Audio fingerprinting for song recognition

Federico Bartsch[1]

1. Introduction

The aim of the project is to go through the An Industrial-Strength Audio Search Algorithm paper, in order to deeply understand how signal processing can be very useful to implement an audio recognition system.
A straightforward approach would be to systematically shift the audio sample across the entire track and assess whether it aligns or corresponds at any given point in time. This method essentially involves exhaustively exploring all possible alignments to identify potential matches, which is very expensive to do considering that there are millions of songs to which the sample has to be compared to. This is why the implemented solution extracts a fingerprint of each song, which is nothing else than a summary of the song, in order to speed up the matching process.

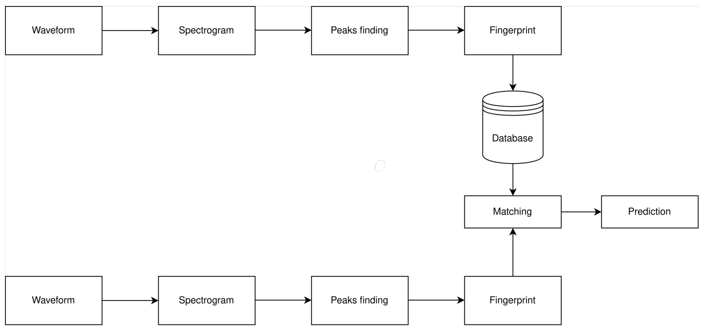

Figure 1: Steps

As illustrated in Figure 1, the sections of this report will present the following procedural steps:

1. Calculating the spectrogram
2. Finding peaks
3. Fingerprint generation
4. Matching

2. Calculating the spectrogram

After making clear that working with the raw audio file isn't a good idea, let's introduce what is a spectrogram and why it is so important in audio recognition. A spectrogram is a visual representation of the spectrum of frequencies in a signal as they vary with time. It provides a two-dimensional view of how the frequency content of a signal changes over a certain period. In particular:

- X-axis represents time
- Y-axis represents frequencies
- The value of a point (x, y) is called **amplitude** and represents the intensity of a frequency y at time x.

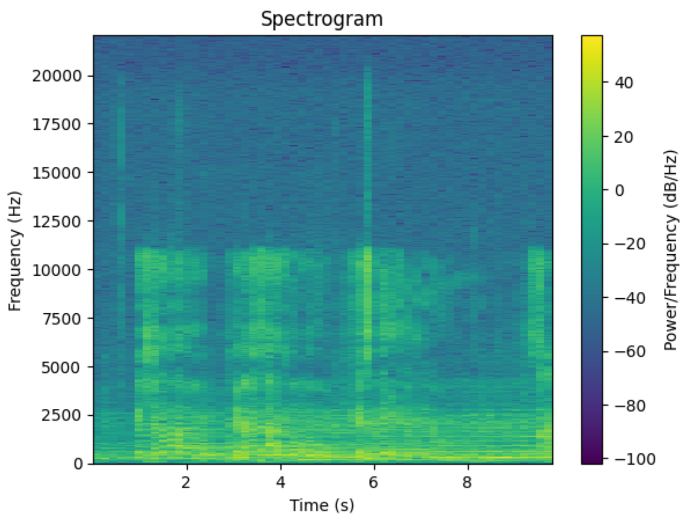

Figure 2: Spectrogram

In Figure 2 the spectrogram of a 10 seconds chuck of a song.

The spectrogram allows the signal to be broken down into its frequencies while also maintaining the information of the time in which they appear. Let's delve into the process of computing a spectrogram step by step:

2.1. *Fourier transform*

The Fourier Transform is a mathematical operation that decomposes a time-domain signal into its constituent frequency components:

$$X(f) = \int_{-\infty}^{\infty} x(t) \cdot e^{-j2\pi ft}\, dt \tag{1}$$

2.2. *Short-Time Fourier Transform*

The STFT is an extension of the Fourier Transform that is particularly useful for analyzing signals that vary over time. It involves breaking down the signal into short, overlapping segments and applying the Fourier Transform to each segment. Mathematically, the STFT is defined as:

$$X(\tau, w) = \int_{-\infty}^{\infty} x(t) \cdot w(t-\tau) \cdot e^{-jwt}\, dt \tag{2}$$

2.3. Spectrogram

The spectrogram is a visual representation of the power of the STFT:

$$S(t, f) = |X(t, f)|^2 \tag{3}$$

3. Finding peaks

The subsequent phase involves identifying peaks within the spectrogram, which consists in finding time-frequency pairs that have a high amplitude in the spectrogram. This task can be done by performing the following steps:

1. Apply a maximum filter to the original spectrogram, generating an augmented spectrogram with accentuated peaks. By varying the size of the filter, one can selectively determine the number of peaks to retain. A larger filter results in the preservation of only a few prominent peaks, whereas a smaller filter allows for the retention of a majority of the peaks.

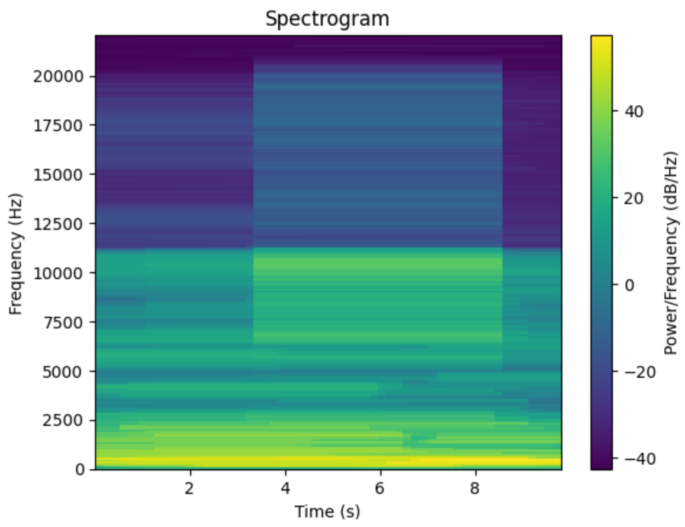

Figure 3: Filtered Spectrogram

2. Overlay the two spectrograms, retaining only those time-frequency pairs where the amplitudes align. The result is the so called constellation map, that can be seen as a scatter plot containing only the most important information of the song:

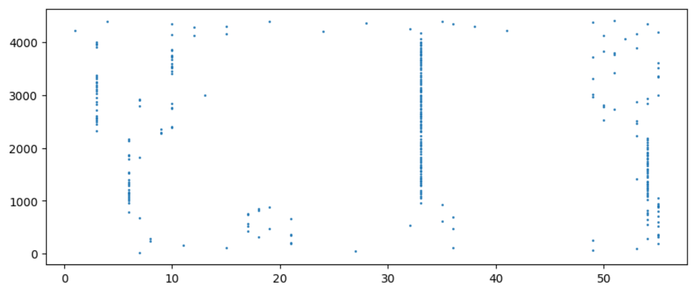

Figure 4: Constellation map

4. Fingerprint generation

Having the constellation map, the only thing that remains to do is to generate the fingerprint of the song. This is done by taking frequency pairs and their time difference and calculating the hash of them.

Considering that the comparison involves an entire song against a brief sample, the algorithm refrains from calculating the hashes of frequencies with excessively high time differences, because they will not both be present in a ten-second recording of the song. Therefore the solution to choose the frequency pairs is to define a frequency A called anchor point, a target window (not to small or too large), connect frequency A to all the frequencies B falling into the target window and calculate the hash as follows:

$$\text{hash}((A, B, \Delta t_{\text{AB}})) \tag{4}$$

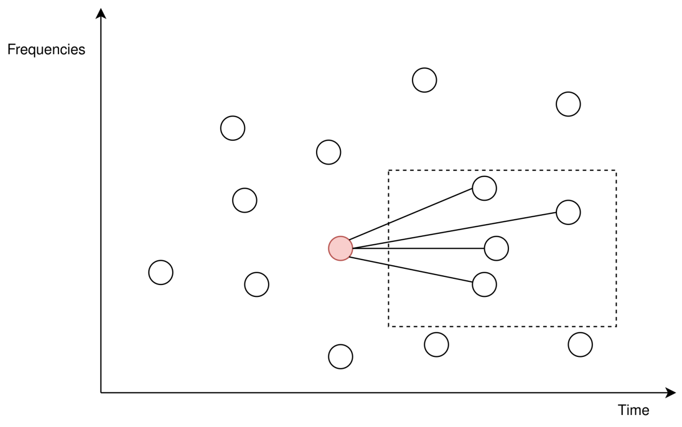

Figure 5: Hashing frequencies

Figure 5 shows how frequency A (red point) is connected to each point inside the target window.

This step has to be done iteratively for each frequency of the constellation map.
Each hash is then associated to the time at which the frequency A appears in the spectrogram, that is important for the matching. After computing all the hashes for a song, the song's fingerprint (a collection of hashes associated to the time of the frequency A) can be saved into a database. Once all the tracks are inserted into the database, a match between these and the given recording can be searched.

5. Matching

The last step is to predict which song is playing by searching for matches in the database. To do this, the above described method is repeated for the new recording:

1. Calculate the spectrogram.
2. Find peaks
3. Generate the fingerprint

Once the hashes of the song are generated, the number of matches for each song in the database is computed. The optimal match will be the one with the highest number of matches. For an ideal match, all the hashes of the recorded track should match to all the hashes of the real track, however this is not possible due to background noise, quality of the microphone and other forms of audio corruption, meaning that only part of the hashes will correspond to the ones of the original track. Even if it is not possible, the consistent number of hashes generated by pairing frequencies are enough to obtain a sufficient number of matches.
This approach works quite well, but sometimes some matches that should not be counted are considered.

5.1. Problem of this approach

Matching hashes are expected to exhibit consistent time differences between them in both the sample and the original song, which implies that two hashes should occur in the same sequential order and with identical time intervals in both the sample and the original song. However, there may be instances where matches occur in a different order. In such cases, it becomes crucial to retain only the pertinent matches, disregarding those that do not align with the expected temporal consistency.

5.2. Solution

This is the reason why, as mentioned in Section 4, the hashes are saved together with the time at which the first frequency occurs. Considering that each hash is linked to the time of the first frequency occurrence, the process involves calculating the time difference for every hash pair between the original song and the sample. By counting the occurrences of each unique time difference, it becomes possible to determine the time difference that maximizes the number of matches. Consequently, the count of matches corresponds to the occurrences associated with this specific optimal time difference.
Below an examples for the right match:

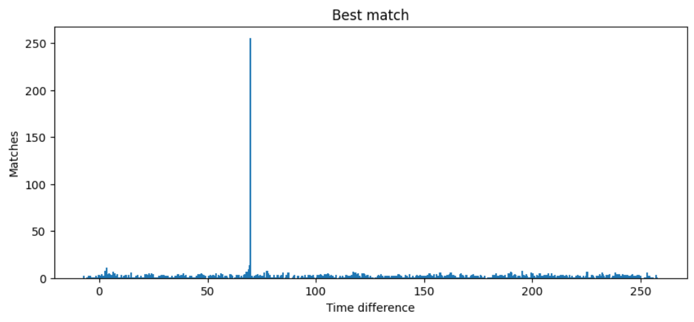

Figure 6: Score Histogram

Its clear that the relevant hashes are exclusively those associated with the time difference corresponding to the peak bar in the histogram. We can conclude that this is a way more robust method of matching.

6. Results

Running the algorithm on a database of 30 songs, and recording from the microphone a brief sample of a song (7-10 seconds), a small enough size for the maximum filter (15-30), and choosing properly the size of the target window, the accuracy of the algorithm is 100%. Moreover, the algorithm demonstrates resistance to noise, but it is sensitive to variations in time since time difference between frequencies is used to calculate the hashes. Below the result for a 10 seconds recorded song:

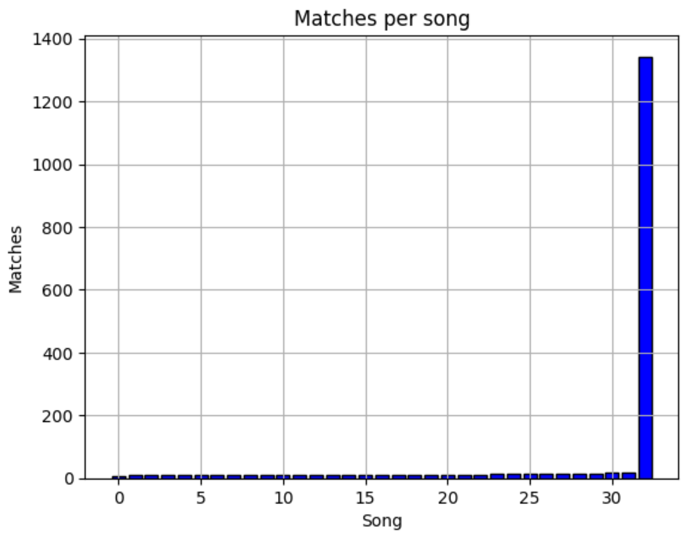

Figure 7: Matches per song

7. Summary and conclusions

The report shows how Short-Time Fourier Transforms can be used to decompose an audio signal into its frequencies, maintaining also the time information, how to use maximum filter in order to find the peaks of the spectrogram and how to generate the fingerprint of an audio.