Acoustic Fingerprinting

Soundz

Jake Runzer

June 28, 2018

Outline

Overview of Acoustic Fingerprints

An audio fingerprint is a compact signature that summarizes an audio signal.

Requirements for Acoustic Fingerprints

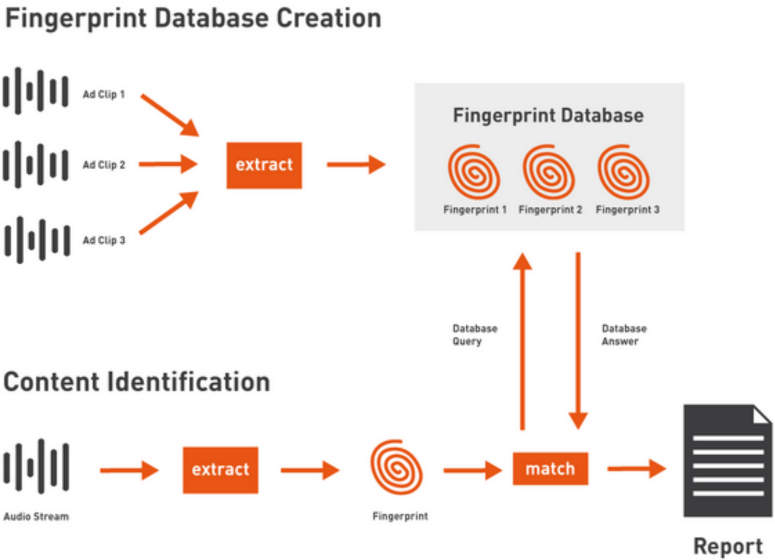
A fingerprint should have the following properties

- Is unique to that specific audio signal
- Does not depend on the binary representation of the audio
- Represents how humans hear the audio

Overview of Music Identification (I)

Use a database of fingerprints belonging to known sources to identify a fingerprint belonging to an unknown source.

Overview of Music Identification (II)



Examples

You have probably have used or know of apps that used audio fingerprinting for music identification

- Shazam
- Soundhound

Requirements for Music Identification

Music identification is often done on mobile devices in noisy environments

- Size of data generated is as small as possible
- Low computational footprint
- Length of audio required to get match is short (<10 sec)
- Noise/distortion agnostic

Typical Pipeline

1. Capture audio on mobile device
2. Create fingerprint on device and send to matching server
3. Database is normally inverted index of fingerprint -> song
4. Approximate nearest neighbour search is performed to find best candidates
5. Temporal alignment step applied to most similar matches
6. Return best matched song to mobile device

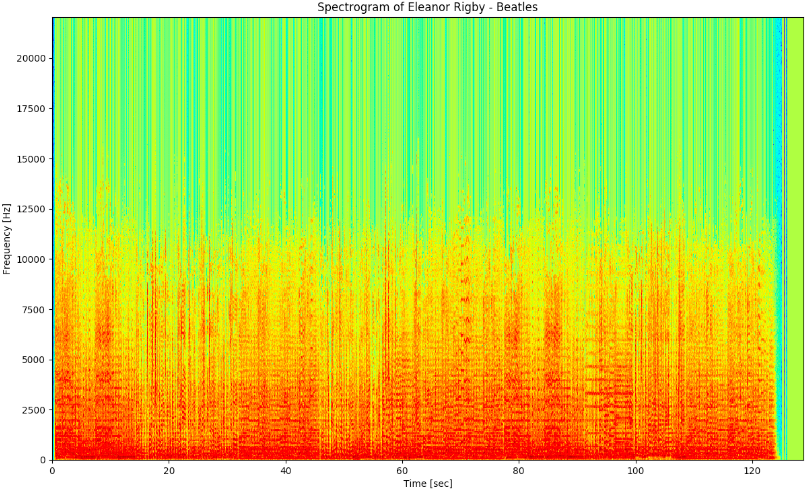
Why Spectrograms?

Almost all fingerprinting techniques rely on audio spectrograms

- More closely represents how humans hear audio compared to the binary representation
- Time and frequency resolution can be adjusted to make algorithm more robust to noise

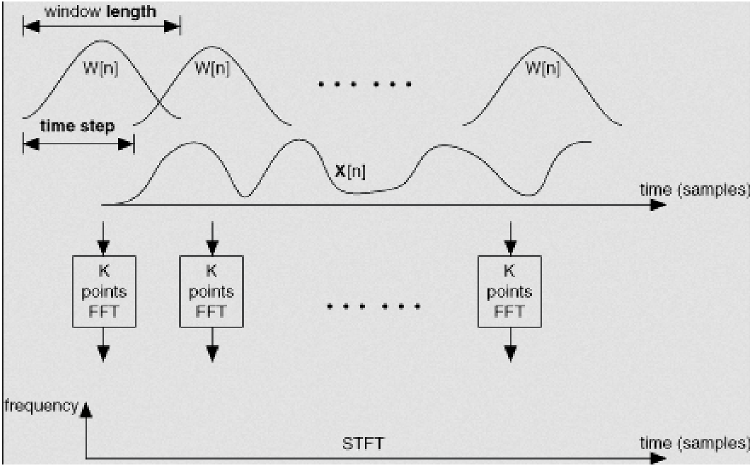
What are Spectrograms?

Visual representation of the spectrum of frequencies of sound as they vary with time.



Short-Time Fourier Transform (STFT)

Used to determine the frequency of local sections of a signal as it changes over time. An overlapping window is moved over the audio. At each step the Fourier Transform is computed using FFT.



STFT Parameters

These values can be configured and modified to change how the spectrogram is generated

- Window length
- FFT Length
- Overlap amount

Approaches

A few common approaches that have been used. All rely on audio spectrograms.

- Computer vision based
- Wavelet based
- Peak based

Computer Vision for Music Identification

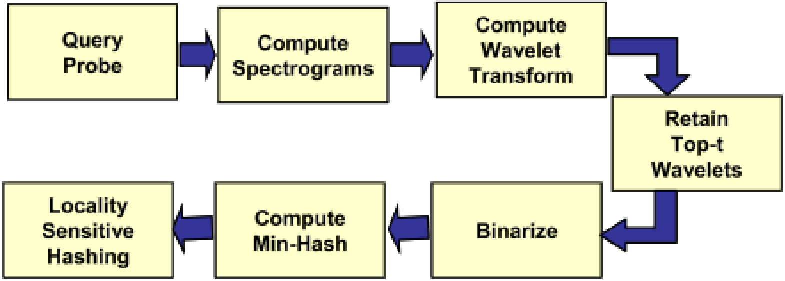
Intuition: 1D audio signals can be processed as conventional images when viewed in the time-frequency spectrogram representation.

- Spectrogram is treated as set of overlapping images
- Train AdaBoost classifiers on box-filters
- Output of classifier is binary value representing the differences between values aggregated in two sub-rectangular regions
- Use concatenated output of classifier as fingerprint

https://ieeexplore.ieee.org/document/1467322/

Wavelet-Based

- Compute overlapping spectrogram images
- Decompose images using multi-resolution Haar wavelets
- Retain only top-t wavelets, where t is much smaller than the size of spectrogram
- Only keep sign information
- Compare two spectrograms by computing byte wise Hamming distance



https://www.sciencedirect.com/science/article/pii/S0031320308001702

Peak-Pair Hashing

The original Shazam algorithm. Look only at spectrogram peaks

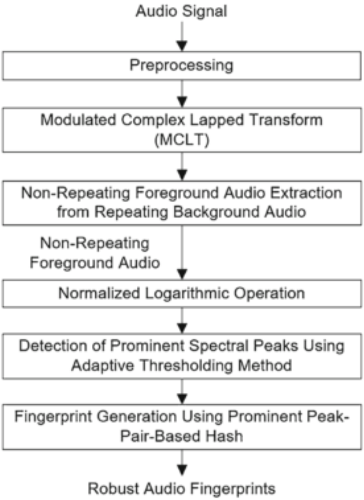
- Peaks are more likely to survive ambient noise
- A peak analysis of music and noise together will contain spectral peaks due to the music and noise as if they were analyzed separately
- Look at pairs of peaks and create lots of fingerprints per audio sample

https://www.ee.columbia.edu/~dpwe/papers/Wang03-shazam.pdf

Peak-Pair Improvements

Improvement on Wang's algorithm (peak-pair hashing)

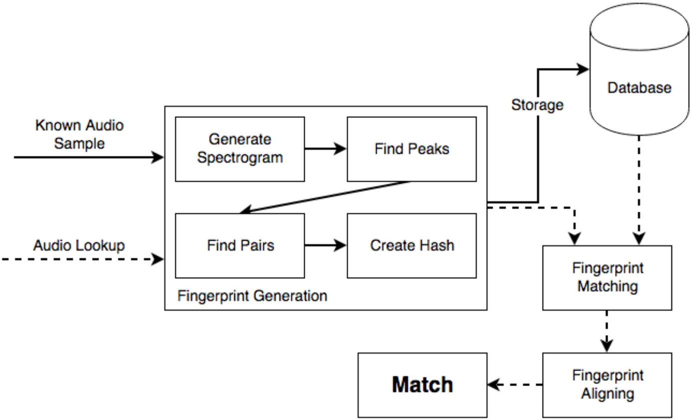
"Fingerprints are generated using a modulated complex lapped transform-based non-repeating foreground audio extraction and an adaptive threshold method for prominent peak detection".



My Implementation

I implemented the music identification using peak-pair hashing (Shazam original algorithm) in Python using the Numpy and Scipy libraries.

Architecture



Audio Fingerprinting and Identification

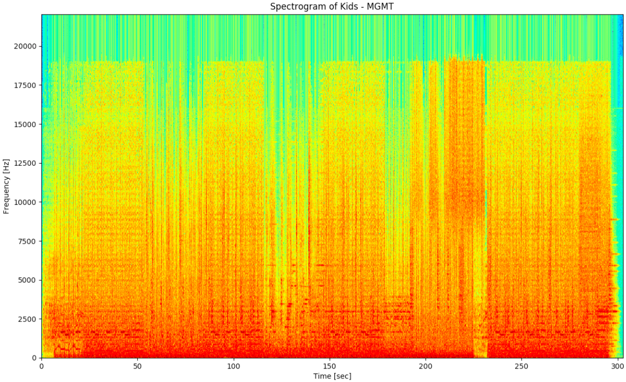
Working Example

Throughout the next slides we will look at the song "Kids" by "MGMT"
https://www.youtube.com/watch?v=aBd46BbdTfs

Spectrogram Creation

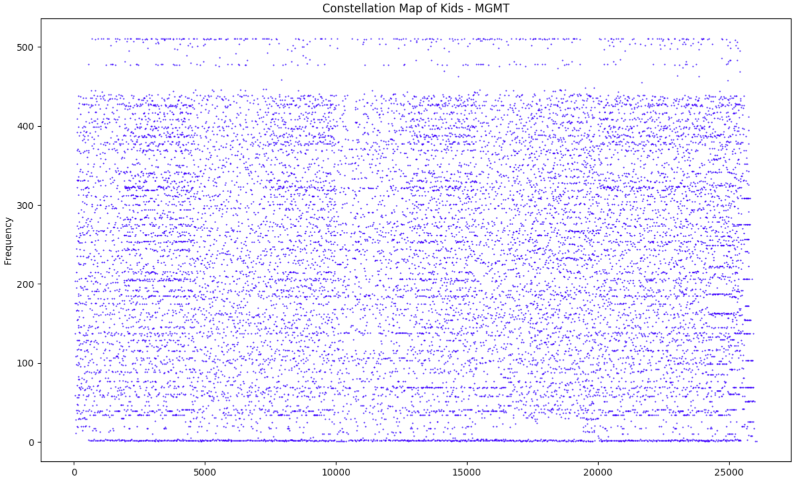
I use the following parameters to create the spectrogram

- Window: Hamming
- Window size: 1024
- Overlap: 0.5
- FFT size: 1024



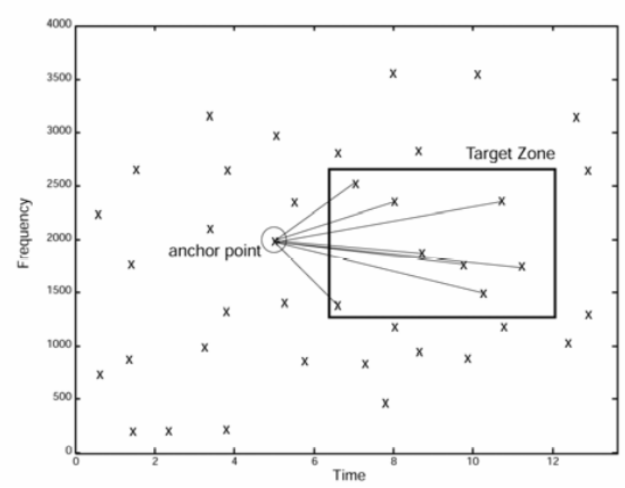
Constellations

Time-frequency peaks are found using an image local maxima filter with a neighbourhood of 15 pixels (freq + time axes). For *Kids*, there are 14425 peaks.



Finding Pair

For each peak, the closest 15 neighbouring peaks within 200 seconds create a pair. For *Kids*, there are 8514 fingerprints.



Creating Hashes (I)

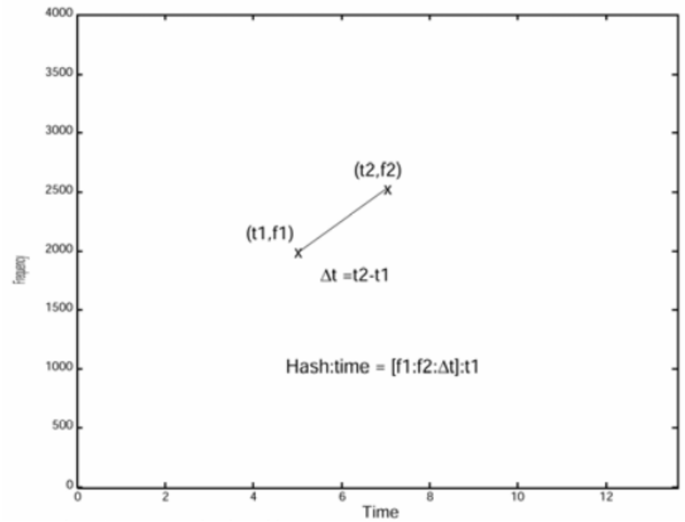
A hash is created for each pair (not a cryptographic hash). Each has is composed of

- the frequency of point 1
- the frequency of point 2
- the difference in their times

The hash is combined with the time offset of the first point, as it will be necessary for matching, to create a fingerprint.

```
fingerprint = hash:time = [f1, f2, t2 - t1]:t1
```

Creating Hashes (II)



Database

Information about the source songs and each fingerprint are stored in a PostgreSQL database.

Song

- Id
- Artist
- Album
- Title
- Track
- Year
- Duration

Fingerprint

- Id
- Hash
- Time offset
- Song Id

Identification

When an unknown audio sample needs to be identified,

- Fingerprints are created
- Matching fingerprints are retrieved from the database
- Fingerprints are aligned
- Song associated with best matched set of fingerprints is returned

Fingerprint Aligning (I)

- We cannot know the time offset the unknown audio was recorded at
- We can find matched fingerprints that occur successively after each other
- The time offsets from the unknown fingerprints are subtracted from the time offsets of the matched fingerprints

Fingerprint Aligning (II)

Diagonal is present where matched fingerprints occur successively after each other.

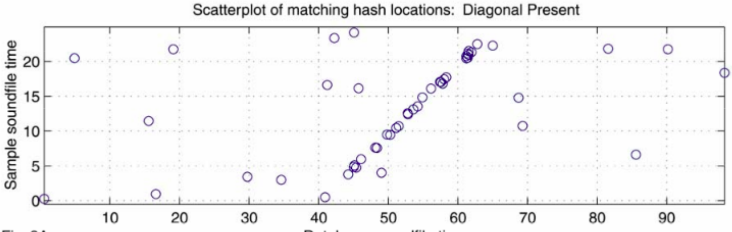


Fig. 3A

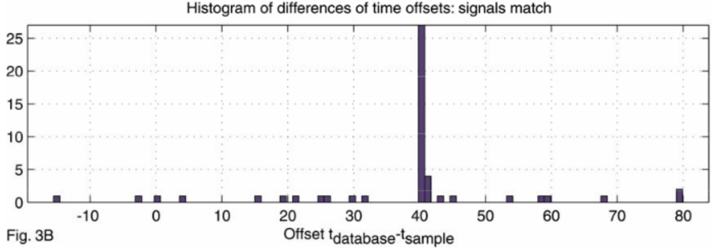
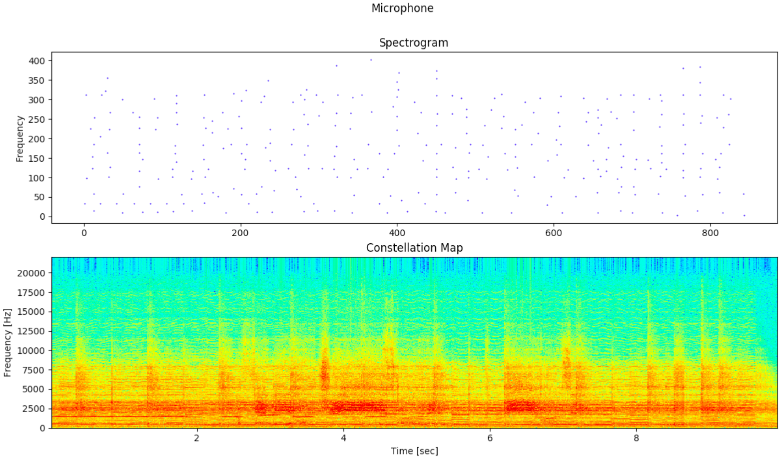


Fig. 3B

A Match!


Microphone
Spectrogram
Frequency
400
350
300
250
200
150
100
50
0
0
200
400
600
800
Constellation Map
Frequency [Hz]
20000
17500
15000
12500
10000
7500
5000
2500
0
2
4
6
8
Time [sec]

Source

The source code can be found on Github.
github.com/coffee-cup/soundz

Demo

And now... a demo!

Thanks

Thanks for listening!

References

- A review of audio fingerprinting
- Computer Vision for Music Identification
- Waveprint: Efficient wavelet-based audio fingerprinting
- A Review of algorithms for audio fingerprinting
- Survey and evaluation of audio fingerprinting schemes for mobile query-by-example applications
- Landmark-based music recognition systems optimisation using genetic algorithms
- An Industrial Strength Audio Search Algorithm
- Robust audio fingerprinting use peak-pair-based hash of non-repeating foreground audio in a real environment