Audio Fingerprinting for Music Identification

Osagioduwa Edo-Osagie

Abstract

Over the past decade, digital music has really taken off. It would be great if it was possible to identify any song you heard live, even if it was not heard on a digital platform. Having identified the song, it would then be possible to acquire or access the song via digital (or other) means. Among many things, the internet is a vast digital library of music. This paper looks at ways to take advantage of a large database of music to dynamically identify music clips. It presents a review of various approaches to measuring the similarity of music and how they can be applied to music identification.

1 Introduction

Fingerprint systems have been around for a very long time. Chinese records from the Qin Dynasty (221-206 BC) include details about using *handprints* as evidence during burglary investigations. From that time, the idea that fingerprints could be used for unique identification persisted even if only as hypotheses and in works of fiction. In 1900, the United Kingdom Home Secretary Office performed some research into the identification of Criminals by fingerprints. Mr Edward Richard Henry (later Sir ER Henry) appeared before the inquiry committee to explain the system published in his book - “The Classification and Use of Fingerprints”. By the next year, Scotland Yard accepted his system for identifying people’s fingerprints (Onin (2015)). This system relies on the pattern of dermal ridges on the fingertips and still forms the basis of all human fingerprinting techniques today. One thing that is important to note of a fingerprint is that it can be used to represent a person without giving (or needing) any other aspects from the person. An example of this is the fact that a person’s fingerprint does not convey any information on a person’s height or hair colour, nor does it need to take any of these things into consideration for it to be used.

More recently, there has been a growing interest in computing fingerprints for multimedia objects. (Yang (2001), Neuschmied (2001)). The objective of multimedia fingerprinting is to obtain an efficient method to establish the perceptual (rather than mathematical) equality of two multimedia objects; not by comparing the relatively large objects themselves, but by comparing their associated fingerprints, which are small by design. This paper investigates and reviews a number of possible approaches to audio fingerprinting and how they can be applied to music identification.

2 Applications of Audio Fingerprinting

Audio fingerprinting has many applications in both the private business and public domains. The most popular use is through software developed by companies like Shazam[1] and SoundHound[2] who both offer mobile solutions for music identification. However, uses for audio fingerprinting extend beyond the general public wanting to identify songs. Audio fingerprinting can be used by composers and musicians to check that parts of their songs and riffs are not already copyrighted. Police and forensic investigators can, and do, use audio fingerprinting to identify stolen music on the computers of those suspected of copyright infringement. If a track of unreleased music is leaked to the public, music producers can use audio fingerprinting to identify the exact revision of the track to help identify when and who leaked the track (Palmer (2010)).

[1]http://www.shazam.com
[2]http://www.soundhound.com

3 Concepts of Audio Fingerprinting

3.1 Definition

An audio fingerprint can be seen as a concise description of an audio object. This means that a fingerprint function F should map an audio object X, which consists of a large number of bits, to a fingerprint of only a limited number of bits (Haitsma and Kalker (2002)). In this way, audio fingerprints are very similar to hash functions. In fact, fingerprinting is often referred to as *perceptual hashing* in a lot of literature (Haitsma et al. (2001)). A hash function enables the comparison of two large objects X and Y, by just comparing their respective hash values $H(X)$ and $H(Y)$. Mathematical equality of the hash pair implies equality of the objects themselves (with a very low probability of error).

Because of this, at first sight, hash functions may seem like a good candidate for fingerprint functions. However, we are interested in perceptual equality rather than mathematical equality, which is what hash functions offer. An illustration of the difference between perceptual and mathematical equality is as follows: an mp3 version of a song and an original CD quality version of a song that are sampled at different bit-rates will sound the same to a listener, but have different waveforms. While the two versions are perceptually similar, they are mathematically different. In addition to this, perceptual similarity is not transitive. Perceptual similarity of a pair of objects X and Y and of another pair of objects Y and Z does not necessarily imply the perceptual similarity of objects X and Z. However, modeling perceptual similarity by mathematical equality of fingerprints would lead to such a relationship (Haitsma and Kalker (2002)).

3.2 Audio Fingerprinting Parameters

Having outlined a definition for an audio fingerprint, we can now discuss the different parameters possessed by an audio fingerprinting system. (Haitsma and Kalker (2002)) propose the following as the main parameters:

- **Robustness -** how well a signal can be identified after degradation.
- **Reliability -** how often a song is incorrectly identified.
- **Fingerprint Size -** how much storage is needed to store a fingerprint. (This is important to note because fingerprints are usually stored on RAM (which is more limited) to enable fast searching).
- **Granularity -** how many seconds of audio is needed to identify an audio clip.
- **Search Speed and Scalability -** how long it takes to find a fingerprint in a database containing x songs.

3.3 Application to Music Identification

Music Identification refers to the process of obtaining some data for a given audio recording, sample or excerpt. A simple summary of the principle is as follows: A database of songs along with relevant meta-data is collected. A query recording or sample is compared against the database for a match. Once a match is found, the meta-data for the match can be returned. However, rather than storing reference songs in the database, only the songs' fingerprints are stored. This is intended to reduce the size of the items being stored and should reduce the search time. Hence, the query audio signal is first converted into an audio fingerprint before the database comparisons are made.

In essence, for an audio fingerprinting system to be useful for a music identification application, it must fulfill the following requirements:

- **Discriminative Power -** A good audio fingerprint must make it possible to accurately identify an item within a huge number of other items. That is, it should be informative and contain the core defining characteristics of the song such that the system has a low probability of false positives.
- **Invariance to Distortions -** A good audio fingerprint should not be sensitive to distortions to the song that do not actually change it. Examples of such distortions are compression artifacts, pitching and background noise.
- **Compactness -** A good fingerprint should compress the size of the songs (or multimedia objects) involved to be as small as possible. This will allow for a faster search.
- **Computational Simplicity -** Extraction of fingerprint should be simple and efficient. It should not place an unwanted overhead on the music identification process.

4 Review of Audio Fingerprinting Methods

This paper looks at two kinds of methods for audio fingerprinting. The first employs intelligent feature vectors for classification and the second involves the use of spectral analysis techniques to generate fingerprints. While spectral analysis methods seem to be more popular (probably due to the fact that the commercial music identification platform, Shazam uses this method), feature vector classification methods are not without their merits. In this section, we will analyze and review both approaches.

4.1 Audio Fingerprinting Using Chroma Features

This approach is an extension of a popular audio processing technique that makes use of ***"feature vectors"*** to represent a piece of audio. A feature vector is a mathematical representation of some feature or attribute possessed by the audio signals in question. There are many different kinds of features used in various applications from speech recognition to similarity measures for music recommendations. A very popular feature (used in both of the applications mentioned previously) is the Mel Frequency Cepstral Coefficients (MFFC) (Logan et al. (2000)). The approach that we will be looking at for audio fingerprinting makes use of *Chroma Features* to identify music.

To fully understand what chroma features are, there are some music terminology and concepts that we must familiarize ourselves with.

Pitch
Pitch is an auditory perceptual property that allows the ordering of sounds on a frequency-related scale (Klapuri and Davy (2007)). Pitches can be compared as "higher" and "lower" in the same sense as musical melodies (Plack et al. (2006)). Pitch is one of the auditory attribute of musical tones, along with duration, loudness, and timbre (Patterson et al. (2010)).

Octave
In music, an octave is the interval between one musical pitch and another with half or double its frequency. The octave relationship is a natural phenomenon that has been referred to as the "basic miracle of music", the use of which is "common in most musical systems" (Applebaum (1974)).

Pitch Class
In music, a pitch class is the set of all pitches that are a whole number of octaves apart. For example, the pitch class C consists of the Cs in all octaves. The pitch class C represents all possible C sounds, in whatever octave position. This quality of pitch is known as ***chroma***.

A chroma feature consists of a twelve-element vector with each dimension representing the intensity associated with a particular semitone, regardless of octave. For each song in the database, a file containing the chroma vectors representing the song is created. An example of a chroma vector is:

0.98892, 0.95418, 0.89027, 0.93907, 0.75066, 0.77229, 0.71884, 0.72182, 0.89031, 0.9806, 0.85977, 1

Each vector can be seen as a point in an n-dimensional space where n is equal to 12. We can take advantage of this by using n-dimensional metrics to calculate similarity as the distance between the chroma features of our query song and the chroma features of the songs in the database.

The distance metric used to measure similarity between chroma features is a modified version of the ***Euclidian distance***. Given two n-dimensional points $p = (p_1, p_2, ..., p_n)$ and $q = (q_1, q_2, ..., q_n)$, the euclidean distance between p and q can be defined as follows:

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_1)^2 + ... + (p_n - q_n)^2} \qquad (1)$$

However, to eliminate similar differences which exist on different octaves in two chromas, the relative euclidean distance is used. The relative euclidean distance between two points p and q is defined as:

$$d_r(p,q) = \sqrt{(p_1 - q_1)^2 + ... + (p_n - q_n)^2 - (n-1) \times min_i(p_i - p_q)^2} \qquad (2)$$

Note that a single song will contain thousands of chroma features. With scalability in mind, it is not efficient to calculate distances between all the vectors in all the songs in the database. To overcome this, the search space needs to be reduced. Instead of considering all lines in a chroma feature file associated with a song, only the *minimum*, *average* and *maximum* values for all octaves are selected. In this way, instead of considering around 30,000 lines for a song from the database and possibly around 5,000 lines for an excerpt of a song to be identified, only three lines need to be considered for both songs in the database and the excerpt to be classified (Iftene et al. (2011)). Thus for two songs s_1 and s_2, the similarity can efficiently be calculated as:

$$d_r(s_1, s_2) = d_r(s_1min, s_2min) + d_r(s_1avg, s_2avg) + d_r(s_1max, s_2max) \qquad (3)$$

To classify a song, the similarity between the songs from the database and the query song (d_r) is calculated. The song with the minimum distance (and hence the greatest similarity) can be selected as a match to the query. The meta-data associated with the query result can be returned to give information on the identified song.

4.2 Audio Fingerprinting Using Spectral Analysis: The Shazam Algorithm

This method was developed by Shazam Entertainment, Ltd. which was started in 2000 with the intention of providing a service that could connect people to music by recognizing music in the environment by using mobile phones to recognize the music directly (Wang et al. (2003)). The algorithm makes use of ***spectrogram peaks*** as its determinant feature upon which fingerprints are built. Spectrogram peaks were selected presumably due to their robustness in the presence of noise and approximate linear superposability (Wang and Smith III (2000)).

The algorithm can be divided conceptually into two stages - *Fingerprinting* and *Matching and Searching*.

4.2.1 Fingerprinting

The point of this step is to reduce the audio signal or file into a small yet robust representation. In this sense, robust means being as insensitive to noise and recording quality as possible. The spectral analysis method achieves this by representing the audio signal as a "constellation" of spectrogram peaks. A spectrogram can be visualised as a plot of time against frequency showing the intensity of the signal as a point on a colour scale. The algorithm identifies peak points in the spectrogram based on some criterion, usually density. A time-frequency point is chosen as a peak if it has a higher energy content (or intensity) than all its neighbors in a region centered around the point. This not only ensures that the constellation has reasonable coverage but, by choosing the highest amplitude peaks - the peaks which are most likely to survive distortions and degradations - the constellation of peaks is reasonably robust.

These spectrogram peaks form a constellation of dots. This pattern of dots should be the same for matching segments of audio.

4.2.2 Matching and Searching

Following the conclusion from the fingerprinting stage that the resulting pattern of dots should be the same for matching segments of audio, this stage aims to find a match between two audio signals that are the same.

Imagine we are given the constellation map for a song from the database on a piece of paper and the constellation map for a matching query audio sample on a **transparent** piece of paper or plastic. If we slide the transparent collection of dots of the query audio sample over the collection of dots of the database song, at some point, a significant number of points will coincide. This idea can be applied to constellation fingerprints in order to match them.

A fingerprint is extracted from the query and compared to the fingerprints of the database songs. The query fingerprint is shifted along time against every database fingerprint. The number of peaks that are matching is counted and saved for every possible shift. A high count indicates a match, and the corresponding reference is identified (Rafii (2014)). Figure 1 shows an illustration of this process.

However, finding the correct registration offset directly from constellation maps in this way can be rather slow. Hence, Wang et al. developed a fast way of indexing constellation maps using *fast combinatorial hashing.* Using this, fingerprint hashes are formed from the constellation map, in which pairs of time-frequency points are combinatorially associated. Spectrogram peaks are chosen as *anchor points.* Each anchor point is assigned an associated *target zone.* Pairs of peaks are formed by choosing an anchor point and associating it with each point in its target zone. For every pair of peaks, a hash is formed using their two frequency values and their time difference. Hashes from a query are compared to hashes from every reference in the database, given their offset times. Figure 2 shows an illustration of this process.

These hashes are reasonably reproducible, even in the presence of noise and voice codec compression. Also, by forming pairs instead of searching for matches against each individual

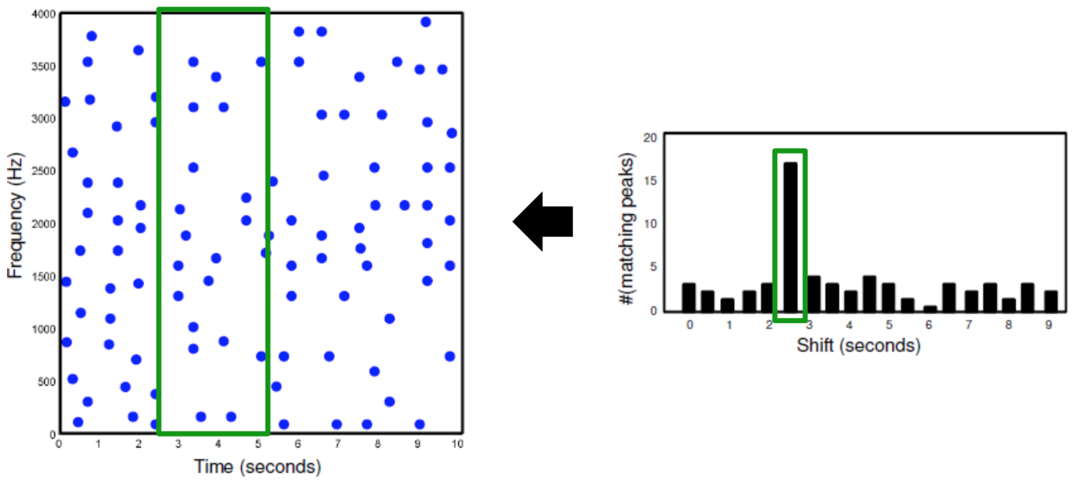

Figure 1: Fingerprint matching

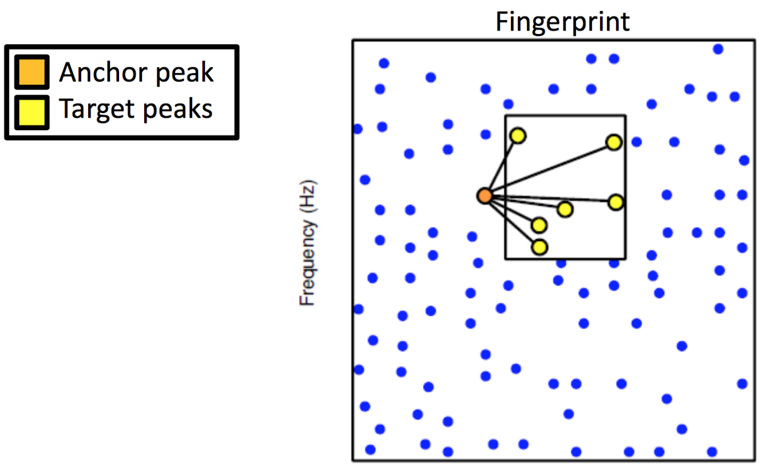

Figure 2: Fast combinatorial hashing

constellation point, we gain a tremendous acceleration in the search process.

5 Comparisons and Conclusion

In terms of speed, the spectral analysis method outperforms the chroma features method. For a database of about 20 thousand tracks implemented on a PC, the search time of the spectral analysis method is on the order of 5-500 milliseconds, depending on the application (Wang et al. (2003)). However, when the search space reduction optimization technique outlined in **section 3.1** is employed on the chroma feature method and fast combinatorial hashing is not applied to the spectral analysis method, the speed of the chroma features approach is the same or better than that of the spectral analysis method. This is because without fast combinatorial hashing, each point in the constellation map will need to be checked whereas the chroma features only makes checks within a smaller search space. This means that if the speed of the spectral analysis method is to be taken advantage of, the fast combinatorial hashing technique ***must*** be implemented.

The spectral analysis fingerprinting algorithm performs well with significant levels of noise. It can correctly identify music in the presence of voices and even other music. However, the algorithm was designed specifically to target recognition of sound files that are already present in the database. It is not expected to be used for live music recordings. While the chroma features method is not as robust to noise as the spectral analysis method, it is extremely robust to tempo and pitch. This means that it can be used to identify music not only from live recordings, but also covers and music played with different instruments.

References

Applebaum, E. (1974). Perspectives in music theory: An historical-analytical approach by paul cooper. new york: Dodd, mead & company, 1973. 282 pp. music examples, index. hard cover. *Music Educators Journal*, 61(1):124–127.

Haitsma, J. and Kalker, T. (2002). A highly robust audio fingerprinting system. In *ISMIR*, volume 2002, pages 107–115.

Haitsma, J., Kalker, T., and Oostveen, J. (2001). Robust audio hashing for content identification. In *International Workshop on Content-Based Multimedia Indexing*, volume 4, pages 117–124. Citeseer.

Iftene, A., Rusu, A., and Leahu, A. (2011). Music identification using chroma features. In *CLEF (Notebook Papers/Labs/Workshop)*, volume 2011.

Klapuri, A. and Davy, M. (2007). *Signal processing methods for music transcription.* Springer Science & Business Media.

Logan, B. et al. (2000). Mel frequency cepstral coefficients for music modeling. In *ISMIR*.

Neuschmied, Mayer, B. (2001). Identification of audio titles on the internet.

Onin (2015). History of the fingerprint. `http://onin.com/fp/fphistory.html`.

Palmer, N. (2010). Audio fingerprinting: Review of audio fingerprinting algorithms and looking into a multi-faceted approach to fingerprint generation. `http://www.palmnet.me.uk/uni/FYP/Audio%20Fingerprinting.pdf`.

Patterson, R. D., Gaudrain, E., and Walters, T. C. (2010). The perception of family and register in musical tones. In *Music perception*, pages 13–50. Springer.

Plack, C. J., Oxenham, A. J., and Fay, R. R. (2006). *Pitch: neural coding and perception*, volume 24. Springer Science & Business Media.

Rafii, Z. (2014). Machine perception of music & audio. `http://www.zafarrafii.com/doc/Rafii%20-%20Audio%20Fingerprinting%20-%20NU%20EECS%20352%202014.pdf`.

Wang, A. et al. (2003). An industrial strength audio search algorithm. In *ISMIR*, pages 7–13.

Wang, A. L.-C. and Smith III, J. O. (2000). Wipo publication wo 02/11123a2, 7 february 2002.

Yang, C. (2001). Music database retrieval based on spectral similarity.