t,k)/(\tilde{Y}_h(t,k)+\varepsilon)\big) \geq \beta\\ M_r^\beta(t,k) &:= 1-\big(M_h(t,k)+M_p(t,k)\big),\end{aligned}$$

where $\varepsilon$ is a small constant to avoid division by zero, and the operators $\geq$ and $>$ produce a binary result from $\{0,1\}$. Applying these masks to the original spectrogram $X$ yields the spectrograms for the harmonic, percussive, and residual components

$$\begin{aligned}X_h^\beta(t,k) &:= X(t,k)\cdot M_h^\beta(t,k),\\ X_p^\beta(t,k) &:= X(t,k)\cdot M_p^\beta(t,k),\\ X_r^\beta(t,k) &:= X(t,k)\cdot M_r^\beta(t,k).\end{aligned}$$

These can be transformed back into the time domain by means of the inverse short-time Fourier transform (ISTFT), yielding the signals $x_h^\beta$, $x_r^\beta$, and $x_p^\beta$ (see [69] for details).

We can summarize the HRP procedure through an operator $\Gamma^\beta$ such that $\Gamma^\beta[x] := (x_h^\beta, x_r^\beta, x_p^\beta)$. The operator $\Gamma^\beta$ takes a signal $x$ as input and delivers three component signals $(x_h^\beta, x_r^\beta, x_p^\beta)$ that were obtained by the HRPSS procedure, using $\beta$ as a separation factor. Note that $x = x_h^\beta + x_r^\beta + x_p^\beta$ (up to small signal reconstruction errors).

### 5.3.2 Cascaded HRP Procedure

In order to apply HRPSS in a cascaded fashion (CHRPSS), we require a list of separation factors $(\beta_1 > \beta_2, > \ldots > \beta_B)$, where $B \geq 1$ is the number of cascading stages. The following equations describe the stages in the procedure:

$$(x_h^{(1)}, x_r^{(1)}, x_p^{(1)}) := \Gamma^{\beta_1}[x]$$
$$(x_h^{(b)}, x_r^{(b)}, x_p^{(b)}) := \Gamma^{\beta_b}[x_r^{(b-1)}],$$

for $b = 2, \ldots, B$. In the first equation, HRP is applied to the original signal $x$ using a separation factor $\beta_1$. The second equation describes the cascade at stage $b$, where HRP is applied to the residual component from the previous stage ($x_r^{(b-1)}$) with a separation factor $\beta_b$.

### 5.3.3 Energy-Based CHRP Features

Once all the required cascading stages are complete, the component signals are sorted on an axis that goes from harmonic, through residual, to percussive:

$$x_h^{(1)}, x_h^{(2)}, \ldots, x_h^{(B)}, x_r^{(B)}, x_p^{(B)}, \ldots, x_p^{(2)}, x_p^{(1)}.$$

In general, CHRP will produce $(2B+1)$ component signals—their sum is equal to $x$ (up to a small error). For the examples shown in this chapter we have chosen $B = 3$, which produces seven component signals H, RH, RRH, RRR, RRP, RP, and P. Motivated by tasks such as musical event density and structure analysis, we now introduce a mid-level feature representation. To this end, we compute the local energy for each of the $2B+1$ component signals using a window $w^e$ and stack them into a feature matrix $V = (v_1, \ldots, v_M)$ such that $v_m \in \mathbb{R}^{2B+1}$, for $m \in [1 : M]$ (where $M$ is the number of elements or observations along the time axis). Furthermore, if we take $||V||_1$, where $||\cdot||_1$ denotes the $\ell$–1 norm, it follows that the columns $v_n \in [0,1]^{2B+1}$. Depending on the task at hand and the nature of the input signal, different energy window lengths may be chosen—we used a window length of 1.86 s for all the examples in this chapter.

### 5.3.4 Efficiency and Implementation Issues

For the sake of simplicity, our treatment of the HRP procedure (particularly, the $\Gamma^{\beta}$ operator) has assumed both input and output as time-domain signals. However, if the desired output is the CHRP feature matrix, energies can be computed in the time-frequency (TF) domain; it makes more sense (efficiency-wise) to only transform the original signal $x$ into the TF domain for the first pass, and remain there for all following stages. In this case, the length of the energy window $w^e$ becomes bound to the time resolution induced by the hop length $h^{\text{STFT}}$.

Throughout the processing pipeline are many parameters that may be modified to achieve task-specific results. To analyze finer-grained events (such as drum playing technique, Section 5.4.1), it is more

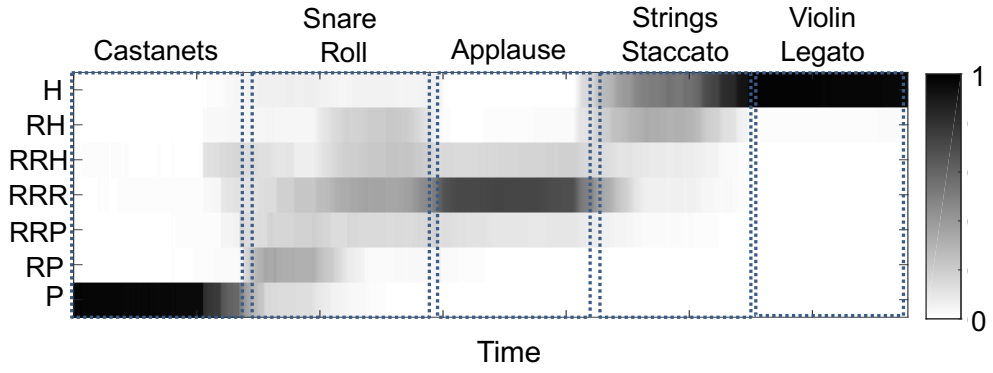



Figure 5.2: CHRP feature for a signal made with five non-overlapping sound samples. Note the ramp-like behavior beginning at the percussive component (castanets, bottom left) and ending at the harmonic component (violin-legato, top right).

meaningful to pick smaller STFT frame sizes and hop lengths, shorter median filter lengths (especially along the horizontal axis), and a more localized energy computation. For tasks at a larger timescale (such as structure analysis), the opposite parameter choices would be more adequate. This assortment of parameters induces a CHRP feature family that can be tailored to different scenarios. Finally, the CHRP feature matrix itself may benefit from different types of normalization and logarithmic compression.

In the following, we give examples of analysis tasks that show the potential of our CHRP feature and demonstrate the role that energies on the HRP axis play in different musical phenomena.

## 5.4 Applications

Two driving forces that shape the structure and tension in musical traditions throughout the world are event density and timbral homogeneity. In this section we use CHRP to study harmonic and percussive elements both in isolation and in relation to each other, at different timescales.

Before illustrating specific use cases for the CHRP feature, we wish to show the general capabilities of our feature. Figure 5.2 contains the CHRP feature matrix for a signal consisting of five non-overlapping sound samples: castanets, snare roll, applause, staccato strings and legato violin. The castanets have the highest percussive energy and are well confined to the P component. The snare roll is predominantly composed of RP and RRR. Indeed, snare drums have a percussive onset and a decay curve which is both noisy and tonal (according to the drum's tuning frequency). When struck in rapid succession, the noisy decay tails overpower the percussive onsets. Applause is centered around RRR and well-confined to the residual region. The staccato strings are predominantly harmonic, with an additional RH component which corresponds to the noisy onsets that emerge in this playing technique. Violin legato is confined to the H component, since the stable, harmonic signal properties dominate all other components.

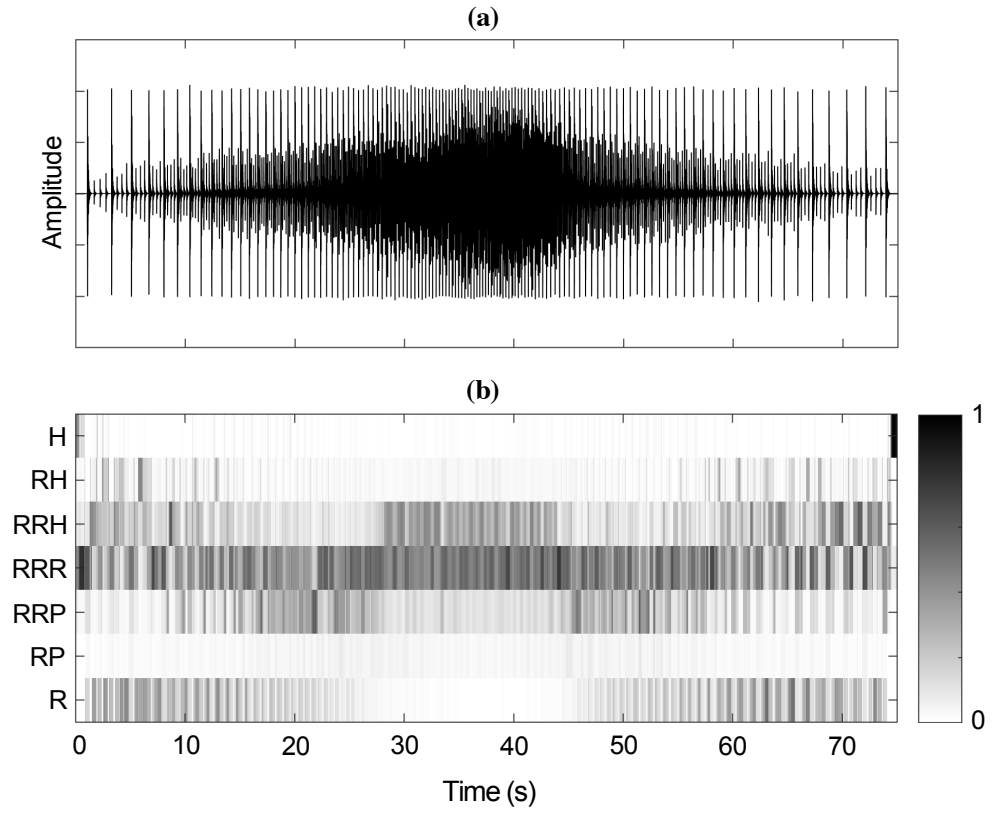



Figure 5.3: **(a)**: Waveform of *paradiddles* on snare drum. Onset frequency increases between 0–40 s and decreases again from 40–75 s. **(b)**: Cascaded Harmonic-Residual-Percussive energy feature (CHRP). Darker colors denote higher energy. Note that increasing onset frequency causes energy to migrate from the percussive components into the residual components.

### 5.4.1 Percussive Event Density

Our first example shows the migration of energy from percussive to residual in a snare-playing technique known as *paradiddles*. Figure 5.3a shows the waveform of paradiddles played on a snare drum; first with increasing speed (0–40 s), and then with decreasing speed (40–75 s). Figure 5.3b contains the corresponding CHRP feature matrix; notice how there is very little remaining P energy after a certain onset frequency or playing speed has been reached (around 25 s). This is due to the fact that the noise-like tails reach a relative proximity that overpowers the individual percussive onsets, centering the energy around the residual components in the feature matrix.

### 5.4.2 Musical Structure

Musical structure is manifested through different types of relationships among the sections of a song. One such type of relationships is homogeneity: the fact that a certain musical property (e. g., instrumentation

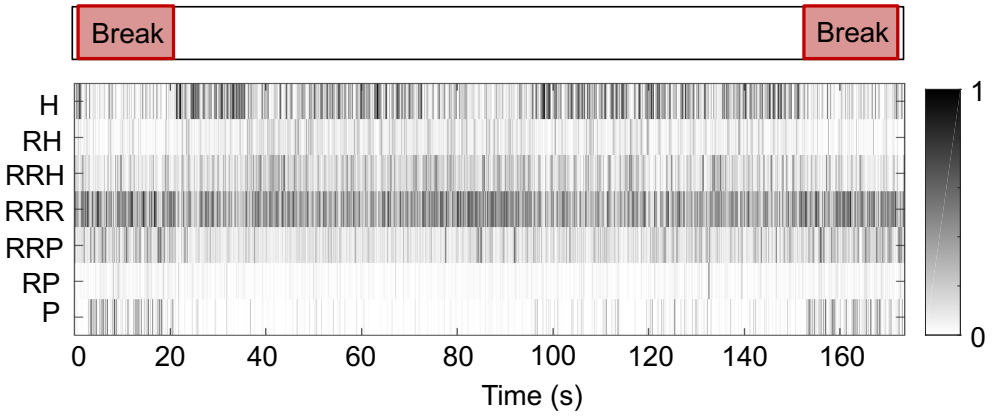



Figure 5.4: Structure annotation (top) and CHRP (bottom) for "Funky Drummer" by James Brown. Red rectangles enclose percussive passages known as *breaks*. Darker colors denote higher energy. Note the high P energy and low H energy in the parts annotated as *drum breaks*.

or tempo) is similar within the section. The focus of homogeneity analysis lies in the identification of longer passages that are coherent with respect to some musical property [97].

For example, hip hop producers often seek purely percussive passages (also known as *drum breaks* or *breaks*) within funk and soul recordings in order to sample them and create new musical material [118]. Figure 5.4 (top, red regions) shows a structural annotation of "Funky Drummer" by James Brown, focusing on the location of the drum breaks. Figure 5.4 (bottom) contains the corresponding CHRP feature matrix, where the drum breaks—a timbrally homogeneous musical structure—are visible at the beginning and end of the piece: notice how the annotated parts correspond to low H energy and high P energy.

Butler [14] describes phenomena in electronic music that are relevant to structure in terms of timbre and homogeneity. For instance, a typical resource to induce anticipation in the listener is to "withhold the beat", which occurs when the bass (kick) drum is temporarily removed from a track—having both textural and metrical influence on the piece. Furthermore, in many types of electronic music, *loops* (short audio fragments) are overlaid in different combinations to shape the track structure [90]. This results in locally homogeneous regions which are sometimes connected through what Butler calls *articulative* sounds: "brief and intermittent; they usually appear at or near structural boundaries". Thus, an approaching structural change is often *announced* by introducing noise filtered with an increasing cutoff frequency, by increasing the envelope release time of cymbal sounds (e. g., hi-hats and rides), or by activating a long-tailed reverb effect.

The CHRP feature can be used to visualize both kick drum removals and instances of noise as an articulative sound—they appear as a shift of energy into the residual components. In Figure 5.5 we show the CHRP feature matrix for the track "Liberum Spirita (Original Mix)" by Nicole Moudaber. We annotated three prominent timepoints (blue triangles, top of figure): (1) indicates the removal of the bass drum and the beginning of a section dominated by cymbal sounds and an increasing level of

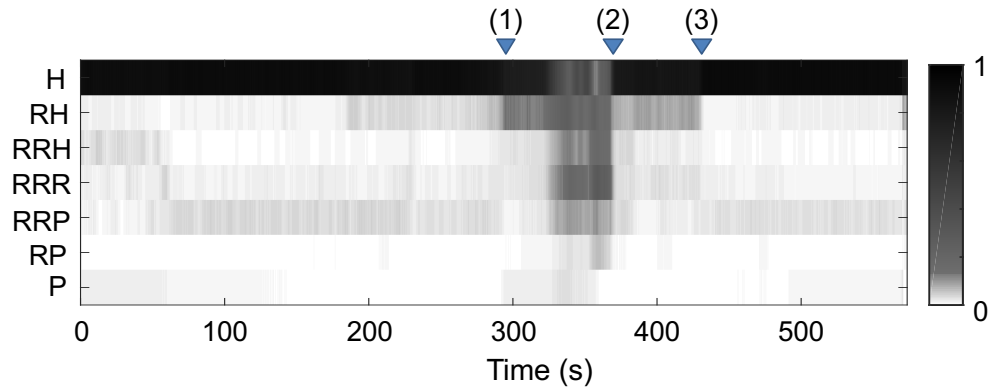



Figure 5.5: CHRP for "Liberum Spirita (Original Mix)" by Nicole Moudaber. The colormap has been shifted to enhance visibility. The annotations (blue triangles) indicate points of timbral change, seen as shifts in energy between harmonic and residual components.

noise-like components (notice the increase in all R components and the decrease in the H component); (2) indicates the reintroduction of the bass drum. The CHRP feature captures this event—somewhat counterintuitively—as an increase in H energy. This is explained by the fact that kick drum sounds contain most of their energy after the initial percussive attack, and electronic music producers often overlay them with low-pitched sinusoids to reinforce their effect. The region between (2) and (3) contains an articulative sound in the form of a noise burst. Timepoint (3) also marks the removal of a long-tailed ride cymbal sound, which can be seen as a drop in RRR energy. The decrease in RH energy at (3) is primarily due to the removal of a synthesizer sound consisting of both tonal and noise-like components.

## 5.5 Conclusions and Future Work

We introduced cascaded harmonic-residual-percussive features which capture timbral properties on the HRP axis. Through musical examples, we showed their capabilities to illustrate onset density and structural changes. The CHRP feature is a helpful extension of other features (e. g., pitch chroma or MFCCs) for two reasons. Firstly, because it is often difficult to classify an instrument's sound as purely harmonic or percussive. For instance, a synthesizer might be configured to play a melody with a mixture of sinusoids and white noise—while a human listener would be capable of recognizing the sound's tonal properties, it is very unlikely that a pitch-related feature (such as chroma) could accurately capture them. Our proposed feature is impervious to this fact and allows analysis on the HRP scale. Secondly, the CHRP representation is easy to interpret visually and can therefore assist timbre-related musical analysis. The main drawbacks of our current approach are the manual selection of feature dimensionality (or cascading steps) and the choice of separation factors at each stage. In future work, we plan to investigate the feasibility of finding optimal separation factors adaptively, and consider the CHRP feature alongside other features to test its potential in tasks like breakbeat detection and homogeneity-based structure analysis.

# Chapter 6

# Finding Drum Breaks in Digital Music Recordings

My main contribution in this chapter—based on the work in [92]—was identifying the task, preparing the dataset, and conducting the experiments. Christian Dittmar made major contributions to the dataset, provided critical insight about random forests (including evaluation issues), and supplied the original implementation (tailored for singing voice detection, SVD). Together with Meinard Müller, we formulated this task as a binary classification and identified the opportunity of testing whether the system for SVD could be *ported* to the task of drum break detection.

DJs and producers of SBEM use breakbeats as an essential building block and rhythmic foundation for their artistic work. The practice of reusing and resequencing sampled drum breaks critically influenced modern musical genres such as hip hop, drum'n'bass, and jungle. While SBEM artists have primarily sourced drum breaks from funk, soul, and jazz recordings from the 1960s to 1980s, they can potentially be sampled from music of any genre. In this chapter, we introduce and formalize the task of automatically finding suitable drum breaks in music recordings. By adapting an approach previously used for singing voice detection, we establish a first baseline for drum break detection. Besides a quantitative evaluation, we discuss benefits and limitations of our procedure by considering a number of challenging examples.

The main contributions of this chapter are as follows. In Section 6.2 we introduce the task of drum break detection and some of the difficulties that arise when trying to define it as a binary classification problem concerned with finding percussion-only passages. In Section 6.3 we present related work, the features we used, and a baseline approach adapted from a machine learning method for singing voice detection. By doing so, we explore how well machine learning techniques can be transferred to a completely different

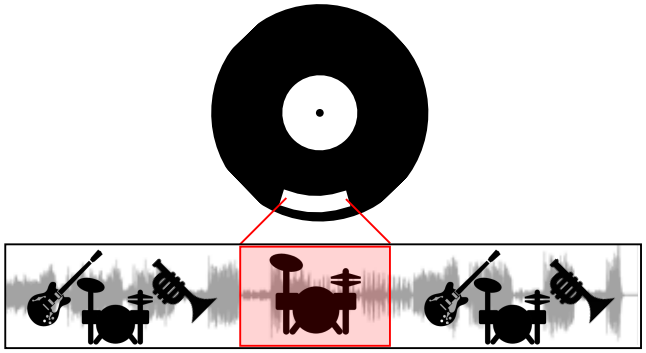


Figure 6.1: *Top*: Schematic illustration of a drum break on a vinyl record. *Bottom*: Location of drum break (enclosed in red) within waveform.

domain. In Section 6.4 we introduce our dataset and elaborate on its most important statistical properties, as well as our annotation process. The dataset represents the real-world music typically sampled in this SBEM scenario. Together with a statistical overview of the results, we also go into greater detail, analyzing two difficult examples we found in our dataset. In Section 6.5 we give conclusions and future work for this chapter.

## 6.1 Drum Breaks

Musical structure arises through the relationships between segments in a song. For instance, a segment can be characterized as *homogeneous* with respect to instrumentation or tempo [97, p. 171]. Structure is also driven by introducing or removing certain instruments, as is the case of solo sections. Music information retrieval (MIR) research has studied these phenomena through tasks such as structure analysis [107] and singing voice detection [85, 41]. In this chapter we focus on finding *drum break* sections, which are homogeneous with respect to instrumentation (i. e., they only contain drums) and often contrast with neighboring segments, where additional instruments are active.

Based mainly on [118], we present the notion of *drum break*, together with its history and usage. *Drum breaks*, *breaks*, or *breakbeats* are percussion-only passages typically found in funk, soul, and jazz recordings. Breaks first came into use within early hip hop DJ practice: by taking two copies of the same record (on separate turntables) along with a mixer, DJs could isolate and loop these sections, which were particularly popular with with the dancers and audience. When digital sampling technology became affordable for use at home and small studios, producers started using breaks to create their own tracks by adding further musical material and rearranging the individual drum hits into new rhythms. These musical practices are a cornerstone of genres like hip hop, jungle, and drum'n'bass; they helped develop a considerable body of knowledge about the nature and location of breakbeats, fostering a culture that valued finding rare and unheard breaks.

At this point, we need to make a distinction regarding the term *break*. The sections that producers choose for looping and sampling are not always exclusively made up of percussion instruments: many famous breaks also contain non-percussion instruments, such as bass or strings. Under a broader definition, a

break can be any musical segment (typically four measures or less), even if it doesn't contain percussion [118]. In this chapter we use *break* to designate regions containing only percussion instruments, for two reasons. First, percussion-only breaks afford producers the greatest flexibility when incorporating their own tonal and harmonic content, thus avoiding dissonant compositions (also known as "key clash"). Second, it allows us to unambiguously define our task: given a funk, soul, or jazz recording, we wish to identify passages which only contain percussion, i.e., detect the drum breaks. In the top portion of Figure 6.1 we show, in a simplified manner, the location of a drum break on a vinyl record. On the bottom we illustrate our task in this chapter, which is finding percussion-only regions in digital music recordings. Originally, vinyl records were the prime source for sampling material. When looking for rare breaks, artists visit record stores, basements and flea markets (known as "[crate] digging"). Once they acquire a record, artists carefully listen to the entire content, sometimes skipping at random with the needle on the turntable, until they find an appealing section to work with. Motivated by the current predominance and size of digital music collections, we propose a method to automate digging in the digital context.

## 6.2 Task Specification

In Section 4.1 we set our task as finding percussion-only passages in a given musical piece—a seemingly straightforward problem definition. Detecting breaks can thus be reduced to discriminating between percussion instruments—which contribute exclusively to rhythm—and all other instruments which contribute (mainly) to melody and harmony. We will now examine why this distinction is problematic from the perspective of prevalent music processing tasks. *Harmonic-percussive source separation* (HPSS) is a technique that aims to decompose a signal into its constituent components, according to their spectral properties [54, 97]. HPSS methods usually assign sustained, pitched sounds to the harmonic component, and transient, unpitched sounds to the percussive component. From an HPSS standpoint, our class of desired instruments has both harmonic and percussive signal components: many drum kit pieces such as kicks, snares, and toms have a discernible pitch, although they are not considered to contribute to the melody or harmony. Thus, drum break retrieval is difficult because of overlapping acoustic properties between our desired and undesired instruments—in other words, it is very hard to give an intensional definition of our target set's characteristics.

Our task lies between the practical definition of *drum break* and a technical definition used for automated retrieval—with a significant gap in between. On the technical side, we have opted for the term *percussion-only passage* instead of *drum break* due to the presence of percussion instruments which are not part of a standard drum kit, such as congas, bongos, timbales, and shakers. As a final note, we also distinguish between [*drum*] *breaks* and *breakbeats*: we interpret the former as a percussion-only segment within a recording (in its "natural" or "original" state), and the latter as a [drum] break which has been spotted and potentially manipulated by the artist.

## 6.3 Baseline System and Experiments

### 6.3.1 Related Work

There are relatively few publications in the MIR field on drum breaks and breakbeats. In [137] and [5], the authors investigate automatic sample identification for hip hop via fingerprinting. A multifaceted study of breakbeats which covers beat tracking, tempo induction, downbeat detection, and percussion identification can be found in [72]. Furthermore, given a certain breakbeat, the authors of [73] automatically retrieve hardcore, jungle, and drum'n'bass (HJDB) tracks where it is present. On the musicological side, [112] proposes a typology of sampled material in EDM, where breakbeats are considered at fine and coarse temporal scales. Putting our work in context, we can think of the typical artistic workflow in two steps: drum break discovery and manipulation. To the extent of our knowledge, research has mainly focused on the second phase—after breakbeats have been extracted and manipulated. Following this analogy, our task would be at the first stage, where the artist wishes to filter useful musical material. In a fully automated pipeline, our proposed system could be inserted before any of the tasks mentioned above.

### 6.3.2 Baseline System

Our baseline system follows [85], an approach for singing voice detection (SVD). SVD is used to determine the regions of a music recording where vocal activity is present. In [85], the authors address the problem that automated techniques frequently confuse singing voice with other pitch-continuous and pitch-varying instruments. They introduce new audio features—*fluctogram*, *spectral flatness/contraction*, and *vocal variance* (VOCVAR)—which are combined with mel-frequency cepstral coefficients (MFCCs) and subjected to machine learning. VOCVAR is strongly related to MFCCs—it captures the variance in the first five MFCCs across a number of consecutive frames. Spectral contraction and spectral flatness (FLAT) are extracted in logarithmically spaced, overlapping frequency bands. The spectral contrast features OBSC [77] and SBSC [3] encode the relation of peaks to valleys of the spectral magnitude in sub-bands. In general, both variants can be interpreted as harmonicity or tonality descriptors.
In this chapter we use a subset of the features from [85], along with a set of novel features derived from *harmonic-residual-percussive source separation* (HRPSS). HRPSS is a technique used to decompose signals into tonal, noise-like, and transient components [48]. The *cascaded harmonic-residual-percussive* (CHRP) feature was recently proposed in [93]; by iteratively applying HRPSS to a signal and measuring the component energies, this feature captures timbral properties along the HRP axis. We have included a seven-dimensional variant of CHRP for our experiments. Concatenating all features results in a vector with 83 entries (dimensions) per spectral frame. The set of all vectors makes up our feature matrix, which is split into training, validation, and test sets for use with machine learning.
Again following [85], we employ random forests (RF) [9] as a classification scheme. RF deliver a

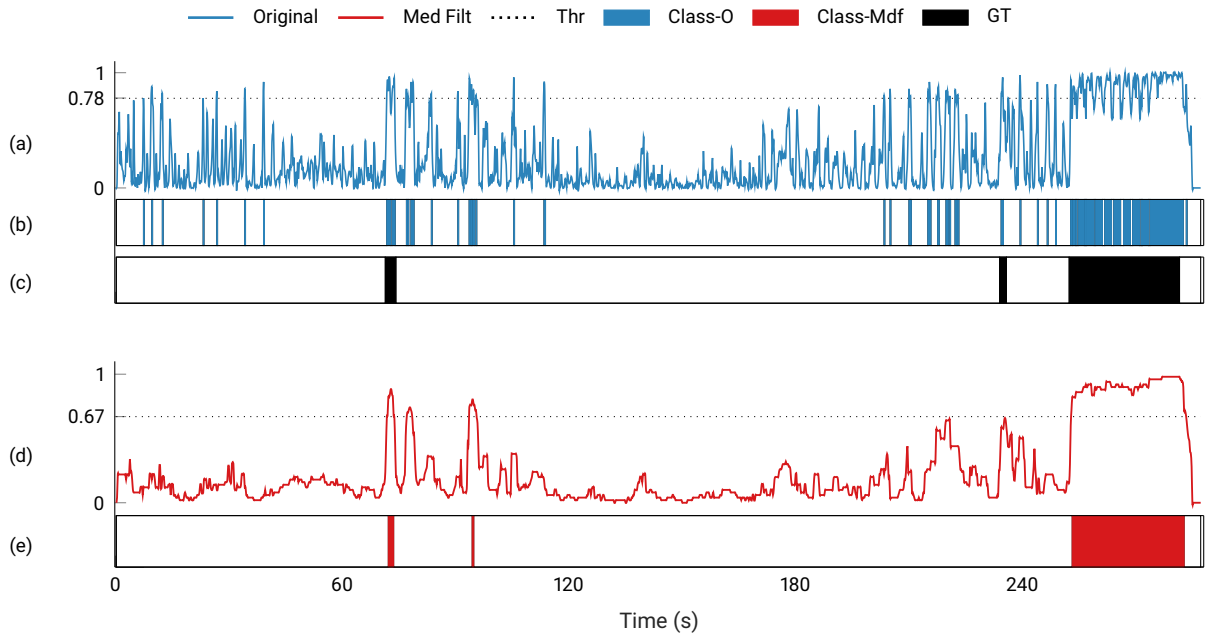



Figure 6.2: **(a)**: Original, unprocessed decision function (the output of the random forest classifier, interpreted as the confidence that a frame belongs to the *percussion-only* class, solid blue curve); optimal threshold value (0.78, dotted black line). **(b)**: Binary classification for original decision function (blue rectangles). **(c)**: Ground truth annotation (black rectangles). **(d)**: Decision function after median filtering with a filter length of 2.2 s (solid red curve); optimal threshold (0.67, dotted black line). **(e)**: Binary classification for median-filtered decision function (red rectangles).

frame-wise score value per class that can be interpreted as a confidence measure for the classifier decision. In our binary classification scenario, the two score functions are inversely proportional. We pick the one corresponding to our target *percussion-only* class and refer to it as *decision function* in the following. A decision function value close to 1 indicates a very reliable assignment to the percussion-only class, whereas a value close to 0 points to the opposite. Only frames where the decision function value exceeds the threshold will be classified as belonging to the percussion-only class. Prior to binarization, the decision function can be smoothed using a median filter, helping stabilize the detection and preventing unreasonably short spikes where the classification flips between both classes. Figure 6.2 illustrates the concepts mentioned above. Figure 6.2a (blue curve) shows the original, unprocessed decision function for "Funky Drummer" by James Brown. The ground truth (black rectangles, 6.2c) has three annotated breaks: shortly after 60 s, shortly before 240 s, and after 240 s (black rectangles). In 6.2d (red curve) we show the median-filtered decision, using a filter length of 2.2 s. In both 6.2a and 6.2d, the dotted black line represents the decision threshold (0.78 and 0.67, respectively). In 6.2b (solid blue rectangles) and 6.2e (solid red rectangles) we show the classification results for the original and median-filtered decision functions. In the remainder of this chapter, all plots related to the original decision function are blue; the ones corresponding to median filtering are red.

Table 6.1: Statistical overview for audio and annotation data.

| *Measure* | *Track Length* (s) | *Break Length* (s) | *Breaks (per track)* |
|---|---|---|---|
| min | 77.00 | 0.84 | 1.00 |
| max | 1116.00 | 224.84 | 9.00 |
| mean | 277.11 | 10.23 | 1.55 |
| median | 251.80 | 6.83 | 1.00 |
| std. dev. | 128.49 | 17.43 | 1.18 |
| Dataset total | 77600.00 | 4500.00 | 433.00 |

## 6.4 Evaluation

### 6.4.1 Dataset

Our dataset consists of 280 full recordings, covering funk, soul, jazz, rock, and other genres. All audio files are mono and have a sampling rate of 22050 Hz. Each track has a corresponding annotation with timepoints that enclose the breaks.[1] Two main principles guided our annotation style. First, we included regions containing strictly percussion, disregarding metrical structure. For example, if a break contained trailing non-percussive content from the previous musical measure (bar), we set the starting point after the undesirable component had reasonably decayed. Our second principle regards minimum duration: although we mostly annotated breaks spanning one or more measures, we also included shorter instances. The criterion for considering shorter fragments has to do with *sampleability*—if a percussive section contains distinct drum hits (for instance, only kick or snare), it is included. On the other hand, a short fill (such as a snare roll or flam) would not be annotated. Table 6.1 has an overview of the statistics for our audio and annotation data. The shortest break, an individual cymbal sound, lasts 0.84 s. The longest break, corresponding to an entire track, has a length of 224.84 s. The median track length is 251.80 s, and the median break length is 6.83 s. The relative rarity of drum breaks as a musical event is noteworthy: 5.81% of the entire dataset is labeled as such.

During the annotation process, we made interesting observations on how humans treat the breakbeat retrieval problem. We used Sonic Visualiser [16] to annotate the start and end of percussion-only passages. At the dawn of SBEM, when vinyl was the only medium in use, artists devotedly looking for unheard breakbeats would listen to a record by skipping with the needle through the grooves.[2] With the help of Sonic Visualiser, we found it very effective to scrub through the audio, moving the playhead forward at short, random intervals, also using visual cues from the waveform and spectrogram.
For our particular task, this fragmented, non-sequential method of seeking breaks seemed to be sufficient for listeners with enough expertise. This leads us to believe that a frame-wise classification approach provides a satisfactory baseline model for this task. Of course, in order to refine the start and end of the

[1] A complete list of track titles, artists, and YouTube™ identifiers is available at the accompanying website, along with annotations in plaintext. `https://www.audiolabs-erlangen.de/resources/MIR/2017-CMMR-Breaks`

[2] In [11, p. 247], the authors relate that "[Grand Wizard] Theodore could do something amazing: he could find the beginning of a break by eye and drop the needle right on it, with no need to spin the record back."

percussion-only passages, we had to listen more carefully and visually inspect the waveform. As we will show, precise localization also poses a major challenge for automatic break retrieval.

### 6.4.2 Results

We now discuss the evaluation results for our experiments conducted on the entire dataset. We use the *framewise F-measure* as an evaluation strategy. Frames with a positive classification that coincide with an annotated break are counted as *true positives* (TP), frames with a negative classification that coincide with an annotated break are *false negatives* (FN), and frames with a positive classification that do not coincide with an annotated break are considered *false positives* (FP). The three quantities are represented in the F-measure by

$$F = \frac{2 \cdot \mathrm{TP}}{2 \cdot \mathrm{TP} + \mathrm{FP} + \mathrm{FN}}. \tag{6.1}$$

In order to reduce the *album effect*, we discarded tracks from our dataset, arriving at a subset with one track per unique artist (from 280 to 220 tracks). We performed a ten-fold cross validation: for each fold, we randomly chose 70% of the tracks for training (155 tracks), 15% (33 tracks) for validation, and 15% (34 tracks) for testing. Since the classes *percussion-only* and *not only percussion* are strongly unbalanced (see statistic in Section 6.4.1), training was done with balanced data. That means that for each track, all *percussion-only* frames are taken as positive examples, and an equal number of *not only percussion* frames are taken as negative examples. Validation and testing are done with unbalanced data [20]. During validation, we perform a parameter sweep for the decision threshold and median filter length.

Figure 6.3 gives an overview of experiments with threshold and median filter length. Figure 6.3a is a parameter sweep matrix across all folds, where each row corresponds to a certain threshold value, and each column is a median filter length. Each entry in the matrix is the mean F-measure across all ten folds, for the testing phase. Darker entries represent a higher F-measure—the colormap was shifted to enhance visibility. The red circle denotes the optimal configuration: a threshold of 0.67 and a median filter length of 4.6 s yield an F-measure of 0.79. The blue triangle at threshold value 0.78 indicates the highest F-measure (0.68) without median filtering. Figures 6.3c and 6.3d contain curves extracted from this matrix. In Figure 6.3c, the curve corresponds to the highlighted row in the matrix (i. e., for a fixed threshold and varying median filter length). Figure 6.3d is the converse: we show the highlighted column of the matrix, with a fixed median filter length and varying threshold. In both 6.3c and 6.3d, the light red area surrounding the main curve is the standard deviation across folds. Figure 6.3b compares the F-measures between the original (unprocessed) binarized decision curve (blue) and after median filtering (red)—with respect to increasing thresholds (horizontal axis).

In Figure 6.3a we can see that the choice of threshold has a greater effect on F-measure than the median filter length: the differences between rows are more pronounced than between columns. Indeed, Figure 6.3b shows that *original* and *median filtered* have a similar dependency on the threshold. The solid red curve in Figure 6.3c starts at an F-measure of 0.67 (without median filtering), reaches a peak

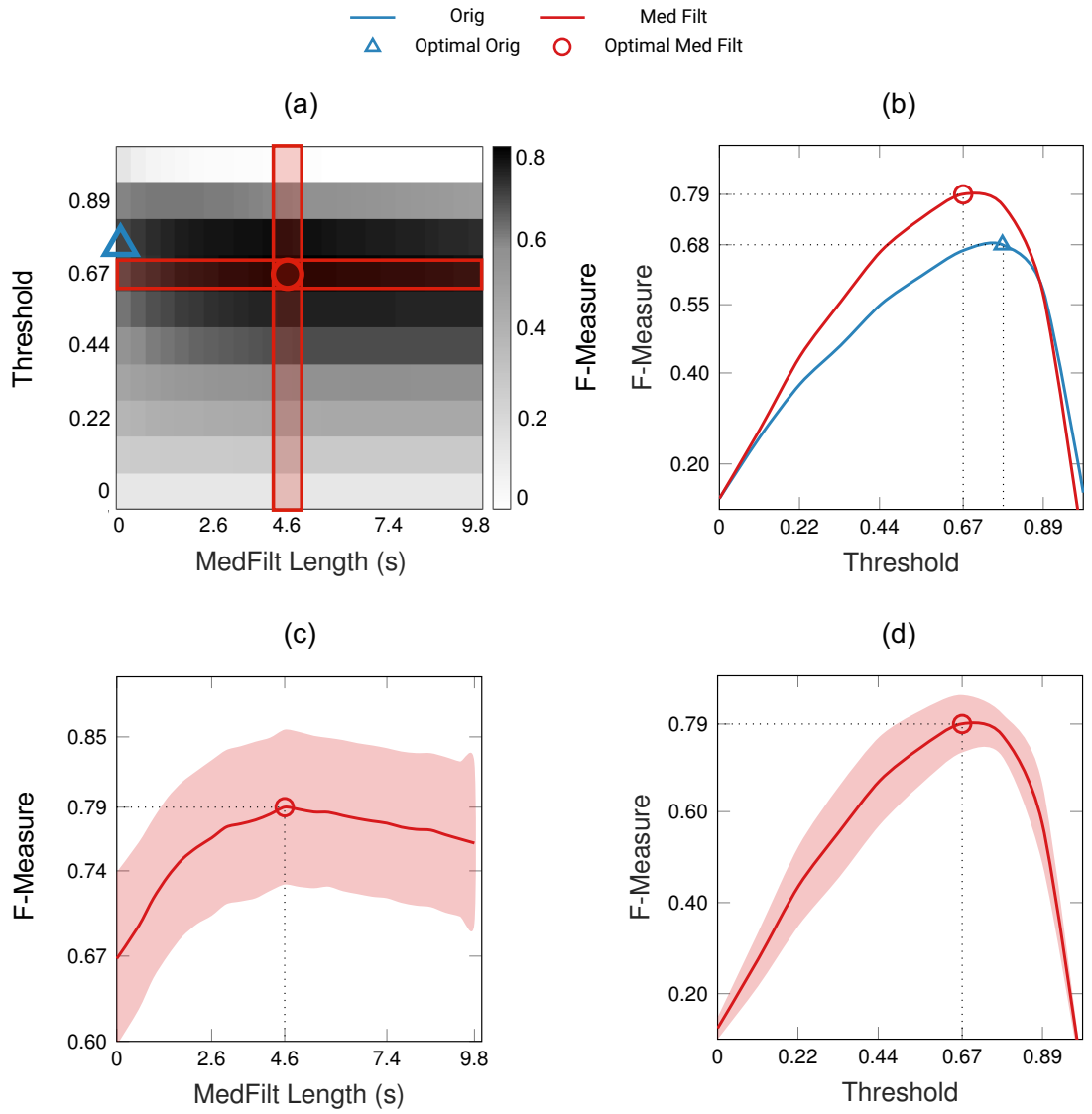



Figure 6.3: **(a)**: Parameter sweep for median filter length (horizontal axis) and threshold (vertical axis). Darker entries of the matrix have a higher mean F-measure. The colormap has been shifted to enhance visibility, markers denote optimal configurations. **(b)**: F-measure for unprocessed (blue) and median filtered (red) decisions, depending on threshold (horizontal axis). **(c)**: Highlighted row of parameter sweep matrix. **(d)**: Highlighted column of parameter sweep matrix.

at F-measure 0.79 (median filter length 4.6 s) and then drops to 0.76 for the longest tested median filter window (9.8 s). The standard deviation is stable across all median filter lengths, amounting to about 0.06. In Figure 6.3d, the mean F-measure goes from below 0.2 (at threshold 0), through the optimal value (0.79 at threshold 0.67), and decays rapidly for the remaining higher threshold values. The standard deviation widens in proximity to the optimal F-measure. It is interesting that the optimal median filter length (4.6 s) is about half the mean annotated break length (10.23 s). This median filter length seems to offer the best

Table 6.2: Evaluation results with optimal parameters. Rows are statistical measures, columns are experimental configurations.

| *Measure* | *Original* | *MedFilt* | *Random* | *Biased* |
|---|---|---|---|---|
| mean | 0.6816 | 0.7923 | 0.0963 | 0.1170 |
| median | 0.6636 | 0.7997 | 0.0951 | 0.1133 |
| std. dev. | 0.0620 | 0.0636 | 0.0180 | 0.0259 |

trade-off between closing gaps in the detection (as seen in the last break of Figure 6.2e), removing isolated false positives (seen throughout Figure 6.2e), and reducing true positives (seen in the two first short breaks in Figure 6.2e).

Table 6.2 summarizes statistics over multiple experimental configurations. Each column contains the mean, median and standard deviation (SD) for the F-measure across ten folds. The mean F-measure for the original (unprocessed) decision function is 0.68, and it corresponds to a threshold value of 0.78. Median filtering with a length of 4.6 s, together with a threshold of 0.67, yields an optimal mean F-Measure of 0.79. Generating a random decision function leads to a mean F-Measure of 0.09; labeling all frames as *purely percussive* (seen in the *biased* column) delivers 0.11.

The first important result from Table 6.2 is that variants of our approach (original and median-filtered) yield F-measures between six and seven times higher than randomly generating decision functions, or simply labeling all frames as *percussion-only* (biased). Second, we can see that median filtering increases the F-measure by about 0.11 (from 0.68 to 0.79). As seen in Figure 6.2b and 6.2e, median filtering removes most false positives (boosting precision), but can also diminish (or completely remove) true positives, as is the case with the short break after 240 s. Finally, we can also be confident of the usefulness of median filtering because it increases the mean F-measure without affecting the standard deviation (0.06 in both cases).

### 6.4.3 Some Notorious Examples

Beyond the large-scale results from Section 6.4, we now show some examples that posed specific challenges to our classification scheme. The first case is "Ride, Sally, Ride" by Dennis Coffey (Figure 6.4), recorded in 1972. From top to bottom, Figure 6.4 shows the decision function output by the RF (solid blue curve), the threshold value optimized during validation and testing (black dotted line), the frames estimated as percussion-only (blue rectangles), and the ground truth annotation (black rectangle). Focusing on the annotated segment (shortly after 60 s), we can see that the decision curve oscillates about the threshold, producing an inconsistent labeling. We attribute this behavior to eighth-note conga hits being played during the first beat of each bar in the break. These percussion sounds are pitched and appear relatively seldom in the dataset, leading to low decision scores. Our second example, seen in Figure 6.5, is "Dusty Groove" by The New Mastersounds. On this modern release from 2009, the production strives to replicate the "vintage" sound found on older recordings. Around the annotated region (240 s) we highlight two issues: during the break, there are few frames classified as percussion-only, and the decision curve maintains a relatively high mean value well after the break. Again, we ascribe the misdetection during the

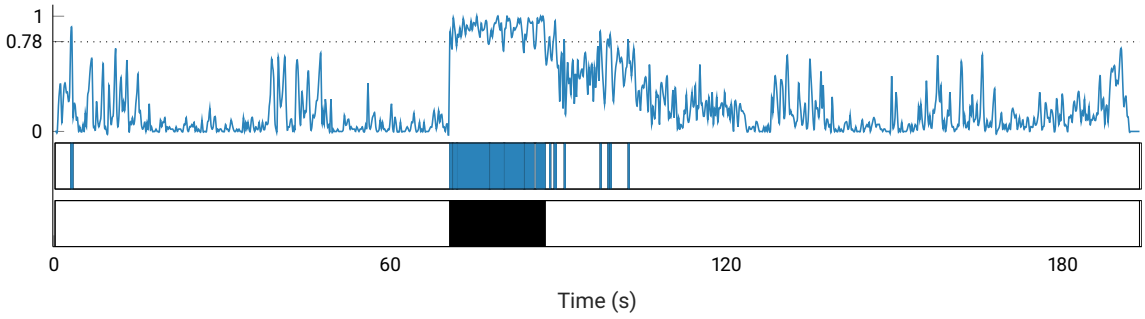



Figure 6.4: Results for “Ride, Sally, Ride” by Dennis Coffey. From top to bottom: unprocessed decision function (solid blue curve) and threshold (dotted black line), classification (blue rectangles), GT annotation (black rectangle). Note the high decision function values immediately following the annotated break.

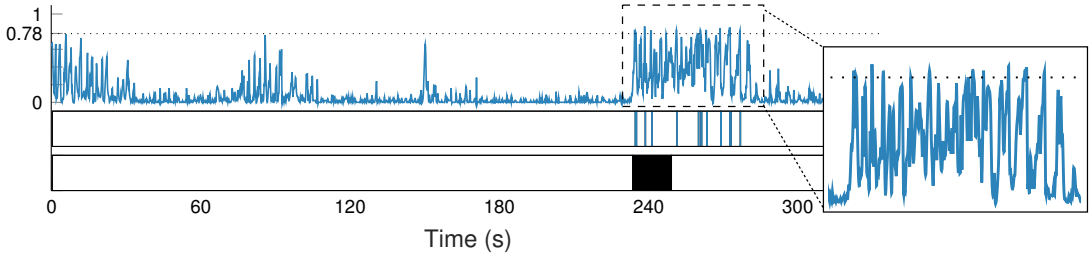



Figure 6.5: “Dusty Groove” by The New Mastersounds. Note the number of false negatives during the annotated break and the high values in the decision function after the break.

break to the presence of drum hits with strong tonal content. Especially the snare has a distinct sound that is overtone-rich and has a longer decay than usual, almost reminiscent of timbales. After the annotated break the percussion continue, but a bass guitar playing mostly sixteenth and syncopated, staccato eighth notes appears—upon closer inspection, we observed that the onsets of the bass guitar synchronize quite well with those of the bass drum. When played simultaneously, the spectral content of both the bass drum and bass guitar overlaps considerably, creating a hybrid sound closer to percussion than a pitched instrument.

## 6.5 Conclusions and Future Work

We presented a system to find percussion-only passages (drum breaks) in digital music recordings. To establish a baseline for this binary classification task, we built our system around the work of [41] and [85]. With this chapter we investigated to which extent binary classification methods are transferable across tasks.

Having established this baseline, in future work we wish to improve detection for difficult examples

(as described in Section 6.4.3), and include genres beyond the ones studied here. When implementing machine learning techniques, it is important to address the issue of overfitting. We are aware that our dataset induces a strong genre-related bias, but it reflects the real-world bias (or practice) that SBEM artists follow when selecting sampling material. Going beyond results with the F-measure, an interesting alternative to evaluate our approach would be to conduct user experience tests, measuring the potential speedup in drum break location. As for applications, DJs and producers could use our system to retrieve drum breaks from large digital collections, considerably reducing the time needed for *digging*. Our procedure can also be used as a pre-processing step for breakbeat identification tasks, as outlined in [5] and [73]; or for structure analysis of loop-based EDM [90]. Finally, MIR researchers could use this system to compile datasets for other tasks such as beat tracking and drum transcription.

# Chapter 7

# Estimating Sixteenth Note Swing Ratio in Drum Break Recordings

This chapter is the result of a collaboration with Christian Dittmar, Andrew V. Frane, and Meinard Müller. For this chapter I identified—together with Christian Dittmar—the potential for reproducing the manual findings of Frane [56] with the automatic swing ratio estimation authored mainly by Christian Dittmar. My main contributions are as follows. First, I transformed Frane's dataset from proprietary formats to standard formats for MIR. Second, together with Christian Dittmar, I prepared the experimental setup and incorporated drum source separation into the swing ratio estimation task. Andrew V. Frane provided input for the whole text, but especially regarding the correct formulation of musical concepts and the analysis and interpretation of results.

*Swing* is a microrhythmic trait that can be heard in performances of many types of music. The key element of swing is an unequal subdividing of the pulse at a particular metrical level, so that odd-numbered divisions are slightly longer than their adjacent, even-numbered counterparts. For instance, in jazz, swing is typically applied at the eighth-note level, so that the first eighth-note subdivision of each quarter-note beat is longer than the second eighth-note subdivision of that beat. In some other music genres, such as funk, soul, and rock, swing is more commonly applied at the sixteenth-note level. These genres are often the source of *drum breaks*, i.e., percussion-only passages that are frequently sampled from older recordings and repurposed within new compositions. In this chapter, we semi-automatically estimate the amount of sixteenth-note swing in drum breaks, using the concept of *swing ratio* (the ratio of odd to even subdivision duration). Our starting point is a recent study by Andrew Frane, who computed swing

ratios in 30 drum breaks based on manual annotations. We contribute to this line of research in several ways. First, we formalize Frane's method of swing ratio computation as an algorithm. Second, we adapt an automatic swing ratio estimation method (which had been developed for eighth-note swing) for use at the sixteenth-note level. Third, we use state-of-the-art drum source separation to examine the role of individual drum kit components in swing ratio estimation. Finally, we go beyond the initial dataset used by Frane and analyze swing ratios on a larger corpus of over 500 drum breaks.

We now summarize our contributions and describe how this chapter is organized. In Section 7.2 we review and formalize the methodology from [56] for computing sixteenth-note swing ratios based on onset annotations. In Section 7.3 we extend and adapt an automatic swing ratio estimation procedure based on pattern matching of the log-lag autocorrelation function (LLACF) by [46] and [45]. We investigate enhancing the required signal representations with an eighth-note grid, as well as estimating swing ratios of individual drum kit components obtained with drum source separation methods. We also give critical insight as to why LLACF-based methods deliver erroneous swing ratio estimates for drum breaks under certain circumstances. In Section 7.4 we show how we validated a dataset from [56] and reproduced previous manual results with the automated methods discussed in Section 7.3. As a further contribution, we go beyond the initial 29-item annotated dataset from Section 7.2 and perform an explorative statistical analysis on a corpus of more than 500 unannotated drum breaks. We conclude the chapter with some criticism about quantifying swing as a ratio and discuss potential future work.

## 7.1 Context and Related Tasks

This chapter involves the intersection of three research topics: *swing*, *drum breaks*, and *drum source separation*. Swing is a type of systematic microtiming (defined in Section 7.1.1) that is present in many styles of music, but has gone relatively unnoticed outside of jazz research. It can be quantified by the *swing ratio* (also defined in Section 7.1.1). Drum breaks are percussion-only passages (often found in funk, soul, rock, and jazz-funk recordings from the 1960s to 1980s) that are highly relevant to contemporary music, but have received little attention from the fields of musicology, music cognition/perception, and music information retrieval (MIR). Drum source separation is used to isolate individual drum kit components (e.g., kick drum, snare, and hi-hat) from a recording of those elements in combination. In this chapter, we build on the work of [56] by formalizing his method of swing ratio computation and by using automated audio processing methods to reproduce his findings regarding swing ratios in drum breaks. Moreover, we adapt the drum source separation method of [44] to examine the swing ratios of individual drum kit components.

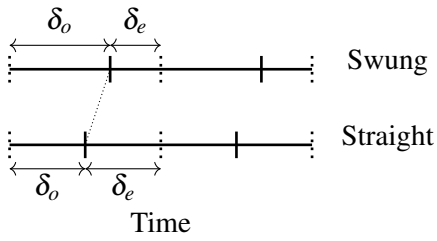



Figure 7.1: Musical time-grids illustrating swing. On both axes, odd-numbered onsets are shown as dotted vertical lines; even-numbered onsets are solid lines. The bottom time-grid is for a *straight* (swing-less) rhythm, i.e., the odd-numbered interval length $\delta_o$ is equal to the even-numbered interval length $\delta_e$. The top time-grid is for a rhythm with swing, i.e., $\delta_o > \delta_e$ (because each even-numbered division's onset is delayed).

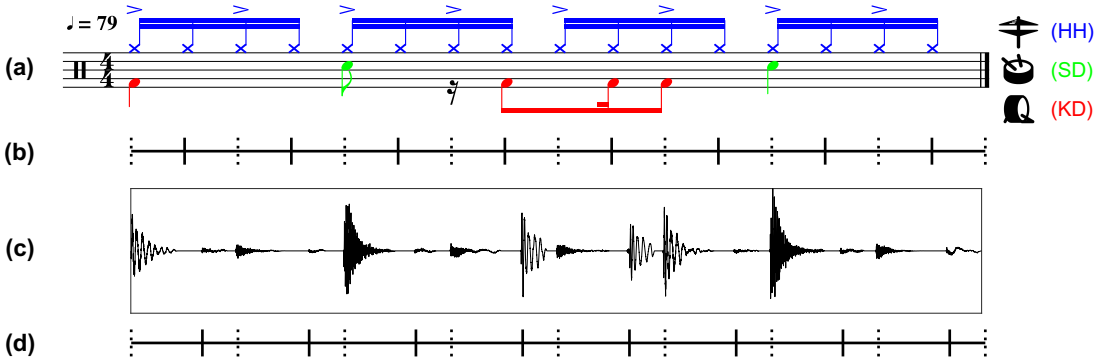



Figure 7.2: Schematic overview of the rhythmic phenomenon of swing for a drum break. **(a)**: Drum score notation with kick drum (red, bottom of staff), snare drum (green, mid-staff), hi-hat (blue, top of staff). **(b)**: Straight sixteenth-note time-grid. **(c)**: Waveform for excerpt of "Theme From the Planets" by Dexter Wansel, modified to emphasize the swing. **(d)**: Time grid with swung onsets corresponding to the waveform.

### 7.1.1 Swing and the Swing Ratio

Musicians apply *swing* to a rhythm by slightly delaying ("swinging") the onsets of notes that occur at even-numbered beat-divisions (at a particular metrical level, typically the eighth-note or sixteenth-note level). This effectively lengthens the odd-numbered divisions and shortens the even-numbered divisions, as illustrated in Figure 7.1. The magnitude of swing can be quantified by the swing ratio (SR): $s := \delta_o / \delta_e$, where $s$ is the swing ratio, $\delta_o$ is an odd-numbered division's duration (i.e., the time interval from the odd-numbered division's onset to the ensuing division's onset), and $\delta_e$ is an even-numbered division's duration. Thus, $s = 1$ indicates a *straight* (swing-less) rhythm, and greater values of $s$ (at a given tempo) indicate greater delays of the even-numbered division onset, i.e., greater discrepancy between $\delta_o$ and $\delta_e$.

For illustrative purposes, we have generated a semi-synthetic running example—a version of Dexter Wansel's "Theme From the Planets" which we manually modified to have a swing ratio of 2 throughout the entire drum break (the original swing ratio being 1). This provides a controlled example drum break

to optimally illustrate the concepts and procedures that we discuss in the chapter. Figure 7.2a shows the drum score notation for the break. In Figure 7.2b we have placed a straight grid, which lines up with the notation in 7.2a. Figure 7.2c shows the waveform for our modified signal. Figure 7.2d is a grid with onsets swung at a ratio of 2, aligned with the waveform. Figure 7.2 can be conceptually divided into two realms: the realm of musical notation, where the exact swing ratio is not expressed or encoded (7.2a and 7.2b), and the realm of musical performance, where swing is applied by musicians (7.2c and 7.2d).

Swing ratio has been studied prominently in jazz research [see 109, 143]. For example, Friberg and Sundström [58] manually annotated ride cymbal onsets in spectrogram plots of jazz recordings to quantify eighth-note swing ratios. Dittmar et al. [46, 45] later developed automated methods that largely confirmed previous findings. Beyond the relatively well-studied characteristics of microtiming in jazz music in general and swing ratio estimation in particular, there is a growing interest in investigating microtiming in other styles of music, such as funk and soul. For instance, Räsänen et al. [111] conducted a case study of Michael McDonald's "I Keep Forgettin'", an emblematic soul recording featuring drummer Jeff Porcaro. The researchers presented an in-depth audio analysis of Porcaro's one-handed hi-hat playing technique, revealing characteristic patterns of accents and sixteenth-note inter-onset intervals (IOIs). In their study of the effects of microtiming on subjective evaluations of rhythms, Davies et al. [35] included rhythmic stimuli that were based on the microtiming from Clyde Stubblefield's drumming in Jame's Brown's "Funky Drummer". In [95], Marchand and Peeters propose an autocorrelation-based SRE method, along with a large dataset of swing ratio annotations at the eighth-note level. Senn et al. [122] used recordings of bass-and-drums duos, some of which were in a funk style, to study the effects of microtiming on listeners' emotional response and perception of "groove". Frane and Shams [57] studied the threshold of swing detection among drummers and non-drummers, using genre-nonspecific electronic drum loops.

### 7.1.2 Drum Breaks

The term *break* has a rich history and a somewhat flexible definition, recounted here based on the work of [118]. In the end of the 1970s, at the dawn of hip-hop, party DJs noticed that the audience reacted most enthusiastically to breaks—sections of a record with isolated (or nearly isolated) percussion. In fact, a great break could come from an otherwise uninteresting song. By using two copies of the same record on two separate turntables connected to a mixer, DJs could alternate back and forth between copies, effectively looping the break and extending it at will. The next major advance in the use of breaks came with the arrival of affordable digital sampling technology in the late 1980s. Producers started sampling interesting parts of their favorite records in order to rearrange and layer them with other musical material. These musical practices of looping and sampling are pillars of music styles such as hip-hop and drum'n'bass. Throughout the years, these genres have gained popularity and mainstream attention. Furthermore, the practice of sampling breaks has transcended these seminal styles, with pop artists such as Justin Bieber, Hanson, and Alanis Morissette also incorporating sampled breaks into their songs.

Although the term *break*, like the similar term *breakbeat*, is often considered as referring specifically to percussion-only passages, Schloss [118, p. 36] noted that "today, the term 'break' refers to *any* segment of music (usually four measures or less) that could be sampled and repeated." If one accepts that looser definition, then the term *drum break*—which unambiguously implies isolated (or nearly isolated) percussion—may be considered as a special class of break that holds a uniquely celebrated status among breaks at large. Not only are drum breaks especially revered by DJs and sampling enthusiasts, drum breaks are also in some ways ideal materials for studying swing and other microrhythmic properties. Indeed, drum breaks are unhampered by harmonic and melodic information, and their extensive historical use ensures their ecological validity as source material. Therefore, it is perhaps somewhat surprising that drum breaks have so rarely been the topic of scientific study. An exception is the work by [92], which approached the task of automatically finding drum breaks in digital music recordings. They treated the task as a framewise classification problem, reduced to deciding whether a given short-time frame belonged to the class "percussion-only" or to the class "not only percussion". Another exception is the study by [56], discussed in Section 7.1.3.

Some publications that touch upon drum breaks and breakbeats in a wider sense are briefly listed in the following. A multifaceted study of breakbeats covering diverse production aspects and MIR tasks can be found in [72]. In [73], the authors developed a method to retrieve HJDB (hardcore, jungle, and drum'n'bass) tracks that incorporate a certain break supplied as a query. Concerning musicology and sampling practices, [112] proposes a typology of sampled material in electronic dance music, where breakbeats are discussed in various roles and capacities. Educational books such as [2] indicate that there is also a growing interest among musicians to practice and play authentic drum breaks.

### 7.1.3 Study by Frane

Frane [56] investigated microrhythmic properties of classic drum breaks. In this context, "classic" means that these particular breaks have been used in large number of derivative recordings and are even known by name among aficionados of these styles. At the core of Frane's study is a dataset of 30 drum breaks with meticulously annotated drum onsets. Using this annotated dataset, Frane computed local swing ratios (LSRs) according to a particular methodology which we have formalized as an algorithm in Section 7.2.2. He also described properties such as *swing density* and *dual swing ratio*. Swing density refers to the proportion of even-numbered division onsets (at the swing level) that are marked by note onsets. Dual swing ratio refers to the two competing swing ratios that arise when two different drum kit components are struck at slightly different times, but at the same nominal beat-division, thereby creating ambiguity regarding the precise onset of that division. Frane's study differed from previous studies on swing ratio for two main reasons. First, Frane studied sixteenth-note swing and considered multiple drum kit components, whereas previous studies of drummers' microtiming, e.g., [58] and [46], mainly studied eighth-note swing and focused on the ride cymbal. Secondly, Frane's dataset was comprised of drum breaks from "hip-hop's

breakbeat canon" (i.e., drum breaks that have become especially iconic due to their widespread use in hip-hop), whereas most previous studies focused on jazz.

## 7.2 Annotated Dataset

We were able to source 29 of the 30 drum breaks used by [56]. The dataset consists of audio files, as well as two types of annotations. The audio files are drum breaks from hip-hop's breakbeat canon. They are one to four musical measures (bars) in length, in 4/4 time, with tempi ranging from 78 to 138 beats per minute (BPM). Each excerpt starts at the onset of a quarter-note beat (usually beat 1 of the 1$^{st}$ measure), and ends at the same beat onset an exact integer number of measures later. Thus, from a music production perspective, each excerpt could be looped without further modification. The first type of annotation gives drum onset timepoints, as is common in MIR tasks. The second type gives LSRs, which were computed based on the onset annotations, using a procedure that we formalize in Section 7.2.2.

### 7.2.1 Annotation Procedure

Frane manually annotated 30 drum breaks. For each drum break, wherever a note (i.e., an annotated onset) occurred at what was judged to be an even-numbered division's onset, Frane computed an LSR at the sixteenth-note level. He then computed the global swing ratio of each drum break as the arithmetic mean of its LSRs.

In the jazz literature, swing ratio has conventionally been computed by comparing the given $\delta_e$ to the preceding $\delta_o$. However, one could also compare the given $\delta_e$ to the ensuing $\delta_o$. In fact, [15] argued that using the ensuing $\delta_o$ may often be more functionally relevant, in part because the swung note and the note at the ensuing division may be considered as an iambic pair. That is, a swung note often acts as an *anacrusis* that creates motional energy leading into the ensuing note: *da-DUM*. On the other hand, one could argue that such motional energy is effectively resolved upon the onset (rather than upon the conclusion) of the second note in the iamb, making the duration of that second note less perceptually critical. Frane's compromise was to average the preceding $\delta_o$ with the ensuing $\delta_o$ when computing the LSR at a given even-numbered division (except when one of the two $\delta_o$ values could not be measured due to rhythmic sparseness, in which case he used whichever of the two was available).

That compromise was based on a threefold rationale. First, the divisions on either side of the swung division are both likely to affect local swing perception to some extent, and there is no definitive reason to assume that one is far more perceptually relevant than the other. Second, incorporating both neighboring $\delta_o$ values into the computation uses more of the available information and thus may make measurements less noisy. Lastly, neither the preceding nor the ensuing $\delta_o$ alone could be consistently used (because some divisions in some breaks were not explicitly marked by event onsets, meaning some $\delta_o$ values could not

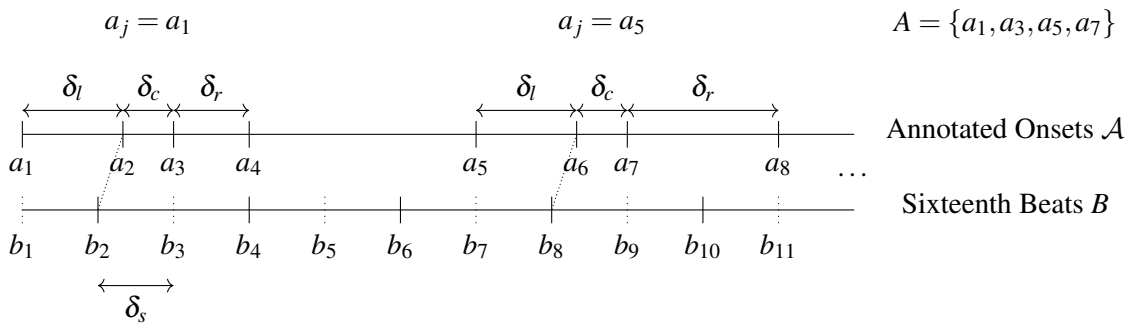



Figure 7.3: Local swing ratio annotation method.

be defined), so a policy of using all available neighboring divisions was a more consistently enforceable approach. We formalize Frane's method in Section 7.2.2 (see Algorithm 1).

### 7.2.2 Annotation Method Formalization

Disregarding some particularities, we can describe the LSR calculation in simple terms as follows. Given a list of annotated onsets and an idealized (isochronous) sixteenth-note grid, we step through the annotated onsets, checking each one to determine whether the annotated onset corresponds to an even-numbered division onset. Each time such an annotated onset is identified, we compute the LSR at that division. That procedure is formalized as follows.

To describe the sixteenth-note grid we use the notation from [97] for "beat" positions; note that in this section, the term *beat* refers to sixteenth-note division onsets, rather than to quarter-note beats. Let $B := (b_1, b_2, \ldots, b_L)$ be a sequence of length $L$ consisting of strictly monotonically increasing beat positions $b_\ell \in \mathbb{R}$ for $\ell \in [1 : L]$. Our formalization has two prerequisites regarding the beat grid. The first requirement is that $b_1$ marks the beginning of the drum break. The second requirement is that there are an even number of beat positions, distributed across the entire drum break (e.g., a one-measure drum break would have 16 beat positions, i.e., $L = 16$). We now construct a sub-list $B^{\text{odd}} := (b_1, b_3, b_5, \cdots, b_{L-1})$, containing the odd-numbered beat positions, which are very unlikely to coincide with a swung drum-onset. Furthermore, let $\mathcal{A} := (a_1, a_2, \cdots, a_J)$ be the list of annotated drum-onsets, with length $J$. We also define a list $A \subseteq \mathcal{A}$ such that $A := (a \in \mathcal{A} \mid \exists b \in B^{\text{odd}} \ : \ |b - a| \leq \varepsilon)$, for a suitable $\varepsilon$. Using automated methods that follow deterministic rules, it is easier to track the mapping between odd-numbered beats and their corresponding annotated onsets. In other words, given a tolerance value $\varepsilon$, it is more plausible to expect that $A$ will—under real-world conditions—contain all the intended onsets from $\mathcal{A}$. On the other hand, given the variability of swung onsets, it would be harder to formulate a deterministic rule assigning swung onsets to their even-numbered grid counterparts. Finally, we need a way to determine whether an interval between two adjacent annotated onsets also maps to an interval between two adjacent beats, or if one or

more beats are "skipped". To that end, we define $\delta_s$ as the theoretical IOI of one straight sixteenth note at the tempo in consideration. To decide whether the length of a given interval $\delta$ is *valid*, we require the function $D : \mathbb{R}^2 \to \mathbb{B}$, where $\mathbb{B} := \{0, 1\}$ is the set of truth values (equivalent to false and true, respectively). For tolerance parameters $\tau_1$ and $\tau_2$, we define the function as

$$D(\delta, \delta_s) := \begin{cases} 1, & \text{if } \tau_1 \cdot \delta_s \leq \ \delta \ \leq \tau_2 \cdot \delta_s\,; \\ 0, & \text{otherwise.} \end{cases} \tag{7.1}$$

In other words, the function $D$ takes a positive time interval $\delta$ and a theoretical straight sixteenth-note interval $\delta_s$, and outputs a truth value indicating whether $\delta$ is sufficiently close to $\delta_s$. We found empirically that $\tau_1 = 0.325$ and $\tau_2 = 1.3$ deliver results very close to those reported by Frane.

**Algorithm 1** Compute local swing ratios

**procedure** LSR($A, \mathcal{A}, \delta_s$)
  $\mathsf{S} \leftarrow \varnothing$ ▷ Initialize empty list of LSRs
  **for** $a_j \in A$ **do** ▷ Loop over non-swing annot.
    $\mathfrak{L} \leftarrow \varnothing$ ▷ Initialize empty list for lookaround
    $\delta_l \leftarrow a_{j+1} - a_j$ ▷ Lookbehind
    $\delta_c \leftarrow a_{j+2} - a_{j+1}$ ▷ Current length
    $\delta_r \leftarrow a_{j+3} - a_{j+2}$ ▷ Lookahead
    **if** $D(\delta_l, \delta_s) \wedge D(\delta_c, \delta_s)$ **then** ▷ If $\delta_l$ and $\delta_c$ are of valid length (lookbehind)
      $\mathfrak{L} \leftarrow \{\mathfrak{L},\ \delta_l/\delta_c\}$ ▷ Add their ratio to the list
    **end if**
    **if** $D(\delta_c, \delta_s) \wedge D(\delta_r, \delta_s)$ **then** ▷ If $\delta_c$ and $\delta_r$ are of valid length (lookahead)
      $\mathfrak{L} \leftarrow \{\mathfrak{L},\ \delta_r/\delta_c\}$ ▷ Add their ratio to the list
    **end if**
    $\mathsf{S} \leftarrow \{\mathsf{S},\ \text{mean}(\mathfrak{L})\}$ ▷ Mean of left and right
  **end for**
  **return** S
**end procedure**

We will now go through an example constructed to illustrate the two main cases encountered when calculating LSR. In the first case, all conditions are met for ratios to be measurable on both sides of a swung interval. In the second case, only one adjacent interval has a valid length, while the other spans two sixteenth notes (and is therefore invalid.) This example follows Figure 7.3 and Algorithm 1. In Figure 7.3, time runs from left to right. The bottom axis contains the beat positions from $B$; odd-numbered onsets are shown as dotted vertical bars, and even-numbered onsets are shown as solid bars. The top axis contains the annotated onsets from $\mathcal{A}$. Because the beat grid is defined to be regular, the interval $\delta_s$, which is shown as the distance between $b_2$ and $b_3$, could alternatively be placed between any other two adjacent beat positions on the axis. Our formalization assumes, without loss of generality, that the first

annotated onset $a_1$ coincides with $b_1$ and both are at time position 0 s. The algorithm begins with the first annotated onset, $a_1$ (seen at the left of Figure 7.3 and line 3 of Algorithm 1). We now set the three IOIs that potentially contribute to the LSR: the "left" interval $\delta_l = a_2 - a_1$ (line 5 of Algorithm 1), the "center" interval $\delta_c = a_3 - a_2$ (the potentially swung interval, as it coincides with the even-numbered beat $b_2$; line 6 of Algorithm 1), and the "right" interval $\delta_r = a_4 - a_3$ (line 7 of Algorithm 1). The next step (line 8 of Algorithm 1) is to determine whether both $\delta_l$ and $\delta_c$ have a valid length with respect to $\delta_s$, $\tau_1$, and $\tau_2$; if so, we can compute the *lookbehind* swing ratio ($\delta_c/\delta_l$); In this case, as is evident in Figure 7.3, both intervals comply, so we proceed to append the swing ratio $\delta_l/\delta_c$ to the temporary *lookaround* list $\mathfrak{L}$ (line 9 of Algorithm 1). We now determine whether the length conditions hold for the *lookahead* ratio (line 11 of Algorithm 1) and, seeing that they do, append $\delta_r/\delta_c$ to $\mathfrak{L}$. Because valid intervals were found on both sides, the LSR associated with $a_2$ will be the mean of $\delta_l/\delta_c$ and $\delta_r/\delta_c$, which is stored in $\mathsf{S}$, the list that holds all the LSRs for the drum break (line 14 of Algorithm 1). The algorithm then progresses to $a_3$ (i.e., $a_j = a_3$ in line 3 of Algorithm 1). For $\delta_l = a_4 - a_3$, the validation function $D(\delta_l, \delta_s)$ holds true, but not for $\delta_c = a_5 - a_4$ (line 8 of Algorithm 1). The next $a_j$ where conditions are met is $a_5$. Here, $\delta_l$ and $\delta_c$ are both valid (and their ratio is appended to $\mathfrak{L}$), but $D(\delta_r, \delta_s)$ does not hold, because $\delta_r \gg \tau_2 \cdot \delta_s$. Thus, only $\delta_l/\delta_c$ is appended to $\mathfrak{L}$, and this quantity is then stored in $\mathsf{S}$ without taking the arithmetic mean.

Having formalized Frane's annotation method, we now discuss the reference annotations available for the dataset. Table 7.1, with data taken from [56], gives an overview of this dataset. The first column (ID) contains identifiers of the form BPM-Shortname, which we will use throughout the chapter. The second column contains the original full title of each drum break. The third column specifies the number of bars or musical measures that each break spans—either 1, 2, or 4. The fourth column contains the tempo in BPM. The fifth column (SR) contains the arithmetic mean of the annotated LSRs for each break, with values ranging from 1.0 to 2.1. Since our automated method estimates global swing ratio, we will be using this annotated value as a reference for evaluation. The sixth column (Max LSR) shows the highest value of LSR registered for each item. Finally, in the rightmost column (S. Dens.), we present the swing density, defined in Section 7.1.3. As we will discuss in Section 7.4, swing density also plays an important role in evaluation since a low swing density can be a potential source for discrepancies between annotated and estimated SRs.

It is interesting to note that 12 items from the dataset (almost half) have a swing density of 25% or lower, and 20 items (two thirds) have a density of 50% or lower. The dataset contains items that have relatively low swing ratios (close to 1, as seen in the SR column), and the swing is often applied sparsely throughout the drum break. This is especially relevant in Section 7.4, where we look closely at the correspondence between annotated and estimated values.

| ID | Song Title | # of Bars | BPM | SR | Max LSR | S. Dens. (%) |
|---|---|---|---|---|---|---|
| 138-amen | Amen, Brother | 4 | 138 | 1.2 | 1.6 | 43.8 |
| 100-hihache | Hihache | 1 | 100 | 1.1 | 1.2 | 25.0 |
| 85-breakthru | Breakthrough | 4 | 85 | 1.1 | 1.2 | 96.9 |
| 109-jam | The Jam | 1 | 109 | 1.2 | 1.3 | 50.0 |
| 92-kissing | Kissing My Love | 4 | 92 | 1.3 | 1.5 | 96.9 |
| 84-imgonna | I'm Gonna Love You Just A Little More Baby | 2 | 84 | 1.1 | 1.2 | 100.0 |
| 94-herecomes | Here Comes The Meter Man | 2 | 94 | 1.2 | 1.4 | 62.5 |
| 98-ashley | Ashley's Roachclip | 1 | 98 | 1.3 | 1.4 | 100.0 |
| 104-koolis | Kool Is Back | 4 | 104 | 1.2 | 1.4 | 46.9 |
| 104-youlllike | You'll Like It Too | 4 | 104 | 1.6 | 2.0 | 96.9 |
| 117-apache | Apache | 2 | 117 | 1.1 | 1.2 | 25.0 |
| 79-themeplanets | Theme From The Planets | 2 | 79 | 1.0 | 1.1 | 93.8 |
| 106-dofunkypenguin | Do The Funky Penguin Part II | 1 | 106 | 1.3 | 1.3 | 25.0 |
| 99-nt | N.T. | 1 | 99 | 1.3 | 1.5 | 87.5 |
| 94-godmademe | God Made Me Funky | 4 | 94 | 1.2 | 1.4 | 31.2 |
| 101-funkydrummer | Funky Drummer | 2 | 101 | 1.1 | 1.2 | 100.0 |
| 92-dontchange | Don't Change Your Love | 1 | 92 | 1.6 | 1.7 | 37.5 |
| 96-syntheticsubs | Synthetic Substitution | 2 | 96 | 1.3 | 1.7 | 50.0 |
| 95-itsanew | It's A New Day | 1 | 95 | 1.7 | 1.7 | 25.0 |
| 94-youregett | You're Getting A Little Too Smart | 2 | 94 | 1.1 | 1.3 | 18.8 |
| 103-youcan | You Can Make It If You Try | 1 | 103 | 1.0 | 1.2 | 50.0 |
| 81-imglad | I'm Glad You're Mine | 2 | 81 | 1.0 | 1.2 | 50.0 |
| 121-youandlo | You & Love Are The Same | 2 | 121 | 1.2 | 1.3 | 12.5 |
| 96-love | Love's Theme | 1 | 96 | 1.0 | 1.0 | 100.0 |
| 82-funkyworm | Funky Worm | 1 | 82 | 1.2 | 1.2 | 12.5 |
| 93-longred | Long Red | 1 | 93 | 1.5 | 1.5 | 25.0 |
| 101-singa | Sing A Simple Song | 1 | 101 | 2.1 | 2.1 | 12.5 |
| 91-getoutmylife | Get Out My Life Woman | 2 | 91 | 1.6 | 1.7 | 25.0 |
| 95-impeach | Impeach The President | 1 | 95 | 1.4 | 1.5 | 12.5 |
| — | Fool Yourself | 1 | 78 | 1.5 | 1.7 | 25.0 |

Table 7.1: Reference table, based on [56].

## 7.3 Automated SR Estimation Approach

In Section 7.2 we discussed Frane's annotation method and dataset for LSR. While the method formalized in Section 7.2.2 is suitable to be applied to manually created onset annotations, [46] showed that SR estimation based on automatic onset detection can be an error-prone process. For this reason, they proposed a more robust and efficient method based on periodicity analysis at a global scale, which we closely follow in our experiments. Since the original works were conceived mainly for eighth-note swing in ride cymbals, we also describe modifications that accommodate sixteenth-note swing in multiple drum kit components. The modifications include an eighth-note pulse grid, as well as the application of drum sound separation as described in [44]. In essence, we are investigating to which degree an automated, global SR estimation methods can reproduce the results obtained from manual LSR annotation.

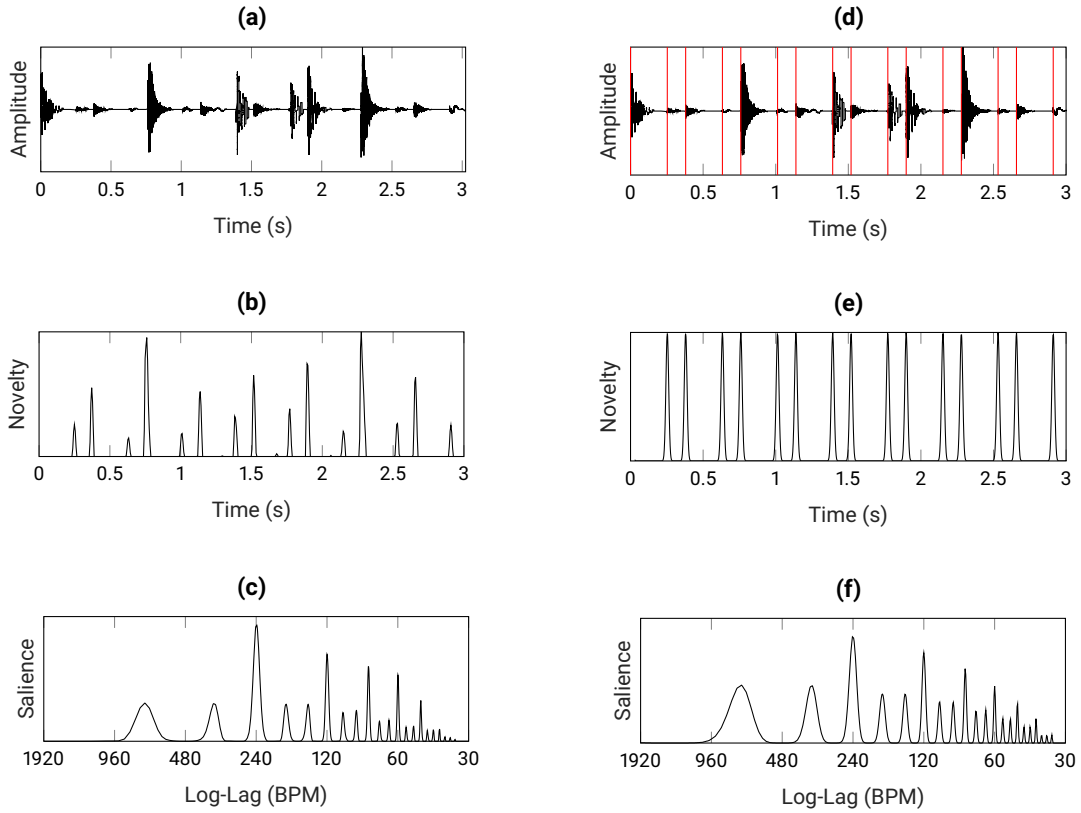



Figure 7.4: **(a)** Waveform for excerpt of "Theme From the Planets" by Dexter Wansel. **(b)** Novelty curve extracted from the waveform. **(c)** LLACF $x$ resulting from the novelty curve. **(d)**: Waveform (black) and ground truth onset annotation (red vertical lines). **(e)**: Oracle novelty curve from annotation points. **(f)**: LLACF curve computed from oracle novelty.

In practically all MIR tasks, an important first step is to compute audio features that capture relevant properties of a signal for further processing [97, p. 117]. One approach to feature-based information retrieval is to compare the features obtained for real-world input data with features computed for idealized synthetic data with known characteristics. Our swing ratio estimation method uses such an approach. On the one hand, we compute audio features from drum break recordings, and on the other hand, we have a series of prototypical *reference features* for which we know the swing ratio. The task then becomes a problem of pattern matching, i.e., comparing the features estimated from the input data with the reference features.

The family of features typically used for rhythm-related tasks is based on novelty (e.g., sudden events such as drum onsets) and periodicity (e.g., at which rate these events repeat) in the underlying signal. Dittmar et al. developed methods for swing ratio estimation via periodicity analysis in [46, 45]. The procedure which we will describe in this section is based mainly on those works.

Figure 7.4 is our running example for this section. The main idea is to first convert waveforms (Figure 7.4a) into a novelty curve representation emphasizing onset candidates (Figure 7.4b), and then into a log-lag

representation that implicitly encodes swing ratio (Figure 7.4c). In parallel, we have a synthetically generated reference set containing pre-computed log-lag representations (or patterns) for various swing ratios within a plausible range. The task is then to compare the estimated pattern (obtained by processing the input waveform) with all the patterns in the reference matrix, selecting the reference which is maximally similar (see Figure 7.5) and thus retrieve the underlying swing ratio. In the following, we examine each of these stages in greater detail.

### 7.3.1 Novelty Curve

Our first goal is to obtain a feature representation that contains the temporal pulse characteristics of the input signal $S$. Because swing ratio is defined by the relative positions of onsets, we compute a novelty curve to expose salient peaks that we can interpret as onset candidates. We start with mono audio files sampled at 44 100 Hz. As an example, Figure 7.4a shows an excerpt of the waveform for our swing-modified version of "Theme from the Planets" by Dexter Wansel. The first step is to compute a short-time Fourier transform (STFT) with a block size of 2048 samples and a hop size of 256 samples, using a Hann window. Following [46, 45], we apply log-compression to the resulting magnitude spectrogram and accumulate changes between consecutive frames to obtain a suitably normalized novelty curve (Figure 7.4b).

#### 7.3.1.1 Oracle Novelty

Using the audio signal $S$ as input can produce imperfect novelty curves. On the one hand, drum hits that are played very softly can lead to missing peaks, and on the other hand, spurious peaks might occur due to noisy input signals. Furthermore, in order to find out whether the SR annotations from [56] are fundamentally compatible with our method, and if so, to what extent they agree, we establish a *glass ceiling* (i.e., estimate of the upper bound) for our algorithm by using the onset annotations as oracle data. In Figure 7.4d–f we illustrate how to produce an oracle novelty curve from annotated onsets. At the top, Figure 7.4d shows the waveform for our running example in black, and the ground truth onset annotations as red vertical bars; Figure 7.4e shows the oracle novelty curve, obtained by placing a short Gauss-shaped pulse centered at each of the annotated onset time-points. Figure 7.4f contains the oracle LLACF, computed from the oracle novelty by using our method as described above. We will revisit the oracle novelty in Section 7.4.1, where it will help put our estimated SR results into context.

### 7.3.2 Log-Lag Autocorrelation Function (LLACF)

Having computed the novelty curve, we could use peak-picking and beat-tracking to select onset candidates within a sixteenth-note grid, corresponding to the algorithmic formalization in Section 7.2.2 and Frane's LSR computation. However, Dittmar et al. [46] observed that such onset detection is an error-prone

process, and that the log-lag autocorrelation function (LLACF) can help circumvent it. The LLACF is a tempo-normalized, mid-level feature representation that encodes the periodicity and salience of the novelty curve at different metrical levels. Computing LLACF patterns (or vectors) is a two-step process. First, the conventional autocorrelation function is applied to the novelty curve. The ACF values are then max-normalized to lie between 0 and 1. As mentioned before, the method in [46] works on the eighth-note level; our adaptation to the sixteenth-note level consists of defining the tempo axis for the autocorrelation one *tempo octave* higher—that is, at "double the tempo". In typical autocorrelation functions, lags that correspond to salient peaks hint at IOIs present in the signal (at different metrical levels). Thus, we denote lag values with $\delta$, the same symbol we used for IOIs. The relationship between a tempo $t$ (in BPM) and a lag $\delta$ (in seconds) is given by $t = 60 \,/\, \delta$. Continuing from the autocorrelation function, we use linear interpolation to map the autocorrelation function values onto a log-lag axis that has equal spacing between tempo octaves, with a reference tempo of 240 BPM at a defined position. The LLACF pattern for our example is shown in Figure 7.4c. For the following extraction steps, let $x \in \mathbb{R}^{K\times 1}$ be an LLACF vector, with $K$ being the number of elements of the discrete log-lag axis. For all experiments in this chapter we use $K = 721$, which results from sampling 120 bins per tempo octave in the range from 30 to 1920 BPM. To simplify notation, we define $\Lambda$ as an operator that encapsulates the steps necessary to arrive at a log-lag representation: transform the signal $S$ to a time-frequency representation, compute the novelty, and apply LLACF. In the following, $\Lambda(S)$ denotes the LLACF pattern obtained from the audio signal, and $\Lambda(O)$ is the pattern obtained from the oracle novelty.

#### 7.3.2.1 LLACF Prototype Patterns

As mentioned before, we can estimate the swing ratio of an LLACF by comparing it against a set of reference LLACF patterns with known swing ratios and selecting the one with highest similarity. Figure 7.5a shows the matrix of LLACF reference patterns, which we call $Y$. From left to right, the columns in $Y$ contain LLACF patterns with increasing swing ratio, each one obtained by applying the method from Section 7.3.2 to a synthetically created novelty curve that has the particular swing ratio present in its IOIs. Thus, $Y := (y_1, y_2, \ldots, y_M) \in \mathbb{R}^{K\times M}$. In practice, we have $M = 402$, resulting from a uniform sampling of 400 swing ratios $s_m \in [1, 2.5]$, along with two additional columns at the end; these columns correspond to straight quarter- and eighth-notes, and are useful when dealing with novelty curves that only exhibit salient periodicities at those metrical levels. In Figure 7.5a, we have enclosed the column number 257 in a red dotted rectangle, corresponding to the swing ratio 1.96, which closely matches the swing ratio 2 enforced in our running example.

#### 7.3.2.2 Pattern Matching and Similarity

Having extracted the LLACF vector $x$ and the reference matrix $Y$, we need to use pattern matching to select a reference vector $y$ that is maximally similar to $x$. We cannot yet directly compare the estimated

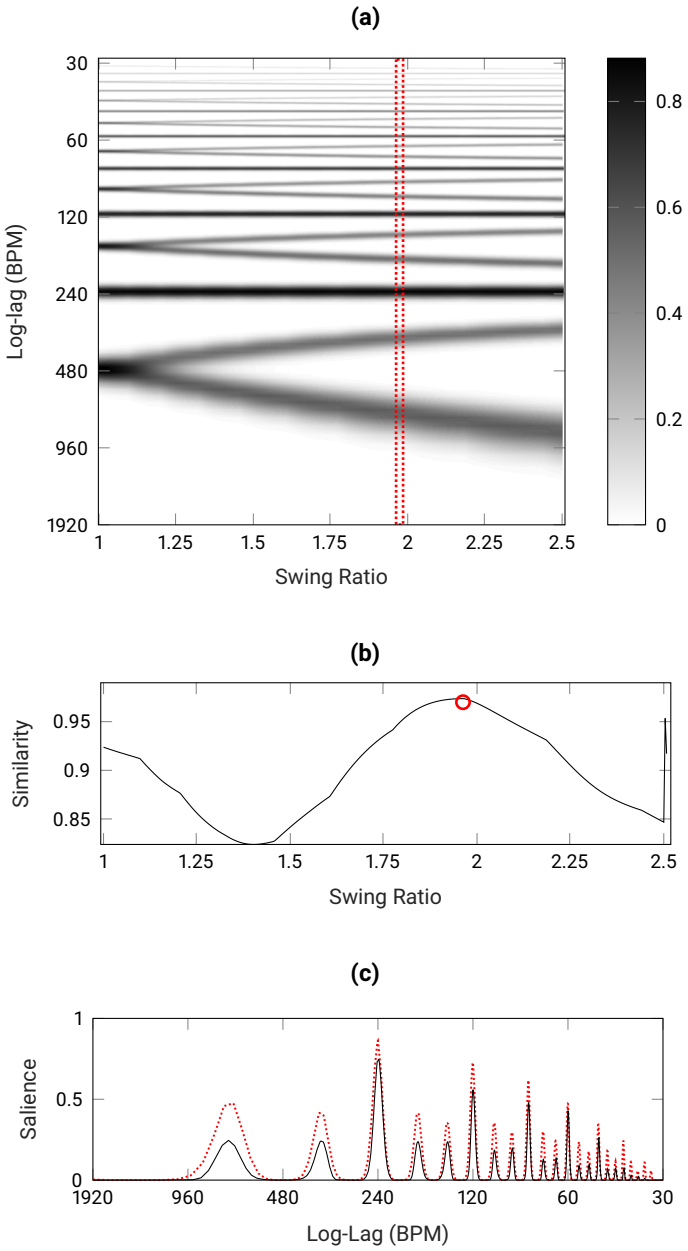



Figure 7.5: **(a)** Reference LLACF patterns for swing ratios from 1 to 2.5. Each column is a synthetically generated LLACF vector corresponding to a certain swing ratio. The column for SR 1.96 is enclosed in a dotted red rectangle, and is shown in (c). **(b)** Similarity that results from comparing the computed LLACF vector for our running example (Figure 7.2c) with all the reference patterns from the matrix. A maximal similarity of 0.97 is found at SR 1.96, enclosed in a red circle at the top-left corner. **(c)** Estimated LLACF (black, solid) and maximally similar reference LLACF (red, dotted).

and reference LLACF vectors, as they may correspond to slightly different underlying tempi. Tempo differences lead to linear shifts in the resulting LLACF pattern along the log-lag axis. To be able to compare $x$ with reference patterns $y_m$, a shift-invariant similarity measure $\psi : \mathbb{R}^K \times \mathbb{R}^K \to \mathbb{R}$ is used. Thus, following [46], we transform the estimated $x$ and each reference column $y_m$ into equivalent representations by computing their discrete Fourier transform (DFT) and taking the absolute value. By preserving the magnitude and discarding the phase, we capture their overall characteristics while ignoring shifts along the log-lag axis. We normalize these DFT-magnitude vectors by subtracting their arithmetic mean and

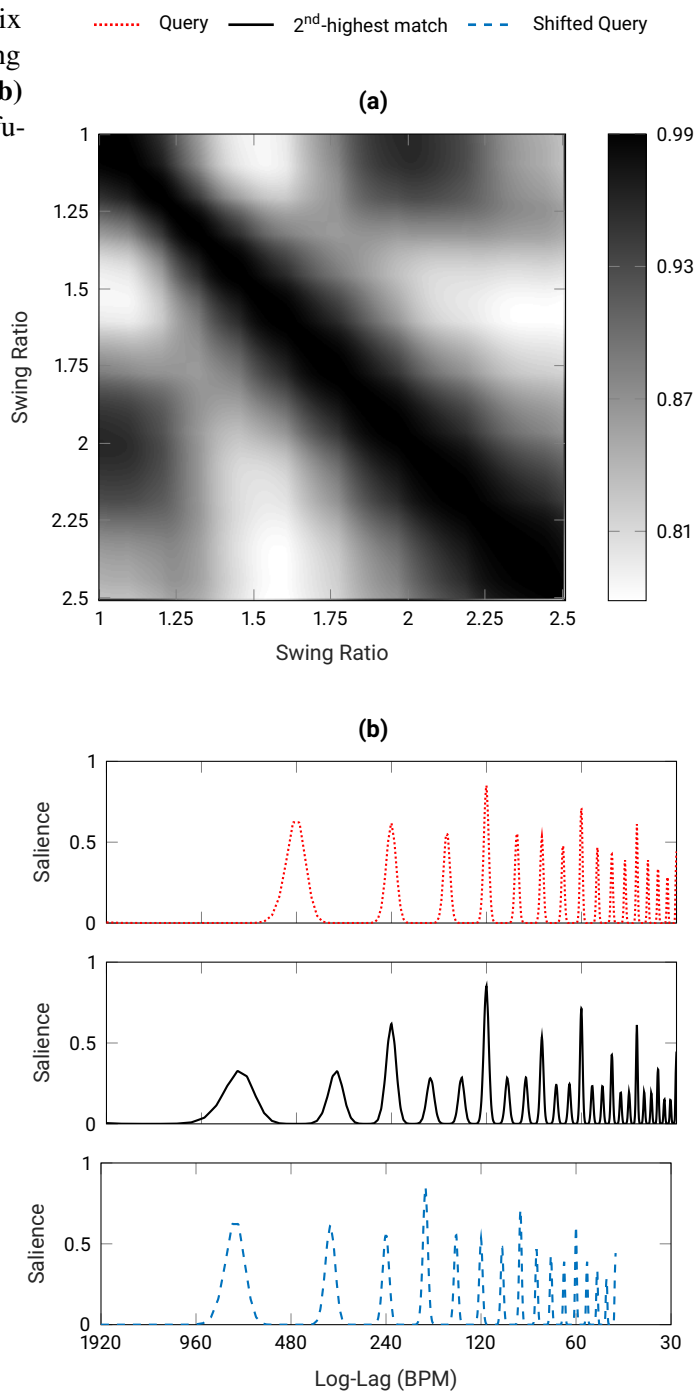



Figure 7.6: **(a)** Self-similarity matrix for the LLACF reference patterns, using Eq. 7.2. Darkness indicates similarity. **(b)** Curves illustrating LLACF pattern confusion.

dividing the result by the standard deviation, resulting in an $\hat{x}$ vector and a series of $\hat{y}$ vectors. We can now define the similarity between $x$ and a given $y$ as:

$$\psi(y,x) := \langle \hat{y} | \hat{x} \rangle, \tag{7.2}$$

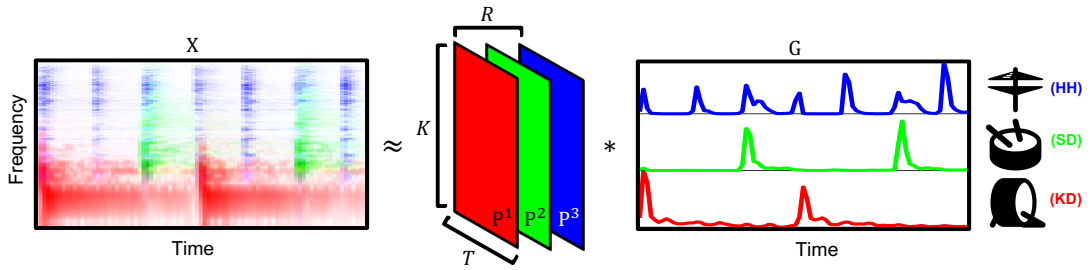



Figure 7.7: Schematic illustration of NMFD drum component separation.

with $\langle \cdot | \cdot \rangle$ being the dot product between two vectors.

Figure 7.5b shows the result of the dot product between $x$ and each column in the matrix $Y$. We can see that $x$ is maximally similar to the 257$^{\text{th}}$ entry in the matrix (similarity 0.97, swing ratio 1.96, indicated by red circle). However, the vector is also considerably similar to patterns in the vicinity of swing ratio 1, as well as the two additional vectors at the end (straight quarter and eighth notes, with swing ratio 1)—this *ambiguity* prompts further evaluation.

We need to critically assess the pattern matching—it is important to determine the potential for confusion among reference patterns. To this end, we have computed a self-similarity matrix $S$ for the matrix $Y$, where $S(i,j) := \psi(\hat{y}_i, \hat{y}_j)$ for $i, j \in [1:M]$. Figure 7.6a shows $S$, with darkness representing higher values (i.e., greater similarity). Ideally, the plot would have a thin, dark main diagonal (because each $\hat{y}_m$ is maximally similar to itself) and should be considerably lighter everywhere else. But in practice, as shown in the figure, the main diagonal is somewhat wide, indicating a swing ratio uncertainty of roughly 0.25 for any given value. Furthermore, there are separate regions with high confusion: Swing ratios between 1 and 1.25 are very likely to be confused with ratios between 1.8 and 2.2 (and vice-versa), as is evident from the dark spots near the bottom-left and top-right of the plot. In Figure 7.6b, we illustrate the reason for this ambiguity, which is a byproduct of the similarity computation described above. As one would intuitively expect, the reference LLACF with swing ratio 1 (in dotted red) is very dissimilar to the reference LLACF with swing ratio 2 (solid black). However, if we deliberately shift it to the left along the log-lag axis (equivalent to increasing the tempo), we reach a point where it matches quite well to the LLACF with swing ratio 2 (blue curve). This particular log-lag shift corresponds to multiplying the original tempo by a factor of 1.5. In other words, our similarity computation can deliver ambiguous results under certain circumstances. For example, a ternary swing pattern with SR 2 at 120 BPM matches a straight pattern at 180 BPM. The tempo-invariant method was developed for swing ratio analysis over the course of longer jazz recordings [45], where its ability to deal with greater tempo variations is an asset. For short drum breaks with steady tempo, the tempo-invariance property can have a detrimental effect on swing ratio estimation.

### 7.3.3 Separating Drum Components

Our first contribution to the swing ratio estimation method is applying drum component source separation as a preprocessing step, following [44]. Here we offer a brief overview of how that method works; for a more detailed description we refer the reader to the original publication. In general terms, drum source separation is a technique that takes a signal containing a mixture of drum kit components, such as kick drum, snare, and hi-hat, and attempts to deliver individual signals for each component as if it were recorded in isolation. Dittmar and Müller [44] approached drum source separation using an informed variant of the non-negative matrix factor deconvolution (NMFD) method proposed by [124].

As its main premise, NMFD considers the original signal to be a linear mixture of component signals that correspond to the individual sound sources. In this case, the original "mixed" signal is a drum break ($S$), and the component signals are kick drum (KD), snare drum (SD), and hi-hat (HH). Thus, we make the simplifying assumption that $S = S_{\mathrm{KD}} + S_{\mathrm{SD}} + S_{\mathrm{HH}}$, where $S$ is the original signal as introduced in Section 7.3.1. The NMFD variant that we are using (from [44]) is informed by timbral templates from a database of prototypical KD, SD, and HH sounds. Seeded by this initialization, the NMFD makes an iterative approximation to the input signal, learning its particularities. Thus, for example, although we are initializing one component with the spectral characteristics for HH, this component might learn the timbral information of similar drum kit components such as ride cymbals. As illustrated in Figure 7.7, the NMFD algorithm considers the original signal as a *target* that can be approximated as the convolutional sum of a tensor with an activation matrix. The target V is represented by the leftmost part of Figure 7.7, which shows an excerpt of the magnitude spectrogram of a drum break. The spectral information is color-coded according to our standard throughout the chapter: KD is red, identified by its characteristic energy in the lower frequencies; SD is green, with predominance in the mid- and higher frequency regions; and HH is blue, with wide-band spectral characteristics. The tensor P, represented by the middle part of Figure 7.7, has three templates, each corresponding to a prototypical magnitude spectrogram for one of the three components: KD ($\mathrm{P}^1$), SD ($\mathrm{P}^2$), and HH ($\mathrm{P}^3$). The activation matrix H, represented by the rightmost part of Figure 7.7, has three rows, each of which essentially encodes when and how intensely a given drum kit component was struck.

NMFD recovers the templates and activations from the mixture, in an iterative process, where the number of iterations is a fixed parameter. We initialize each template in P with a prototypical instance of each type of drum sound, learned from a large database of one-shot sounds (recorded in isolation), and we initialize H with constant values. In each iterative pass, the algorithm performs the convolution $\mathrm{P} * \mathrm{H} = \hat{\mathrm{V}}$ and attempts to minimize the Kullback–Leibler divergence between the current estimate $\hat{\mathrm{V}}$ and the target V by updating P and H. Ideally, after the specified number of updates, the algorithm has learned the timbral details of each drum kit component, as well as the locations and intensities of their onsets. To recover the individual magnitude spectrograms for the separated drum components, we perform the convolutional approximation but only between the selected spectral template in P and its corresponding row in H. Since NMFD yields only a rough approximation of the component spectrograms we apply generalized Wiener

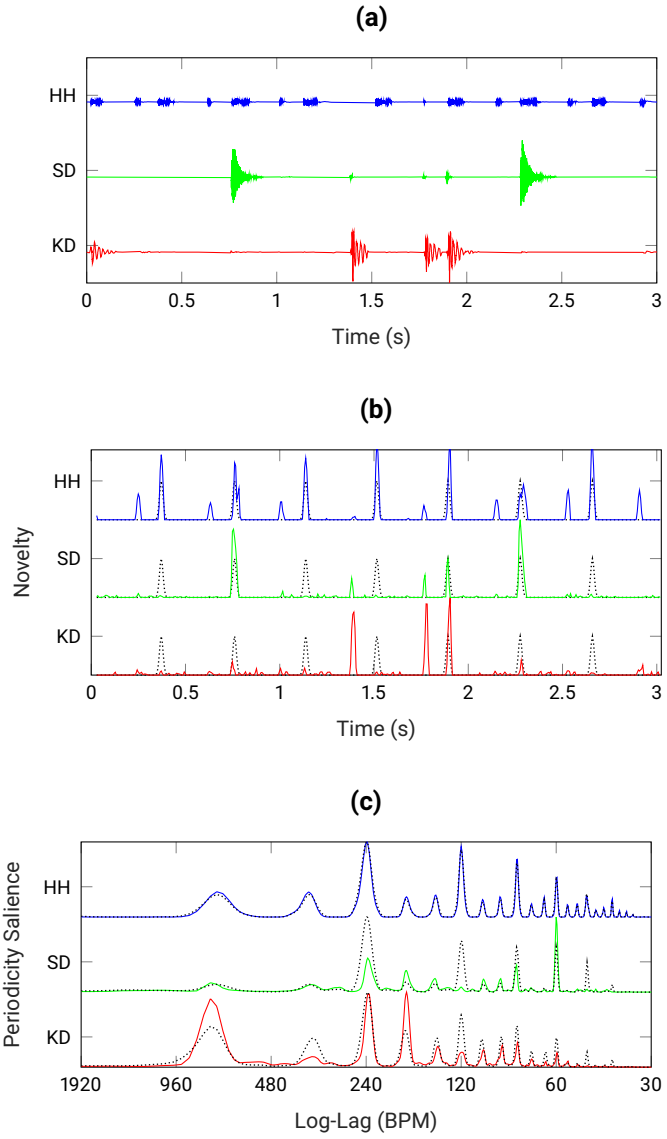



Figure 7.8: **(a)** Waveforms for separated drum kit components: kick drum (KD, bottom, red), snare drum (SD, middle, green), and hi-hat (HH, top, blue). **(b)** Novelty curves for drum kit components (red, green, and blue), overlaid with an eighth-note pulse grid (dotted black). **(c)** LLACF patterns for separated drum kit components ($\Lambda(S_{\mathrm{KD}})$ in red, $\Lambda(S_{\mathrm{SD}})$ in green, $\Lambda(S_{\mathrm{HH}})$ in red). The LLACF patterns with grid support $\Lambda(S_{\mathrm{KD}}, \mathcal{G})$, $\Lambda(S_{\mathrm{SD}}, \mathcal{G})$, $\Lambda(S_{\mathrm{HH}}, \mathcal{G})$ are overlaid as black dotted plots.

filtering as described by [88], to preserve small spectral nuances that otherwise might be lost. Finally, we follow the same steps for the three component spectrograms as we did for the the original mixture signal in Section 7.3.1. We refer to the LLACF patterns for these components as $\Lambda(S_{\mathrm{KD}})$, $\Lambda(S_{\mathrm{SD}})$, and $\Lambda(S_{\mathrm{HH}})$. In Section 7.3.4 we describe a further modification that helps stabilize swing ratio estimates for drum components that exhibit relatively sparse onset activity, such as KD and SD.

### 7.3.4 Eighth-Note Pulse Grid

In most drum breaks from Frane's dataset (and also, we contend, in drum breaks at large), KD and SD have sparser onsets than HH. This is exemplified in Figure 7.8a, which shows the waveforms for the individual drum components obtained by the method from Section 7.3.3: KD has four onsets and SD has

| ID | $r$ | $s(\Lambda(O))$ | $r - s(\Lambda(O))$ | $s(\Lambda(S))$ | $r - s(\Lambda(S))$ | $s(\Lambda(S,\mathcal{G}))$ | $r - s(\Lambda(S,\mathcal{G}))$ | S.Dens. |
|---|---|---|---|---|---|---|---|---|
| 138-amen | 1.1774 | 1.1729 | 0.0045 | 1.0000 | 0.1774 | 1.0376 | 0.1398 | 43.8 |
| 100-hihache | 1.1002 | 1.0940 | 0.0062 | 1.0000 | 0.1002 | 1.0301 | 0.0702 | 25.0 |
| 85-breakthru | 1.0710 | 1.0602 | 0.0109 | 1.0000 | 0.0710 | 1.0000 | 0.0710 | 96.9 |
| 109-jam | 1.1939 | 1.1805 | 0.0134 | 1.0602 | 0.1337 | 1.0639 | 0.1300 | 50.0 |
| 92-kissing | 1.2628 | 1.2481 | 0.0146 | 1.2068 | 0.0560 | 1.2105 | 0.0522 | 96.9 |
| 84-imgonna | 1.0985 | 1.0752 | 0.0233 | 1.0977 | 0.0007 | 1.0000 | 0.0985 | 100.0 |
| 94-herecomes | 1.1890 | 1.1654 | 0.0236 | 1.2105 | −0.0215 | 1.1203 | 0.0687 | 62.5 |
| 98-ashley | 1.2577 | 1.2293 | 0.0283 | 1.1353 | 0.1223 | 1.1353 | 0.1223 | 100.0 |
| 104-koolis | 1.1503 | 1.1805 | −0.0301 | 1.1203 | 0.0300 | 1.0677 | 0.0827 | 46.9 |
| 104-youlllike | 1.5748 | 1.5414 | 0.0335 | 1.6617 | −0.0868 | 1.8158 | −0.2409 | 96.9 |
| 117-apache | 1.0723 | 1.1128 | −0.0404 | 1.0000 | 0.0723 | 1.0714 | 0.0009 | 25.0 |
| 79-themeplanets | 0.9864 | 1.0451 | −0.0588 | 1.0000 | −0.0136 | 1.0000 | −0.0136 | 93.8 |
| 106-dofunkypenguin | 1.2967 | 1.2368 | 0.0598 | 1.0000 | 0.2967 | 1.0000 | 0.2967 | 25.0 |
| 99-nt | 1.2971 | 1.2368 | 0.0603 | 1.2519 | 0.0452 | 1.2218 | 0.0753 | 87.5 |
| 94-godmademe | 1.1776 | 1.1165 | 0.0611 | 1.0000 | 0.1776 | 1.0000 | 0.1776 | 31.2 |
| 101-funkydrummer | 1.0590 | 1.1241 | −0.0650 | 1.0564 | 0.0026 | 1.1241 | −0.0650 | 100.0 |
| 92-dontchange | 1.5958 | 1.6617 | −0.0659 | 1.0000 | 0.5958 | 1.0000 | 0.5958 | 37.5 |
| 96-syntheticsubs | 1.2986 | 1.2180 | 0.0806 | 1.0000 | 0.2986 | 1.0000 | 0.2986 | 50.0 |
| 95-itsanew | 1.7121 | 1.8008 | −0.0886 | 1.0263 | 0.6858 | 1.0451 | 0.6670 | 25.0 |
| 94-youregett | 1.1007 | 1.0000 | 0.1007 | 1.0000 | 0.1007 | 1.0000 | 0.1007 | 18.8 |
| 103-youcan | 1.0398 | 1.1429 | −0.1030 | 1.0526 | −0.0128 | 1.0564 | −0.0166 | 50.0 |
| 81-imglad | 1.0246 | 1.1316 | −0.1070 | 1.0564 | −0.0318 | 1.0677 | −0.0431 | 50.0 |
| 121-youandlo | 1.2420 | 1.1165 | 0.1254 | 1.0338 | 0.2081 | 1.0451 | 0.1968 | 12.5 |
| 96-love | 0.9652 | 1.1241 | −0.1589 | 1.0150 | −0.0498 | 1.0526 | −0.0874 | 100.0 |
| 82-funkyworm | 1.1607 | 1.0000 | 0.1607 | 1.0000 | 0.1607 | 1.0000 | 0.1607 | 12.5 |
| 93-longred | 1.5305 | 1.3684 | 0.1621 | 1.0000 | 0.5305 | 1.0000 | 0.5305 | 25.0 |
| 101-singa | 2.0957 | 1.8985 | 0.1972 | 1.9962 | 0.0995 | 2.0263 | 0.0694 | 12.5 |
| 91-getoutmylife | 1.5799 | 1.8195 | −0.2396 | 1.0000 | 0.5799 | 1.0000 | 0.5799 | 25.0 |
| 95-impeach | 1.4188 | 1.0000 | 0.4188 | 1.0000 | 0.4188 | 1.0226 | 0.3962 | 12.5 |

Table 7.2: Detailed overview of SR comparisons per item in Frane's set of drum breaks. Rows are sorted by magnitude difference between annotation and oracle SR (third column). The two items with highest absolute deviation between estimation and expert annotation are shaded in gray.

two onsets, whereas HH is in a continuous stream of 16 sixteenth-notes. The sparsity of onsets in KD and SD leads to LLACF patterns that match highly with many prototype templates, making the swing ratio ambiguous. To address this problem, we further adapt the method of [45] by applying an eighth-note pulse grid that helps *stabilize* the novelty curves of sparse elements. The eighth-note pulse grid is obtained by processing the original signal's novelty curve with a dynamic programming beat tracker, following [51] and [97, p. 333]. In the following, we designate the use of this grid as $\Lambda(\cdot,\mathcal{G})$, with $\mathcal{G} = \text{true}$.

Figure 7.8b shows the separated drum component novelty curves, each overlaid with the eighth-note pulse grid in dotted black. Focusing on the KD component (bottom row of Figure 7.8b), we can see that the eighth-grid introduces previously non-existent novelty peaks (such as the first peak, shortly before 0 s) and amplifies weaker peaks, such as the one shortly before 2.5 s. In Figure 7.8c, the LLACF patterns $\Lambda(S_{\text{KD}})$, $\Lambda(S_{\text{SD}})$, and $\Lambda(S_{\text{HH}})$ are shown in their original colors; $\Lambda(S_{\text{KD}},\mathcal{G})$, $\Lambda(S_{\text{SD}},\mathcal{G})$, and $\Lambda(S_{\text{HH}},\mathcal{G})$ are overlaid as dotted black plots.

## 7.4 Evaluation

The general goal of this section is to show the degree of correspondence between the LSR values that Frane computed in [56] by using Algorithm 1 and our own estimation of global SR computed with the LLACF-based method discussed in Section 7.3. However, we do not condense this assessment into a single number, but instead provide diverse, detailed views on the agreement between expert annotation and automatic estimation. We first present simple pairwise differences (or deviations) between an annotated SR and an SR estimated with a particular configuration, seen in Table 7.2. For each drum break[1] with index $n \in [1:N]$, $N = 29$, we define the deviation $\varepsilon_n := s_n - r_n$, where $r_n$ is the corresponding reference swing ratio from Frane's annotations. To put the results into context, recall from Section 7.3 that our search space for SR estimates is defined between 1 and 2.5.

### 7.4.1 Oracle Novelty Evaluation

The first two columns of Table 7.2 contain our internal drum break ID and the SR as annotated by Frane. In the following, we will also frequently refer to the last column of the table, which contains the annotated swing density (S. Dens.). The third column $s(\Lambda(O))$ contains the SR values that we estimated semi-automatically from oracle novelty curves generated using manually annotated onsets. In [56], Frane did not make any distinctions between drum components when annotating. For this reason, the oracle values are computed taking into account all annotated onsets, regardless of the drum kit component. The fourth column contains the difference $r - s(\Lambda(O))$ (i.e., the difference between the previous two columns); this column was used to sort the rows in order of increasing absolute value. Thus, the first item (138-amen) has the smallest difference between annotated SR and oracle SR ($1.1774 - 1.1729 = 0.0045$). A total of 22 dataset items (around two thirds of the dataset) have a difference $|r - s(\Lambda(O))|$ smaller than 0.11. Interestingly, the 7 bottom items (which have $|r - s(\Lambda(O))| > 0.11$) mostly have swing densities of either 12.5% or 25%. This discrepancy can be attributed to the fact that the LLACF method favors global swing applied consistently throughout the input signal. Furthermore, out of these 7 items, "Impeach the President" is an especially difficult case because of dual SR, meriting more detailed discussion in Section 7.4.5. The difference $r - s(\Lambda(O))$ gives an upper bound for results estimated from audio data—having established this upper bound we proceed to discuss results computed from audio features.

### 7.4.2 Audio-Based Novelty Curves

The columns $s(\Lambda(S))$ and $r - s(\Lambda(S))$ summarize results with mixture audio signals as input, without grid support. In this category of our experiments, a total of 16 items (around half the dataset) have a deviation $|r - s(\Lambda(S))| < 0.11$, and only 7 have a difference greater than 0.2. In this "problematic" group, the item 96-syntheticsubs has the highest swing density (50%)—all others have lower densities. This further

[1] Recall from Section 7.2 that we were only able to source 29 of the 30 breaks from Frane's original dataset.

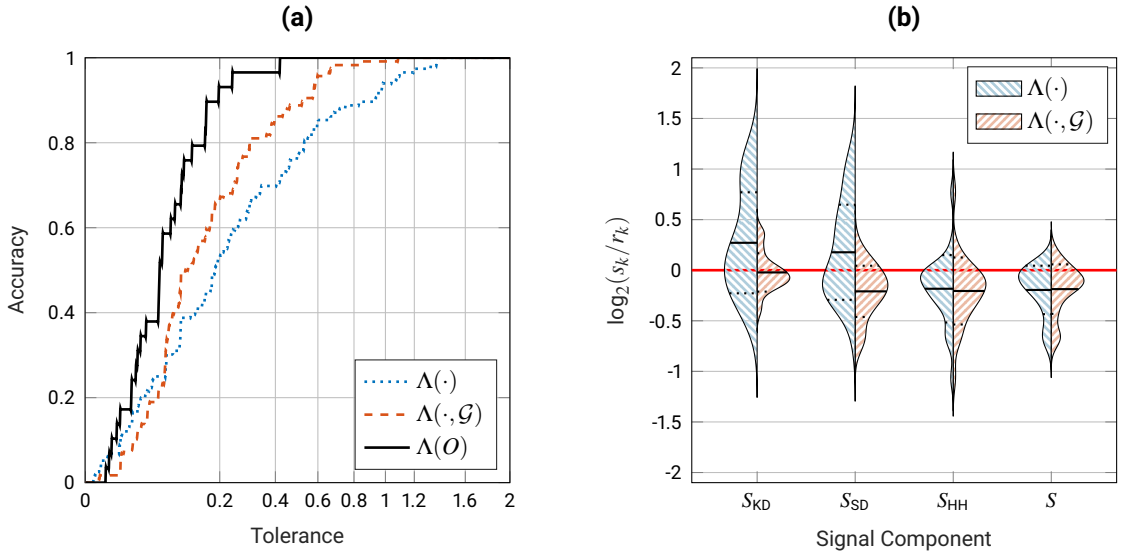



Figure 7.9: **(a)**: Mean accuracy for increasing tolerance value (horizontal axis with non-uniform spacing). The blue curve shows results for original method $\Lambda(\cdot)$, the red curve represents the eighth-grid-enhanced method $\Lambda(\cdot,\mathcal{G})$ (both curves are means across all input signals $S$, $S_{\text{KD}}$, $S_{\text{SD}}$, $S_{\text{HH}}$). Black curve is accuracy for oracle data as input, $\Lambda(O)$. **(b)**: Relative distributions of $\log_2$-ratios for both estimation methods, according to input signal.

illustrates that our results coincide to a high degree with Frane's, except when faced with low swing densities. "Don't Change Your Love" (92-dontchange) and "It's a New Day" (95-itsanew), shaded gray in Table 7.2, were both estimated to have SR close to 1. Upon closer inspection, the estimated novelty curves for these examples closely resembled the oracle novelties, indicating that the misestimation happened at the LLACF pattern matching stage. Indeed, the estimated SR of 1 is highly confusable with the annotation and oracle values around 1.7, as seen in Figure 7.6.

### 7.4.3 Eighth-Note Grid Enhanced Novelty Curves

We continue with columns $\Lambda(S,\mathcal{G})$ and $r - s(\Lambda(S,\mathcal{G}))$, which correspond to the mixture signal as input, and using the eighth-note grid support. Since we introduced grid support to investigate whether it offers an improvement to signals containing sparse drum onsets, we will present the results for this category in comparison to the previous category, $s(\Lambda(S))$. Adding grid support improved performance for 9 items, decreased performance for another 9 items, and left the remaining 11 unchanged. Interestingly, the items with decreased performance have swing densities of 46% and higher—5 of them are at 80% or above. The potential of grid support unfolds when applied to the sparser component signals obtained through drum component source separation, which we discuss in Section 7.4.4.

### 7.4.4 Further Metrics

As a first higher-level performance metric, we present the accuracy $\alpha$: the proportion of items whose absolute deviation $|\varepsilon_n|$ falls within a certain tolerance $\tau$. That proportion gives us the accuracy $\alpha_\tau$, described in Equation 7.3:

$$\alpha_\tau := |\{n \in [1:N] : -\tau \leq \varepsilon_n \leq \tau\}| \,/\, N. \tag{7.3}$$

Figure 7.9a shows the mean $\alpha_\tau$ (averaged across the results for all input signals $S$, $S_{\mathrm{KD}}$, $S_{\mathrm{SD}}$, $S_{\mathrm{HH}}$) as a function of $\tau$, based on $s_n$ values computed with eighth-note grid support (dashed red curve) and without eighth-note grid support (dotted blue curve), as well as the accuracy obtained for the oracle novelty curves (solid black curve). Recall that the oracle values (based on Frane's original annotations) were computed using only $S$ (and not the component signals $S_{\mathrm{KD}}$, $S_{\mathrm{SD}}$, $S_{\mathrm{HH}}$)—thus, since the solid black curve is based on only a quarter of the items, it is more step-like. We can see that introducing eighth-note grid support notably improves mean accuracy, notwithstanding extreme values of $\tau$. It is important to note the oracle accuracy of around 0.9 at a low tolerance value of 0.2, implying that there is a high agreement between Frane's annotations and the LLACF-based method applied to oracle novelty curves. We extend the tolerance sweep to a total value of 2 (which could seem exaggerated) in order to illustrate the point at which $\Lambda(\cdot)$ (dotted blue curve) reaches an accuracy of 1.

Figure 7.9b shows the the relative distributions of $\log_2(s_n/r_n)$ for the four signal categories used in our experiments: $S_{\mathrm{KD}}$, $S_{\mathrm{SD}}$, $S_{\mathrm{HH}}$, and $S$. Overestimation ($s_n > r_n$) results in positive $\log_2$ ratios, whereas underestimation ($s_n < r_n$) results in negative $\log_2$ ratios. The ideal score is zero (shown as a red horizontal line), because if $s_n = r_n$ it follows that $\log_2(s_n/r_n) = \log_2(1) = 0$. The red distributions (hatched with positive slope, on the right of each item) are based on $s_n$ values computed with eighth-note grid support ($\Lambda(\cdot, \mathcal{G})$), and the blue distributions (hatched with negative slope, on the left of each item) are based on $s_n$ values computed without eighth-note grid support ($\Lambda(\cdot)$). Note that the blue (left-hand) distributions for $S_{\mathrm{KD}}$ and $S_{\mathrm{SD}}$ have larger variances than their red (right-hand) counterparts. This confirms that eighth-note grid support helps to stabilize swing ratio estimations for $S_{\mathrm{KD}}$ and $S_{\mathrm{SD}}$ (i.e., for the components that tended to have sparser onsets). In contrast, the blue and red distributions are nearly identical to each other for $S_{\mathrm{HH}}$, and also for $S$. This is not surprising, because $S_{\mathrm{HH}}$ and $S$ tended to have onsets at most eighth-note divisions already, thus leaving little room for enhancement from eighth-note grid support. Other relevant trends in Figure 7.9b include the tendency, on average, to overestimate swing ratio with $S_{\mathrm{KD}}$ and SD without eighth-note grid support, and to underestimate swing ratio once eighth-note grid support has been introduced.

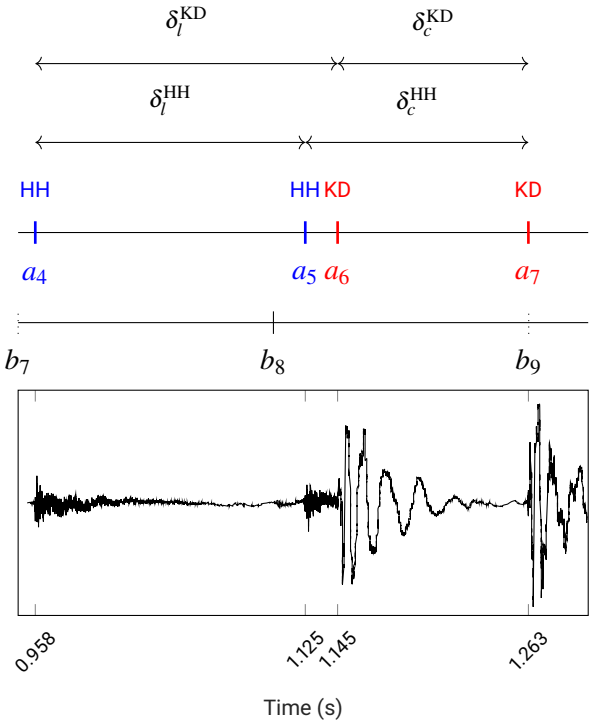



Figure 7.10: Excerpt from "Impeach the President" by the Honey Drippers. The audio waveform is shown in black; the blue vertical bars at 0.95 and 1.12 s are HH onsets; the red vertical bars at 1.14 and 1.26 s are KD onsets. This is a problematic case because the swing ratio differs considerably, depending on whether the HH onset at $a_5$ or the KD onset at $a_6$ is used in the computation, even though those two onsets are only 0.02 s apart.

### 7.4.5 Problematic Example: Dual Swing

In "Impeach the President", the drummer plays hi-hat at an even-numbered sixteenth-note division, immediately followed by kick drum. The onsets of those two events are sufficiently far apart (0.02 s) that one could say they were played in a "loose" style, yet they are sufficiently close together that they clearly should be mapped to the same nominal division. Because we are dealing with ratios of small numbers, and because changing the onset of an even-numbered division affects not only the length of that division but also the length of the preceding division, a small variation in onset can substantially alter the swing ratio computation. In this case, considering the hi-hat onset as the division onset results in an LSR of 1.6, whereas considering the kick drum as the division onset results in an LSR of 1.8. Figure 7.10 illustrates this dual swing ratio. Following the formalization in Algorithm 1, the main issue is that the function $D$ holds true for all four intervals $\delta_c^{KD}$, $\delta_l^{KD}$, $\delta_c^{HH}$, and $\delta_l^{HH}$. In other words, both $a_5$ (shown in blue, corresponding to a hi-hat onset at 1.125 s) and $a_6$ (shown in red, the kick drum onset at 1.145 s) are mapped to the same even-numbered beat, $b_8$. Because both choices yield a valid result for $D$, this leads to two potential pairs of $\delta_c$ and $\delta_l$.

### 7.4.6 Explorative Statistical Results on Large Dataset

In this section we report statistical trends observed when applying our swing ratio estimation method to a larger corpus of drum break recordings. We use a superset of the corpus from [92], comprising a total

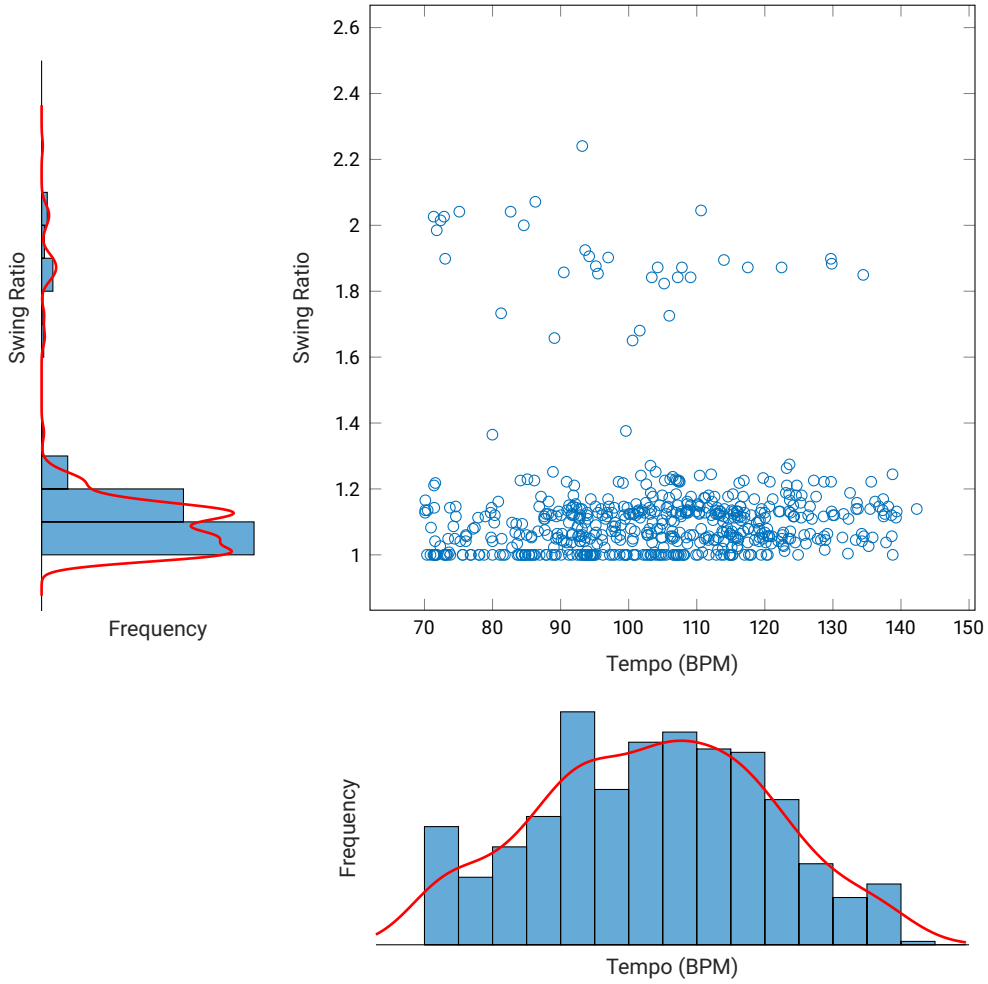



Figure 7.11: Statistical model of swing ratio vs. tempo for our large corpus of drum breaks. Each point in the scatter plot corresponds to one drum break recording, giving a rough estimate of the joint distribution of swing ratio and tempo. The histograms provide approximations to the marginal distributions of each variable independently.

of 575 drum breaks. For this large-scale study we allow breaks containing percussion, as well as further instrumentation. Similarly to the jazz analyses conducted in [45], we were interested to see if there is a common pattern of swing ratio given different tempi in this kind of data.

As it turns out, there are only two conclusions that can be safely drawn with this perspective on the data. The first finding is that drum breaks in our dataset have a tempo range between 80 and 120 BPM, with a mode close to 110 BPM. The second finding is that sixteenth-note swing is used very subtly, rarely exceeding an SR of 1.2; we give visual evidence for these two findings in Figure 7.11, which shows a scatter plot representing all drum break items as single points in the swing ratio vs. tempo plane. The corresponding histograms support our conclusions.

As a final note, we suggest that Figure 7.11 could serve as a basis for an interactive application, enabling audiovisual exploration of our dataset. This would be particularly interesting as future work, analyzing outliers in the dataset (i.e., the point-cloud gathered close to swing ratio 2).

## 7.5 Conclusion

In this chapter we investigated swing ratio estimation in drum break recordings. Starting from a previous study that was based on expert annotations of drum sound onsets, we formalized the algorithm used to quantify LSR. We then adapted a method that was originally intended for swing ratio estimation in jazz solo recordings to the drum breaks. Through a series of experiments, we showed a satisfactory correspondence between mean LSR computed as per [56] and global SR computed automatically with LLACF. We observed that swing ratio estimation is very sensitive, as a few milliseconds in onset deviation can strongly influence the result. In contrast to jazz music, this is even more problematic with drum breaks, since they tend to exhibit a much more subtle use of swing. It should be acknowledged that swing ratio is not the only metric of swing magnitude. Swing can also be quantified by *onset displacement*, i.e., the delay of even-numbered divisions, which can be heuristically defined as $(\delta_o - \delta_e)/2$; see [57]. However, unlike swing ratio, onset displacement in a given recording changes when the recording is played at different speeds (unless the onset displacement is zero). Thus, given that sampled drum breaks are rarely used at their original tempi, swing ratio is generally a more relevant metric for drum breaks. Despite these issues, we applied our automated method to a considerably larger corpus, resulting in an explorative study of the statistical distribution of swing in drum breaks. We hope that our speculative findings serve as a foundation for further research in the domain of drum breaks, rhythm, and groove in general.

# Chapter 8

# Summary and Future Work

In this thesis we have developed methods to analyze SBEM as well drum breaks—one of its most prominent sampling sources. In the following we will go through each chapter, revisiting key findings and contributions, and putting them into the context of potential applications and future work.

In Chapter 3 we studied using peak-map-based audio fingerprinting to recognize samples under various types of modifications and degradations commonly present in SBEM. Although this task is not new, and our experiments were performed in a controlled, synthetic environment, our contribution was to quantify the effects of modifications such as overlaying multiple samples, changing the STFT framing offset, or adding white noise. Furthermore, we studied the relationship between the sonic properties of a query loop (classified as either harmonic, noise-like, or percussive) and the feasibility of retrieving it from a mixture. We learned that peak maps respond differently for combinations of these sound categories, suggesting that it would make sense to incorporate this information into future algorithms. If we consider that state-of-the-art methods which are robust to timescale modifications (such as the one by Sonnleitner et al. [129]) are still challenged by overlapping musical sources, we can see the importance of our investigation. This use case of fingerprinting (or automated sampling detection/analysis) is interesting and useful for many reasons. Perhaps most prominently, there is great monetary value in being able to automatically determine sampling cases, for royalty payments and legal proceedings. However, scholars and SBEM lovers can gain a lot from exploring the connections between sampling *sources* and *targets*, as made evident by websites such as `WhoSampled.com`.

In Chapter 4, inspired by how SBEM artists structure their compositions, we developed a model and a method (based on NMFD) to answer the following question: If we are given the downmix of a SBEM track, along with the patterns that were used to create it, can we determine the points in time where each pattern was activated? Although we used a controlled environment and a priori information that is seldom available, we were able to achieve promising results in the task of finding activation points. A related question, based on our initial study, was investigated by Smith and Goto [125], who used non-negative tensor factorization paired with downbeat estimation to approach this task with less a priori information.

The art of structuring tracks is difficult to master—in contrast to other musical skills (such as piano or guitar playing technique), there are remarkably few resources that allow aspiring producers to improve their understanding of structure. Future work along these lines could yield interactive tools for analyzing structure in electronic music.

In Chapter 5 we developed a mid-level audio feature based on cascaded harmonic-residual-percussive (CHRP) source separation. Although it requires manual parameter selection, we were able to show that we can use this type of feature to visualize important musical phenomena, such as the joint role of timbre and event density in a track's structure. Given the ease of interpreting the CHRP representation, future work could incorporate this type of visualization into a GUI with easily controllable parameters. Such a GUI could be useful for aspiring musicians wishing to master density-related techniques, such as paradiddles with varying speeds on a snare drum.

In Chapter 6 we designed a system for finding drum breaks by classifying short audio frames as *percussion-only* or *not only percussion*. The relative rarity of drum breaks as a musical event (5–6% of our dataset), coupled with the amount of time and effort that SBEM artists invest in finding and manipulating them, indicate that drum breaks are a compelling musical construct warranting more attention from researchers. Although our method does not deliver ready-to-use drum breaks, we were able to show that our system, based on random forests and a bag-of-timbral-features, can effectively perform this binary classification. By contributing an annotated dataset of drum break regions (in the form of segmentations) we call attention to an important musical construct that is pervasive across many genres beyond SBEM. Music producers looking in digital collections would benefit from a full-blown system that can present relevant suggestions for sampling new drum breaks.

In Chapter 7 we formalized an algorithm by Frane [56] to compute local swing ratio (LSR) based on both the preceding and following interval lengths. Then we used Frane's expert-annotated dataset to investigate the correspondence between LSR computed manually and an automatic estimation of global swing ratio. This is an important contribution mainly for two reasons. First this study departs from the more numerous studies on swing ratio and microtiming in jazz. Second, the dataset is interesting because drum breaks are a natural occurrence of solo percussion, and thus estimation algorithms can operate unhindered by further instrumentation. Despite the small dataset, we were able to observe the joint role of swing ratio and swing density (the number of intervals that are actually swung, divided by the total number of intervals). Given the painstaking effort that it takes to annotate drum onsets at the sixteenth-note level (which is required to compute swing ratios, it would be non-trivial to categorize or sort a large collection of drum breaks based on their swing ratios. Studying drum breaks from the perspective of swing ratio is not only interesting from a musicological perspective, but it also opens up new forms of interaction for producers looking to sample material based on *groove* and other microrhythmic properties.

# Appendix A

# NMF Toolbox

I collaborated with Christian Dittmar to develop the `NMF Toolbox`: a collection of `MATLAB` code and examples that span multiple publications, such as [49], [44], [90], and [89]. The formal definitions and mathematical notation follow [97] and [44].

## A.1 Introduction

Non-negative matrix factorization (NMF) is a family of methods widely used in MIR and audio processing, but also many other fields that deal with signal processing and information retrieval. In general terms, NMF decomposes a non-negative matrix V into the multiplication of two other non-negative matrices W and H, such that $V \approx \tilde{V} := W \cdot H$.

In the case of audio signal processing and MIR, one popular choice for the matrix V is the magnitude spectrogram of an audio signal containing a number of musical events spread across time. Intuitively speaking, we would like to learn two non-negative matrices W and H that capture *what* happened (i.e., which particular sounds were produced, which drum kit parts were struck, which piano notes were sounded), and *when* each one of these sound events was active in time. Intuitively, we can say that the template (or spectral basis) matrix W contains the "what"—whereas H, called the gain or activation matrix, contains the "when". Keeping in mind the SBEM production setup described in Section 2.2, we can draw parallels between SBEM production and the NMF framework. Recall that a sequencer sends abstract musical instructions to the sampler (and the other instruments), telling it when to activate a pattern, and with which intensity. Following this SBEM production metaphor, we can think of the activation matrix H as representing the sequencer; H encodes the information about the location in time and relative intensity

of the musical events. On the other hand, we can think of the template matrix W as the sampler, since it contains a representation of the actual sounds or spectral information that is to be realized.

### A.1.1 NMF and NMFD Models

For the sake of easier reading, in this section we reproduce portions from [44] and Chapter 4.

NMF is based on iteratively computing a low-rank approximation $\tilde{\mathrm{V}} \in \mathbb{R}_{\geq 0}^{K \times M}$ of the mixture spectrogram V, where $K \in \mathbb{N}$ is the feature dimensionality and $M \in \mathbb{N}$ represents the number of elements or frames along the time axis. Specifically, $\tilde{\mathrm{V}}$ is defined as the linear combination of the templates $\mathrm{W} \in \mathbb{R}_{\geq 0}^{K \times R}$ and activations $\mathrm{H} \in \mathbb{R}_{\geq 0}^{R \times M}$ such that $\mathrm{V} \approx \tilde{\mathrm{V}} := \mathrm{W} \cdot \mathrm{H}$. The rank $R \in \mathbb{N}$ of the approximation (i. e., number of components) is an important but generally unknown parameter.

NMF typically starts with a suitable initialization of the matrices W and H. For example, both matrices could be populated with non-negative random numbers—however, depending on the task and availability of prior information, it might make more sense to use other initialization strategies. After initialization, both W and H are iteratively updated to approximate V with respect to a cost function $\mathcal{L}$. A standard choice is the generalized Kullback-Leibler Divergence (KLD) [83], given as

$$\mathcal{L} = \mathcal{D}_{\mathrm{KL}}(\mathrm{V} \mid \tilde{\mathrm{V}}) = \sum \left( \mathrm{V} \odot \log \left( \frac{\mathrm{V}}{\tilde{\mathrm{V}}} \right) - \mathrm{V} + \tilde{\mathrm{V}} \right). \tag{A.1}$$

The symbol $\odot$ denotes element-wise multiplication; the logarithm and division are to be performed element-wise as well. The sum is to be computed over all $K \cdot M$ elements of V. To minimize this cost, an alternating scheme with multiplicative updates is used [83]. The respective update rules are given as

$$\mathrm{W} \leftarrow \mathrm{W} \odot \frac{\frac{\mathrm{V}}{\tilde{\mathrm{V}}} \cdot \mathrm{H}^{\top}}{\mathrm{J} \cdot \mathrm{H}^{\top}}, \tag{A.2}$$

$$\mathrm{H} \leftarrow \mathrm{H} \odot \frac{\mathrm{W}^{\top} \cdot \frac{\mathrm{V}}{\tilde{\mathrm{V}}}}{\mathrm{W}^{\top} \cdot \mathrm{J}}, \tag{A.3}$$

where the symbol $\cdot$ denotes the matrix product. Furthermore, $\mathrm{J} \in \mathbb{R}^{K \times M}$ denotes a matrix of ones. Since this is an alternating update scheme, it should be noted that Eq. (A.2) uses the latest update of H from the previous iteration. In the same vein, (A.3) uses the latest update of W. These update rules are typically applied for a limited number of iterations $L \in \mathbb{N}$, with the iteration index $\ell \in [0 : L]$.

NMFD extends NMF to the convolutive case by using two-dimensional templates so that each of the $R$ templates (or spectral bases) can be interpreted as a magnitude spectrogram snippet consisting of $T \ll M$ spectral frames. We assume that the track (and therefore the track's magnitude spectrogram) was constructed from a set of $R$ patterns $\mathrm{P}^r \in \mathbb{R}^{K \times T^r}$, $r \in [0 : R-1] := \{0, \ldots, R-1\}$. The parameter $T^r \in \mathbb{N}$ is the number of feature frames or observations for pattern $\mathrm{P}^r$. In practice, the patterns can

have different lengths—however, without loss of generality, we define their lengths to be the same $T := T^0 = \ldots = T^{R-1}$, which could be achieved by adequately zero-padding shorter patterns until they reach the length of the longest. Based on this assumption, the patterns can be grouped into a pattern tensor $\mathrm{P} \in \mathbb{R}^{K \times R \times T}$. Consequently, the subdimension of the tensor which refers to a specific pattern with index $r$ is $\mathrm{P}^r := \mathrm{P}(\cdot,\ r,\ \cdot)$, whereas $\mathrm{P}_t := \mathrm{P}(\cdot,\ \cdot,\ t)$ refers to frame index $t$ simultaneously in all patterns. Thus, the model for the feature representation becomes

$$V \hat{=} \sum_{t=0}^{T-1} \mathrm{P}_t \cdot \overset{t\rightarrow}{\mathrm{H}}, \tag{A.4}$$

where $\overset{t\rightarrow}{(\cdot)}$ denotes a frame shift operator [124].

## A.2 SBEM Example: Learning a Track Model with Minimal Information

We saw in Chapter 4 that we can use NMFD to learn the activation points for a SBEM track if we have a downmix of the track and one copy of each loop present in the mix (i. e., V and W, respectively). As we discussed in Section 4.6, it is unrealistic to expect that the individual loops are available, and thus the question arises: *How does NMFD perform with the absolute minimum amount of information?* In other words, what can we learn if we have only the target spectrogram V, the number of loops or patterns $R$, and the template length $T$? In the following we will go through the code example in Listing L.1 to examine this scenario.

```
inpPath = 'data/';
filename = '120_jd.wav';
[xTr,fs] = audioread([inpPath filename]);
xTr = mean(xTr, 2);

% spectral parameters
paramSTFT.blockSize = 4096;
paramSTFT.hopSize = 2048;
paramSTFT.winFunc = hann(paramSTFT.blockSize);
paramSTFT.reconstMirror = true;
paramSTFT.appendFrame = true;

% STFT computation for Track returns complex-val, mag-spec, phase-spec
[XTr, ATr, PTr] = forwardSTFT(xTr, paramSTFT);

% get dimensions and time and freq resolutions
[numBinsTr, numFramesTr] = size(XTr);
deltaT = paramSTFT.hopSize / fs;
deltaF = fs / paramSTFT.blockSize;

% set common parameters
numComp = 3;
numIter = 15;
```

```
24 numTemplateFrames = 87;
25
26 % generate initial activations
27 paramActivations.numComp = numComp;
28 paramActivations.numFrames = numFramesTr;
29 initH = initActivations(paramActivations, 'uniform');
30
31 % NMFD parameters
32 paramNMFD.numComp = numComp;
33 paramNMFD.numFrames = numFramesTr;
34 paramNMFD.numIter = numIter;
35 paramNMFD.numTemplateFrames = numTemplateFrames;
36 paramNMFD.initH = initH;
37 paramNMFD.numBins = numBinsTr;
38
39 % generate initial templates
40 paramTemplates.numComp = numComp;
41 paramTemplates.numBins = numBinsTr;
42 paramTemplates.numTemplateFrames = numTemplateFrames;
43 initW = initTemplates(paramTemplates, 'random');
44 paramNMFD.initW = initW;
45
46 %% visualize initialization data
47 paramVis = [];
48 paramVis.deltaF = deltaF;
49 paramVis.deltaT = deltaT;
50 fh1 = visualizeComponentsNMF(ATr, initW, initH, [], paramVis);
51
52 %% NMFD core method
53 [nmfdW, nmfdH, nmfdV, divKL] = NMFD(ATr, paramNMFD);
54
55 %% visualize after NMFD
56 fh2 = visualizeComponentsNMF(ATr, nmfdW, nmfdH, nmfdV, paramVis);
```

Listing L.1: Code example to learn templates and activations using a randomly initialized NMFD model.

In lines 1–3 of Listing L.1 we set up the audio path, read the file for the mixture track into the variable `xTr` (the suffix `Tr` for "Track" is appended to all relevant variables) and make it monaural (line 4). In lines 7–11 we use the structure `paramSTFT` to set the STFT parameters: a block size of 4096 samples, a hop size of 2048 samples, a Hann window, a Boolean value indicating that we wish to discard the mirror spectrum (`paramSTFT.reconstMirror`), and a Boolean indicating that we want to pad the entire signal with one-half block size at the beginning and end (`paramSTFT.appendFrame`). In line 14 we compute the STFT by calling `forwardSTFT`, which returns the complex-valued spectrogram `XTr`, the magnitude spectrogram `ATr`, and the phase spectrogram `PTr`. In lines 17–19 we obtain the spectrogram's dimensions and set `deltaT` (time resolution) and `deltaF` (frequency resolution) for later use. In lines 22–24 we set the first parameters relative to NMFD. First we set the number of components `numComp` $\hat{=} R := 3$, since we know a priori that the track was produced using 3 loops or patterns (line 22). In line 23 we set the initial number of iterations `numIter` $\hat{=} L := 15$. In line 24 we set the number of template frames

`numTemplateFrames` $\hat{=} T := 87$ because we know beforehand that each loop instance used to produce the track has a length of 8 seconds (equivalent to 16 quarter-note-beats at 120 BPM, or 87 STFT frames under the current settings). The next steps are to prepare the initial states for the activations and the templates. In lines 27–29 we prepare the activation matrix `initH` $\hat{=}$ H. The function `initActivations()` initializes the activation matrix with dimensions specified in `paramActivations` (lines 27 and 28), and values specified as a second parameter. The output matrix will consist of all ones, since we pass the string parameter `uniform` to `initActivations()`. All the previous NMFD parameters are assigned to the parameter structure `paramNMFD` in lines 32–37—this structure will later be passed to the function `NMFD()`. In lines 40–42 we assign the three dimensions to the pattern tensor P, using `paramTemplates`. The function `initTemplates()` takes these parameters, along with the string `random`, and initializes the template tensor `initW` $\hat{=}$ W with random values. In line 44 we append the initialized template tensor `initW` to `paramNMFD`. In order to visualize the initial states of the templates and activations, we define the variable `paramVis` in lines 47–50. We then call `visualizeComponentsNMF()` with the track spectrogram `ATr`, the initialized templates `initW`, the initialized activations `initH`, and `paramVis`—this results in Figure A.2. The core function in this example, `NMFD()` (line 53), takes the target spectrogram `ATr` $\hat{=}$ V together with the the parameter structure `paramNMFD` and performs the learning. The output arguments for `NMFD()` are the learned pattern tensor `nmfdW`, the learned activation matrix `nmfdH`, the component spectrograms `nmfdV`, and the Kullback-Leibler divergence across all iterations, `divKL`. Finally, in line 56 we use these outputs from `NMFD()` to call `visualizeComponentsNMF()`—this time generating the plots seen in Figure A.3.

In Figure A.1a we show the ground truth (GT) for the track presented originally in Chapter 4 and which we use for further examples in this appendix. Figure A.1b shows the GT templates, corresponding to the three loops used in the track. The first template (colored red), corresponds to the drums; the second template (green) corresponds to the melody; the third template (blue) corresponds to bass. In Figure A.1c we show the GT activations, color-coded and numbered to match the templates on the left.

Figure A.2a shows W, the vertical axis represents log-frequency. Note that although the experiments were conducted with linear frequency axes, we show log-frequency to enhance visual properties. The templates are color-coded in red (template 1), green (template 2), and blue (template 3). In Figure A.2b we show the activation matrix H, whose rows are colored and numbered to match the templates on the left. For visualization (and also in subsequent figures), each row in H is normalized to have values between 0 and 1—since H is initialized to the constant 1 and we have colored the area under the curve, this results in each row appearing as a horizontal bar.

Figure A.3 contains four panels—it is a snapshot of the NMFD model at iteration 15 (from a total of 30). In Figure A.3a we show the loss function $\mathcal{L}$ as a blue curve, for the first 15 iterations (denoted by a red circle at iteration 15). The vertical axis, which is logarithmic, represents the KLD between the target spectrogram V and the approximation $\tilde{\mathrm{V}}$. We can see that the fastest learning happened within the first 3 iterations, where the slope is most pronounced. After the third iteration, learning becomes

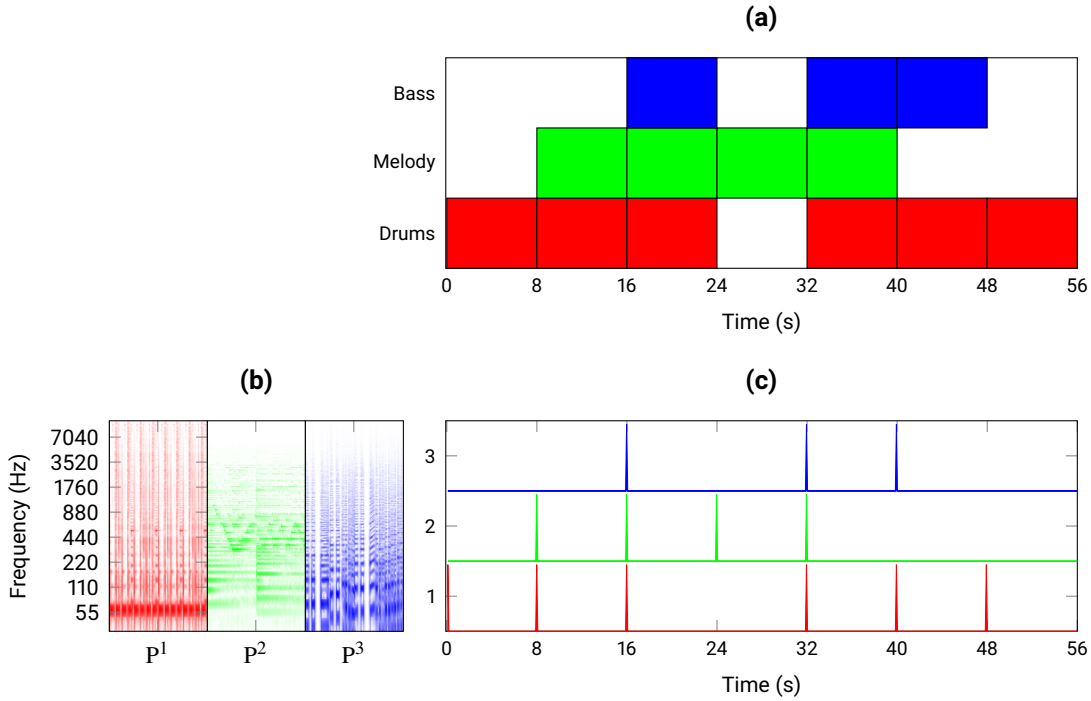



Figure A.1: **(a)** Production structure for example track. **(b)** Ground truth templates and **(c)** ground truth activations for NMFD model to learn structure and patterns of a SBEM track.

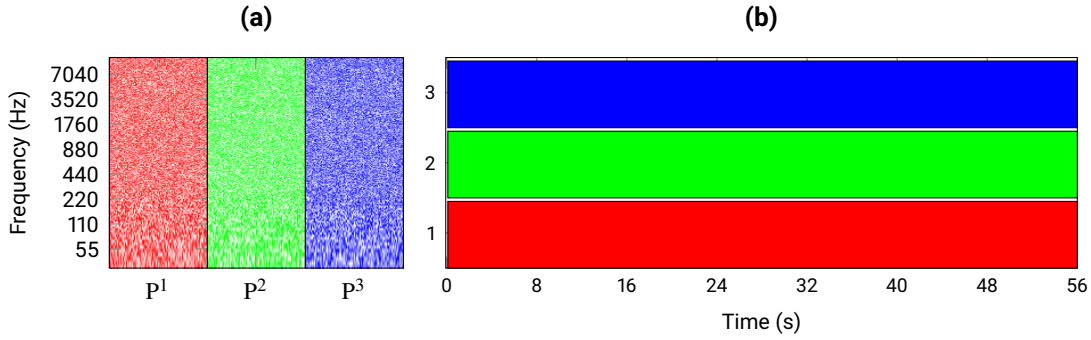



Figure A.2: **(a)** Initial state for templates in an uninformed NMFD model to learn structure and patterns of a SBEM track. **(b)** Initial state for activations.

slower, stabilizing at a divergence close to $1 \cdot 10^{-6}$. Figure A.3b contains H— notice that the rows are very similar to each other: there are higher gains close to the activation points (at multiples of the loop length, 8 seconds) and lower periodic peaks throughout. In Figure A.3c (showing W), we can see that the model has learned some salient timbral properties mostly in the lower frequencies, but the templates have not yet differentiated from each other—each template is still a mixture of all three loops used to produce the track. Finally, in Figure A.3d, which shows $\tilde{V} := W \cdot H$, we can see coarse structural properties beginning to appear, like the higher density between 16–24 s and 32–40 s, and a lower density from 24–32 s, where only the melody pattern is active.

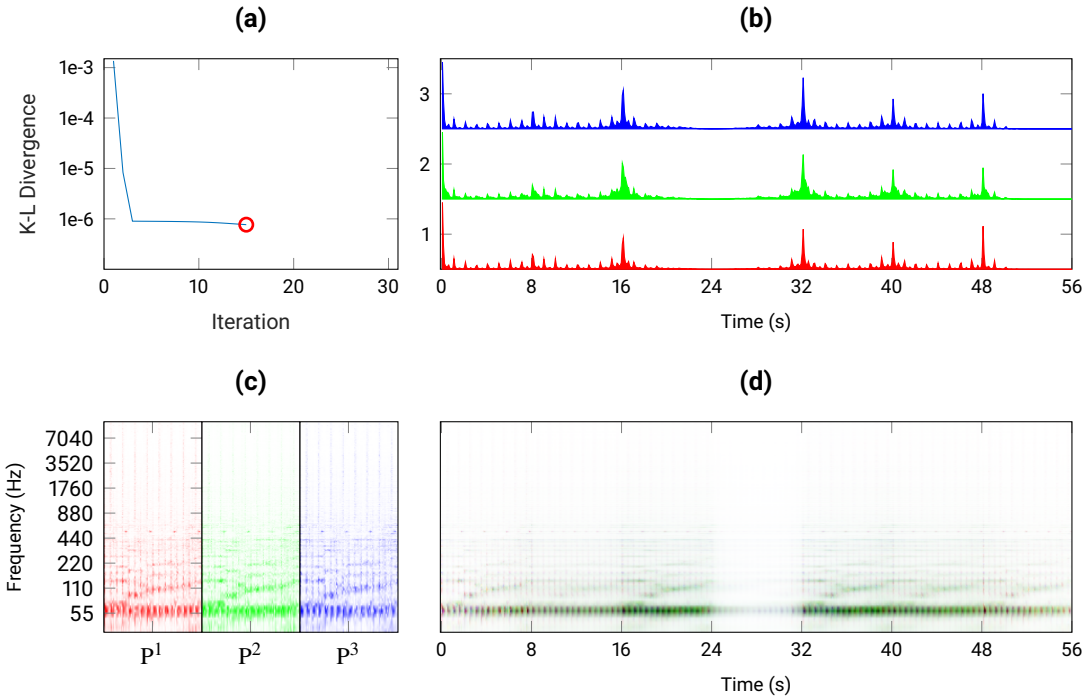



Figure A.3: NMFD model with random initialization at iteration 15.

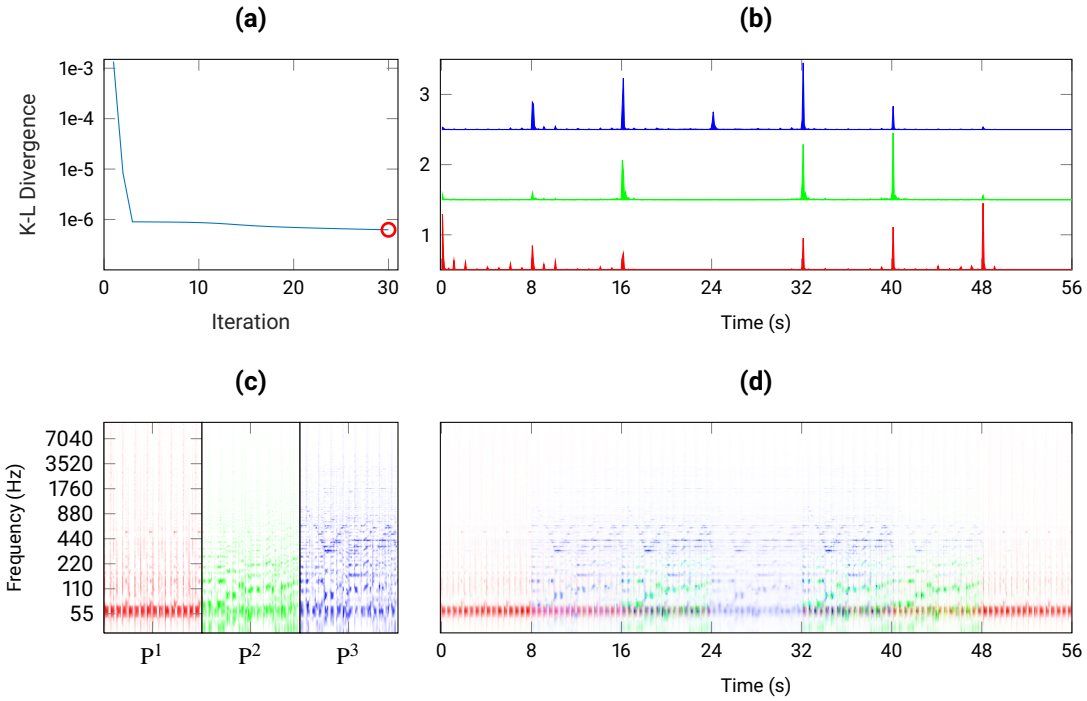



Figure A.4: NMFD model with random initialization at iteration 30.

Figure A.4 shows our NMFD model at the final state, after 30 iterations. In Figure A.4a we see the KLD has reached a value slightly below $1 \cdot 10^{-6}$; the model has learned the track structure and templates remarkably well. We will now go through the visual characteristics of Figure A.4b and A.4c. The first spectral template $P^1$ has best approximated the drum loop—this is apparent in the vertical structures, as well as the activation peaks in H (Figure A.4b), which largely coincide with the GT activations. $P^2$ best approximates the bass pattern: we can see spectral information in the lower frequencies of $P^2$ and the activations in the second row of H largely coincide with the GT. Finally, we come to $P^3$, which closely resembles the melody pattern. We can see the up-and-down main melody line centered at around 400 Hz, and almost all activations in the top row of H coincide with the GT. The only exception is the last activation (around 40 s), which does not appear in the original track. This spurious peak tells us that $P^2$ and $P^3$ contain information from both the melody and bass patterns, leading to this *confusion* in the model. Despite this erroneous peak, NMFD has correctly captured the track's most salient characteristics, without prior knowledge about the spectral content.

## A.3 Diagonality-Enhanced NMF

In contrast to NMFD, where temporal continuity of the spectral bases is enforced through convolutive template activation, standard NMF algorithms offer no such guarantees. For certain tasks, such as SBEM decomposition, it is desirable to enforce temporal continuity by activating contiguous templates. In [49], Driedger et al. describe a set of update rules for NMF that support the development of sparse diagonal structures in the activation matrix. The intent of these rules is to encourage the activation of consecutive templates to obtain sonically pleasing results in a task known as *audio mosaicing*. In this section we present an example of diagonality-enhanced NMF (DE-NMF) with fixed templates—a different approach for learning the activations of a SBEM track with known loops.

```
inpPath = 'data/';
filename = '120_jd.wav';
filenameBass = '120_jd_bass_1.wav';
filenameMelody = '120_jd_melody_1.wav';
filenameDrums = '120_jd_drums_1.wav';

[xTr,fs] = audioread([inpPath filename]);
xTr = mean(xTr, 2);
[xBass, fsBass] = audioread([inpPath filenameBass]);
[xMelody, fsMelody] = audioread([inpPath filenameMelody]);
[xDrums, fsDrums] = audioread([inpPath filenameDrums]);
xBass = mean(xBass, 2);
xMelody = mean(xMelody, 2);
xDrums = mean(xDrums, 2);

% spectral parameters
paramSTFT.blockSize = 4096;
paramSTFT.hopSize = 2048;
paramSTFT.winFunc = hann(paramSTFT.blockSize);
```

```
20  paramSTFT.reconstMirror = true;
21  paramSTFT.appendFrame = true;
22
23  % STFT computation for Track returns complex-val, mag-spec, phase-spec
24  [XTr, ATr, PTr] = forwardSTFT(xTr, paramSTFT);
25
26  [XBass, ABass, PBass] = forwardSTFT(xBass, paramSTFT);
27  [XMelody, AMelody, PMelody] = forwardSTFT(xMelody, paramSTFT);
28  [XDrums, ADrums, PDrums] = forwardSTFT(xDrums, paramSTFT);
29  numTemplateFrames = size(ABass, 2);  % we know that all loops are same length
30
31  % get dimensions and time and freq resolutions
32  [numBinsTr, numFramesTr] = size(XTr);
33  deltaT = paramSTFT.hopSize / fs;
34  deltaF = fs / paramSTFT.blockSize;
35
36  W0 = cat(2, ADrums, AMelody, ABass);
37  numSourceFrames = size(W0, 2);
38  numTargetFrames = numFramesTr;
39  % initialize activations randomly
40  paramActivations.numComp = numSourceFrames;
41  paramActivations.numFrames = numTargetFrames;
42  H0 = initActivations(paramActivations, 'random');
43
44  paramNMFdiag.fixW = 1;
45  paramNMFdiag.numOfIter = 20;
46  paramNMFdiag.continuity.polyphony = 3;
47  paramNMFdiag.continuity.length = floor(numTemplateFrames / 2);
48  paramNMFdiag.continuity.grid = 1;
49  paramNMFdiag.continuity.sparsen = [1, floor(numTemplateFrames / 2)];
50
51  % call the reference implementation as provided by Jonathan Driedger
52  [nmfdiagW, nmfdiagH] = NMFdiag(ATr, W0, H0, paramNMFdiag);
53  nmfdiagV = nmfdiagW * nmfdiagH;
54
55  %% visualize
56  paramVis = [];
57  paramVis.deltaF = deltaF;
58  paramVis.deltaT = deltaT;
59  fh1 = visualizeComponentsNMF(nmfdiagV, nmfdiagW, nmfdiagH, [], paramVis);
```

Listing L.2: Code example to learn NMF model with diagonality-enhanced activation matrix and fixed templates.

The code in Listing L.2 shows DE-NMF with fixed templates, mainly following [49]. In lines 1–5 we set up the paths where the audio files are located: the mixture track, as well as instances of the drums, melody, and bass loops. In lines 7–14 we load the audio files and make them monaural for further processing (variables `xTr`, `xBass`, `xMelody`, and `xDrums`). In lines 17–21 we prepare the parameter structure `paramSTFT` with the necessary fields (identical to those described in Section A.2). In line 24 we compute the STFT for the mixture track, obtaining the complex-valued spectrogram `XTr`, the magnitude spectrogram `ATr`, and the phase spectrogram `PTr`. In lines 26–28 we compute the same three types

of spectrograms for the individual loops, prefixing the variable names with `X`, `A`, and `P`, respectively. Since we know that all loops have the same length, in line 29 we use the number of frames in the bass loop to set the template length `numTemplateFrames`, in order to set a smoothing parameter later on in line 49. In lines 32–34 we save the dimensions of the track spectrogram, as well as the time and frequency resolutions `deltaT` and `deltaF`, respectively. To construct the spectral bases, we concatenate the loop spectrograms along the time dimension into a single matrix, called `W0` (line 36). In line 37 we save the size of the newly constructed template matrix `W0` as `numSourceFrames`, and in line 38 we save the number of STFT frames for the mixture track as `numTargetFrames`. In lines 40 and 41 we use the previously defined dimensions to create the parameter structure `paramActivations`, setting the number of spectral components `numComp` to `numSourceFrames` and the number of frames (or observations along time) `numFrames` to `numTargetFrames`. In line 42 we call `initActivations` with the parameter structure `paramActivations` and the string `random`, which outputs `H0`, the initial activation matrix with random values. In lines 44–49 we set the fields of `paramNMFdiag`, the structure that contains the parameters for the DE-NMF learning. In line 44 we use the Boolean `fixW` to specify that we want fixed templates—in other words, we disallow updates to the template matrix `W0`. In line 45 we set the number of iterations to 20. In line 46 we set `continuity.polyphony` to 3: this means that each column in the activation matrix can have up to three active (salient) entries, taking into account that our track has up to 3 concurrently active loops. In line 47 we set `continuity.length` to half the length of one loop instance; this parameter controls the length of the smoothing window enhancing the diagonals of the activation matrix. In line 48 we set `continuity.grid` to `1`, which ensures that the continuity constraints will be applied at each iteration of the algorithm. In line 49, the field `continuity.sparsen` specifies the length of a max-filter used to make activations more sparse, by preserving local maxima in the matrix. We have chosen one-half of the loop length, since we want to discourage diagonal structures that are too close together—i. e., repeating activations at the beginning of a loop instead of learning the full length. In line 52 we call the main function for this example, `NMFdiag()`, with `ATr` as the target spectrogram, `W0` as the initial (and fixed) template matrix, `H0` as the randomly initialized activation matrix, and `paramNMFdiag` as the parameter structure. `NMFdiag()` returns `nmfdiagW` and `nmfdiagH`, which we multiply in line 53 to obtain `nmfdiagV`, the learned spectrogram approximating the track. In lines 56–59 we prepare a structure with visualization parameters, `paramVis`, and then call `visualizeComponentsNMF()`, which results in the plot shown in Figure A.5.

Figure A.5 shows the state of our DE-NMF model after 20 update iterations. Figure A.5a contains H, the activation matrix learned with the constraints set in lines 44–49 of Listing L.2. The horizontal axis represents time and the vertical axis corresponds to frame indices from the matrix W in Figure A.5b. The red dashed horizontal lines are intended to simplify understanding the correspondence between diagonals and their counterparts in the template matrix—these annotations are added for illustrative purposes and are not generated by the function `visualizeComponentsNMF()` (line 59 of Listing L.2). The main characteristic of DE-NMF is that it enforces consecutive activations of the templates—which is desirable because we know a priori that full-length loop activations were used to produce the track. Recall

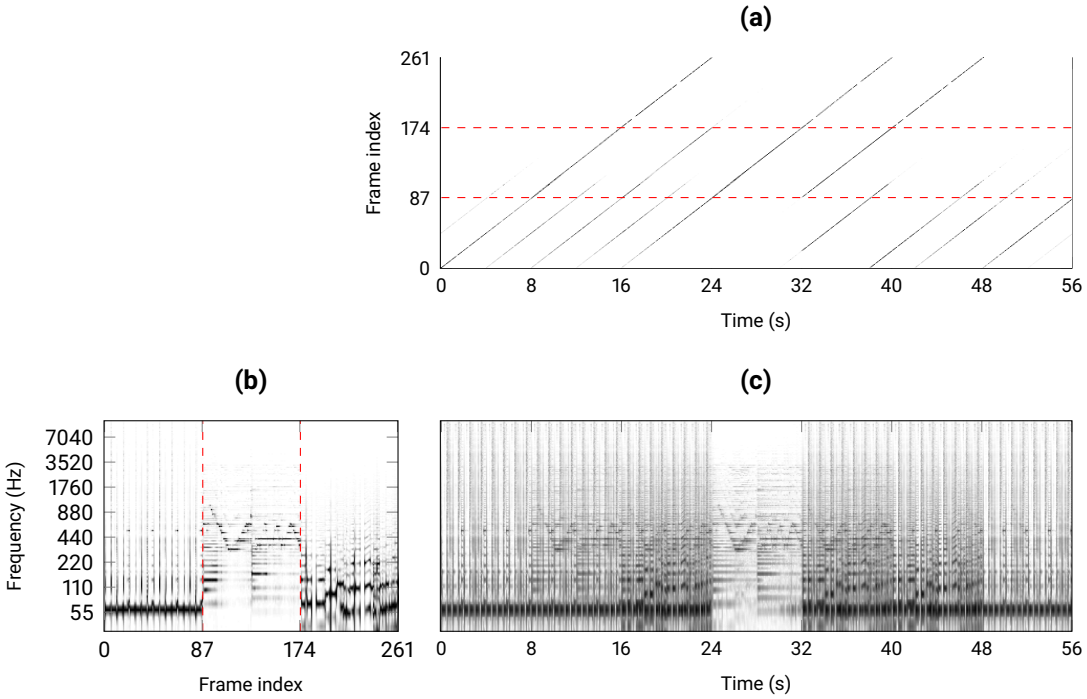



Figure A.5: Diagonal NMF.

that our running example track starts with an isolated activation of the drum loop from 0–8 s. We can see in the template matrix (Figure A.5b) that the region labeled 1 corresponds to the drum loop, which, in turn, corresponds to the bottom portion of the activation matrix, shown in Figure A.5a. Focusing on the drum loop activations, the expected behavior is to have a diagonal structure starting at the lower-left corner of the activation matrix and ending at the dashed line at frame index 87; this means that all template vectors from the drum loop were activated consecutively in the first 8 seconds of the track. Ideally, the bottom portion of the activation matrix would have 6 diagonals, starting at 0, 8, 16, 32, 40, and 48 s, respectively. In practice, due to the high self-similarity within the drum loop, further diagonals emerge between the expected ones. The same principle holds for the other regions in the activation matrix corresponding to melody and bass. In Figure A.5a we can see that DE-NMF has correctly learned the main structural aspects of our example track.

Figure A.5b shows the template matrix W, made by horizontally concatenating the drums (frames 0–86), melody (frames 87–173), and bass (frames 174–260). Each loop has a duration of 8 seconds, as indicated by the red dashed vertical lines which separate the loops in the matrix for better visibility; the vertical axis corresponds to log-frequency. Figure A.5c shows the model $\tilde{V} := W \cdot H$ obtained through the code in Listing L.2.

## A.4 MATLAB Function Reference

| Filename | Main parameters | Description |
|---|---|---|
| `NMFD.m` | `V,`<br>`numComp,`<br>`numIter,`<br>`numTemplateFrames,`<br>`initW,`<br>`initH,`<br>`paramConstr,`<br>`fixH` | Non-Negative Matrix Factor Deconvolution with Kullback-Leibler-Divergence and fixable components. The core algorithm was proposed in [124], the specific adaptions are used in [44]. |
| `NMF.m` | `V,`<br>`costFunc,`<br>`numIter,`<br>`numComp` | Given a non-negative matrix V, find non-negative templates W and activations H that approximate V. Multiple cost functions are supported. |
| `NMFdiag.m` | `V,`<br>`W0,`<br>`H0,`<br>`distMeas,`<br>`numOfIter,`<br>`fixW,`<br>`continuity.length,`<br>`continuity.grid,`<br>`continuity.sparsen,`<br>`continuity.polyphony` | Given a non-negative matrix V, find non-negative matrix factors W and H such that $V \approx W \cdot H$. If specified, also enforce continuity constraints, as in [49]. |
| `NMFconv.m` | `V,`<br>`numComp,`<br>`numIter,`<br>`numTemplateFrames,`<br>`initW,`<br>`initH,`<br>`beta,`<br>`sparsityWeight,`<br>`uncorrWeight` | Convolutive Non-Negative Matrix Factorization with Beta-Divergence and optional regularization parameters as described in Chapter 3.7 of [21]. The averaged activation updates are computed via the compact algorithm given in paragraph 3.7.3. For the sake of consistency, we use the notation from [44] instead of the one from [21]. |
| `semiFixedComponentConstraintsNMF.m` | `W,`<br>`H,`<br>`iter,`<br>`numIter,`<br>`initW,`<br>`initH,`<br>`adaptDegree,`<br>`adaptTiming` | Implements a simplified version of the soft constraints in [90]. Pass this function pointer to `NMFD()` in order to fix (or partially fix) templates. |
| `convModel.m` | `W,`<br>`H` | Convolutive NMF model implementing Eq. (4) from [44]. Note that it can also be used to compute the standard NMF model in case the number of time frames of the templates equals one. |

| | | |
|---|---|---|
| shiftOperator.m | A,<br>shiftAmount | Shift operator as described in Eq. (5) from [44]. It shifts the columns of a matrix to the left or the right and fills undefined elements with zeros. |
| initActivations.m | numComp,<br>numFrames,<br>deltaT,<br>pitches,<br>onsets,<br>durations,<br>drums,<br>decay,<br>onsetOffsetTol,<br>tolerance,<br>strategy | Implements different initialization strategies for NMF activations. The strategies 'random' and 'uniform' are self-explanatory. The strategy 'pitched' places gate-like activations at the frames, where certain notes are active in the ground truth transcription [47]. The strategy 'drums' places decaying impulses at the frames where drum onsets are given in the ground truth transcription [44]. |
| initTemplates.m | numComp,<br>numBins,<br>numTemplateFrames,<br>pitches,<br>drumTypes,<br>strategy | Implements different initialization strategies for NMF templates. The strategies 'random' and 'uniform' are self-explaining. The strategy 'pitched' uses comb-filter templates as described in [47]. The strategy 'drums' uses pre-extracted, averaged spectra of desired drum types [44]. |
| visualizeComponentsNMF.m | V,<br>W,<br>H,<br>compV,<br>deltaT,<br>deltaF,<br>startSec,<br>endeSec | Given a non-negative matrix V, and its non non-negative NMF or NMFD components, this function provides a visualization. |
| NEMA.m | lambda | This function takes a matrix of row-wise time series and applies a non-linear exponential moving average (NEMA) to each row. This filter introduces exponentially decaying slopes and is defined in Eq. (3) from [42]. |
| midi2freq.m | midi | Converts a given MIDI pitch to the corresponding frequency in Hz. No sanity checks on the validity of the input are performed. |
| freq2midi.m | freq | Converts a given frequency in Hz to the corresponding MIDI pitch, applying a quantization to semitone steps on the equal tempered scale. No sanity checks on the validity of the input are performed. |
| logFreqLogMag.m | A,<br>deltaF,<br>binsPerOctave,<br>upperFreq,<br>lowerFreq | Given a magnitude spectrogram, this function maps it onto a compact representation with logarithmically spaced frequency axis and logarithmic magnitude compression. |

| | | |
|---|---|---|
| `forwardSTFT.m` | `x,`<br>`blockSize,`<br>`hopSize,`<br>`winFunc,`<br>`reconstMirror,`<br>`appendFrame` | Given a time signal as input, this computes the spectrogram by means of the Short-time Fourier transform. |
| `inverseSTFT.m` | `X,`<br>`blockSize,`<br>`hopSize,`<br>`anaWinFunc,`<br>`reconstMirror,`<br>`appendFrame,`<br>`synWinFunc,`<br>`analyticSig,`<br>`numSamples` | Given a valid STFT spectrogram as input, this reconstructs the corresponding time-domain signal by means of the frame-wise inverse FFT and overlap-add method described as LSEE-MSTFT in [69]. |
| `alphaWienerFilter.m` | `alpha,`<br>`binarize` | Given a cell-array of spectrogram estimates as input, this function computes the alpha-related soft masks for extracting the sources. Details about this procedure are given in [88], further experimental studies in [39]. |

# Appendix B

# Break-Informed Audio Decomposition for Interactive Redrumming

This section is based on [89]: Patricio López-Serrano, Matthew E. P. Davies, Jason Hockman, Christian Dittmar, and Meinard Müller. Break-informed audio decomposition for interactive redrumming. In *Late Breaking and Demo Session of the International Society for Music Information Retrieval Conference (ISMIR)*:, 2018.

In this appendix we describe a further scenario where audio decomposition techniques are used for a creative application. *Redrumming* or *drum replacement* is used to substitute or enhance the drum hits in a song with one-shot drum sounds obtained from an external collection or database. In an ideal setting, this is done on multitrack audio, where one or more tracks are dedicated exclusively to drums and percussion. However, most non-professional producers and DJs only have access to mono or stereo downmixes of the music they work with. Motivated by this scenario, as well as previous work on decomposition techniques for audio signals, we propose a step towards enabling full-fledged redrumming with mono downmixes.

## B.1 Proposed Method

Figure B.1 gives an overview of our method for break-informed redrumming, inspired by [99]. The figure consists of two sides: the left side contains the track that will be redrummed, and the right side contains the track that will provide new timbral information. We describe each side in the following.

As input for the track that will be redrummed, we need the monaural (mono) downmix of a song that contains a drum break, as well as a segmentation indicating the location of the drum break. A method for

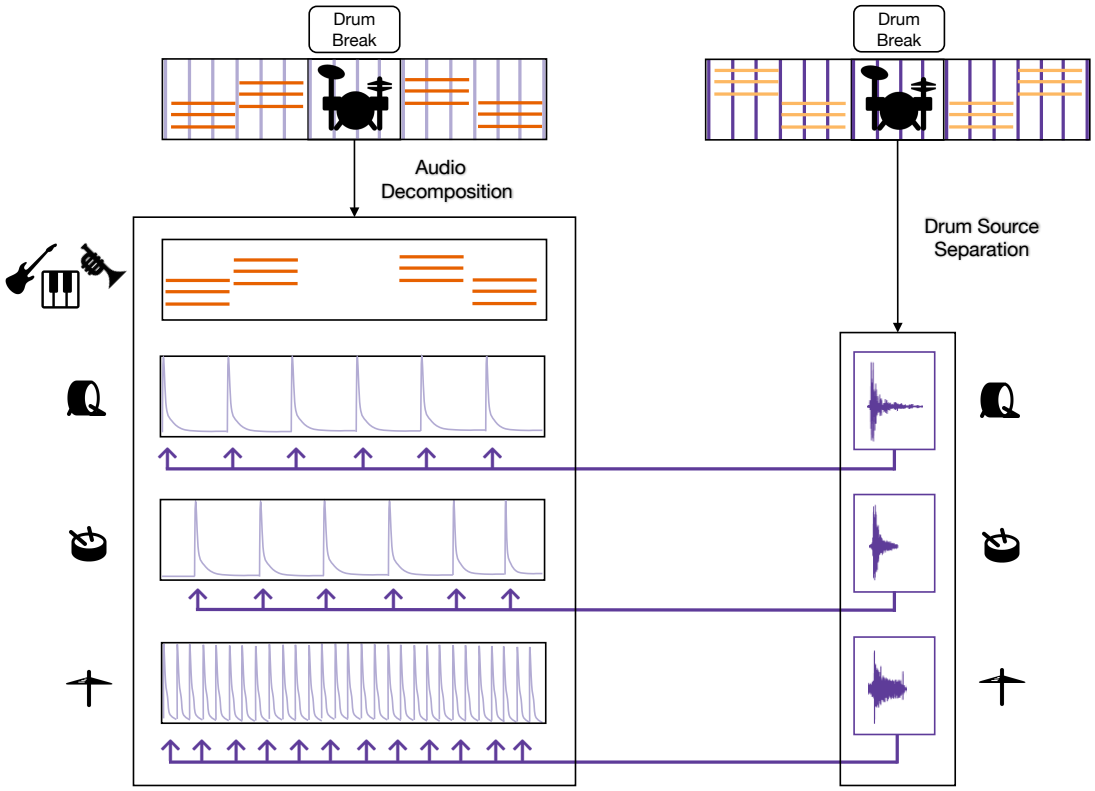



Figure B.1: Overview of our break-informed redrumming method. For a detailed explanation, see Section B.1.

automatically finding percussion-only regions in digital music recordings is described in [92]. Since we are working with mono input data, we need a way to extract multiple drum kit tracks from the mixture. At this stage we use the drum source separation (DSS) method by Dittmar and Müller [44] (which is based on non-negative matrix factor deconvolution, NMFD) to learn timbral templates for the drum kit elements present in the break.

Following the assumption that the drum timbre remains largely unchanged throughout the track, we fix the drum templates and learn a further non-negative matrix factorization (NMF) for the entire track, following [79]. In principle, this is equivalent to NMF-based harmonic-percussive source separation (HPSS), such as recently proposed by [81]. We now have spectral information for all the non-percussive instruments (or the *harmonic* part, shown as horizontal orange bars), which includes lead melody as well as accompaniment instrumentation. Our NMF model has also learned activations for the fixed drum templates—these activations are shown as light purple curves for kick drum (KD), snare drum (SD), and hi-hat (HH), and will be used as insertion positions for the redrumming.

The right side of Figure B.1 shows a second song containing a drum break, along with an indication of where the break is. This drum break will provide the timbral information for the redrum, with sounds being "copied" onto appropriate timepoints in the left-hand track. Again, we use NMFD-based DSS [44]

to learn the timbral properties of individual drum hits in the break, in order to combine these spectral templates with the activations found in the left-hand track.

## B.2 Real-World Scenario

Fully automatic redrumming is a difficult task. Especially when working with audio mixtures, the main challenge lies in avoiding crosstalk—not only between the harmonic and percussive parts, but also among the individual drum kit parts themselves. In a studio setting, redrumming is usually done by selecting (one-shot) drum hit sounds from a collection or database—in this contribution, instead of having a ready-made drum sound database, we construct this collection from a drum break of the user's choosing. Thus, for instance, we can imagine that a user selects James Brown's "Funky Drummer" as a track to be redrummed, using the sounds (or timbral properties) from "Amen, Brother" by The Winstons.

Redrumming is also usually done manually or semi-manually, using a DAW such as `Logic Pro` or `Cubase`, or a specialized plugin, like `Drumagog` or `Superior Drummer`. Assuming that a user wishes to substitute KD hits, the procedure would be to step through all the transients in the drum track, placing the cursor at the beginning of each KD hit. Then, the user would select a sound from the library and use it to either replace the current KD hit, or place it in a new track to enhance the timbral properties of the existing one. An advantage of our system is that all hits of a certain type (e.g., all KD hits) are tracked simultaneously by our decomposition model, enabling a once-through automatic substitution.

Furthermore, in a professional setting, musicians often request a certain *sound* or *aesthetic* for the mixing process, wishing to sound like other artists that they admire. Thus, another advantage of our method is that users can directly achieve this effect by selecting a drum break recording with their desired properties.

# Bibliography


[1] Ableton. Live. `https://www.ableton.com/en/live/` (web source, retrieved March 2016), 2016.

[2] Mike Adamo. *The Breakbeat Bible*. Hudson Music, Briarcliff, NY, USA, 2010.

[3] Vincent Akkermans and Joan Serrá. Shape-based spectral contrast descriptor. In *Proceedings of the Sound and Music Computing Conference (SMC)*, pages 143–148, Porto, Portugal, 2009.

[4] Anna Aljanaki, Mohammad Soleymani, Frans Wiering, and Remco C. Veltkamp. Mediaeval 2014: A multimodal approach to drop detection in electronic dance music. In *Working Notes Proceedings of the MediaEval 2014 Workshop, Barcelona, Catalunya, Spain, October 16-17, 2014.*, 2014.

[5] Jan Van Balen, Martín Haro, and Joan Serrà. Automatic identification of samples in hip hop music. In *International Symposium on Computer Music Modeling and Retrieval (CMMR)*, pages 544–551, London, UK, June 2012.

[6] Jan Van Balen, Ethan Hein, and Dan Brown. Why hip-hop is interesting. In *Tutorials of the International Society for Music Information Retrieval Conference (ISMIR)*, New York City, USA, 2016.

[7] Gilberto Bernardes, Matthew E. P. Davies, and Carlos Guedes. Automatic musical key estimation with adaptive mode bias. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 316–320, New Orleans, Louisiana, USA, 2017.

[8] Rachel M. Bittner, Minwei Gu, Gandalf Hernandez, Eric J. Humphrey, Tristan Jehan, Hunter McCurry, and Nicola Montecchio. Automatic playlist sequencing and transitions. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 442–448, Suzhou, China, 2017.

[9] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[10] Bill Brewster and Frank Broughton. *The Record Players: DJ Revolutionaries*. Grove/Atlantic, Inc., 2011.

[11] Bill Brewster and Frank Broughton. *Last Night a DJ Saved My Life: The History of the Disc Jockey*. Grove Press, 2014.

[12] William L. Briggs and Henson Van Emden. *The DFT - an owner's manual for the discrete Fourier transform*. SIAM, 1995.

[13] Nicholas J. Bryan and Ge Wang. Musical influence network analysis and rank of sample-based music. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 329–334, Miami, Florida, USA, 2011.

[14] Mark J. Butler. *Unlocking the Groove: Rhythm, Meter, and Musical Design in Electronic Dance Music*. Profiles in popular music. Indiana University Press, 2006.

[15] Matthew W. Butterfield. Why do jazz musicians swing their eighth notes? *Music Theory Spectrum*, 33(1):3–26, 2011.

[16] Chris Cannam, Christian Landone, and Mark B. Sandler. Sonic visualiser: An open source application for viewing, analysing, and annotating music audio files. In *Proceedings of the International Conference on Multimedia*, pages 1467–1468, Florence, Italy, October 2010.

[17] Estefanía Cano, Mark Plumbley, and Christian Dittmar. Phase-based harmonic percussive separation. In *Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech)*, pages 1628–1632, Singapore, September 2014.

[18] Michael A. Casey and Malcolm Slaney. Fast recognition of remixed music audio. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2007.

[19] Jeff Chang. *Can't Stop Won't Stop: A History of the Hip-Hop Generation*. Picador, 2005.

[20] Chao Chen, Andy Liaw, and Leo Breiman. Using random forest to learn imbalanced data. Technical report, July 2004.

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

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

[23] Karen Collins. *Game sound: an introduction to the history, theory, and practice of video game music and sound design*. MIT Press, 2008.

[24] Nick Collins. Algorithmic composition methods for breakbeat science. In *Proceedings of Music Without Walls*, pages 21–23, De Montfort University, Leicester, UK, 2001.

[25] Nick Collins. Further automatic breakbeat cutting methods. In *Proceedings of Generative Art*, Milan, Italy, 2001.

[26] Nick Collins. Interactive evolution of breakbeat cut sequences. In *Proceedings of Cybersonica Symposium*, London, UK, 2002.

[27] Nick Collins. Influence in early electronic dance music: An audio content analysis investigation. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 1–6, Porto, Portugal, October 2012.

[28] Nick Collins, Peter Manning, and Simone Tarsitani. A new curated corpus of historical electronic music: Collation, data and research findings. *Transactions of the International Society for Music Information Retrieval*, 1(1):34–55, 2018.

[29] Nick Collins, Margaret Schedel, and Scott Wilson. *Electronic Music*. Cambridge Introductions to Music. Cambridge University Press, Cambridge, United Kingdom, 2013.

[30] Anne Danielsen. *Presence and pleasure: The funk grooves of James Brown and Parliament*. Wesleyan University Press, 2006.

[31] Anne Danielsen. Continuity and break: James Brown's 'Funky Drummer', 2010.

[32] Laurent Daudet. Sparse and structured decompositions of signals with the molecular matching pursuit. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(5):1808–1816, September 2006.

[33] Matthew E. P. Davies, Philippe Hamel, Kazuyoshi Yoshii, and Masataka Goto. Automashupper: An automatic multi-song mashup system. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 575–580, Curitiba, Brazil, 2013.

[34] Matthew E. P. Davies, Philippe Hamel, Kazuyoshi Yoshii, and Masataka Goto. Automashupper: automatic creation of multi-song music mashups. *IEEE/ACM Transactions on Audio, Speech & Language Processing*, 22(12):1726–1737, 2014.

[35] Matthew E. P. Davies, Guy Madison, Pedro Silva, and Fabien Gouyon. The effect of microtiming deviations on the perception of groove in short rhythms. *Music Perception*, 30(5):497–510, 2013.

[36] Matthew E. P. Davies, Adam M. Stark, Fabien Gouyon, and Masataka Goto. Improvasher: A real-time mashup system for live musical input. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 541–544, London, United Kingdom, 2014.

[37] Robert Davis. *Who Got Da Funk?: An Etymophony of Funk Music from the 1950s to 1979*. PhD thesis, Université de Montréal, 2005.

[38] Dennis DeSantis. *Making Music: 74 Creative Strategies for Electronic Music Producers*. Ableton AG, 2015.

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

[40] Christian Dittmar, Kay F. Hildebrand, Daniel Gärtner, Manuel Winges, Florian Müller, and Patrick Aichroth. Audio forensics meets music information retrieval – a toolbox for inspection of music plagiarism. In *Proceedings of the European Signal Processing Conference (EUSIPCO)*, pages 1249–1253, Bucharest, Romania, August 2012.

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

[42] Christian Dittmar, Patricio López-Serrano, and Meinard Müller. Unifying local and global methods for harmonic-percussive source separation. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Calgary, Canada, April 2018.

[43] Christian Dittmar and Meinard Müller. Towards transient restoration in score-informed audio decomposition. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, pages 145–152, Trondheim, Norway, December 2015.

[44] Christian Dittmar and Meinard Müller. Reverse engineering the Amen break – score-informed separation and restoration applied to drum recordings. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(9):1531–1543, 2016.

[45] Christian Dittmar, Martin Pfleiderer, Stefan Balke, and Meinard Müller. A swingogram representation for tracking micro-rhythmic variation in jazz performances. *Journal of New Music Research*, 47(2):97–113, 2017.

[46] Christian Dittmar, Martin Pfleiderer, and Meinard Müller. Automated estimation of ride cymbal swing ratios in jazz recordings. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, pages 271–277, Málaga, Spain, October 2015.

[47] 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.

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

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

[50] Dan Ellis. Robust landmark-based audio fingerprinting. `http://labrosa.ee.columbia.edu/matlab/fingerprint/`, 2009. Web resource, last consulted in January 2017.

[51] Daniel P.W. Ellis. Beat tracking by dynamic programming. *Journal of New Music Research*, 36(1):51–60, 2007.

[52] Ángel Faraldo, Emilia Gómez, Sergi Jordà, and Perfecto Herrera. Key estimation in electronic dance music. In Nicola Ferro, Fabio Crestani, Marie-Francine Moens, Josiane Mothe, Fabrizio Silvestri, Giorgio Maria Di Nunzio, Claudia Hauff, and Gianmaria Silvello, editors, *Advances in Information Retrieval*, pages 335–347. Springer International Publishing, 2016.

[53] Ángel Faraldo, Sergi Jordà, and Perfecto Herrera. A multi-profile method for key estimation in edm. In *Proceedings of the Audio Engineering Society (AES) Conference on Semantic Audio*, Erlangen, Germany, June 2017.

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

[55] Derry FitzGerald, Eugene Coyle, and Matt Cranitch. Using tensor factorisation models to separate drums from polyphonic music. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, Camo, Italy, September 2009.

[56] Andrew V. Frane. Swing rhythm in classic drum breaks from hip-hop's breakbeat canon. *Music Perception*, 34(3):291–302, 2017.

[57] Andrew V. Frane and Ladan Shams. Effects of tempo, swing density, and listener's drumming experience, on swing detection thresholds for drum rhythms. *Journal of the Acoustical Society of America*, 141(6):4200–4208, 2017.

[58] Anders Friberg and Andreas Sundström. Swing ratios and ensemble timing in jazz performance: Evidence for a common rhythmic pattern. *Music Perception*, 19(3):333–349, 2002.

[59] 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)*, pages 445–449, Shanghai, China, March 2016.

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

[61] Daniel Gärtner. Tempo detection of urban music using tatum grid non negative matrix factorization. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 311–316, Curitiba, Brazil, 2013.

[62] Roman B. Gebhardt, Matthew E.P. Davies, and Bernhard U. Seeber. Harmonic mixing based on roughness and pitch commonality. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, pages 185–192, Trondheim, Norway, December 2015.

[63] Roman B. Gebhardt, Matthew E.P. Davies, and Bernhard U. Seeber. Psychoacoustic approaches for harmonic music mixing. *Applied Sciences*, 6(5), 2016.

[64] 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.

[65] 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.

[66] Nikolay Glazyrin. Towards automatic content-based separation of DJ mixes into single tracks. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 149–154, Taipei, Taiwan, 2014.

[67] Daniel Gómez-Marín. *Similarity and style in electronic dance music drum rhythms*. PhD thesis, Universitat Pompeu Fabra, Barcelona, Spain, 2018.

[68] Daniel Gómez-Marín, Sergi Jordà, and Perfecto Herrera. Drum rhythm spaces: from global models to style-specifc maps. In *Proceedings of the International Symposium on Computer Music Multidisciplinary Research (CMMR)*, pages 68–79, Porto, Portugal, September 2017.

[69] 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.

[70] Siddharth Gururani and Alexander Lerch. Automatic sample detection in polyphonic music. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 264–271, Suzhou, China, 2017.

[71] Jason Hockman, Matthew E. P. Davies, and Ichiro Fujinaga. One in the Jungle: Downbeat Detection in Hardcore, Jungle, and Drum and Bass. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 169–174, Porto, Portugal, October 2012.

[72] Jason A. Hockman. *An ethnographic and technological study of breakbeats in Hardcore, Jungle, and Drum & Bass*. PhD thesis, McGill University, Montreal, Quebec, Canada, 2012.

[73] Jason A. Hockman, Matthew E. P. Davies, and Ichiro Fujinaga. Computational strategies for breakbeat classification and resequencing in Hardcore, Jungle and Drum & Bass. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, pages 337–342, Trondheim, Norway, December 2015.

[74] Aline Honingh, Maria Panteli, Thomas Brockmeier, David Iñaki López Mejía, and Makiko Sadakata. Perception of timbre and rhythm similarity in electronic dance music. *Journal of New Music Research*, 44(4):373–390, 2015.

[75] 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.

[76] Martin James. *State of bass: Jungle: the story so far*. Boxtree, 1997.

[77] Dan-Ning Jiang, Lie Lu, Hong-Jiang Zhang, Jian-Hua Tao, and Lian-Hong Cai. Music type classification by spectral contrast feature. In *Proceedings of the International Conference on Multimedia and Expo (ICME)*, volume 1, pages 113–116, Lausanne, Switzerland, 2002.

[78] Thor Kell and George Tzanetakis. Empirical analysis of track selection and ordering in electronic dance music using audio feature extraction. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 505–510, Curitiba, Brazil, 2013.

[79] Minje Kim, Jiho Yoo, Kyeongok Kang, and Seungjin Choi. Nonnegative matrix partial co-factorization for spectral and temporal drum source separation. *IEEE Journal of Selected Topics Signal Processing*, 5(6):1192–1204, 2011.

[80] Peter Knees, Ángel Faraldo, Perfecto Herrera, Richard Vogl, Sebastian Böck, Florian Hörschläger, and Mickael Le Goff. Two data sets for tempo estimation and key detection in electronic dance music annotated from user corrections. In *Proceedings of the International Society for Music Information Retrieval Conference, ISMIR 2015, Málaga, Spain, October 26-30, 2015*, pages 364–370, 2015.

[81] Clément Laroche, Matthieu Kowalski, Hélène Papadopoulos, and Gaël Richard. Hybrid projective nonnegative matrix factorization with drum dictionaries for harmonic/percussive source separation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(9):1499–1511, 2018.

[82] Chuan-Lung Lee, Yin-Tzu Lin, Zun-Ren Yao, Feng-Yi Lee, and Ja-Ling Wu. Automatic mashup creation by considering both vertical and horizontal mashabilities. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 399–405, Málaga, Spain, 2015.

[83] 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, Colorado, USA, November 2000.

[84] Simon Leglaive, Romain Hennequin, and Roland Badeau. Singing voice detection with deep recurrent neural networks. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 121–125, Brisbane, Australia, 2015.

[85] 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.

[86] 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.

[87] Henry Lindsay-Smith, Skot McDonald, and Mark B. Sandler. Drumkit transcription via convolutive NMF. In *Proceedings of the International Conference on Digital Audio Effects Conference (DAFx)*, York, UK, September 2012.

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

[89] Patricio López-Serrano, Matthew E. P. Davies, Jason Hockman, Christian Dittmar, and Meinard Müller. Break-informed audio decomposition for interactive redrumming. In *Late Breaking and Demo Session of the International Society for Music Information Retrieval Conference (ISMIR):*, 2018.

[90] Patricio López-Serrano, Christian Dittmar, Jonathan Driedger, and Meinard Müller. Towards modeling and decomposing loop-based electronic music. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, pages 502–508, New York City, USA, August 2016.

[91] Patricio López-Serrano, Christian Dittmar, Andrew V. Frane, and Meinard Müller. Estimating sixteenth note swing ratio in drum break recordings. 2018.

[92] Patricio López-Serrano, Christian Dittmar, and Meinard Müller. Finding drum breaks in digital music recordings. In *Proceedings of the International Symposium on Computer Music Multidisciplinary Research (CMMR)*, pages 68–79, Porto, Portugal, September 2017.

[93] Patricio López-Serrano, Christian Dittmar, and Meinard Müller. Mid-level audio features based on cascaded harmonic-residual-percussive separation. In *Proceedings of the Audio Engineering Society Conference on Semantic Audio (AES)*, pages 32–44, Erlangen, Germany, June 2017.

[94] Pedro Solórzano Manzano. Audio fingerprinting techniques for sample identification in electronic music. Master's thesis, Friedrich-Alexander Universität Erlangen-Nürnberg and International Audio Laboratories Erlangen, Germany, 2016.

[95] Ugo Marchand and Geoffroy Peeters. Swing ratio estimation. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, pages 423–428, Trondheim, Norway, December 2015.

[96] Taro Masuda, Kazuyoshi Yoshii, Masataka Goto, and Shigeo Morishima. Spotting a query phrase from polyphonic music audio signals based on semi-supervised nonnegative matrix factorization. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 227–232, Taipei, Taiwan, 2014.

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

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

[99] Tomohiko Nakamura, Hirokazu Kameoka, Kazuyoshi Yoshii, and Masataka Goto. Timbre replacement of harmonic and drum components for music audio signals. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7520–7524, Florence, Italy, May 2014.

[100] Cárthach Ó Nuanáin. *Connecting time and timbre: computational methods for generative rhythmic loops in symbolic and signal domains*. PhD thesis, Universitat Pompeu Fabra, Barcelona, Spain, 2018.

[101] Nobutaka Ono, Kenichi Miyamoto, Hirokazu Kameoka, Jonathan Le Roux, Yuuki Uchiyama, Emiru Tsunoo, Takuya Nishimoto, and Shigeki Sagayama. Harmonic and percussive sound separation and its application to mir-related tasks. In *Advances in Music Information Retrieval*, volume 274 of *Studies in Computational Intelligence*, pages 213–236. Springer Berlin Heidelberg, 2010.

[102] 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.

[103] 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.

[104] Maria Panteli, Niels Bogaards, and Aline K. Honingh. Modeling rhythm similarity for electronic dance music. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 537–542, Taipei, Taiwan, 2014.

[105] Maria Panteli, Bruno Rocha, Niels Bogaards, and Aline Honingh. A model for rhythm and timbre similarity in electronic dance music. *Musicae Scientiae*, 21(3):338–361, 2017.

[106] 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, October 2015.

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

[108] Ángel Faraldo Pérez. *Tonality estimation in electronic dance music: a computational and musically informed examination*. PhD thesis, Universitat Pompeu Fabra, Barcelona, Spain, 2018.

[109] Martin Pfleiderer. *Rhythmus: Psychologische, theoretische und stilanalytische Aspekte populärer Musik*. Transcript, 2006.

[110] Joseph A Prögler. Searching for swing: Participatory discrepancies in the jazz rhythm section. *Ethnomusicology*, 39(1):21–54, 1995.

[111] Esa Räsänen, Otto Pulkkinen, Tuomas Virtanen, Manfred Zollner, and Holger Hennig. Fluctuations of hi-hat timing and dynamics in a virtuoso drum track of a popular music recording. *PLOS ONE*, 10(6):1–16, June 2015.

[112] Robert Ratcliffe. A proposed typology of sampled material within electronic dance music. *Dancecult: Journal of Electronic Dance Music Culture*, 6(1):97–122, 2014.

[113] Francois Rigaud, Mathieu Lagrange, Axel Röbel, and Geoffroy Peeters. Drum extraction from polyphonic music based on a spectro-temporal model of percussive sounds. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, (ICASSP)*, pages 381–384, May 2011.

[114] Axel Röbel, Jordi Pons, Marco Liuni, and Mathieu Lagrange. On automatic drum transcription using non-negative matrix deconvolution and Itakura Saito divergence. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 414–418, Brisbane, Australia, April 2015.

[115] Bruno Rocha, Niels Bogaards, and Aline Honingh. Segmentation and timbre similarity in electronic dance music. In *Proceedings of the Sound and Music Computing Conference (SMC)*, pages 754–761, Stockholm, Sweden, 2013.

[116] Tim Scarfe, Wouter M. Koolen, and Yuri Kalnishkan. A long-range self-similarity approach to segmenting DJ mixed music streams. In *Artificial Intelligence Applications and Innovations - 9th IFIP WG 12.5 International Conference, AIAI*, pages 235–244, Paphos, Cyprus, 2013.

[117] Tim Scarfe, Wouter M. Koolen, and Yuri Kalnishkan. Segmentation of electronic dance music. *International Journal of Engineering Intelligent Systems for Electrical Engineering and Communications*, 22(3):4, 2014.

[118] Joseph G. Schloss. *Making Beats: The Art of Sample-Based Hip-Hop*. Music Culture. Wesleyan University Press, 2014.

[119] Hendrik Schreiber and Meinard Müller. A crowdsourced experiment for tempo estimation of electronic dance music. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 409–415, Paris, France, September 2018.

[120] Björn Schuller, Alexander Lehmann, Felix Weninger, Florian Eyben, and Gerhard Rigoll. Blind enhancement of the rhythmic and harmonic sections by NMF: Does it help? In *Proceedings of the International Conference on Acoustics (NAG/DAGA)*, pages 361–364, Rotterdam, The Netherlands, 2009.

[121] Prem Seetharaman and Bryan Pardo. Simultaneous separation and segmentation in layered music. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, pages 495–501, New York City, USA, 2016.

[122] Olivier Senn, Claudia Bullerjahn, Lorenz Kilchenmann, and Richard von Georgi. Rhythmic density affects listeners' emotional response to microtiming. *Frontiers in Psychology*, 8:1709, 2017.

[123] Amanda Sewell. *A Typology of Sampling in Hip-Hop*. PhD thesis, Indiana University, 2013.

[124] 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, September 2004.

[125] Jordan B. L. Smith and Masataka Goto. Nonnegative tensor factorization for source separation of loops in audio. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 171–175, Calgary, Canada, April 2018.

[126] Rick Snoman. *Dance Music Manual: Tools, Toys, and Techniques*. Taylor & Francis, third edition edition, 2014.

[127] Ragnhild Torvanger Solberg. "Waiting for the bass to drop": Correlations between intense emotional experiences and production techniques in build-up and drop sections of electronic dance music. *Dancecult: Journal of Electronic Dance Music Culture*, 6(1):61–82, 2014.

[128] Reinhard Sonnleitner, Andreas Arzt, and Gerhard Widmer. Landmark-based audio fingerprinting for DJ mix monitoring. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, pages 185–191, New York City, New York, USA, August 2016.

[129] Reinhard Sonnleitner and Gerhard Widmer. Robust quad-based audio fingerprinting. *IEEE Transactions on Audio, Speech, and Language Processing*, 24(3):409–421, 2016.

[130] Alexander Stewart. 'Funky Drummer': New Orleans, James Brown and the rhythmic transformation of American popular music. *Popular Music*, 19(3):293–318, 2000.

[131] 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.

[132] Hideyuki Tachibana, Takuma Ono, Nobutaka Ono, and Shigeki Sagayama. Melody line estimation in homophonic music audio signals based on temporal-variability of melodic source. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 425–428, Dallas, Texas, USA, 2010.

[133] Timothy D Taylor. *Strange sounds: Music, technology and culture*. Routledge, 2014.

[134] Nao Tokui. Massh!: a web-based collective music mashup system. In *Proceedings of the International Conference on Digital Interactive Media in Entertainment and Arts, DIMEA*, pages 526–527, Athens, Greece, 2008.

[135] 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.

[136] Christian Uhle, Christian Dittmar, and Thomas Sporer. Extraction of drum tracks from polyphonic music using independent subspace analysis. In *Proceedings of the International Symposium on Independent Component Analysis and Blind Signal Separation (ICA)*, pages 843–847, Nara, Japan, April 2003.

[137] Jan Van Balen. Automatic recognition of samples in musical audio. Master's thesis, Universitat Pompeu Fabra, Barcelona, Spain, 2011.

[138] Jan Van Balen, Joan Serrà, and Martín Haro. *From Sounds to Music and Emotions: 9th International Symposium, CMMR 2012, London, UK, June 19-22, 2012, Revised Selected Papers*, chapter Sample Identification in Hip Hop Music, pages 301–312. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[139] Len Vande Veire and Tijl De Bie. From raw audio to a seamless mix: creating an automated dj system for drum and bass. *EURASIP Journal on Audio, Speech, and Music Processing*, 2018(1):13, 2018.

[140] Richard Vogl and Peter Knees. An intelligent drum machine for electronic dance music production and performance. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 251–256, Copenhagen, Denmark, 2017.

[141] Avery Wang. An industrial strength audio search algorithm. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 7–13, Baltimore, Maryland, USA, 2003.

[142] 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.

[143] Brian C. Wesolowski. *Testing a Model of Jazz Rhythm: Validating a Microstructural Swing Paradigm*. PhD thesis, University of Miami, Miami, Florida, USA, 2012.

[144] Jordan L. Whitney. Automatic recognition of samples in hip-hop music through non-negative matrix factorization. Master's thesis, University of Miami, 2013.

[145] Wikipedia. Exponential smoothing — wikipedia, the free encyclopedia, 2017. [Online; accessed 10-January-2017].

[146] Karthik Yadati, Martha Larson, Cynthia C. S. Liem, and Alan Hanjalic. Detecting drops in electronic dance music: Content based approaches to a socially significant music event. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 143–148, Taipei, Taiwan, 2014.