Ukrainian Catholic University

Faculty of Applied Sciences

Business Analytics & Computer Science Programmes

Shazam Music Recognition

Linear Algebra final project report

Authors:
Dmytro Batko
Anastasiia Petrovych
Yurii Zinchuk

17 May 2023

Contents

Abstract

With the rapid growth of digital music consumption and the increasing availability of diverse audio content, the need for an efficient and accurate audio recognition system has become paramount. Thus, in this project, we address the problem of music recognition or more explicitly Shazam's music recognition. The aim of the project is to develop the software part of the addressed problem, based on Shazam's algorithms, which help to find out the name of the song simply by its recorded part. Although not every recorded piece of audio is perfect (e.g. without noises), the implementation contain the ways to avoid the other information except for the most important one. In addition, a bunch of tests were conducted on diverse audio recordings to check the quality of "filtering" algorithms.

Keywords: Shazam, Music Recognition, Fast Fourier Transform, Filtering, Matching.

1 Introduction

As music is an important and progressive part of the modern world, the need for the tool that could recognise it was only growing over the time which is confirmed by the popularity of Shazam. Suppose the person hear a familiar song, that has listened a thousand times long ago and the one captures 20 seconds of a song or even less. In that case, no matter if it's intro, verse, or chorus, it will create a fingerprint for the recorded sample, consult the database for a match, and use its music recognition algorithm to tell you exactly which song you are listening to.

2 Problem Setting

As it was said before, our aim is to try to develop our own user-friendly application similar to Shazam, which allows us to find the name of the song by its recorded fragment. The main parts of the implementation is Fast Fourier Transform of music piece, applying filtering algorithms for detecting the more valuable information for fingerprinting of the song, its hashing and matching algorithms.

3 Overview of approaches

Indeed, there are many solutions that propose Music Recognition, as Shazam and other apps can do (e.g. SoundHound, Genius, Musicxmatch, Google Assistant & Siri). But the basis of it is transforming audio fragments into the frequency domain by using Fast Fourier Transform. It will help to extract further the fingerprint or signature for the audio fragment providing a static representation of a dynamic signal. From that point, there is a difference between the algorithm of Shazam and some attempts to imitate it. There are many variants of it, but our aim is to highlight two of them.

3.1 Hashing Intervals

To make Music Recognition more effcient, those algorithms use some kind of sliding window, as frequency domain representation does not provide the most important information about the timing of, for example, some peak frequency [5]. Such sources as [5] [7] make Fast Fourier Transform on every chunk of the music (i.e. 44 chunks in one second, given 44100 sample rate) and then, having intervals of frequencies, detect the peaks with the biggest magnitude. This information forms a signature for this chunk of the song, and this signature becomes part of the fingerprint of the song as a whole [5]. Hashing the data is then stored in the database for further matching.

3.2 Hashing with Anchors and target zone (Shazam)

Another way to fingerprint the audio fragment is using Shazam's algorithm. As Fast Fourier Transform has been performed on the song, one needs to find the strongest frequencies in that signal by applying filtering. It is important that they have to be evenly spaced through the spectrogram of the signal to allow fingerprinting of all fragments. Also, it is common knowledge that the recorded song fragment may contain noise; thus, one has to be confident that those peaks are evenly spaced in frequency so that the algorithm can deal with noise and frequency distortion. [4] After detecting those strongest frequencies (anchors) we create the target zone and then hash it.

Although those two approaches end with almost the same algorithms of matching the recorded fragment, we decided to implement Shazam's approach as it is more accurate and more efficient for the task of Music Recognition, which will be discussed in the next paragraph.

4 Algorithm analysis and Implementation

The Shazam's Music Recognition algorithm can be divided into the following main part:

1. Fast Fourier Transform of the song/fragment of the song
2. Finding peaks by applying filtering algorithm
3. Hashing
4. Matching

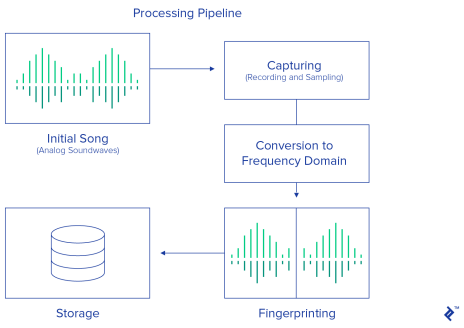


Figure 1: Pipeline of fingerprinting of the song [5]

4.1 Theoretical part

4.1.1 Fast Fourier Transform of the song/fragment of the song

The main part of the Music Recognition Algorithm is Fast Fourier Transform (further FFT), an algorithm used to transform a time-domain signal into its corresponding frequency-domain representation by using the computed discrete Fourier transform (DFT) of a sequence. Let $x_0, \ldots, x_{N-1}$ be complex numbers. Thus, the DFT is defined by the forumula:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi kn/N} \qquad k = 0, \ldots, N-1$$

where $e^{i2\pi/N}$ is a primitive Nth root of 1.

As for the complexity, from definition it directly requires $O(N^2)$ operations: there are N outputs X_k, and each output requires a sum of N terms. But FFT algorithms require only $O(n \log(n))$.

In the case of audio signals, the FFT can be used to analyse a sound's spectral content, allowing us to visualise and manipulate individual frequencies that make up the sound. The basic idea behind the FFT is that any signal can be represented as a sum of sinusoids with different frequencies, amplitudes, and phases. The FFT algorithm takes a time-domain signal as input and produces a frequency-domain representation of that signal as output. The output of the FFT is a set of complex numbers which represent the amplitude and phase of each frequency component of the signal.

4.1.2 Finding peaks by applying filtering algorithms

Also, going further into the algorithm, a Maximum filter algorithm is performed on our obtained frequencies. In a nutshell, every pixel of the spectrogram is being set to the local maximum by looking at its neighbourhood pixels. The example is given in the picture:

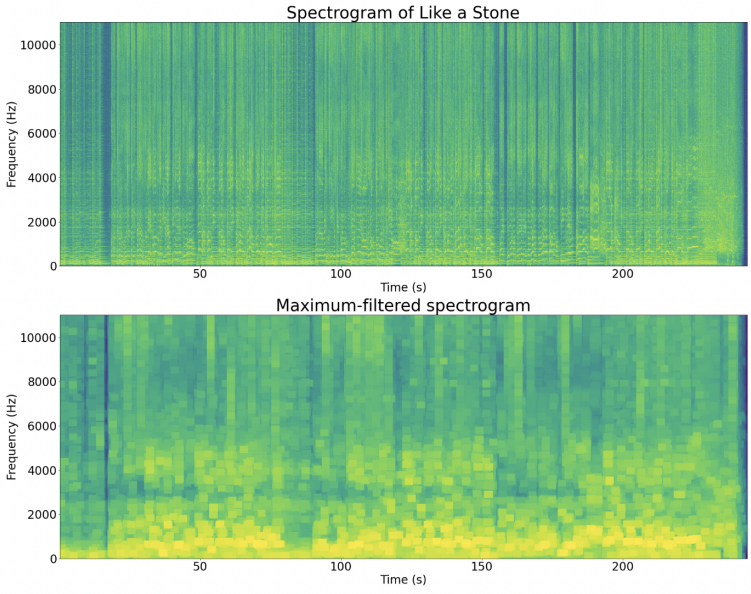


Figure 2: Spectrogram and Maximum-filtered spectrogram [4]

Then one has to find the location of the local peaks, as the Maximum filter algorithm only emphasises them. The concept behind this technique is that the local peaks have been used to replace all the non-peak points in the spectrogram, resulting in a modification of their values. The peaks are the only points in the spectrogram that remain unchanged; therefore, they are candidate peaks for the Constellation map. Below one can see how those peaks (Fig. 3: white ones are chosen from a zoomed section of the above spectrogram).

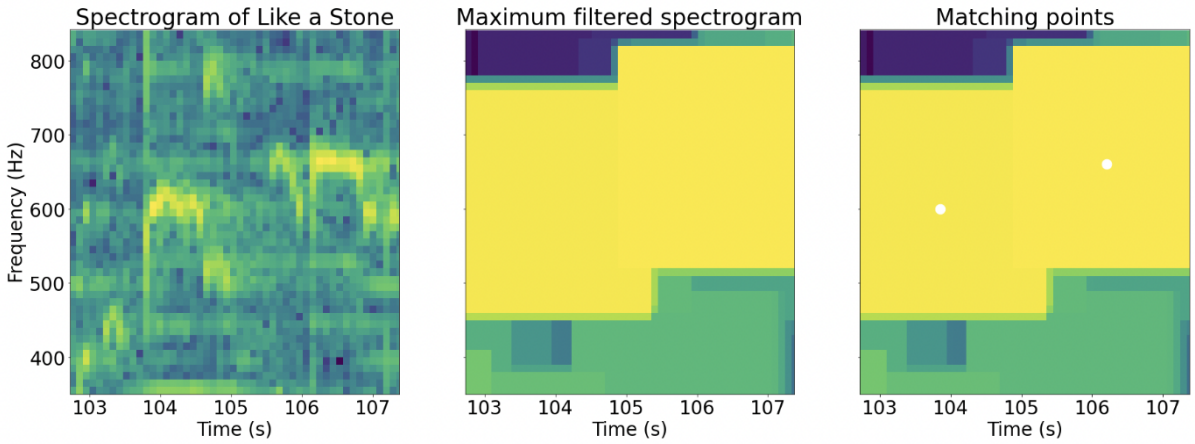


Figure 3: Finding of filtered peaks (frequencies) [4]

If we plot our filtered peaks together we will get the kind of Constellation map (Fig. 4).

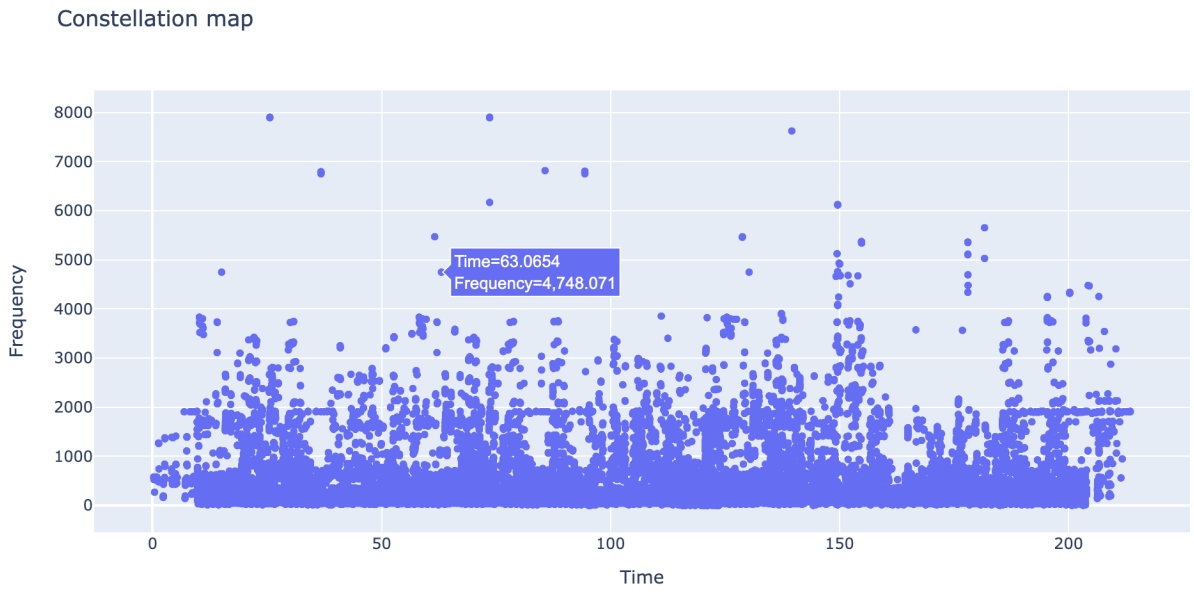


Figure 4: Constellation map

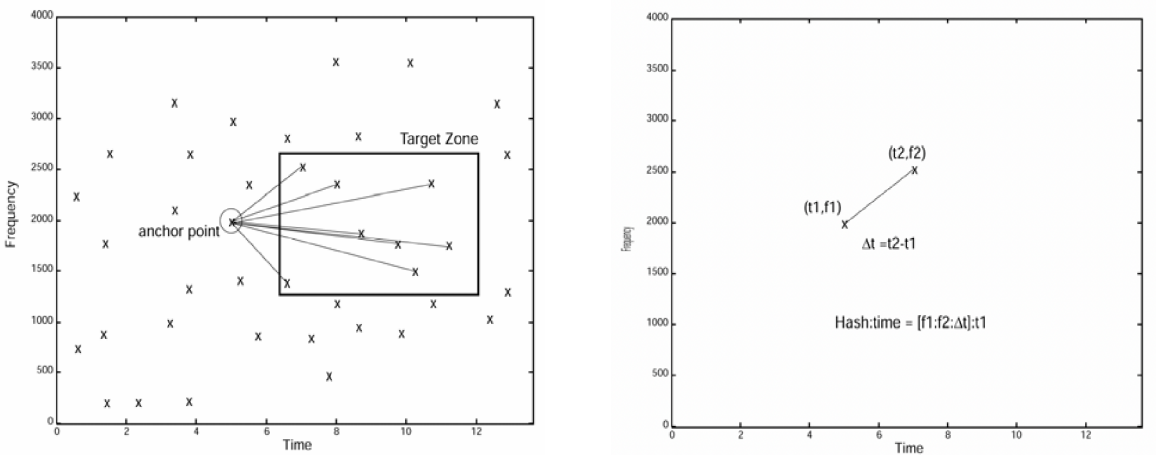
In addition, if the one wants reduce the size of fingerprints which will be stored in the database, especially when there is millions of song, which need to be stored, some peaks can be discarded. Despite the fact that the size of each fingerprint will be smaller, fewer peaks can cause decrease in probability to find a match for the correct song. In our implementation we would omit that part of the filtering in favour of more accurate work of the algorithm.

4.1.3 Hashing

Having a unique fingerprint is crucial as it significantly enhances the speed of searching and enables the identification of more songs. Shazam addresses the challenge of uniqueness by generating hashes from pairs of peaks. Pairing points offers the advantage of significantly enhancing uniqueness compared to individual points. When two points are paired together, the resulting combination becomes much more distinct.

As for Shazam's algorithm of Music Recognition, it suggets the following way of hashing.

1. Take the point from Constellation map, that will be the anchor point.
2. Find out the size of target zone for every anchor point.
3. Pair each point from target zone to the anchor point.



(a) Anchor point with its target zone

(b) Hash details

Figure 5: Hashing [3]

Then, as the one can see from Fig. 5 (b) we need to hash the first three parameters. The fourth would point out the second when the following hash was made. All together, the hashes creates fingerprint for the song, which further are written to database.

- frequency 1 (f_1) - frequency of the anchor point.
- frequency 2 (f_2) - frequency of the point from target zone corresponding to anchor point.
- $\Delta = t_2 - t_1$
- t_1

4.1.4 Matching

Given the database of grouped fingerprints by song, we can now perform the matching. After receiving the fingerprint of the recorded piece of music, we iterate through the songs and their fingerprints to check what hashes coincide with those in database. However, due to much noise we cannot guarantee that it perfectly lines up. Thus, while iteration

through the database, subsequently, we can compare the timing of when the hash appears in the original track with the timing of when the same hash appears in the sample, simply doing the difference of them. Then, the number of coinciding differences is counted and compared to other songs' differences, and possible match is shown as the potential name of song of the given fragment.

4.2 Pros and cons of the algorithm

The pros of Shazam's algorithm is the **high accuracy** in identifying songs, even in the presence of noise and distortions. It is known for its efficiency, providing quick results by matching audio fingerprints against a large database of songs (with appropriate hashing). In addition, it demonstrates robustness by recognizing songs despite slight variations in audio quality, encoding, tempo, or pitch.

There are less cons, but it is limited to recognizing songs that are present in its database, potentially missing rare or lesser-known tracks. Also, the algorithm may face challenges in accurately matching manipulated or distorted versions of songs. But, in general, the algorithm in its way works very well for recognition of all kinds of songs.

5 Pseudocode

As the one can see, the main parts of the algorithm are fingerprinting the song and matching it through database.

Creating database:

Data: Songs

Database of the songs fingerprints

For song in **songs**:

———frequency_domain = **fft**(song)

———pitches = **extract_pitches**(frequency_domain)

———anchors = **maximum_filter**(pitches)

———**For** anchor in **anchors**:

—————target_zone = **target**(anchor)

—————**For** peak in target_zone:

———————-database[song] add hash(anchor_freq, peak_freq, peak_time-anchor_time), anchor_time

end; **end**; **end**;

That pseudocode is also works with recorded fragment, except for the writing to database, instead of that it is processed through the matching algorithm which was discussed above.

6 Testing

What is very important is to test if the algorithm is working on recognition of the fragment of the song. We decided to represent two kinds of test. The main difference between them is that the second option gives more real testing even if the accuracy is not so high, as every user of Shazam just record the song not in very perfect circumstances. On the other hand, the expectation from the tests of first option is much higher accuracy of recognition as it is simply the cut fragment of the song.

1. Take the fragment from the song and check for perfect match.

2. Record the song fragment and check for a match.

In addition, the only benchmark that can compete with our implementation is obviously Shazam app, that with 100% of probability will recognise the song if it is present in its database. In our case the probability will be much less.

7 Conclusions

Solving the problem of Music Recognition, using the approaches of such popular app as Shazam, we reach our goal of imlementation of its algorithm and imitation of database from which we can find matches to our songs or recorded fragments. The main part of Linear Algebra is actually Fast Fourier Transform and its application, that helped a lot with beginning of fingerprinting. Also, it is worth mentioning filtering algorithm, more accurately maximum-filter which however is used mostly for Image Denoising but was also useful for Music Recognition algorithm.

Although it is not the full algorithm of Shazam, it is open for further improvements that can lead to better recognition, fingerprinting and even database. In other words the project is good basis for developing.

References

[1] https://colab.research.google.com/drive/1uLgSVZI9cQo9YKVIHiyRurEbvW2PNoGu?usp=sharing

[2] https://github.com/otecdima/LA_Project

[3] Wang, Avery Li-Chun . "An Industrial-Strength Audio Search Algorithm." https://www.Ee.Columbia.Edu/~Dpwe/Papers/Wang03-shazam.Pdf.

[4] MacLeod, Cameron. "Cameron MacLeod's Profile Picture." https://www.Cameronmacleod.Com/Blog/How-does-shazam-work, 19 Feb. 2022.

[5] Jovanovic, Jovan. "Toptal Engineering Blog." Toptal Engineering Blog, 2 Feb. 2015, www.toptal.com/algorithms/shazam-it-music-processing-fingerprinting-and-recognition

[6] Drevo, Will. "Audio Fingerprinting with Python and Numpy." Audio Fingerprinting with Python and Numpy, 15 Nov. 2013, https://willdrevo.com/fingerprinting-and-audio-recognition-with-python/.

[7] Rijn, Roy V. "Creating Shazam in Java." 1 Jan. 2010, www.royvanrijn.com/blog/2010/06/creating-shazam-in-java/.