Literary Review of Content-based Music Recognition Paradigms

Emmi Siitonen

University of Tampere
Faculty of Natural Sciences
Computer Science
Master's thesis
Supervisor: Martti Juhola
15.6.2018

University of Tampere
Faculty of Natural Sciences
Emmi Siitonen: Literary Review of Content-based Music Recognition Paradigms
Master's thesis, 48 pages
June 2018

Abstract

During the last few decades, a need for novel retrieval strategies for large audio databases emerged as millions of digital audio documents became accessible for everyone through the Internet. It became essential that the users could search for songs that they had no prior information about using only the content of the audio as a query. In practice this means that when a user hears an unknown song coming out of the radio and wants to get more information about it, he or she can simply record a sample of the song with a mobile device and send it to a music recognition application as a query. Query results would then be presented on the screen with all the necessary meta data, such as the song name and artist. The retrieval systems are expected to perform quickly and accurately against large databases that may contain millions of songs, which poses lots of challenges for the researchers.

This thesis is a literature review which will go through some audio retrieval paradigms that allow querying for songs using only their audio content, such as audio fingerprinting. It will also address the typical problems and challenges of audio retrieval and compare how each of these proposed paradigms performs in these challenging scenarios.

Keywords: music, content-based retrieval, approximate matching, music retrieval, music recognition, audio fingerprinting

Contents

1 Introduction

Traditionally, when searching for a song, users would search the database using some already known information, such as the song title or name of the artist, in order to find what they are looking for. A typical query would usually include the name of the artist or the song and the system would then respond by listing songs from this particular artist or which had similar titles.

In order to query a song based on this type of textual metadata information, the user is required to have some prior knowledge about the song he/she is looking for. In other words, the content must be at least somewhat known to the user in order for such search to be successful. As millions of digital audio documents became accessible to everyone through the Internet, a need for novel retrieval strategies for large audio databases emerged. It became essential that the users could search for songs that they had no prior information about using only the content of the audio as a query. In practice this means that when a user hears a song coming out from a radio in a bar, café or a car, and wants to get more information about it, he or she can simply record a sample of the song with a mobile device and send it to a music recognition application as a query. The application would then in return display the search results on the screen with additional information about the song title, artist, release year and other release information. Two popular applications for mobile devices made for solving this challenge are Shazam [Shazam, 2018] and SoundHound [SoundHound, 2018].

The content-based audio retrieval queries are performed using a short snippet of audio, usually only a few seconds long, and the documents that are similar enough to the query snippet are returned to the user. The notion of similarity, i.e. what actually counts as "being similar enough", can alter from high-level of similarity to lower-level of similarity depending on the used approach. The high-level of similarity audio retrieval is often referred to as "audio identification" and the lower-level as "audio matching". Audio identification refers to cases where only exact matches are searched for whereas audio matching can refer to scenarios where the original song is searched for using a live version or when the user wants to find all different performances and versions of a certain song. This can be developed even further by searching for songs that are performed by a certain artist, have the same genre or any other similarities with the query song. Audio identification is a task that has already been largely solved, due to its straightforward "find-me-this-and-exactly-this" type of nature and current state-of-the-art systems such as Shazam are able to retrieve a queried song only

in a few seconds. Audio matching however is still missing adequate solutions that could meet its more challenging performance requirements. This is because the lower we define the similarity level, the more complicated the retrieval task becomes. Audio retrieval systems are expected to perform fast and accurately against large databases that may contain millions of songs. Many approaches have been proposed to overcome the challenges, but the trade-off between scalability and accuracy always exists and the challenge is to find a perfect balance between accuracy and computation times.

It is important to note that it is not computationally reasonable to compare two audio files directly with each other in order to determine their similarity. Even slight audio compressions or distortions caused by noise will alter the audio encodings in a way that makes the direct comparisons insufficient, even though the compressed audio is still aurally similar to the original audio. Instead, compact representations are used to represent the songs in a way that each song has a unique signature, usually referred to as a fingerprint, that is robust against noise and other distortions. When two signatures are similar, it is highly probable that the two songs corresponding to these signatures are also similar if not even identical. The aim is to create signatures that are unique enough that not too many search results are created, but generic enough that it can still find matches despite serious distortions and alterations. This type of balance between specificity and approximate matching is a difficult challenge. Also, audio retrieval queries are often performed using mobile devices and therefore it is crucial that the signatures created are as small as possible in order to reduce network latency. This also means that the length of the audio sample required for a successful match should be relatively short (e.g., less than 10 seconds).

This thesis will go through some query-by-example paradigms that allow searching for songs using only their audio content as a query. It will also address the typical problems and challenges of audio retrieval and compare how each of these proposed paradigms performs in these challenging scenarios. Other possible use cases for audio retrieval include automated commercial detection and broadcast monitoring, but these are not included in this thesis. There are also many audio retrieval techniques specified for speech and other non-music audio detection, but these, too, will be out of this thesis' scope. Although terms such as audio and sound will be used during the thesis, in this context they will be meant to stand for musical audio instead of all audio in general unless otherwise mentioned.

The subject of audio retrieval is wide and lots of research has been done on it which is why it was not possible to include everything in this thesis. The

choices about what to include or exclude were done based on my best judgment and therefore some significant methods may have been excluded unintentionally. The references used in this thesis were found from Google Scholar different keywords, e.g. "audio retrieval", "audio fingerprinting", "music recognition" and "content-based recognition". Due to the large amount of parameters and changing requirements, the comparison of audio retrieval system is not always very straightforward. Comparing the performance evaluations of two different systems is insufficient if the parameters and used datasets aren't the same or even in the same scale, which makes the method evaluation a challenging task.
Section 2 will explain some of the basic characteristics of music while Section 3 goes through some of the basic characteristics and main challenges that audio retrieval paradigms must overcome in order to produce successful queries. Sections 4 and 5 focus on paradigms created for audio identification tasks, whereas Section 6 will introduce some approaches suggested for audio matching and version identification. Finally, Section 7 will conclude the thesis and present some thoughts about the future of audio retrieval applications.

2 Elements of music

Music is a complex, multi-dimensional phenomenon that has no simple, singular definition for its structure or style. Each song may consist of melody, harmony, rhythm, texture and other elements of music, and each of these elements has unique characteristics and structures that change depending on a song. In order to understand some of the principles of audio recognition and retrieval, it is good to know some basic properties of music itself. The following chapter explains some of the very basic musical terms in more detail and many of them are utilized in the methods described in later chapters. [HowMusicWorks.org, 2018].
Musical features can be split roughly into two groups: acoustical features and thematic features [Hsu *et al.*, 1998], and both of these groups are used for music retrieval tasks depending on the use case. Acoustical features include, for instance amplitude, pitch, and tone and the following list describes some of the musical features related to them more closely.

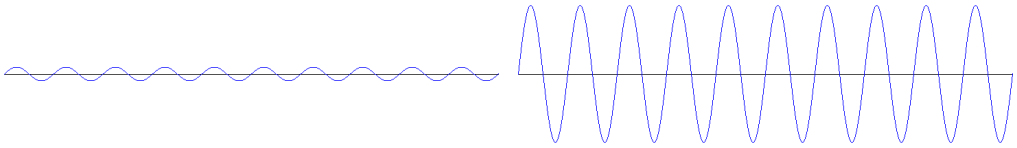
- **Amplitude:** Size of the vibration of sound and therefore also how loud the sound is. See examples in Figures 2.1 (a) and (b).

- **Frequency and Pitch:** Frequency is the speed of the sound vibration and it determines the pitch of the sound, i.e. how low or high the sound is. The higher the frequency the higher the sound. Typical human hearing range is from 20Hz to 20,000Hz, but with age the high frequency limit usually lowers. See examples in Figures 2.1 (c) and (d).

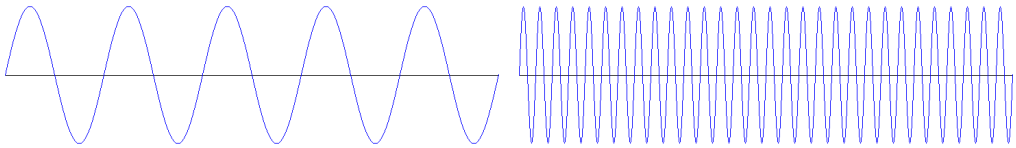
- **Tone:** Different instruments have their own distinct sound called tone. Tone is a combination of different but harmonic frequencies playing simultaneously, mixing up and creating the distinctive sound of an instrument. See example showing the different tones of a guitar and piano compared with a sine wave in Figure 2.2.

- **Pitch class and Chroma:** A set of all pitches that are an integral number of octaves apart is called a *pitch* class. For example, the pitch class C includes all of the Cs in all of the octaves. The tone height of the pitch tells which octave the pitch belongs to. *Chroma* is a quality of pitch, and all the pitches inside a certain pitch class share the same Chroma. Some pitches that are dictated differently still sound the same. For example, C, B♯, and D𝄫, still belong to the same pitch class because they have the same Chroma.

Figure 2.1 Examples of low and high amplitude and frequency



(a) Low amplitude (b) High amplitude



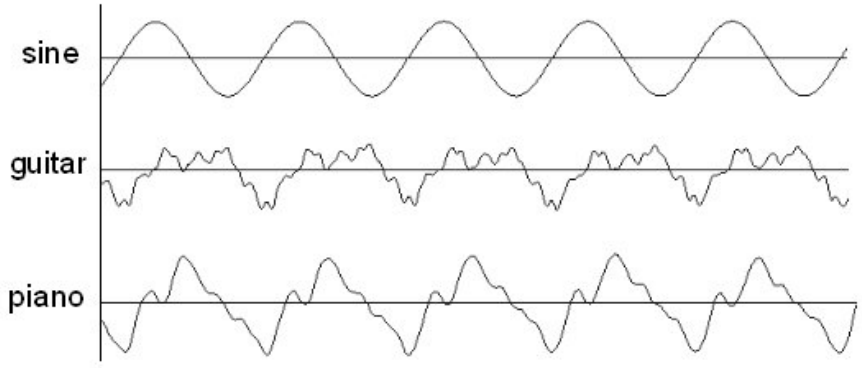
(c) Low frequency (d) High frequency

For human input queries the acoustical features are usually level too low to be used for song querying [Hsu *et al.*, 1998]. Instead, thematic features such as rhythm and melodies are much more accessible to humans, i.e. through tapping, singing and humming. The following list briefly describes some of the most basic thematic features: rhythm, melody and chords. It is worth to note that all of these features can also be presented in string-format, which will be further discussed in Section 5.

- **Rhythm:** The sequence of note durations in the theme. For example "1/2 - 1/2 - 1/2 - 1/2 - 2 - 1/2 - 1/2 - 1/2 - 4", can also be thought as the timing of events over time.

- **Melody:** Combination of pitch and rhythm and the musical line of notes that listeners most often perceive and recognize as a single recognizable and forward moving element of the song.

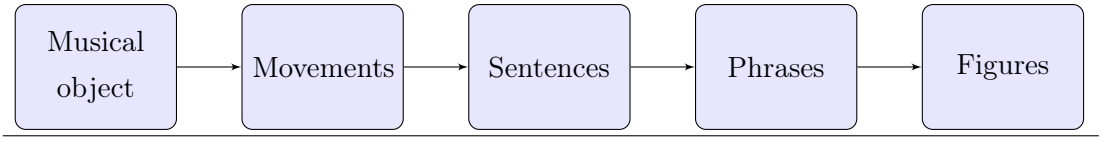
Figure 2.2 Examples of different instrumental tones, [Invisibles, 2014]



- **Chord:** Three or more notes played simultaneously, as a basis for harmony. In western music, most commonly chords are *triads* which are chords with three distinct notes: root, third and fifth.

A very general way to structure a song is to split it into Intro, Verse, Chorus, Bridge, Instrumental and Outro [Maddage *et al.*, 2004]. More universal way to compose musical objects is to follow a structure referred to as the *musical form*. Musical form follows two basic rules: *hierarchical rule* and *repetition rule* [Hsu *et al.*, 1998]. The hierarchical rule means that musical objects follow a specific hierarchy as shown in Figure 2.3. A music object consists of movements, movements consist of sentences, sentences consist of phrases and phrases consist of figures. Figures are the shortest parts in music, short successions of notes. The repetition rule refers to a well-known property of music: music consists of sequences of notes that repeat throughout the music. These repeating sequences are also known as motives. For example, in western music repeating sequences traditionally occur regularly in refrains. Repetition is considered as one of the universal characteristics of music and several researchers have proposed to use only repeating patterns to represent songs in a more compact way instead of using whole musical objects.

Figure 2.3 Musical form of a musical object, [Hsu *et al.*, 1998]



It is not fully understood how human brain encodes music so that a human listener

is capable of recognizing songs and their very different version. Some research has been made in order to discover how experienced musicians transcript music [Hainsworth & Macleod, 2003]. Also some knowledge has been gained about the human brain's sensitivity (or insensitivity) to certain melodic changes [Serra *et al.*, 2010]. The melody seems to be one of the characteristics that is mostly preserved in different versions of the same piece of music. This is why many researches suggest using the melodic properties of music to robustly identify songs and their covers and other versions.

The retrieval task of musical objects is complicated because of the complex nature of music and the demanding performance requirements of the task itself. The retrieval application has to be capable of extracting the most interesting elements of the song, i.e. keeping the most descriptive features while discarding noise and other irrelevant sounds from the audio signal. Determining which parts of the song are the most interesting ones in different scenarios is not a very trivial task. For example, different genres have their own distinguishable characteristics. This is especially true in audio matching where the goal is to find songs that are similar to the queried song according to some measure, instead of simply identifying the song. It is worth to note that the definition of similarity does not only change according to musical aspect because sometimes even contextual or personal aspects can affect what parts of the song are considered meaningful.

3 Music retrieval

3.1 General

A human listener is capable of effortlessly recognizing and identifying a previously known song successfully even after the audio has gone through severe alterations. A person can even recognize a song which is performed by a different performer or which has been changed into something completely new by turning it into a remix or a mix with other songs and new sounds. Sometimes recognizing a song is hard, i.e. when somebody tries to whistle a song that has been stuck in their head for their friends. If the whistling is accurate enough, human ear tends to pick up the familiar tune although connecting the tune to a known song and recalling the song name might still be tricky. Some of the main relevant factors for why humans can recognize and identify thousands of familiar songs are the capacity of a human brain (how many songs a person can remember) and how the brain extracts the most characteristic features of a song and matches them with the songs in the memory [Gutiérrez & García, 2015], but how the human brain exactly handles musical similarities, is still somewhat unknown [Serra *et al.*, 2010].

Music retrieval systems aim to solve the problem of recognizing a previously unknown song using only its musical content as a query by combining the way humans recognize music to more detailed and technical musical comparisons. The audio retrieval task can be split into three different types of retrieval scenarios, audio identification, audio matching and version identification [Grosche *et al.*, 2012] and the use case mostly defines how much the system has to mimic human recognition to achieve wanted accuracy. In *audio identification*, the aim is to recognize an unknown song by retrieving songs that have exactly the same original audio signal as the queried song, i.e. the version of the queried song is the same as the original and it is not for example a remix or a performance made by another artist. This is because even though they are just different versions of the same song, their audio signal encodings are all very different from each other. In practice this means that the queries are performed by recording a sample of the audio with a mobile device and sending that sample to the recognition service. Some alterations are tolerated, as long as the actual audio content is still the same. For example, compressions and other alterations that keep the audio audibly intact are quite well handled in the current audio identification systems, such as Shazam. Shazam is a popular audio retrieval system that was

first founded in 1999 [Wang, 2003]. According to Shazam approximately 150 million people use the service each month and each day 20 million searches are performed by the users [DMR Business statistics, 2018].

In the literature, there are several types of music recognition and retrieval methods. One of the very early methods were so called string-based retrieval methods, which transformed the audio signal into strings that could then be further processed with modified string searching techniques [Liu *et al.*, 1999]. String-based retrieval techniques will be discussed more in Section 5. One of the weaknesses of string-based retrieval techniques is that the music transcription problem, i.e. the transformation of the audio signal into discrete note/chord representations is still a challenge that has not been very successfully solved [Hainsworth & Macleod, 2003]. Also a musical score rarely represents the original song well enough and for example instrumentation and the singer's vocal style are not readable from a musical score. This is why other music recognition methods utilize techniques like audio fingerprinting, where the spectral features of the audio are used to represent the audio instead of simple string representations. Audio fingerprinting will be discussed more in Section 4.

It is not always enough to have the ability to recognize songs based on a recorded sample. Sometimes user might have a song stuck in his/her head for days and have no source from which to record a sample for the recognition service. The user might also be in a concert, listening to a live version of a song and wishing that he/she could record that performance and use it as a query for discovering the name of the song. "Query-by-tapping" and "Query-by-humming" are two different types of queries for *audio matching* tasks, in which the user tries to search for the song either by tapping the rhythm or humming the melody of a song, respectively. Audio matching and a very similar technique called *version identification* both use approximate matching instead of exact matching to find audio documents that are not exactly the same as the queried song but similar enough to count them as matches. In [Serra *et al.*, 2010] researchers display a list of common types of song versions such as remixes and live performances.

Unlike with audio identification, audio matching problems are still missing adequate solutions because of the variety of different use cases and the complexity of determining two non-identical audio files as similar [Grosche *et al.*, 2012]. Version identification is very much similar to the audio matching, but its restrictions are much looser and usually in version identification the system is expected to return more than one result.

Both the audio matching and the version identification include use cases which

are hard to address because in both cases the audio files can go through an endless number of changes and alterations, and there is no way to know how much the query signal differs from the original signal. Most users are really poor at humming a song at a right tempo or pitch, and even with experienced users there is no way to know beforehand how much and what kind of errors should the querying be prepared to handle, especially, if the goal is to find only one result instead of the n most similar. Also, when the similarity measures change continuously according to the situation it becomes problematic to know what should be stored in the music database. Should the database include several kinds of compact representation that can each be used for queries of different kinds? How to know what kind of representations should be included and is it even reasonable to cover all the necessary combinations? Questions like this together with the ones presented in the next chapter are the main reasons why audio retrieval is a challenging and complicated problem and why audio matching is still a task that is missing proper solutions. Other use cases include the automatized detection of audio files that include copyright protected content. The challenges include overcoming issues such as background noise and audio distortions due to bad recording quality, in addition to many intentional changes such as tempo and pitch alterations and different versions of the same song. More detailed descriptions of these challenges are given in Section 3.2.

A typical audio retrieval process can also be split into four parts. First one is the database creation part, which is an offline process that has no strict time limits and will only be computed in rare occasions. The second part includes the query generation, which is a much more time-critical procedure that is performed each time a new query is made. The third part includes finding the candidate matches for the queried song from the database and returning n most similar audio documents as a query result. It is important to note that it is mandatory that the same methods are used in both first two parts of the retrieval process. Otherwise the matching in phase three would be impossible [Wang, 2003]. The fourth and the last phase of the process involves going through the candidate matches and finding the best match. Another way to partition the retrieval process is presented by [Serra, 2011]. According to Serra, many current state-of-the-art systems can be divided into five parts: feature extraction, key invariance, tempo invariance, structure invariance and similarity computation. This division is further discussed in Section 6.

Audio recognition systems involve a great amount of parameters that have an effect on the computation times and storage requirements of the queries, and

therefore also define the final performance and quality of the system. Haitsma et al. [2010] presented a few questions that have to be addressed before building up an audio recognition system. In their research paper, Haitsma et al. only referred to audio fingerprinting systems, but based on literature these questions are valid for other types of audio retrieval systems too.

1. **Feature selection**. Determining which features are actually 'perceptually important' is a challenge in itself and the answer depends highly on the retrieval context. The context changes quite a lot when dealing with different music genres. For example, the distinctive features of western popular music are quite different than those of classical music or jazz. Perceptual similarity is also known to be non-transitive [Haitsma & Kalker, 2010]. In other words, even if a pair of X and Y and another pair of Y and Z are perceptually similar, it does not necessarily mean that the objects X and Z are also perceptually similar. However, the transitivity rule can be reapplied if the audio representations are modeled from the beginning based on the perceptual information of the audio files. This is why for example the fingerprints in audio retrieval systems try to capture the perceptually important features of audio as well as possible.

2. **Fingerprint representation**. In what format should the audio files be represented in the database? When creating an audio fingerprint, what should the resulting fingerprint look like? One option is to utilize vector representations, i.e. a vector of real numbers where each number represents a weight of a certain perceptual element. Another option is to use binary hashes that represent the audio signal using only ones and zeros. The latter usually works better, since comparing real numbers that represent element weight requires real additions, subtractions and sometimes even multiplications, resulting in too cumbersome and heavy similarity computations.

3. **Granularity**. In most cases, the query represents only a small part of the original song that is being searched for. Retrieval systems should be able to identify a whole song based on only a short snippet of audio, usually less than 5 seconds long, which can also be located anywhere in the original audio file. This problem is usually addressed with the overlapping of extracted audio segments. One segment may not contain enough information to identify a song, but by combining several of them, it is possible to reach a fairly reliable identification, assuming that the representations themselves

are good enough. In audio fingerprinting a granularity measure and a basic unit that contains enough information to identify an audio sample is called a fingerprint-block.

3.2 Challenges

The complex nature of music is not the only source of challenges for music retrieval systems. A wide spectrum of challenges coming from many external sources needs to be addressed in order to produce good querying results. This section will describe some of these challenges while explaining why they are critical when dealing with the music retrieval tasks.

3.2.1 Noise & Distortions

When recording a sample for the query, the recorder is prone to pick up sounds that do not belong to the queried song, for example speech, laughter, wind and other music. Especially in public places such as cafés and bars the query recordings tend to suffer from huge amounts of additional sounds. For example, the recording quality of mobile devices is usually also very poor. The audio signal goes through lots of compression and degradations while picking up lots of white noise. White noise is a combination of different, equally intense frequencies which together create a mask for the audio signal, covering much of the meaningful information. The audio encoding of the end result will end up being vastly different from the original audio track.
Music retrieval systems can ignore much of the additional noises by filtering out small frequency intensities from the original audio signal by using, for example, a well-defined threshold value, i.e. keeping only the most interesting values while discarding small values that are more often than not noise. One such threshold value is used in Shazam [Wang, 2003], as described later in Section 4.3. Big values can be considered interesting in this context because they represent large frequency intensities in an audio signal and therefore tend to be the most audible and distinguishable.

3.2.2 Mobile devices

Mobile devices introduce a set of challenges to music retrieval applications due to their limited memory and computation capacity and network issues. Retrieval applications should be fast enough to provide users with pleasant and interactive

experiences so the querying should be quick and feel nearly instantaneous. This means that in order to avoid network latency, the size of the data generated for querying should be as small and compact as possible. Similarly the length of the audio sample required for successful retrievals should be as short as possible when querying with mobile devices, e.g. <10 seconds. The longer the audio sample recorded, the greater the amount of fingerprints generated is.

3.2.3 Parameters

Majority of music recognition techniques involve a set of parameters that need to be high-tuned in order for the algorithms to produce good results. Tuning the parameters requires professional knowledge and research. Therefore the algorithms are usually presented with values that have already been tested and validated, but the problems arise when researchers try to combine existing techniques to other techniques or when trying to develop systems that can switch for example from one specificity level to another. It is not guaranteed that the same parameter values will bring similar results when the algorithms are altered. Rather, in many cases even a slight change in the parameters may cause the algorithm to produce false results. Also, as the number of parametrized variables increases, the amount of possible parameter combinations skyrockets. For example, in [Baluja & Covell, 2006] the researchers tried over 50,400 different parameter combinations when executing experiments for their suggested Wavelet method. 122 of these parameter combinations reached a retrieval accuracies of 97.5% or more.
Due to the vast number of possible parameter combinations, it is not feasible for humans to enter them manually to the systems even during development and research. Instead, for example evolutionary computation has been proposed for finding the perfect value combinations [Gutiérrez & García, 2015] more efficiently. Machine learning is also a field that could be utilized for this purpose.

3.2.4 Copyright protection & designed modifications

Websites such as YouTube and other multimedia platforms where users can freely post content have been dealing with copyright protection issues for years. Various file sharing networks have cost music and movie industries billions in economic losses through piracy and illegal file sharing [Gupta *et al.*, 2010]. According to [Ouali *et al.*, 2015] in March 2015 YouTube claimed on their website that every single minute 300 hours of videos are uploaded to the service. Due to this vast amount of data huge efforts have been made so that copyright violations could

be detected as easily and automatically as possible.
As a counterattack, copyright violators make designed modifications to the audio content so that they would be harder to detect. Anguera et al. [2012] defined these transformations as any alterations to the original audio signal that changes the physical characteristics of the signal while keeping the audio still aurally intact, i.e. keeping the audio similar enough that the human-ear can't really hear any difference between the original audio signal and the modified version. Such transformations include MP3 encoding, sound equalization and adding of noises or external signals.

4 Audio fingerprinting

Most spectral-based audio identification applications utilize *audio fingerprints* as a more compact representation of whole audio files. Here audio fingerprints are defined as unique signatures that allow representing the original audio signal in a very compact format. In other words, a fingerprinting function F should map an audio object X with a varying length of n, to a much smaller format that has a more limited length of m. Usually $m = 32$ bits is used. After construction audio fingerprints are stored in the database together with necessary metadata, from where these fingerprints can then compared with the query fingerprint to find the most similar songs from the database. The key idea is that highly similar audio files should produce highly similar fingerprints whereas highly different audio files should produce highly different fingerprints. [Haitsma & Kalker, 2010].
Some of the guiding principles for creating fingerprints are that they should be temporally localized, translation-invariant, robust and also small and entropic enough. Descriptions for these principles are given below. It is also crucial that the fingerprints in the database and in the query are constructed in an identical manner. [Wang, 2003].

1. **Temporal locality**: only audio samples that are near the corresponding point in time should be used to create a fingerprint hash. This way distant events do not affect the hash.

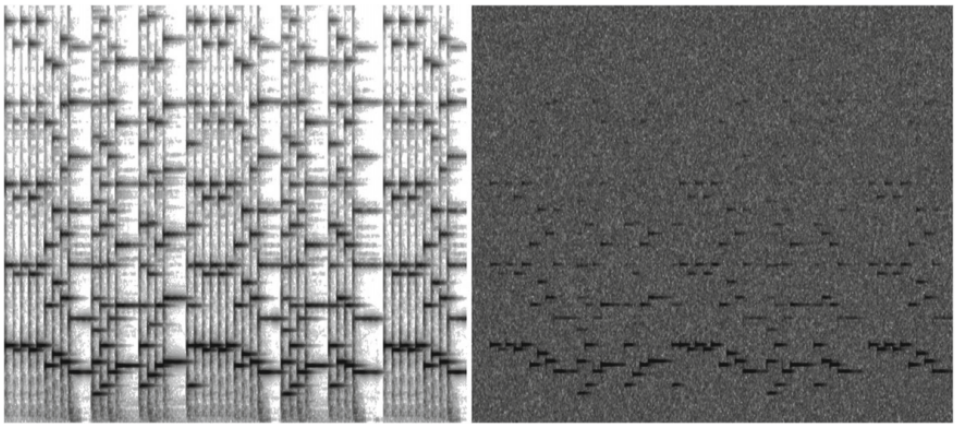
2. **Translation-invariant**: Fingerprint hashes are reproducible regardless of the position inside the audio. This is because the audio sample can be from any point of the original audio.

3. **Robustness**: Noise and other degradations shouldn't affect the reproducibility of hash. In other words hashes created from a clean audio database should be very similar to those created from a degraded copy.

4. **Size**: The amount of memory a fingerprint requires should be as small as possible in order to ensure that comparing two fingerprints together can be performed as fast as possible.

5. **High entropy**: Fingerprints should be unique enough in order to minimize the probability of false positives, i.e. the number of false identifications. However, it is worth noticing that too much entropy may lead to non-reproducibility of fingerprints.

Figure 4.1 *Fur Elise* spectrogram [Gutiérrez & García, 2015]



During the past few decades, several proposals have been presented for constructing adequate fingerprints. Some of them are not very robust against noise or severe audio transformations and some of them do not scale very well to large audio databases, making the querying too slow. The following sub-chapters present a few of the most researched and well-performing audio identification paradigms found in the literature. These paradigms share lots of similarities, since most of them are optimized or further developed versions of each other. The similarities are described here first, starting from the overall structure of the fingerprint generation.

Audio fingerprint construction.

An audio signal is first transformed into a spectrogram using a Short-Term Fourier Transform (STFT). A spectrogram is a visual representation of an audio signal with three dimensions: time, frequency and intensity. The time is represented by the X axis and the frequency corresponds to the Y axis. The intensity value of a specific point corresponds to the intensity of that particular frequency in that point in time. Figure 4.1 shows an example spectrogram of *Beethoven's Fur Elise* together with a noise degraded version on the right. [Gutiérrez & García, 2015]. A reduced frequency range is selected to reduce the number of spectrogram's frequency bins. For example, in [Ouali *et al.*, 2016] a frequency range from 500 Hz to 3000 Hz is split across 257 frequency bins, but also fewer bins with a smaller range (e.g. 33 bins with a range going from 300 Hz to 2000 Hz) have been used [Baluja & Covell, 2007]. More bins means more detailed shape of the signal and therefore also a more unique and descriptive spectrogram image. The

frequency range is limited because the frequencies falling outside the range are usually aurally insignificant to human hearing and therefore redundant to the retrieval process.

Spectrogram can be seen as multiple overlapping audio-images or frames. The frame width which is also referred to as a windows size and the overlapping ratio are definable parameters. The large overlapping of successive frames ensures that there is only a little variation in sub-fingerprints over time [Haitsma & Kalker, 2010]. This is useful because the given audio sample (e.g. audio recording) can come from any part of the audio file and with slowly variating sub-fingerprints the probability of mismatches caused by unlucky alignment is minimized.

In many cases, perceptually the most interesting features are in the frequency domain, and depending on the used approach a set of features is computed for each frame. The aim is that the selected features are robust against noise and unique enough so that the audio sample can be recognized based on them. An ideal fingerprint should be able to identify modified songs, regardless of the severity of the transformations while balancing with acceptable computation times.

Because the direct comparisons of the spectrogram frames are inefficient and unsuitable for audio matching, much more compact representations to each audio segment are needed. The compact representation of one single frame is often referred to as a sub-fingerprint and together the sub-fingerprints form the fingerprint of the song. The way the fingerprints are exactly constructed differs depending on the used approach. Instead of spectrograms, also chromagram representations have been proposed, which will be further discussed in Section 6.

The actual retrieval process changes depending on the used approach, but the main idea is that the database is first queried for candidate matches, i.e. fingerprints that fulfill some defined similarity measures and which can therefore be considered as the most likely matches for the queried song. After this, the candidate matches are further investigated to eliminate the false positive matches and find the fingerprint(s) that match the queried song the best. A so called "time alignment step" can be executed to see if the found matches are consistent over time or just individual hits from arbitrary locations. Techniques used for this step include RANSAC [Ke *et al.*, 2005] and Dynamic Time Warping (DTW) [Gionis *et al.*, 1999]. If enough candidate matches that map to the same audio file pass the defined requirements and thresholds, the respective audio file is returned as a query result to the user.

In many ways, audio fingerprinting resembles hashing functions which are well known and widely used in cryptography. Just like fingerprints, hashing too en-

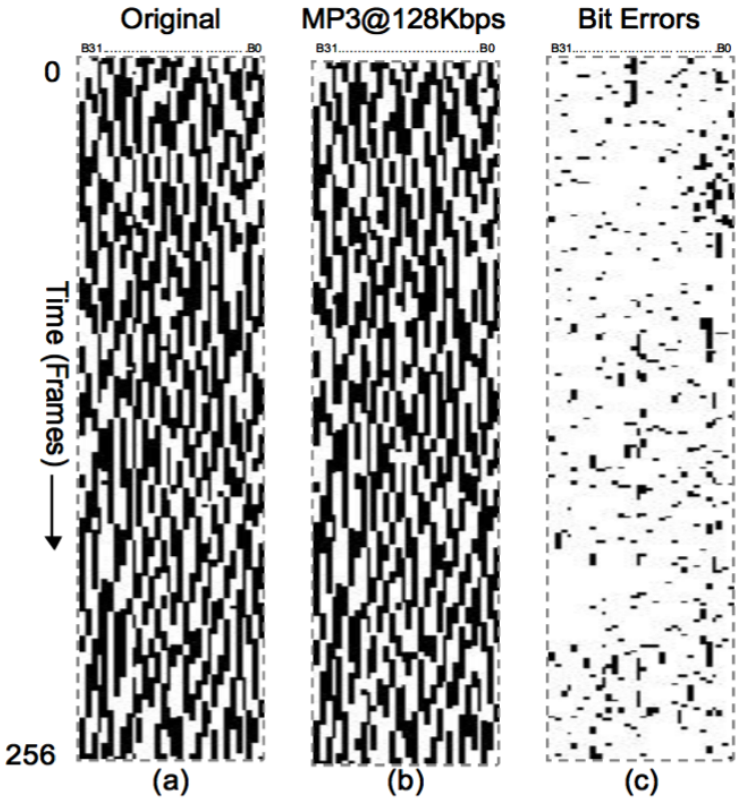
ables that two large objects A and B can be compared with each other by comparing their respective hashes. In cryptography hashing functions map large objects such as e-mails into much smaller hash values, which are used to keep the hashed messages hidden in a way that they can still be transformed back into their original form. Hashing in audio fingerprinting works pretty much the same, but cryptographic hash functions have qualities that make them inefficient for audio hashing. First, cryptographic hashes are highly sensitive to changes, i.e. when only one bit changes in the original file the resulting hash changes drastically, making it impossible to declare two items similar based on their fingerprints. This is of course crucial for cryptography, but for audio identification it is unacceptable. Second, as mentioned in earlier chapters, audio matching most often requires matching two audio files based on their perceptual similarity instead of their mathematical similarity, which also makes cryptographic hashes insufficient for audio comparisons. [Haitsma & Kalker, 2010]

4.1 Haitsma et al. & Ke et al. - Spectrogram features in audio fingerprinting

The approach proposed by Haitsma et al. [2010] uses highly overlapping frames in order to achieve a highly robust retrieval system. The fingerprints are created by segmenting the audio signal first into frames of length 0.37 beginning every 11.6 milliseconds. This means that the overlapping factor of one frame is 31/32 (0.37s / 32 $\approx$ 11.5ms). The high overlapping of sub-fingerprints (compact representations of frames) minimizes the risks of unlucky misalignment of fingerprints, i.e. the situations where the query sample's beginning does not exactly align with the fingerprints in the database. Because the sub-fingerprints are varying slowly overtime they are still very similar to the sub-fingerprints of the same audio clip in the database, even in the worst case scenarios. A single fingerprint block consists of 256 sub-fingerprints, which means that the granularity of the system is about 3 seconds (11.5ms $\times$ 256 = 2944ms).

For each of the extracted frames, the most interesting features are computed by performing a Fourier transform on each frame. The frequency spectrum is also divided into 33 non-overlapping frequency bands ranging from 300Hz to 2000Hz, leaving out frequencies that are not audible to humans. The end result is a set of two-dimensional binary images. The bit derivation process is described in more detail in [Haitsma & Kalker, 2010]. These binary images, which are now also fingerprint blocks, can be compared with other fingerprint blocks fairly easily

Figure 4.2 (a) A fingerprint block of an original audio file, (b) fingerprint block of a compressed version, and (c) the difference between a and b shown as bit errors. [Haitsma & Kalker, 2010]



and the resulting bit error images, as shown in Figure 4.2, are the final similarity measure between the two fingerprint blocks. The bit errors (the differences between the two blocks) are shown as black pixels.

The method proposed by Haitsma et al. [2010] was further improved by Ke et al. [2005] by using a known ensemble machine learning method called AdaBoost, which is short for Adaptive Boosting. The most valuable insight provided by Ke et al. was that when an audio signal is represented as a time-frequency image it can also be processed like a regular image. This allows using widely used computer vision techniques for audio signal processing.

In [Ke *et al.*, 2005] the authors first trained the AdaBoost classifiers with box-filters, which are very popular in face detection. The training for classification is executed by using a set of audio samples and their corresponding noise degraded versions. Each classifiers outputs a binary value as a response. Also, instead of

manually creating suitable filters, Ke et al. apply machine learning techniques to manufacture a large set of filters from which only the best performing ones are selected for the final filter set. The generated filter set includes a default filter set, described in detail in [Ke *et al.*, 2005], with varying bandwidths, frequency bands and time-width to ensure good coverage and the best possible performance. Filters with larger bandwidths and time-width perform better against different types of distortions while the filters with shorter bandwidths and time-widths are better at preserving discriminative information. In the example described by Ke et al. the candidate filter set included approximately 25,000 filters, from which M discriminative filters and a threshold value are picked for generating M-bit vectors to represent audio segments. These vectors are called *descriptors*. The goal is to generate descriptors that enable us to declare that two audio snippets originate from the same position of the same song. The classifier for this could then be something similar to $H(x_1, x_2) \rightarrow y = \{-1, 1\}$ where x_1 and x_2 stand for the two spectrograms. y denotes if two images originate from the same audio source ($y = 1$) or not ($y = -1$).

During retrieval, similarity search is performed for each query descriptors against the database. According to Ke et al. [2005] direct indexing produced so good results that there was no need for more advanced similarity measures, i.e. for Locality Sensitivity Hashing. The idea is that the database is first queried for the near-neighbors of Hamming Distance of 0. After this, the procedure is repeated for finding matches that are within the Hamming Distance of 1 and finally within the Hamming Distance of 2. This approach may first seem unreasonably time-consuming and inefficient, but according to Ke et al. computing similarities this way was much faster than using techniques like LHS. Other advantage of direct indexing is also that it returns exact results instead of approximations.

Finally, once the near-neighbors have been found, the best match is searched for using RANSAC. RANSAC iterates through the possible time alignments and estimates the likelihood that the query signature comes from the same origin as the candidate signature using an EM score. The temporal alignment process of this method is described in more detail in [Ke *et al.*, 2005].

Despite promises, according to the evaluations made by Chandrasekhar et al. [2011] the system proposed by Ke et al. does not perform too well with short audio snippets. In order to produce accurate matches, queries had to be over 10 seconds long, which is not ideal. Also because the system relies highly on the AdaBoost classifiers the performance is very vulnerable if there is much mismatching between the training and test data. Because of the broad range of possible dis-

tortions and levels of noise, training a set of AdaBoost classifiers for each of these changing scenarios is not very practical in the end. Anguera et al. [2012] point out that the system also uses fixed step sizes which is computationally ineffective. Mobile devices have restricted memory and computation capacities, so the amount of data they should need to process and store is much smaller than for example in server-based solutions where it is more desirable to store as many fingerprints as possible.

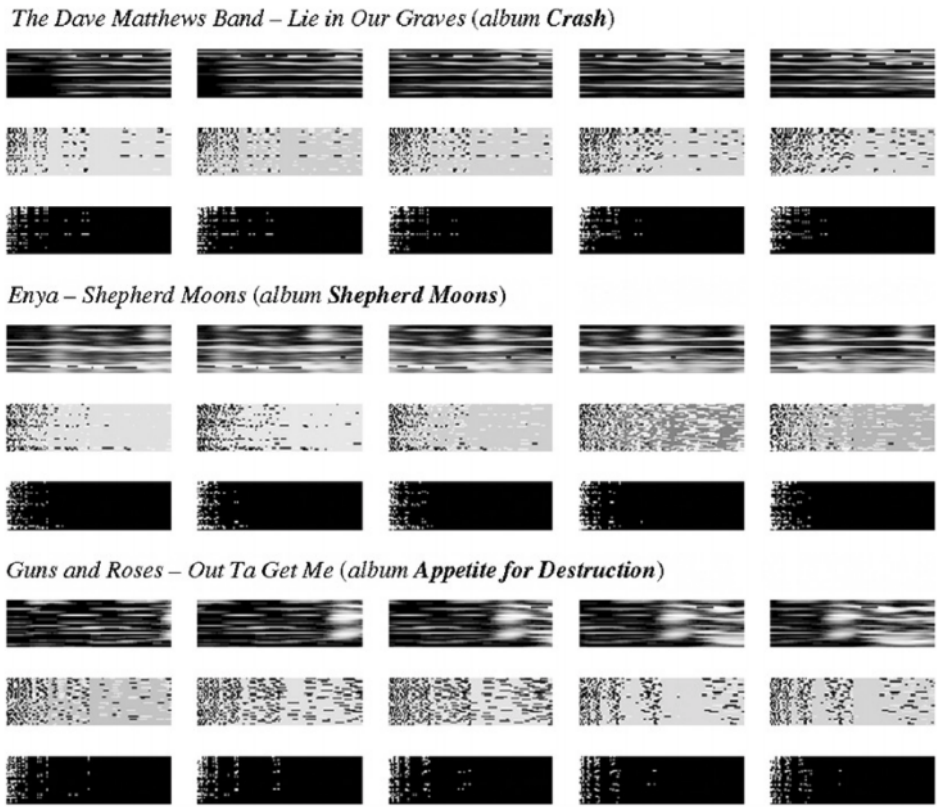
4.2 Baluja & Covell - Wavelets in audio fingerprinting

Baluja and Covell [2006] also proposed applying image retrieval techniques to audio retrieval, but instead of using a learning approach like [Ke *et al.*, 2005], they propose using wavelet-based audio fingerprints. The wavelet-based fingerprints are also referred to as *"waveprints"* by the authors.

In audio retrieval context wavelets are used to extract information from an audio signal. According to Baluja and Covell [2006], wavelets are mathematical tools which are used for the hierarchical decomposition of functions. With them, a function can be described by its overall shape, with varying details. Wavelets have been successfully used in image retrieval tasks and since time-frequency images of audio can be processed as images, wavelets are also suitable for audio retrieval tasks. In experiments done in [Jacobs *et al.*, 1995] the wavelets produced better retrieval results than direct pixel comparisons or histograms when using hand drawn low quality sketches as queries. A good description of a wavelet method can be found in [Stollnitz *et al.*, 1995].

Similar to processes described in Section 4.1 the waveprint creation begins by converting the audio signal into spectrograms. Again, the audio is sliced into highly overlapping frames of length 0.37 seconds starting every 11.6 milliseconds. The frequency range is also reduced to 32 frequency bins ranging from 300Hz to 2000Hz. From the spectral images, top Haar-wavelets are extracted based on their magnitude. The idea is that instead of directly comparing two images with each other, the images are first decomposed using multi-resolution Haar wavelets. This process is described in more detail in [Stollnitz *et al.*, 1995]. In the system described by [Baluja & Covell, 2007], a wavelet signature is created for each spectrogram image. Wavelets aren't resistant to noise and distortions on their own and when a spectrogram image of size $m \times n$ is turned into wavelets there is no compression, $m \times n$ wavelets are returned [Baluja & Covell, 2006]. Instead, the idea is to utilize only the top-t wavelets for each image instead of

Figure 4.3 Three songs represented with five consecutive frames. For each song there are three different representations illustrated: the original spectrogram image, the wavelet magnitude image and the final top-t (t=200) wavelet image. [Baluja & Covell, 2008]



all of them. Top wavelets are much robuster against noise since they include the most distinctive features of the audio as illustrated in Figure 4.3.

Baluja and Covell also mention that one very important discovery made by Jacobs et al. [1995] is that instead of full coefficients only sign bits are needed for each top wavelet. In other words, each top wavelet could be represented using only two bits, either as 10 (positive values) or 01 (negative values). Wavelets that do not belong to the set of top-t wavelets are labeled with 00. The end result is a sparse bit vector that can then be further reduced using a method called *Min-Hash* which is a technique used for computing compact sub-fingerprints from the sparse wavelet-vectors. The aim is that two sub-fingerprints will be highly similar only if the corresponding wavelets are also highly similar.

When comparing two wavelet-vectors together we can refer to four different match

Figure 4.4 Match types of two binary vector bits. [Baluja & Covell, 2006]

Type	Vector 1	Vector 2
a	1	1
b	1	0
c	0	1
d	0	0

types: a, b, c, and d, as shown in Figure 4.4. Given two vectors v_1 and v_2, these two vectors are compared together and depending on the corresponding bits we define match types for each of these bits. For most bit positions the match type will be of type d (bits in both vectors are zeroes). Baluja and Covell define the similarity of two vectors to be the relative amount of a-rows from the other, non-zero rows as shown in Equations (4.1) and (4.2). As an example, two vectors that have the similarity of 8/10 are more likely more similar to each other than vectors with a similarity of only 2/10.

$$Sim(v_1, v_2) = a/(a + b + c) \tag{4.1}$$

which is equal to

$$Sim(v_1, v_2) = (v_1 \cap v_2)/(v_1 \cup v_2) \tag{4.2}$$

In the Min-Hash technique, the wavelet's bit positions are reordered into some random but known order. For each vector, the position of the first occurrence of '1' is calculated for that particular permutation. This process is performed multiple times, each time with a new permutation. Let p be the amount of times the procedure has been repeated and the number of unique permutations. The end result is then p independent descriptions of the bit vector. For a large enough p, the amount of exact matches in signatures indicates similarity between the two original vectors. The resulting Min-Hash computed signatures are then represented as p 8-bit integers which are also the final sub-fingerprints.

So far, the amount of data has been reduced with three different steps. First, only top-t Haar-wavelets were selected for further processing, eliminating most of the noise. Second, these selected top wavelets were reduced to only two bits based on the finding presented by Jacobs et al. [1995]. Third, Min-Hash technique was used to reduce the bit-vectors to p values, which then formed the final sub-fingers. In order to make the matching process faster and more efficient, Baluja and Covell used Locality-Sensitive-Hashing (LSH). Even though the amount of information

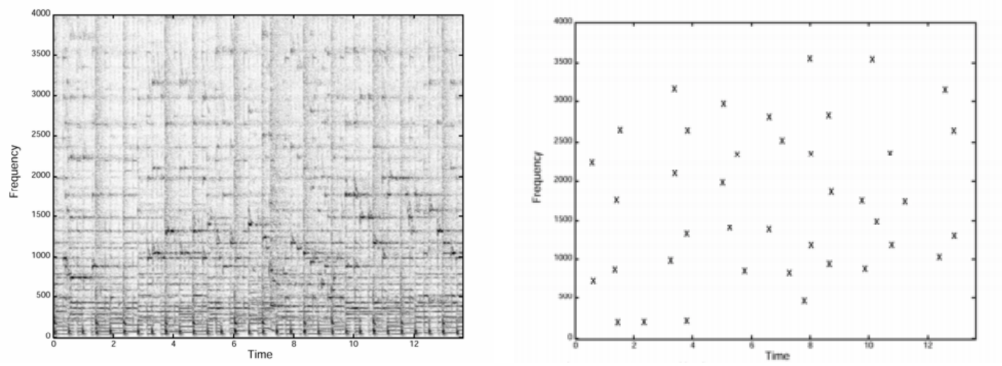
is reduced quite a lot until this point, it is still not computationally trivial to find near-neighbors in p dimensional space [Baluja & Covell, 2008]. At this point of the process, each spectrogram is represented using p 8-bit integers. LSH performs a set of hashes and each of them addresses only a small part of the sub-fingerprint and each point is hashed into separate hash tables. The idea is to use several hash functions for hashing so that the probability of collision is more likely to happen for items that are similar and less likely for those that are not [Gionis *et al.*, 1999]. During retrieval, these hash tables are queried to find sub-fingerprint segments that match the queried song the best. Only sub-fingerprints that have more than i matches (a defined threshold value) are kept for further investigation. For the remaining sub-fingerprints the final similarity measure can be calculated using a simple Hamming distance. Finally, the temporal alignment problem is addressed by using Dynamic Time Warping (DTW) and leaving out the temporal information during the match voting. A good description can be found in [Gionis *et al.*, 1999].

In [Covell & Baluja, 2007], the researchers evaluated the temporal ordered Waveprint by comparing its results with direct spectrogram comparisons. The accuracy of both techniques was not only very high (>98% accuracy) but also surprisingly similar to each other. Since the direct comparisons of spectrogram as computationally so expensive, the Waveprint approach is naturally more preferable. Waveprint also has advantages when compared with other techniques, such as the methods proposed by Ke et al. (see Section 4.1). For example, the generated Waveprint sub-fingerprints represent longer time periods, and based on the evaluations made by [Chandrasekhar *et al.*, 2011], the fingerprinting method proposed by Baluja and Covell requires shorter queries than those of Ke et al. or Wang (see Section 4.3). However, the system proposed by Baluja and Covell is computationally much more expensive and contains more bits per fingerprints than the other proposed methods. This emphasizes the difficulty of finding a balance between matching accuracy, granularity and resource requirements.

4.3 Wang - Spectrogram peaks in audio fingerprinting

Instead of using large and complete spectrogram images, Wang [2003] proposed that only spectrogram peaks should be used. There are several factors that support this idea: First, using only spectrogram peaks instead of whole spectrogram images saves lots of storage resources. Secondly, spectrogram peaks are very robust against noise and are most likely not going to be hidden or buried by

Figure 4.5 Example spectrogram and constellation map from same audio snippet. [Wang, 2003]



(a) Spectrogram (b) Constellation map

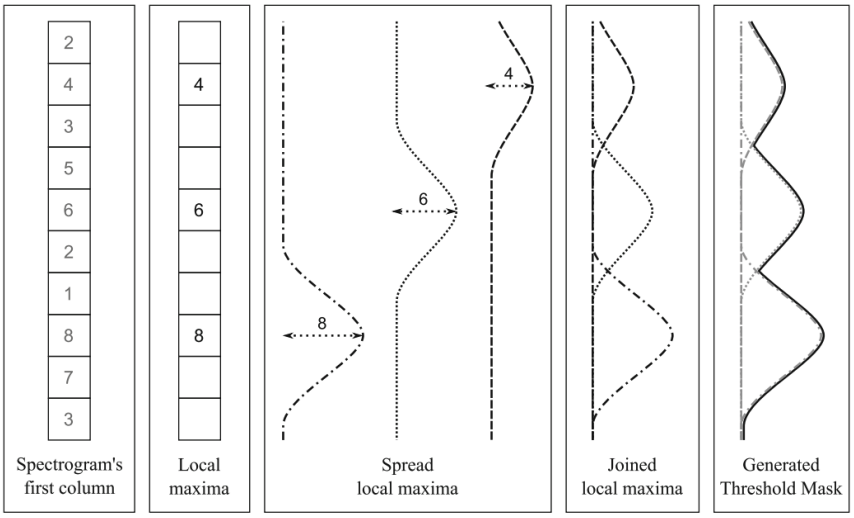
ambient noise. Thirdly, spectrogram peaks satisfy the property of linear superposition. This method proposed by Wang is often also referred to as the Shazam method, as the popular audio recognition service called Shazam uses this method.

Figure 4.5 (a) shows a regular three-dimensional spectrogram. In Figure 4.5 (b) it has been reduced to a much simpler representation, where only a set of individual points are displayed as a constellation map. Note that the image is now two-dimensional, as the amplitude property has been eliminated.

Spectrograms can be seen as two-dimensional images where each (x, y) point represents the frequency intensity with a varying color of gray. The constellation map extraction, also known as candidate peak selection, is achieved by selecting a set of points from the original spectrogram. This is done by running a threshold mask through the spectrogram image and selecting only points that are above the threshold value, ignoring the rest.

The process of peak selection aims to keep up a constant and uniform density along the spectrogram, and a Gaussian spread is used for generating a threshold mask. In Figure 4.6, the threshold mask generation is described graphically. For each spectrogram column, local maximas are extracted (points that are greater than the adjacent neighbors) and then joined together to create a threshold mask. The mask generation process is described in detail in [Gutiérrez & García, 2015], but the idea is that the threshold mask is generated from the first spectrogram column and then ran over each column individually, first from left-to-right and then again, after regenerating the mask from the last column, from right-to-left. Only points that are selected as candidate points in both the backward and

Figure 4.6 Generation of a threshold mask, [Gutiérrez & García, 2015]

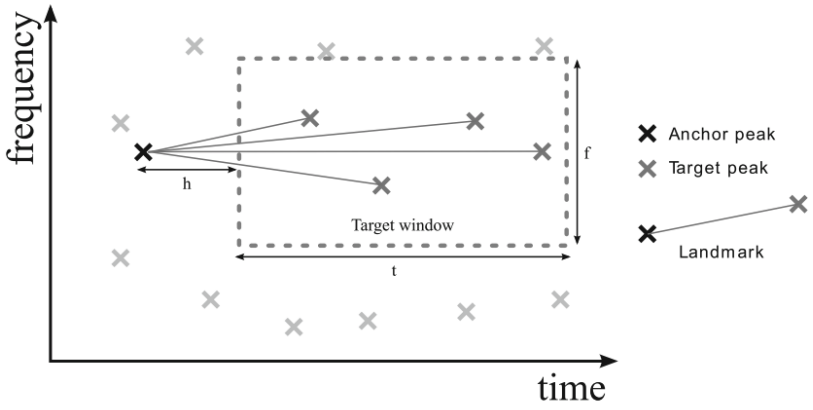


forward computations are selected as final peak points. It is worth to note that every time the mask is moved to the column, all of its values are multiplied by a decay rate $\lambda \in (0, 1)$ in order to make sure that the mask does not get stuck on high values and is able to keep selecting new peaks. The length of the threshold mask is the same as the height of spectrogram, so that it can be easily moved along the x axis.

When two constellation maps, one coming from the original audio signal and the other from the query, are placed together and a significant amount of points coincide and overlap, it indicates that it is highly likely that the maps originate from the same source and that a match has been found. However, simply comparing constellation maps together like this is a very slow and computationally expensive process. Instead, Wang [2003] proposes a fast indexing technique for the constellation maps where pairs of peaks are used for creating landmarks that are much more distinctive than individual points. According to Wang the specificity of hashes generated using point pairs is a million times greater than those of constructed using only individual points, due to the additional 20 extra bits coming from using two points instead of one. Using pairs instead of individual points also saves a lot of computation time, and the losses in the probability of accurate signal detection are insignificant due to the huge savings in storage and computation times.

The following paragraph explains the landmark generation process presented by [Gutiérrez & García, 2015]. A target window with specific size is used to link the

Figure 4.7 The target window, [Gutiérrez & García, 2015]



peak to the points closest to it. The height and width of the window represent frequency and time, respectively. Let $p_1, p_2, ..., p_n$ be the constellation map peak points. Let t and f be the width and height of the target window, as shown in Figure 4.7. Moreover, h is a hop value, which is the gap between the p_i and the most left side of the target window. Furthermore, p_i can also be referred to as *an anchor point*. A group of landmarks F (fingerprint) is then constructed as follows:

1. Sort the peaks in ascending order according to their x coordinates
2. For each p_i
   (a) Let (x_i, y_i) be the coordinates for p_i
   (b) For each point p_j where $x_j > x_i$
      i. If $x_j < x_i + t + h$ and $y_j < y_i + f/2$ and $y_j > y_i - f/2$, then the pair (p_i, p_j) is added to the F.

Because a landmark includes two peaks which both have two (x, y) coordinates, we can calculate the distance measurement between them. Because it desirable to make the final fingerprints as compact as possible, the information about the landmarks should also be compressed as much as possible while holding on to the capability of recreating them. In addition, each landmark needs to be linked with an indexable hash that is as unique as possible. If (t_1, f_1) and (t_2, f_2) are two peak coordinates, we get their differences as $\delta t = |t_2 - t_1|$ and $\delta f = |f_2 - f_1|$. The final landmark hash is constructed from $f_1, \delta t$ and δf and stored in the database

Figure 4.8 An example of a Shazam hash table, [Gizmodo, 2010]

Frequency in Hz	Time in seconds, song information
823.43	53.352, "Song A" by Artist 1
823.44	34.678, "Song B" by Artist 2
823.45	108.65, "Song C' by Artist 3
. . .	. . .
1892.31	34.945, "Song B" by Artist 2

together with t_1 and needed metadata about the song. Figure 4.8 shows an example image about how for example Shazam stores its fingerprints in hash tables. As shown in the image, frequency acts as the key [Gizmodo, 2010].

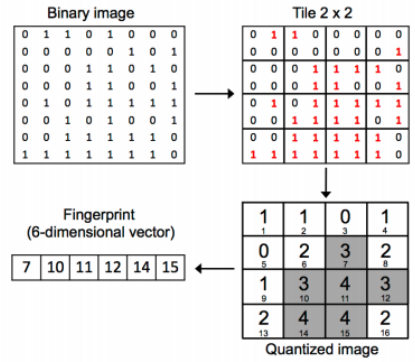
For audio identification purposes the Shazam algorithm has a retrieval accuracy of near 100% as shown in the experiments performed by Williams et al [2017]. Also according to [Chandrasekhar *et al.*, 2011] the Shazam fingerprint is the most compact when comparing with Waveprints [Baluja & Covell, 2006] or fingerprints based on the methods proposed by Ke et al [2005]. One of the disadvantages of the Shazam fingerprint is that the comparisons aren't done directly in the binary format because of the large number of bits used, but instead those are first converted into natural numbers.

4.4 Other techniques

In [Anguera *et al.*, 2012] the researchers propose a fingerprinting method called Masked Audio Spectral Keypoints (MASK) which also relies on the resilience of the spectral peaks. The idea is that the audio signal is first transformed into a spectral domain and then again into Mel-scale which is divided into 18 Mel-bands. The spectral transformation does not differ much from the methods used for example in Section 4.1 and is therefore not discussed more in here.

Salient points are selected from the Mel-scale. Similar to the peak points used in the Shazam method [Wang, 2003] the salient points are the frequency points whose energy is greater than in the adjacent points. Salient points are more resistant to noise and other audio transformation, and also more descriptive and distinct than points with low energy. It is worth to note that a salient point cannot be positioned in the bottom or top-most band in the frequency domain. With 18 bands, this means that only 16 bands are used for salient point selection. According to Anguera et al. [2012] this does not have any significant effect on

Figure 4.9 The feature extraction of binary images, [Ouali *et al.*, 2015]



the information gain quality.

A threshold value is defined in order to reduce the number of selected salient points. For each selected salient point, regions of interest are created around them by grouping some spectral values using a superimposed mask. The selected regions can overlap because one region usually consists of values that have some similar characteristics and which can therefore be grouped together. The final fingerprints are formed by comparing the averaged energies of some certain regions and creating the final fingerprint (a binary descriptor) of a certain length based on these comparisons. The first 4 bits of the fingerprint represent the location of the salient point. More detailed description of the fingerprint construction can be found in [Anguera *et al.*, 2012]. The idea is that the final fingerprints can then be indexed into a hash table as hash keys. The final matching process is very similar to that of Haitsma et al [2010] and will therefore be skipped here.

Compared with Shazam system, which also uses spectral peaks for its fingerprints, the MASK fingerprints are more localized and compact. This is because MASK encodes individual points instead of point pairs while using fewer fingerprints to represent audio files. Also the parameters used can be modified to make the system suitable for a range of different kinds of requirements.

Another approach proposed by Ouali et al. [2015] also uses binary spectrogram images to construct its fingerprints. These binary images are quantized by applying a tile of a specific size to it and each binary image is converted into n-dimensional vectors as described in the following paragraph. The construction of the n-dimensional vectors can be graphically seen in Figure 4.9. In it there is a binary image of size 8×8 and a tile of size 2×2 is applied to it, creating a divided matrix of size 5×5. For each of these small squares, the sum of '1' points

are calculated and the locations of the square values greater than the threshold value are stored in the n-dimensional vector, which is in this case 6-dimensional. This generated vector is the final fingerprint. The matching of fingerprints is performed using a nearest neighbor search and Manhattan distance as described in [Ouali *et al.*, 2015].

In other research, Ouali et al. [2016] propose a method where the search process is accelerated using the graphics processing units (GPU's). The idea is that computations could be performed in parallel, speeding up the process significantly.

5 String-based audio retrieval

During the late 90's string-based audio matching techniques were widely studied and researched. The idea was that once the music objects were transformed into a string format, regular string searching processes could be used to find similarities between two songs [Liu *et al.*, 1999], because as mentioned in Section 2, melodic shapes and contours can be represented as strings. Thoroughly studied string matching methods were therefore naturally considered as an attractive option for audio retrieval. For example one of the very first proposed methods was a technique proposed by Ghias et al. [1995], where music was represented using only three symbols: "U", "D", and "S", where the letter represented when the note was higher, lower, or the same as the previous one, respectively. However, music objects are unfortunately much too complex for direct comparisons and the representation method proposed by Ghias was too rough and simplified. Music has special characteristics that should be taken into consideration when strings like chord strings are being processed. For example, according to *root note rule*, which is also described in [Liu *et al.*, 1999], some notes in a chord are more significant than others, and should therefore weigh more whenever chords are being processed. Also a *harmonic property* of music defines that some notes sound better or more similar together than others, making some note combinations more attractive and similar during processing.

Assume that the music database has two music objects $S1$ and $S2$ whose melody strings are "sol-mi-fa-do-re-mi-re-do" and "sol-fa-fa-do-re-re-mi-do" and the query melody string Q is "do-re-mi". The task is now a substring matching task, where the aim is to find out if $S1$ or $S2$ contain the query melody string Q. There are several well-known string searching algorithms developed for solving this problem. Two of the most important ones are the Knuth-Morris-Pratt algorithm (also known as the KMP algorithm) and the Boyer-Moore algorithm, both of which are exact string matching algorithms. In other words errors between the queried string and the strings in the database are not being tolerated. In reality this is an unreasonable requirement, since even experienced singers tend to make some errors when inputting a query either by singing or humming, and amateur users make them even more. According to Chou et al. [1996] the four most common types of input faults are *duplication*, *elimination*, *disorder*, and *irrelevancy*.

- **Duplication**: A single note gets input more than once. "do-mi-sol" –> "do-do-mi-sol"

- **Elimination**: A single note that should have been input was missed. "do-mi-mi-sol" –> "do-mi-sol"

- **Disorder**: The order of the notes is different from what it was supposed to be. "do-mi-sol" –> "do-sol-mi"

- **Irrelevancy**: Irrelevant notes are input. They serve no purpose. "do-mi-sol" –> "do-mi-re-sol"

One input query can include several same or different kinds of input faults. According to Chou et al. a moderate amount of input faults should be tolerated, since it is not reasonable to expect users to input error-free queries. The exact string matching task now becomes an approximate string matching task, where the task is to find all substrings that are similar enough to the query string within some specific threshold value. Traditionally approximate string matching algorithms use a property called *editing distance* as a similarity measure. An editing number is defined as a minimum number of changes needed to make the two string identical. For example, if we have two strings "ABCD" and "AABCE", the editing distance is two, because in order to make two strings identical two changes have to be performed in the second string: remove the second letter 'A' and change the letter 'D' to 'E'. However, editing distance alone is not enough as a similarity measure for music retrieval. In order to address problems presented by rules like the root note rule and harmonic property, the retrieval system needs to support modified similarity measures that are more specific to music objects. As an example, as described in [Liu *et al.*, 1999], all of the substrings "do-mi", "re-mi" and "do-re" of a string "do-mi-sol-sol-re-mi-fa-fa-do-re-re-mi" are similar to a melody string "do-re-mi" with the editing distance of one, and the three substrings may this way first seem equally similar. However, according to root note rule and the harmonic property as described earlier, note "do" is harmonically more significant than "re" and therefore substrings "do-mi" and "do-re" are also harmonically more similar to "do-re-mi" than the substring "re-mi".
Chou et al. [1996] used chords to represent music as strings. Like mentioned in Section 2, chord is a combination of three or more notes which sound together in harmony. This means that chords are a much more compact way of representing music than individual notes. Chords are also naturally quite tolerant to input faults. For example, measures "do-mi-sol" and "do-mi-mi-sol-sol" would both be represented as a "C" chord, because they both sound similar. So the chord rep-

resentation tolerates missing and duplications of similar sounding notes. Figure 5.1 lists a few of the most frequently-used chords.

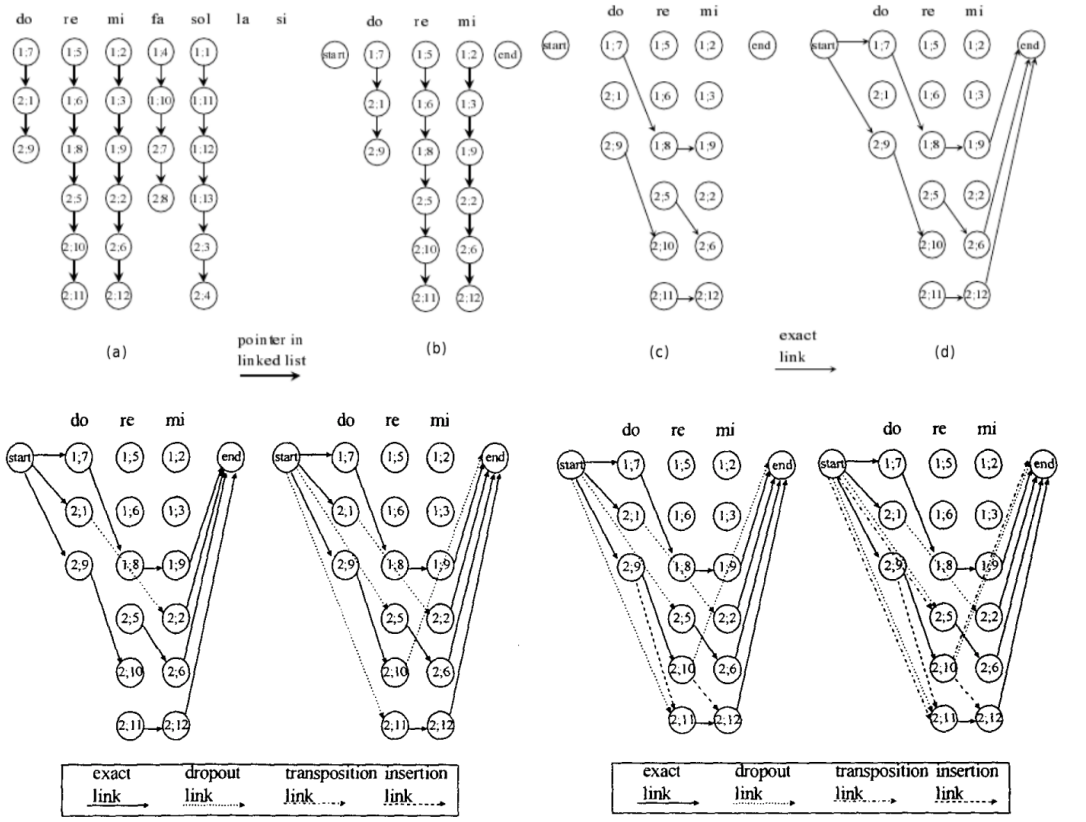
Figure 5.1 Some popular chords. C, F, and G are minor chords and Dm, Em, and Am are major chords.

C: do-mi-sol
Dm: re-fa-la
Em: mi-sol-si
F: fa-la-do
G: sol-si-re
Am: la-do-mi

In [Chou *et al.*, 1996] a set of user-input notes are divided into measures and the chord decision algorithm is then invoked for each measure. There are many different ways to make chord decision, and the paper by Chou et al. presents one of them. A chord decision algorithm simply takes a note representation of music and turns it into chord strings. The resulting strings are then indexed into a suffix tree for storing. One option is to use a PAT-tree, but according to Chou et al. it can get quite large and complicated after several insertions and deletions, so the researchers proposed a tree called B+Tree, which is another modified version of a PAT-tree. The idea is that in order to search for an answer the whole tree does not have to be traversed. Only the first substring and its neighbor buckets need to be checked. The advantage of a PAT-trees is that they allow an unstructured search of chord-strings, which is essential because the query can come from any point in a song and not just from the beginning. The actual matching process is not described in the paper.

In 1999, Liu et al. proposed a more generalized version of an approximate string matching technique where the music strings are extracted from the original music and then stored inside linked lists. Figure 5.2 shows the creation process of these linked lists, the top row for exact string matching and the bottom for approximate string matching purposes. In these linked lists, each node corresponds to a certain note and inside these nodes the note locations are stored in $(x; y)$ format, which corresponds to the x-th note of y-th melody string in the database. Start and end nodes are also added in the beginning and end of the linked list as shown in Figure 5.2 (b). The difference between the exact and approximate string matching

Figure 5.2 String matching example, the top row for exact matching and the bottom row for approximate matching. [Liu *et al.*, 1999]



scenarios is that in the latter, in addition to exact links, dropout, transposition and insertion links are added to the linked lists too. For example, in the Figure 5.2 (d), we can see that the notes "do" and "mi" occur successively in positions $(2;1)$ and $(2;2)$, meaning that there is a dropout error (note "re" is missing) in that particular position. A dropout link is created between these two elements to denote the dropout error. The creation of approximate linked lists is described more in [Liu *et al.*, 1999].

In [Hsu *et al.*, 1998], the researchers utilize the universal property of music, repetition, to discover the most interesting parts of the song. The algorithm introduced in the paper finds and extracts all the repeating patterns found in the musical strings and store them instead of the whole string representation. This of course saves storage resources and benefits from the high probability that the queried piece of music includes some of these repeating patterns. On the contrary, if the query string does not include any repeating patterns, the querying fails.

Brute-forcing the finding of repeating patterns is not computationally reasonable. According to [Hsu *et al.*, 1998] the total complexity of the calculation could in the worst case be $O(n^4)$. This would mean that if a music object consisted of 1000 notes, the algorithm would have to make $O(10^{12})$ comparisons in order to find all repeating patterns. To make the process more efficient, Hsu et al. used a correlative matrix. The idea is that if the i-th and j-th note in the string are the same, the element in the i-th row and j-th column will be set to one. If the $(i+1)$-th and $(j+1)$-th notes are also the same, this means that there is a repeating pattern in this location, and the element in $(i+1)$-th row and $(j+1)$-th column is set to $n+1$ when the n is the value in element (i, j). The value of element inside the correlation matrix tells the length of the found repeating pattern. A very thorough and detailed example can be found in [Hsu *et al.*, 1998].

Even though string-based audio retrieval systems were widely researched throughout the 90's, these paradigms have largely been superseded by more modern retrieval paradigms such as fingerprinting as described in Section 4. String-based retrieval paradigms had many challenges that proved to be much too severe to reach retrieval applications that were accurate and powerful enough.

One of these major issues in string-based audio matching is that accurate polyphonic transcription is still today a largely unsolved problem [Hu *et al.*, 2003]. In other words, software that can accurately turn an audio signal into discrete notes still does not exist. The task is difficult even for humans, and only very experienced and skilled humans are capable of performing the task successfully. The difficulty lies within the complexity of the musical context itself as explained the Section 2. The system would have to determine the style and rhythm of the song, recognize the different instruments used (which can not even be detected from chord-string or sheet music in general) and even perform tasks that require greater understanding about the music domain, such as song and chord structure extractions. Even simply identifying a singular note from an audio signal is a big challenge. [Hainsworth & Macleod, 2003]

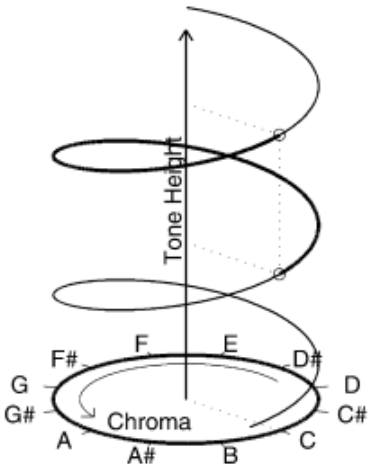
6 Audio matching and Version identification

Unlike audio identification, audio matching and version identification tasks are still missing adequate solutions that are both accurate and computationally reasonable enough. One of the main reasons is the frequently changing notion of similarity, based on which the system retrieves the most similar songs from the database. Because the notion of similarity can change from identical performance to "the same song performed by a different artist" and "any song with a similar theme" the number of parameters involved in the computation have to alter too. This means that in order to successfully retrieve queries of very different kinds, the system has to be highly reconfigurable and flexible. Audio matching systems have to extract the most characteristic features which are more descriptive to the underlying piece of music instead of any particular recording, i.e. ignore the characteristics such as instrumentation. In copy detection, the aim is also to recognize songs that have been intentionally modified to have a different encoding while keeping the audio aurally intact. In practice this means situations where the user is going to upload a song into a website and he/she modifies it first in order to avoid avoid copy detection. These modifications include pitch shifting and tempo changes.

Many retrieval paradigms suggested by the researchers implement their systems using similar techniques as described in Section 4 together with chroma-based features, which are sometimes referred to as pitch class profiles. Definitions for pitch class and Chroma were given in Section 2. In the early 1960s, Shepard [1964] described that the structure of a pitch is best described with two dimensions instead of just one. He also said that a helix is more appropriate for representing how humans perceive pitch than any one-dimensional lines. Figure 6.1 is an illustration of a pitch perception helix which has two dimensions: vertical for the tone height of the pitch and the angular for Chroma. As the pitch of the musical note increases, it moves higher in the helix while rotating through the different pitch classes. This means that when the pitch has traversed through the helix for a full circle, it will be exactly one octave higher or lower than before moving.

A very thorough and one of the most complete cover song recognition systems is described in [Serra, 2011]. According to Serra, current state-of-the-art systems can be divided into five different parts: feature extraction, key invariance, tempo invariance, structure invariance and similarity computation. The following list explains each of these steps briefly and introduces some suggested methods for handling them. For more detailed explanations, see [Serra, 2011]. A very good

Figure 6.1 Shepard's pitch perception helix where the angular dimension is the chroma and the vertical dimension is the tone height. [Bartsch & Wakefield, 2005]



summary table of these methods can be found in [Serra, 2011, p.36].

1. **Feature extraction:** Usually a chroma representation. The idea is to preserve characteristics that describe the underlying piece of music the best, while ignoring the features of a particular performance. Many audio recognition services exploit tonal sequences when extracting meaningful information from an audio signal [Hu *et al.*, 2003]. Tonal sequence is a set of different note combinations played either individually or in harmony.

2. **Key invariance:** Different performances of the same song can be performed in a different key. Most people do not hear much difference between an original song and its transposed version, because humans perceive pitches relative to each other, not in absolute values. Systems that are made for version identification usually handle transposition changes, since it is a very common for different versions to be played in different keys, whereas in an audio identification field it is not very common to consider transposition. One can tackle transposition by going through all possible transpositions [Kurth & Müller, 2008] or selecting a set of specific transpositions [Ahonen, 2010]. Using all possible transpositions increases the retrieval accuracy to its maximum, while it is much faster and computationally cheaper to use only a small set of carefully selected transpositions.

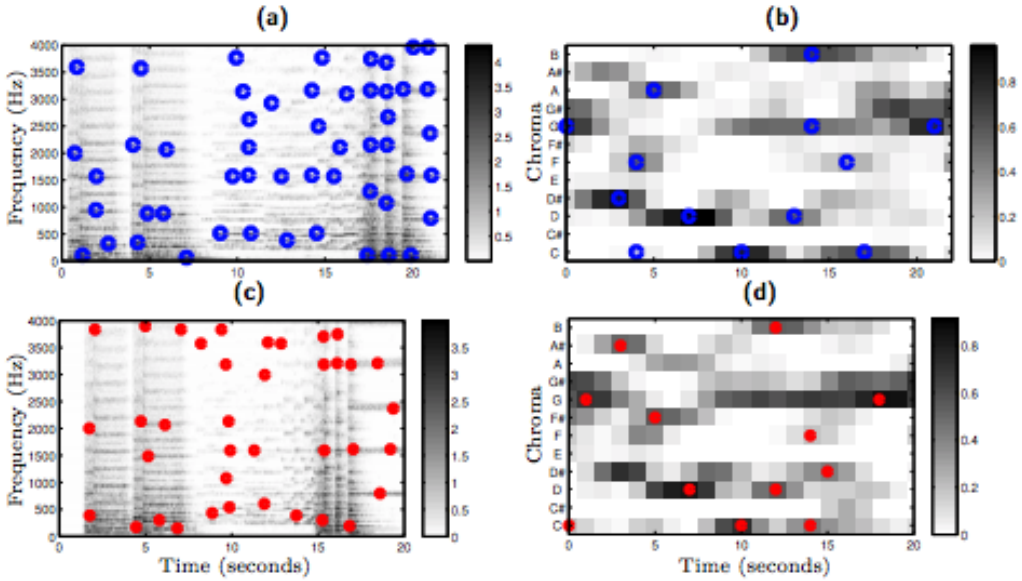
3. **Tempo invariance:** Different versions can have different tempo than the original version. Some systems like [Ahonen, 2010] ignore the tempo changes in their implementation, and use techniques that either fail to retrieve versions that do not have a matching tempo or end up oversimplifying the music representations. The most common way to achieve tempo invariance is to use relative encoding instead of precise durations. Relative encoding can be tracked using either beat tracking or dynamic programming. Alignment of two musical objects can also be done by applying a beat-alignment step to both of them [Bertin-Mahieux & Ellis, 2011].

4. **Structure invariance:** This is an optional step and many retrieval systems have left this part out of their solutions. A common approach is to summarize a song into its most repeated parts. Structure analysis is performed on the song in order to segment sections, like in [Peeters, 2007]. The most repetitive patterns are chosen, discarding the rest. This is however ineffective in scenarios where the most characteristic and memorable feature does not repeat in a song. A method called sequence windowing [Kurth & Müller, 2008], descriptors are segmented into small parts and the similarity is calculated between them. Such algorithms that consider the best possible alignments between two subsequences as similarity measures are widely used in many applications, for example in [Serrà *et al.*, 2008].

5. **Similarity computation:** The final part is to retrieve the n most similar versions from the database. Many conventional similarity measures such as Euclidean distance or more complex algorithms such as time warping can be used to determine the similarities between the query song and the songs in the database.

Chroma features can be computed in many ways. Chroma values can be divided into 12 distinct pitch classes, which correspond neatly to twelve tone scale traditionally used in Western music (C, C♯, D, D♯, ... , B). By transforming an audio signal into a chromagram by using a short-time Fourier transform together with bucketing strategies, the frequency axis of a spectrogram is divided into twelve chromas, each of which corresponds to a particular semitone. A semitone is an interval between two notes in a 12-note scale. This creates a chromagram which is coarser than the original detailed spectrogram for representing the audio signal. Individual chroma features are computed by summing up the pitches for

Figure 6.2 Spectro- and chromagrams and peaks extracted from different recordings of Beethoven's Symphony No. 5. (a) and (b) Spectrogram and chromagram for Bernstein recording. (c) and (d) Spectrogram and chromagram for Karajan recording. [Grosche *et al.*, 2012]



each pitch class, creating a 12-dimensional chroma vector where each entry corresponds to a chroma. In [Kurth & Müller, 2008] each of these chroma vectors corresponded to a window of 200ms while the overlapping ratio was 1/2. According to Grosche et al. these features are very robust against many changes, e.g. caused by differences in instrumentations and also the chroma-based audio features are closely correlated to the harmonic advancement of the underlying music, making them suitable for describing the music. For example, Figure 6.2 from [Grosche *et al.*, 2012] shows two spectrograms and chromagrams with their respective peak fingerprints from two different versions of Beethoven's Symphony No. 5. From the spectrogram images it is hard to see that in both of them the original audio signal is the same piece of music, whereas in the chromagrams the similarities are very visible. Another good description of the chroma fingerprinting system is given in [Malekesmaeili & Ward, 2013].

In their research, Kurth and Müller [2008] present some modifications to the chroma representation. Each chroma vector v is replaced by a vector where each entry represents the signal energy's relative distribution over the 12 chroma bands. This is done in order to enhance the absorbing of dynamic variations.

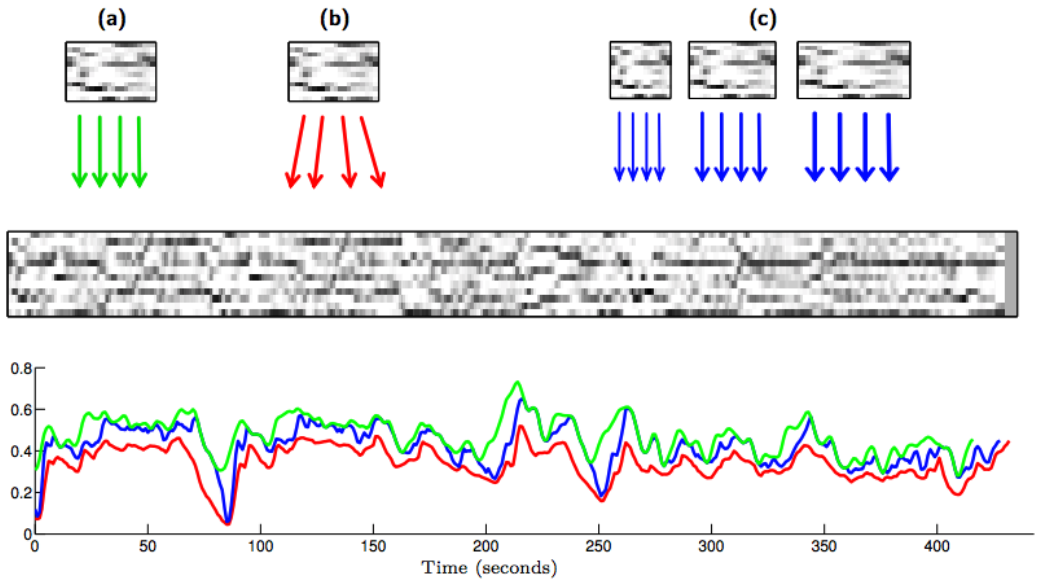
Also, in order to minimize random energy distributions caused by silent moments, a threshold value is used to replace chroma vector v with uniform distribution whenever the values of v are too close to zero. Silent moments and bad alignment of windows are problematic by causing false negative results when two windows are unintentionally misaligned. Unlucky misalignments can also be minimized by performing a beat-alignment step for the audio signal, which uses musical knowledge about the musical tempo to segment the audio signal in a way that it matches with the rhythm of the song [Maddage *et al.*, 2004].

Chromagrams can be further post-processed for additional robustness. The chroma vectors can be normalized, which makes them robust against changes in dynamics and loudness. Robustness against local variations can also be increased by applying a temporal smoothing and downsampling steps to the chroma vectors. More post-processing steps are described in [Grosche *et al.*, 2012]. The important thing to understand is that the post-processing phases in the feature extraction step are the ones that define how well the extracted chroma features perform in different retrieval scenarios. Just like badly constructed fingerprints result in bad querying results, unsuitable processing of chroma features most likely also lead to poor results.

For matching, instead of using peak representations used for example in [Wang, 2003], chroma-based systems employ a subsequence search, where a query chromagram is compared with all the subsequences found in database chromagrams. A curve called a matching curve is obtained as a result, from which one can see in which position the best match is located. Small values in the matching curve indicate that the query sequence is similar to a database subsequence that begins from this particular position. This way the smallest value found is also the best match found. Figure 6.3 shows an example presented by [Grosche *et al.*, 2012], where three different matching curves are obtained by using first a strict subsequence matching and then a DTW, and a multiple query strategy. Peeters [2007] explains that a sequence is defined as a set of successive times that is similar to some other set of successive times. This is quite similar to the definition of melody or chord successions, and these sequences are visualized in similarity matrices as diagonal lines.

An approach called an *inverted file indexing* is used to speed up the matching process even further. It uses a codebook that includes a set of chroma vectors, and which can be obtained through an unsupervised learning process or an exploitation of existing musical knowledge. The codebook is used to classify and index the database chroma vectors to their assigned codebook vectors. Now each

Figure 6.3 Illustration of audio matching process. Three different matching techniques are used to create three different matching curves: (a) strict subsequence matching (b) DTW and (c) multiple query strategy. [Grosche *et al.*, 2012]



codebook vector has its own *inverted list*, which can be used for efficient exact matching. According to [Grosche *et al.*, 2012], the downside of this approach is that the performance of the system is highly dependent on the quality of the codebook, and even at its best it is suitable only for medium sized databases. In order to cope with large databases, [Casey *et al.*, 2008] propose a system that uses utilizes small and overlapping *audio shingles*, a set of chrome feature subsequences. According to Casey et al. the approach is more effective than using individual chrome features for comparisons while being also directly compatible with Locality Sensitive Hashing (LSH). Thomas et al. [2012] also use shingles to create a system that combines diagonal matching approach to subsequence Dynamic Time Warping (DTW) approach in order to create a system that is accurate and computationally faster. Both of these approaches have their strengths and weaknesses: diagonal matching is faster than DTW but because it requires that the two compared sequences have the equal length it does not perform well at handling for example tempo variations. DTW addresses global and local variations much better, but is on the other hand much slower. The resulting index-based DTW is much faster and accurate than the regular DTW [Thomas *et al.*, 2012].

6.1 Sung and hummed queries

Audio matching tasks also include the task of finding a match for queries that have been input by a user through humming, tapping or singing. The user inputs the query by recording a piece of humming or tapping, and the database is then searched for melodies that match this user input query. These *query-by-example* systems (also query-by-humming and query-by-tapping) most often present the user with the n most likely matches, from which the user can check if the song he/she was looking for was found. In case the system does not return the wanted result, the user can input a new query [Zhu & Shasha, 2003]. The idea is that queries like query-by-humming is that they require no musical training of the users, which is why queries of this type usually also include many errors. Because of the high number of input errors, most query-by-example paradigms utilize a melodic contour. According to Ghias and other [1995], a melodic contour describes the relative differences in pitch between notes and is also the method which users most naturally use for determining melodic similarities correctly. The user input query is transcribed into discrete notes and then compared with the melodies found in the database. However, just like in Section 5, the unresolved problem of music transcription makes the method quite unreliable [Zhu & Shasha, 2003].

Similar to other audio matching methods, the techniques found in the literature also use methods like DTW to use audio information itself for comparisons instead of their note representations. Slowness and other performance issues remain strongly in these queries too [Zhu & Shasha, 2003]. The proposed methods do not differ much from those presented earlier with other audio matching scenarios. The used query is just much more primitive and simple although most probably also more prone to include errors.

7 Conclusions

The aim of this thesis was to give a good overview of the field of audio retrieval systems and to present a few of the retrieval methods proposed in the literature. Due to the large number of research papers addressing the subject, everything could not be included in this thesis, and I have used my best judgment when selecting which research papers to include or to exclude.

First some of the basic characteristics of music and music recognition were discussed. The problem of audio retrieval was split roughly into three categories: audio identification, audio matching, and version identification. Audio identification is a problem that has already some working solutions and this thesis focuses on comparing these different methods with each other while keeping an eye on their strengths and weaknesses. Sections 4 and 5 focused on audio identification paradigms called audio fingerprinting and string-based audio retrieval, respectively. The string-based audio retrieval methods are a bit out-dated and not that much researches nowadays because of the challenges in audio transcription, but they were discussed here because it is very common to think of audio retrieval as a string-based retrieval task. Today, popular applications like Shazam use an approach called audio fingerprinting, for which several suggested paradigms were presented, many of which utilized the spectral features of an audio to compute compact and distinct audio fingerprints.

Audio matching and version identification are much broader problems with very challenging requirements that are still very much unsolved today. For these tasks too, suggested approaches were discussed together with their strengths and weaknesses in Section 6. A very popular approach to audio matching is to utilize Chromas. Chroma features are better at presenting the characteristics of the underlying song, e.g. the melody, while ignoring characteristics that have more to do with a specific performance or recording of the audio, e.g. instrumentation or noise. The real challenge of audio matching problems is the constant balancing between specificity and granularity, from highly specific and exact matching to a much broader meaning of similarity. How the system and its parameters should be configured depends highly on the use case and the number of parameter combinations is so high that it is unfeasible for humans to manually configure them. This is why some of the researchers suggest using machine learning for finding the best possible way to match audio files together.

References

[Ahonen, 2010] T. E. Ahonen. Combining chroma features for cover version identification. *Proc. of the 11th International Society for Music Information Retrieval Conference*, pages 165–170, 2010.

[Anguera *et al.*, 2012] X. Anguera, A. Garzon, & T. Adamek. Mask: Robust local features for audio fingerprinting. *Proc. of the IEEE International Conference on Multimedia and Expo*, pages 445–460, 2012.

[Baluja & Covell, 2006] S. Baluja & M. Covell. Content fingerprinting using wavelets. *Proc. of the 3rd European Conference on Visual Media Production*, pages 198–205, 2006.

[Baluja & Covell, 2007] S. Baluja & M. Covell. Audio fingerprinting: Combining computer vision & data stream processing. *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 213–216, 2007.

[Baluja & Covell, 2008] S. Baluja & M. Covell. Waveprint: Efficient wavelet-based audio fingerprinting. *Pattern Recognition*, 41(11):3467–3480, 2008.

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

[Bertin-Mahieux & Ellis, 2011] T. Bertin-Mahieux & D. P. W. Ellis. Large-scale cover song recognition using hashed chroma landmarks. *2011 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 117–120, 2011.

[Casey *et al.*, 2008] M. Casey, C. Rhodes, & M. Slaney. Analysis of minimum distances in high-dimensional musical spaces. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(5):1015–1028, 2008.

[Chandrasekhar *et al.*, 2011] V. Chandrasekhar, M. Sharifi, & D. A. Ross. Survey and evaluation of audio fingerprinting schemes for mobile query-by-example applications. *Proc. of the 12th International Society for Music Information Retrieval Conference*, pages 801–806, 2011.

[Chou *et al.*, 1996] T. C. Chou, A. L. P. Chen, & C. C. Liu. Music databases: Indexing techniques and implementation. *Proc. of the 2nd International Workshop on Multimedia Database Management Systems*, pages 46–53, 1996.

[Covell & Baluja, 2007] M. Covell & S. Baluja. Known-audio detection using waveprint: Spectrogram fingerprinting by wavelet hashing. *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 237–240, 2007.

[DMR Business statistics, 2018] DMR Business statistics. 23 amazing shazam statistics and facts (march 2018), 2018. [Online; accessed April 19, 2018].

[Ghias *et al.*, 1995] A. Ghias, J. Logan, D. Chamberlin, & B. C. Smith. Query by humming: musical information retrieval in an audio database. *Proc. of the 3rd ACM International Conference on Multimedia*, pages 231–236, 1995.

[Gionis *et al.*, 1999] A. Gionis, P. Indyk, & R. Motwani. Similarity search in high dimensions via hashing. *Vldb*, 99(6):518–529, 1999.

[Gizmodo, 2010] Gizmodo. How shazam works to identify (nearly) every song you throw at it, 2010. [Online; accessed May 21, 2018].

[Grosche *et al.*, 2012] P. Grosche, M. Müller, & J. Serrà. Audio content-based music retrieval. *Dagstuhl Follow-Ups*, 3:157–174, 2012.

[Gupta *et al.*, 2010] V. Gupta, G. Boulianne, & P. Cardinal. Content-based audio copy detection using nearest-neighbor mapping. *Proc. of the IEEE International Conference on Acoustics Speech and Signal Processing*, pages 261–264, 2010.

[Gutiérrez & García, 2015] S. Gutiérrez & S. García. Landmark-based music recognition system optimisation using generit algoritms. *Multimedia Tools and Applications*, 75(24):16905–16922, 2015.

[Hainsworth & Macleod, 2003] S. W. Hainsworth & M. D. Macleod. The automated music transcription problemy. *Technical report*, pages 1–23, 2003.

[Haitsma & Kalker, 2010] J. Haitsma & T. Kalker. A highly robust audio fingerprinting system with an efficient search strategy. *Journal of New Music Research*, 32(2):211–221, 2010.

[HowMusicWorks.org, 2018] HowMusicWorks.org. Sound and music, 2018. [Online; accessed April 15, 2018].

[Hsu *et al.*, 1998] J. L. Hsu, C. C. Liu, & A. L. P. Chen. Efficient repeating pattern finding in music databases. *Proc. of the 7th International Conference on Information and Knowledge Management*, pages 281–288, 1998.

[Hu *et al.*, 2003] N. Hu, R. B. Dannenberg, & G. Tzanetakis. Polyphonic audio matching and alignment for music retrieval. *Department of Computer Science, School of Computer Science*, page 521, 2003.

[Invisibles, 2014] Invisibles. Equal temperament in tuning, 2014. [Online; accessed April 19, 2018].

[Jacobs *et al.*, 1995] C. Jacobs, A. Finkelstein, & D. Salesin. Fast multiresolution image querying. *Proc. of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, pages 277–286, 1995.

[Ke *et al.*, 2005] Y. Ke, D. Hoiem, & R. Sukthankar. Computer vision for music identification. *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 597–604, 2005.

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

[Liu *et al.*, 1999] C. C. Liu, J. L. Hsu, & A. L. P. Chen. An approximate string matching algorithm for content-based music data retrieval. *IEEE International Conference on Multimedia Computing and Systems*, pages 451–456, 1999.

[Maddage *et al.*, 2004] N. C. Maddage, C. Xu, M. S. Kankanhalli, & X. Shao. Content-based music structure analysis with applications to music semantics understanding. *Proc. of the 12th Annual ACM International Conference on Multimedia*, pages 112–119, 2004.

[Malekesmaeili & Ward, 2013] M. Malekesmaeili & R. K. Ward. A local fingerprinting approach for audio copy detection. *Signal Processing*, 98:308–321, 2013.

[Ouali *et al.*, 2015] C. Ouali, P. Dumouchel, & V. Gupta. Efficient spectrogram-based binary image feature for audio copy detection. *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1792–1796, 2015.

[Ouali *et al.*, 2016] C. Ouali, P. Dumouchel, & V. Gupta. Fast audio fingerprinting system using gpu and a clustering-based technique. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(6):1106–1118, 2016.

[Peeters, 2007] G. Peeters. Sequence representation of music structure using higher-order similarity matrix and maximum-likelihood approach. *Proc. of the ISMIR*, pages 35–40, 2007.

[Serra *et al.*, 2010] J. Serra, E. Gomez, & P. Herrera. Audio cover song identification and similarity: background, approaches, evaluation, and beyond. *Advances in Music Information Retrieval*, pages 307–332, 2010.

[Serra, 2011] J. Serra. *Identification of versions of the same musical composition by processing audio descriptions*. Phd. thesis, Universitat Pompeu Fabra, 2011.

[Serrà *et al.*, 2008] J. Serrà, E. Gómez, P. Herrera, & X. Serra. Chroma binary similarity and local alignment applied to cover song identification. *IEEE Transactions on Audio, Speech, and Language*, 16(6):382–395, 2008.

[Shazam, 2018] Shazam. Shazam - music discovery charts and song lyrics, 2018. [Online; accessed April 10, 2018].

[Shepard, 1964] R. N. Shepard. Circularity in judgements of relative pitch. *Acoustical Society of America*, 36:2346–2353, 1964.

[SoundHound, 2018] SoundHound. Soundhound inc. 2018. [Online; accessed May 2, 2018].

[Stollnitz *et al.*, 1995] E. J. Stollnitz, T. D. DeRose, & D. H. Salesin. Wavelets for computer graphics: A primer, part 1. *IEEE Computer Graphics and Applications*, pages 76–84, 1995.

[Thomas *et al.*, 2012] V. Thomas, S. Ewert, & M. Clausen. Fast intra-collection audio matching. *Proc. of the Second International ACM Workshop on Music*

Information Retrieval with User-Centered and Multimodal Strategies, pages 1–6, 2012.

[Wang, 2003] A. Wang. An industrial strength audio search algorithm. *Proc. of the 4th International Conference on Music Information Retrieval*, pages 7–13, 2003.

[Williams *et al.*, 2017] D. Williams, A. Pooransingh, & J. Saitoo. Efficient music identification using orc descriptors of the spectrogram image. *EURASIP Journal on Audio, Speech, and Music Processing*, pages 1–17, 2017.

[Zhu & Shasha, 2003] Y. Zhu & D. Shasha. Warping indexes with envelope transforms for query by humming. *Proc. of the 2003 ACM SIGMOD International Conference on Management of Data*, pages 181–192, 2003.