A study of Shazam's Audio Recognition

Presentation · December 2016

DOI: 10.13140/RG.2.2.21768.21766

1 author:

Guendalina Palmirotta
University of Luxembourg

A study of Shazam's Audio Recognition

Seminar in Data Science

PALMIROTTA Guendalina

University of Luxembourg,
Faculty of Science, Technology and Communication,
Master in Mathematics
Winter semester 2016-2017

December 21, 2016

Motivation – Introduction

1. Capture the music for a few seconds (5-15s)
2. Identification of the song
3. Display the information (name, artist, album)

Goal

Recognize our unknown song in a short time using Shazam music application

Difficulty and constraints

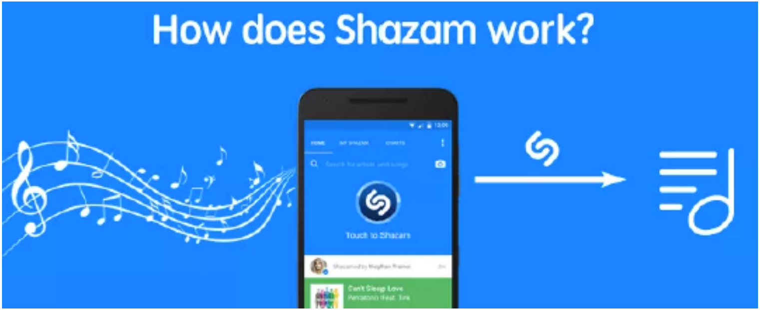
Develop an algorithm that is able to:

- Capture by a little microphone a short sample of music
- Often with mixed heavy ambient noise
- Quick identificaton over a large database of music $\Rightarrow$ 2M tracks

Keywords:

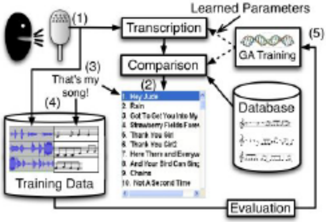
Roboustness, Noise resistanceness and Speedness

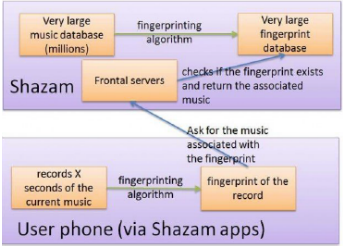
Seems to be magic, but...



Overview

Shazam two sides : 'Client' side and the 'Server' side

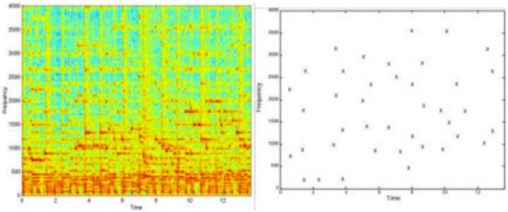




Step by step – Continous signal to discrete signal

Step 1

The song must be transformed in a time-frequency graph, that we call spectrogram. Then we do a kind of filtration, we get a constellation map and keep only the 'important points'.



How to get a spectogram?

Goal: From digital sound to frequency

Discrete Fourier Transform (DFT)

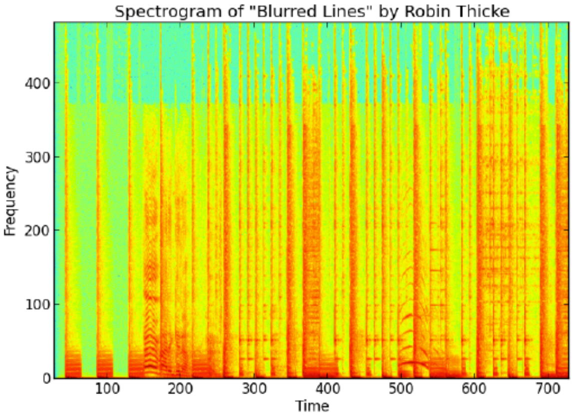
$$X(n) = \sum_{k=0}^{N-1} x[k] e^{\frac{-2\pi ikn}{N}},$$

- N size of (Hamming) window
- $X(n)$ the n-th bin of frequencies
- $x[k]$ the k-th sample of the audio signal

$\Rightarrow$ Use Fast Fourier Tranform (FFT) instead of the DFT

Spectogram filtering

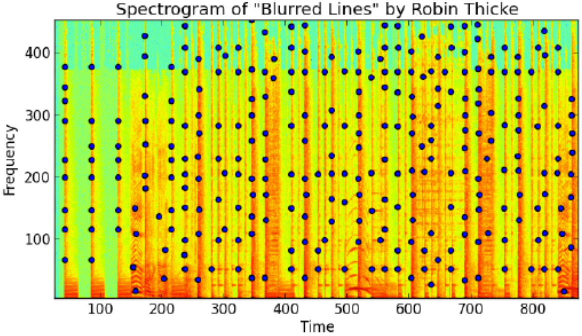
Combination of sinewaves at multiple frequencies



Filtration : Consellation map

What are the important nodes to consider?

- Depends on the coefficients of the bins
- Depends on the number of bands of the strongest time-frequency point



Step by step – Fingerprinting and Hash functions

Step 2

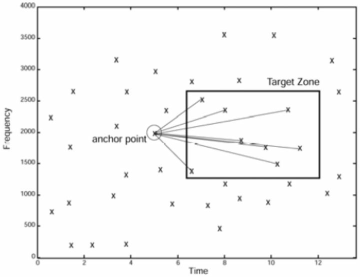
We code the song in a unique acoustic fingerprints and store it in a hash tag table.

How to store?

Use anchor point with their corresponding zone, called target zone

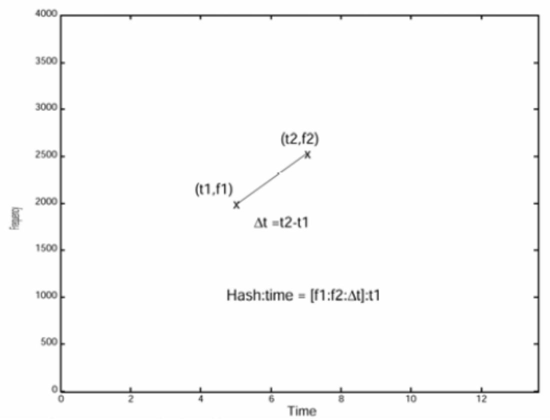
Idea

To look for multiple points at the same time instead of comparing each point one by one.



Hash details

Each time an anchor point lies inside the target zone → a hash is created.



Searching and Scoring the fingerprints

Fingerprint

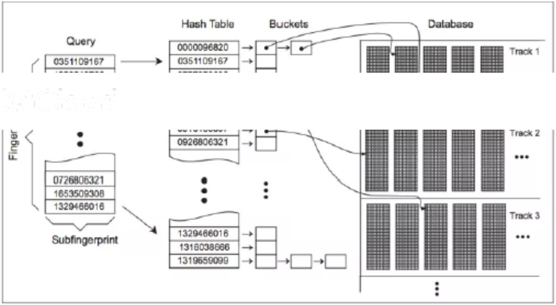
Unknown song: $[(f_1, f_2, \Delta t), t_1] \rightarrow [t_1]$

Update a new song: $[(f_1, f_2, \Delta t), t_1, ID] \rightarrow [t_1, ID]$

- (t_1, f_1) time-frequency at which the anchor point is located,
- (t_2, f_2) time-frequency at which the point in the target zone is positioned,
- $\Delta t = t_2 - t_1$ the time difference between t_1 and t_2,
- *ID* of the song (name, artist, album, of the song).

Hash function and table

$\Rightarrow$ Store in a hash tag table in the database



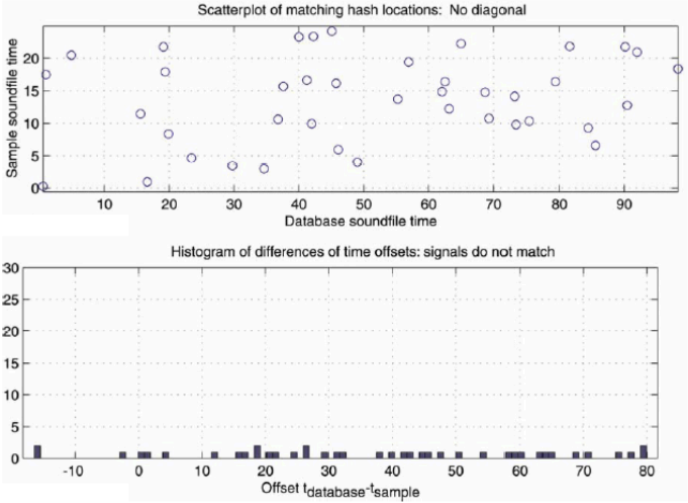
- Hash tag table: $\text{Hash}_{table} = [\text{Hash}(1), \ldots, \text{Hash}(n)]$
- Bucket is a specific location in database

Step by step – Matching of a song

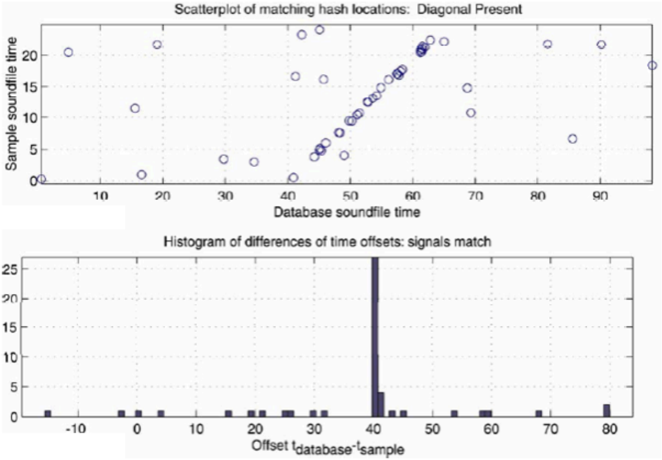
Step 3

For the matching factor we use a scatter graph and the corresponding histogram graph.

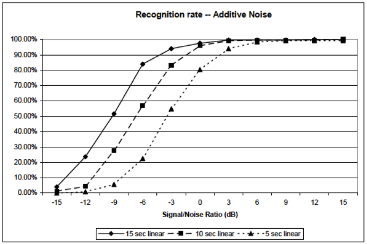
Scatterplot and Histogram of no matching

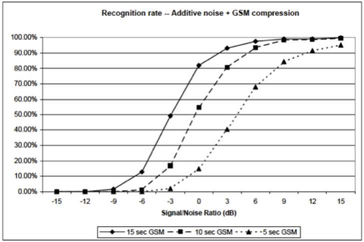


Scatterplot and Histogram of matching



Roboustness, Speedness and Noise resistanceness





- Identify the music with a rate of 90% of correctness in a short time with noise!


Thank you for your attention!