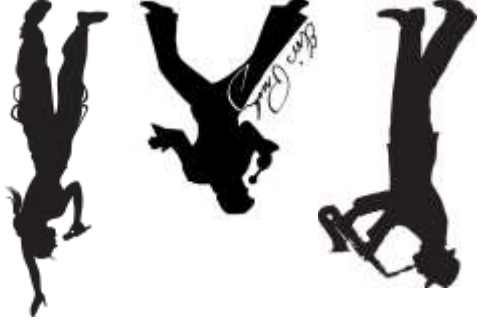


Ok Shazam, "**La-la-lalaa**"!

@itspoma




Google
how shazam wo
All
Images
Safe Search
I think it's Prodigy. . .
581 × 575 - thepoke.co.uk
How Shazam works
How Shazam works
VIA 9GAG.COM

Agenda

- What is a **signal**?
- Where the **fourier transformation**? **DFFT**? **FFT**?**sFFT?** can be used?
- How to get **spectrogram**?
- What is **energy picks** of sound?
- How to get **acoustic fingerprint**?
- So, how does **Shazam** works?
- Will show **a primitive Shazam-like app**

I’m

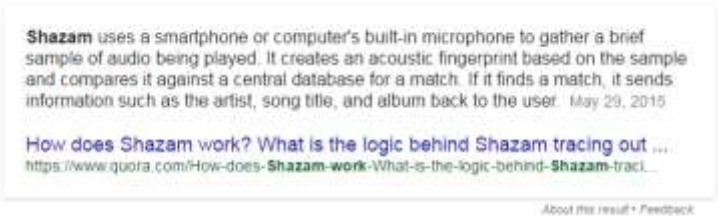
Roman Rodomansky

Software Engineer at Perfectial
FE Trainer at CURSOR Education (http://**cursor.education**/teacher/roman-rodomansky)
Co-Founder of GDG (Google Developers Group) Lviv (http://**lviv.gdg**.org.ua/)
Founder of UASC (Ukrainian Security Community) group (2009)
Founder of 2enota startup (2013)

https://**github**.com/itspoma
https://**facebook**.com/rodomanskyy
https://**linkedin**.com/in/rodomansky

Sound identification **apps**

- **Shazam** founded in 1999
  - uses audio in (ex. built-in microphone) to gather a brief samples (10s) from audio in
  - has more than 100 million monthly active users (\w more 500 million mobile devices)
- Similar apps
  - **SoundHound** (Midomi), Xiaomi Music, Musipedia
  - **Fire Phone** from Amazon
  - **Google** Sound Search, **Bing** Audio, **Yahoo** Music
  - **Sony** TrackID, Lyrics Mania, **Musipedia**, Omusic, Peach, etc
- About **~4000 pattents** https://patents.google.com/?q=music+identification…
  - how to collect, parse, identify, query, etc & etc



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

An Industrial-Strength Audio Search Algorithm

Avery Li-Chun Wang
avery@shazamteam.com
Shazam Entertainment, Ltd.

USA:
2925 Ross Road
Palo Alto, CA 94303

United Kingdom:
375 Kensington High Street
4th Floor Block F
London W14 8Q

We have developed and commercially deployed a flexible audio search engine. The algorithm is noise and distortion resistant, computationally efficient, and massively scalable, capable of quickly identifying a short segment of music captured through a cellphone microphone in the presence of foreground voices and other dominant noise, and through voice codec compression, out of a database of over a million tracks. The algorithm uses a combinatorially hashed time-frequency constellation analysis of the audio, yielding unusual properties such as transparency, in which multiple tracks mixed together may each be identified. Furthermore, for applications such as radio monitoring, search times on the order of a few milliseconds per query are attained, even on a massive music database.

1 Introduction

Shazam Entertainment, Ltd. was started in 2000 with the idea of providing a service that could connect people to music by recognizing music in the environment by using their mobile phones to recognize the music directly. The algorithm had to be able to recognize a short audio sample of music that had been broadcast, mixed with heavy

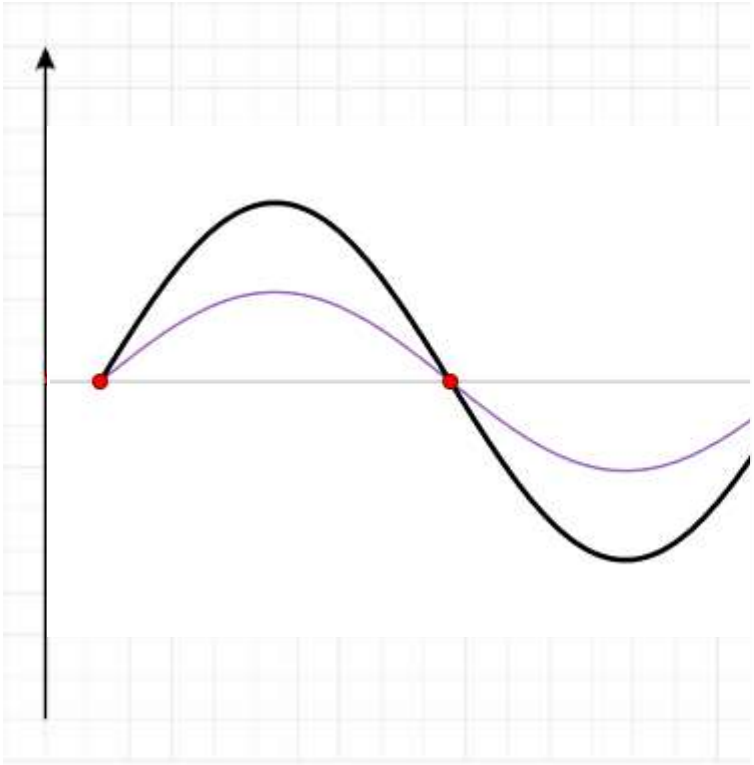
30-second clip of the song to a friend. Other services, such as purchasing an MP3 download may become available soon.

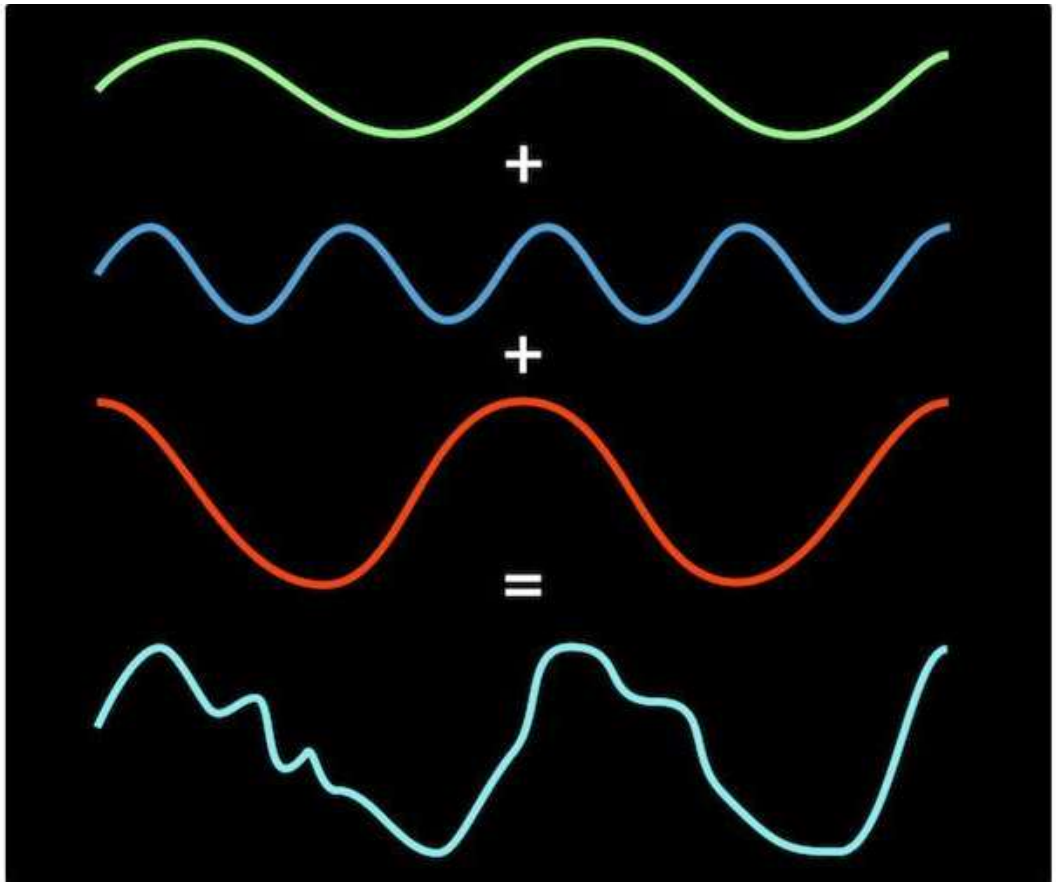
A variety of similar consumer services has sprung up recently. Musiwave has deployed a similar mobile-phone music identification service on the Spanish mobile carrier Amena using Philips' robust hashing algorithm [2-4]. Using the algorithm from Relatable, Neuros has included a

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

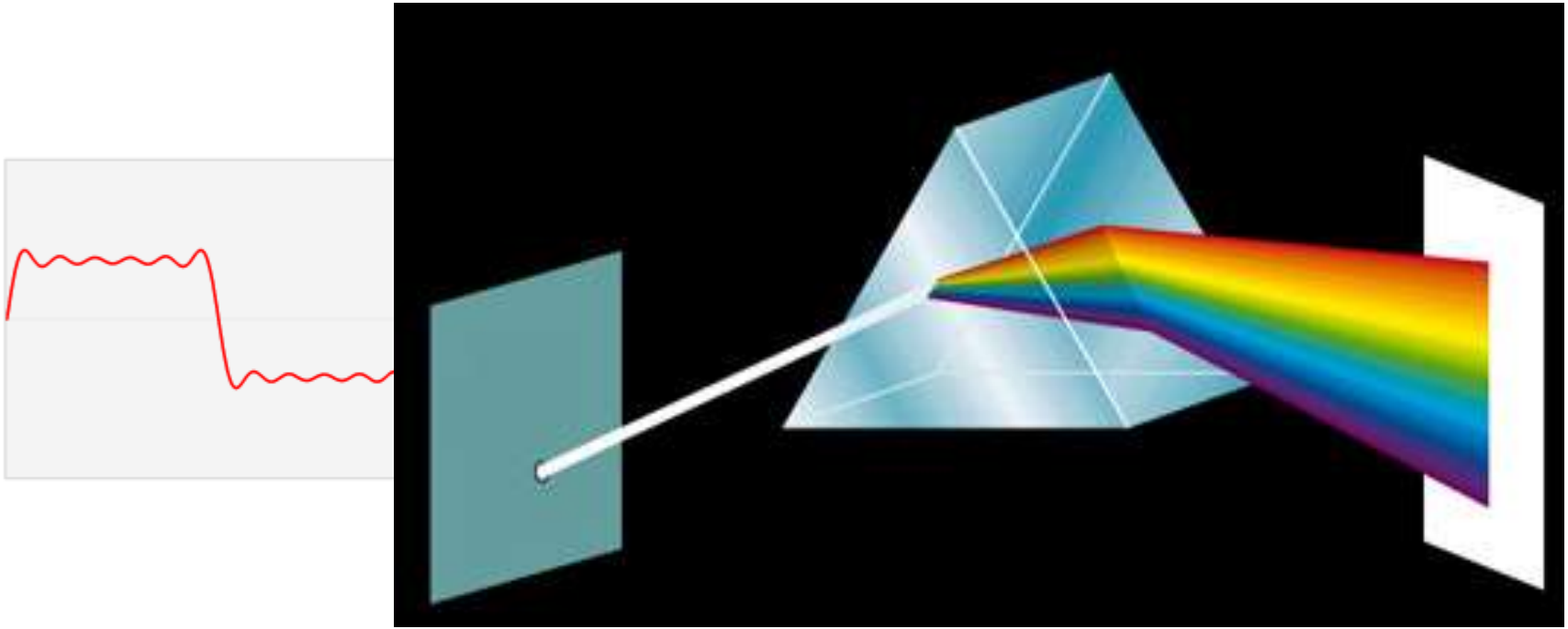
We have developed and commercially deployed a **flexible audio search engine**. The algorithm is **noise and distortion resistant**, computationally efficient, and **massively scalable**, capable of quickly identifying a short segment of music captured through a cellphone microphone in the presence of foreground voices and other dominant noise, and through voice codec compression, out of a database of over a million tracks. The algorithm uses a **combinatorially hashed time-frequency constellation analysis of the audio**, yielding unusual properties such as transparency, in which multiple tracks mixed together may each be identified. Furthermore, for applications such as radio monitoring, search times on the order of a few milliseconds per query are attained, even on a massive music database.

Audio Signal

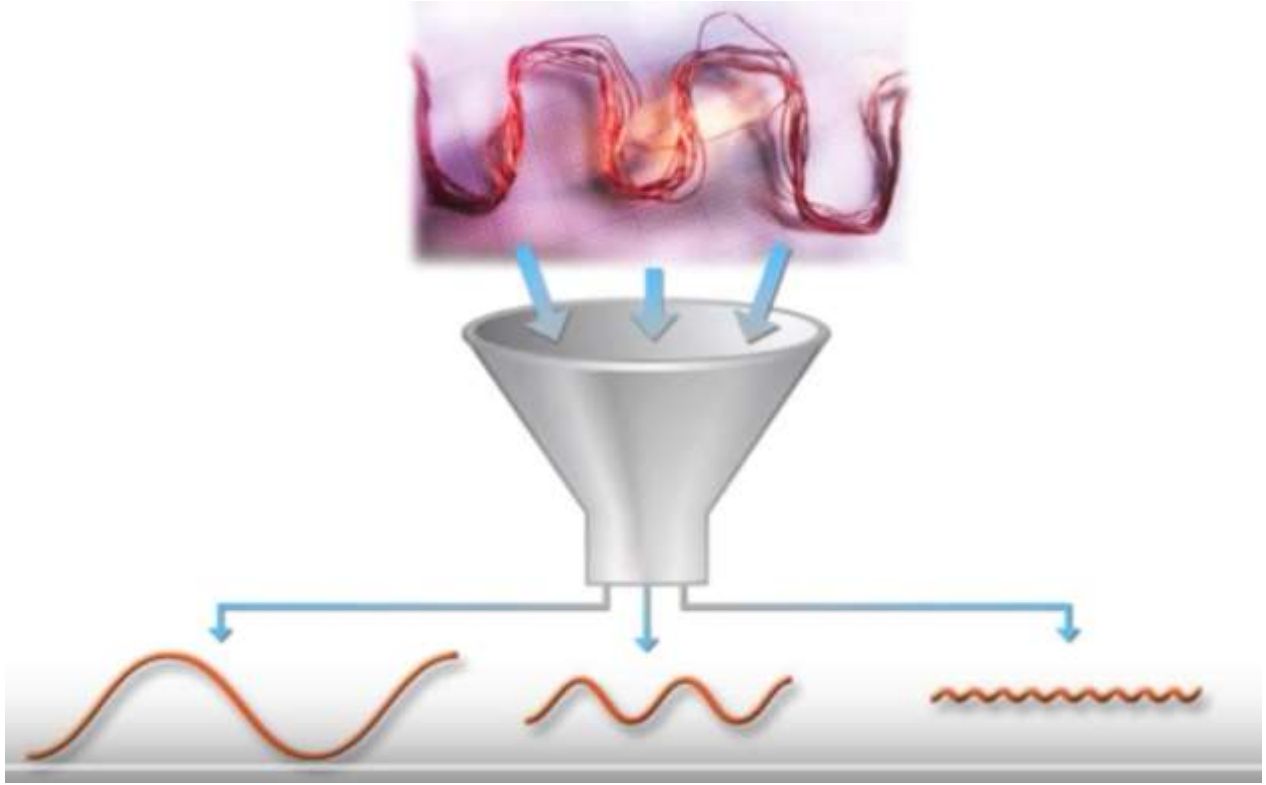




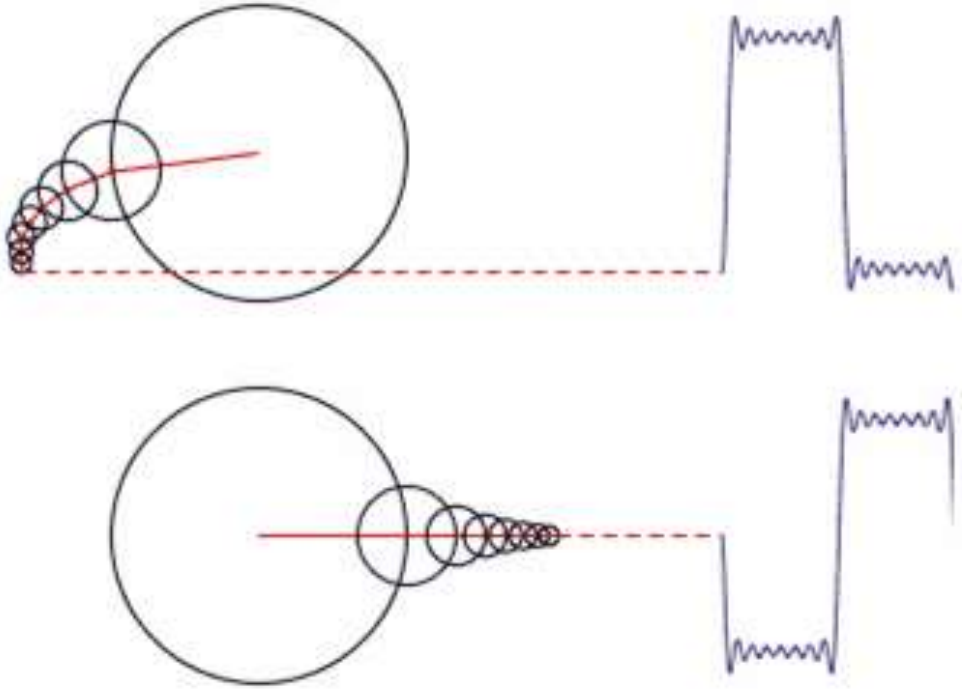
Fourier transform $\mathcal{F}$



Fourier transform $\mathcal{F}$



Fourier transform $\mathcal{F}$



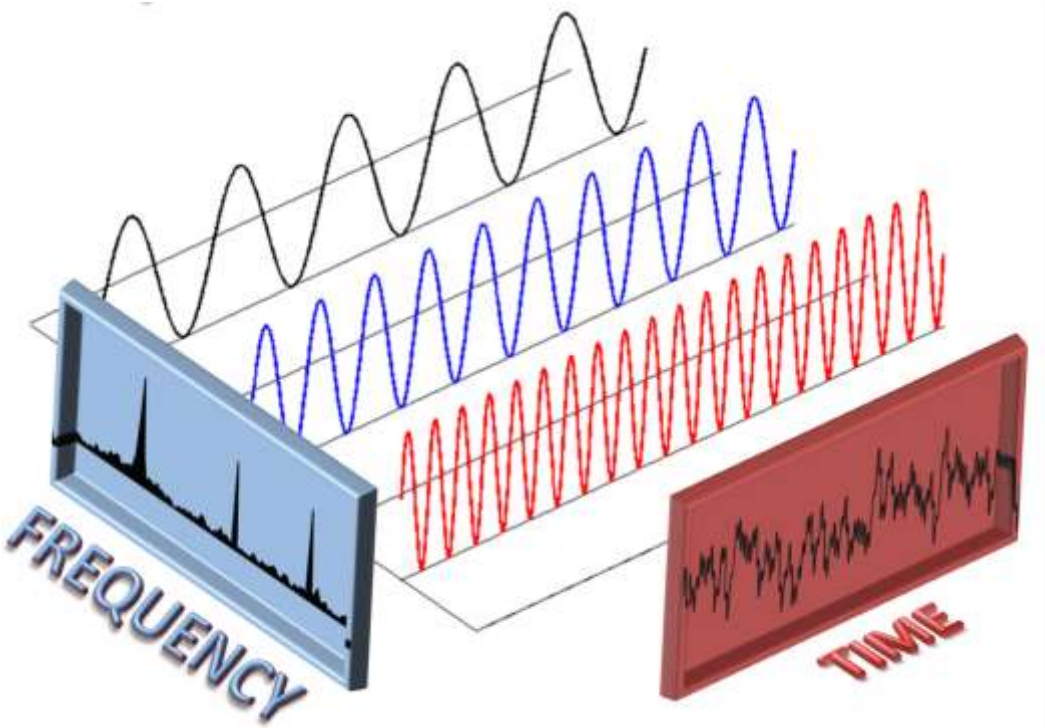
Fourier transform in real life

Discrete Fourier Transform (DFT)

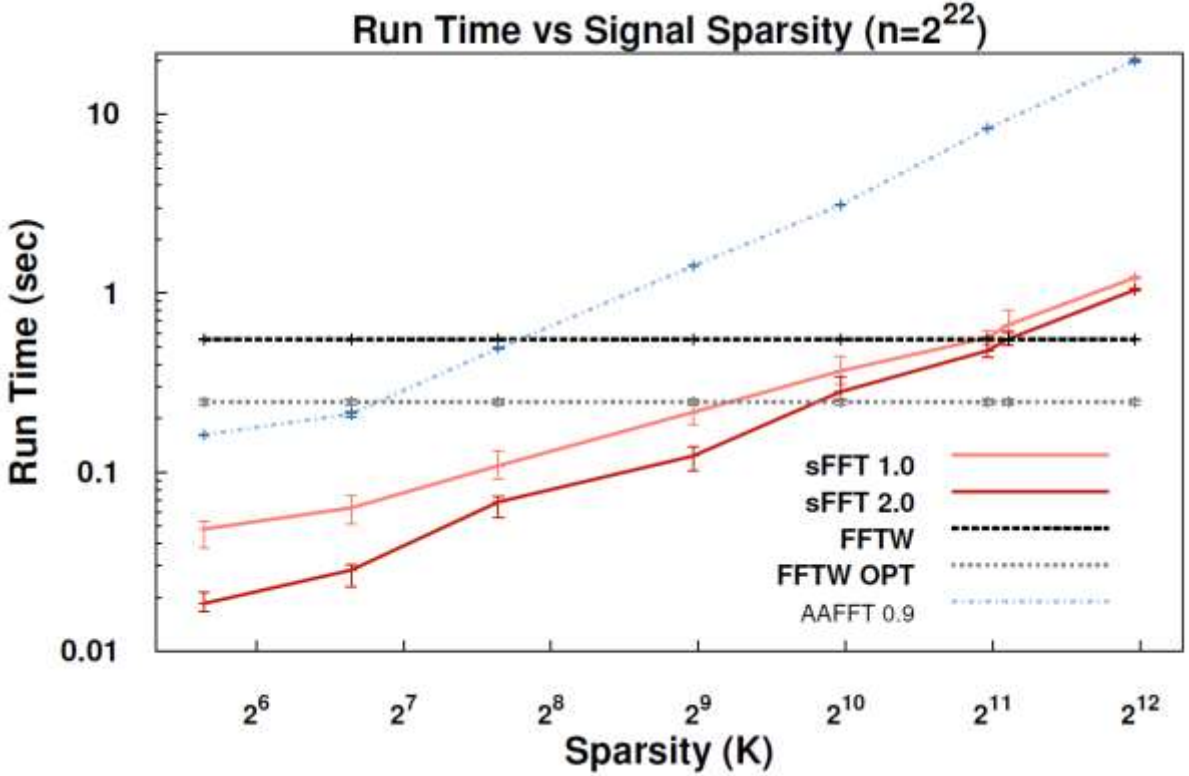
$$X_k = \sum_{n=0}^{N-1} x_n \, e^{-\frac{2\pi i}{N} kn}, k = 0 \dots N-1$$

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k \, e^{\frac{2\pi i}{N} kn}, n = 0 \dots N-1$$

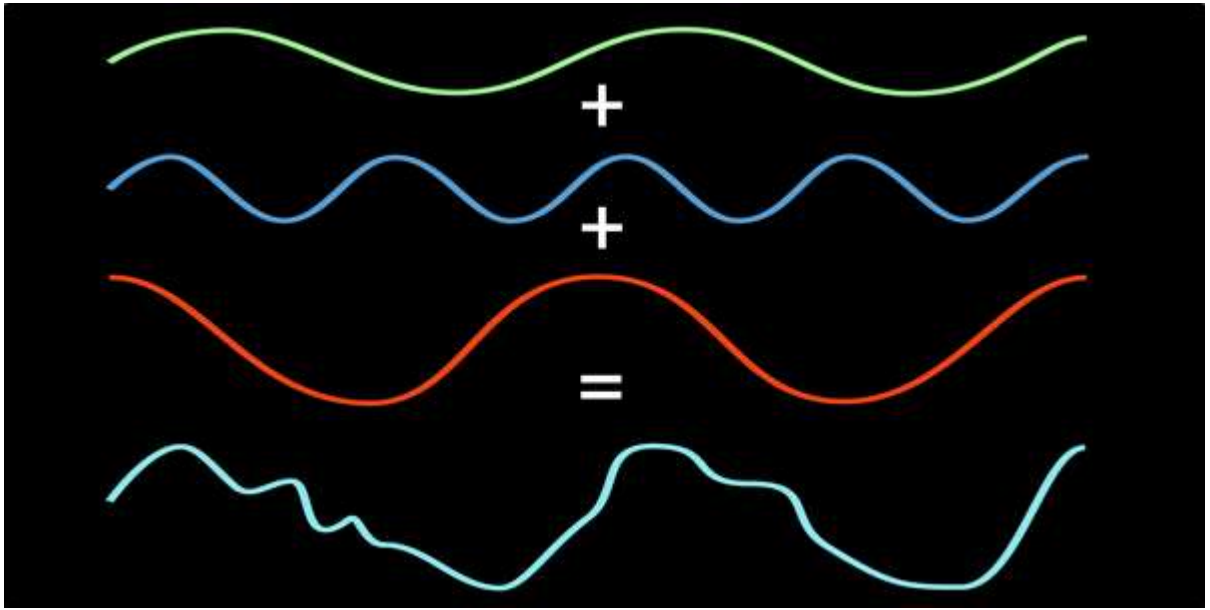
Fast Fourier transform



Fastest Fourier Transform in the West (FFTW)

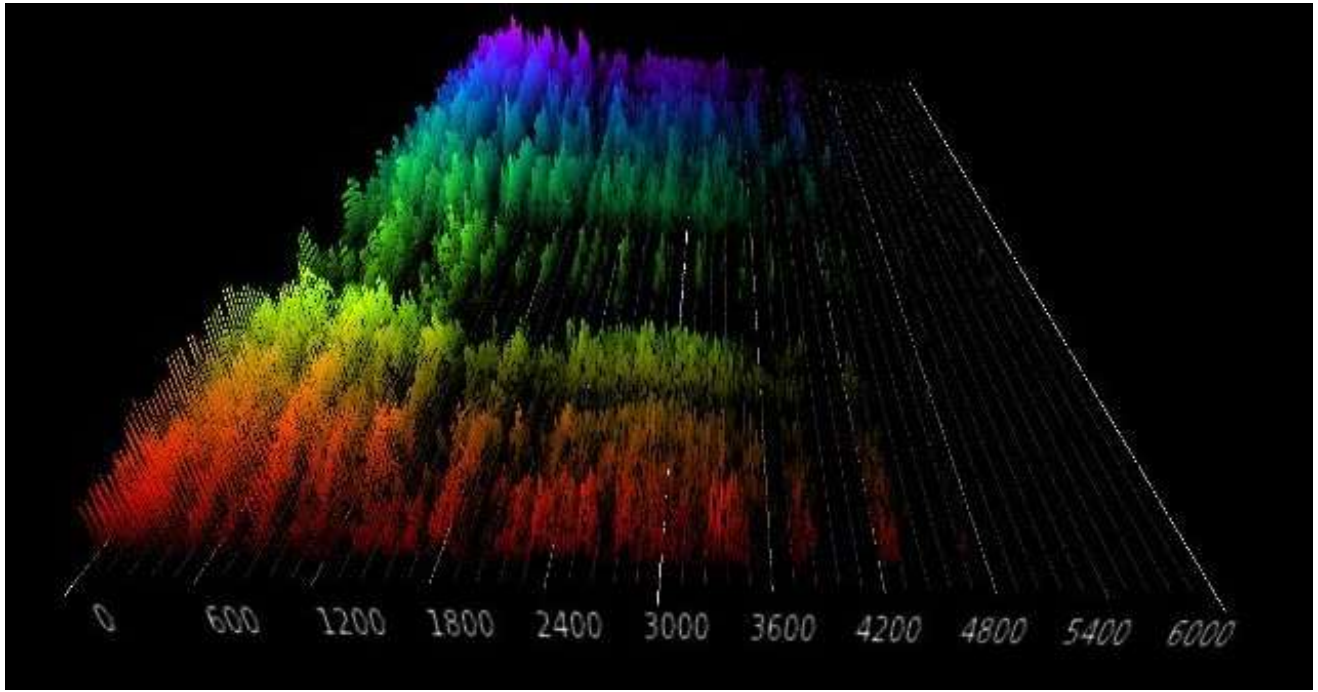


Sparse Fast Fourier Transform (sFFT)



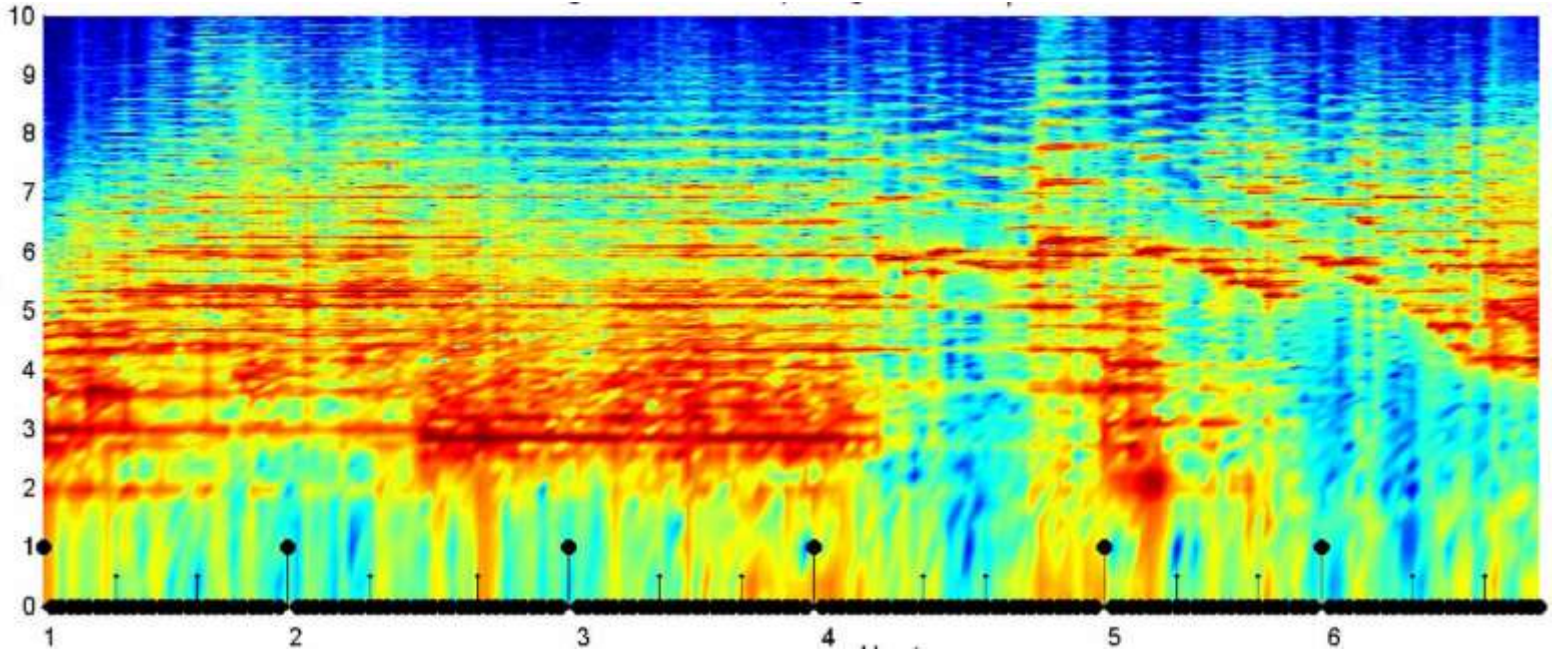
O(k log n)
O(k log k log(n/k))

Spectrogram



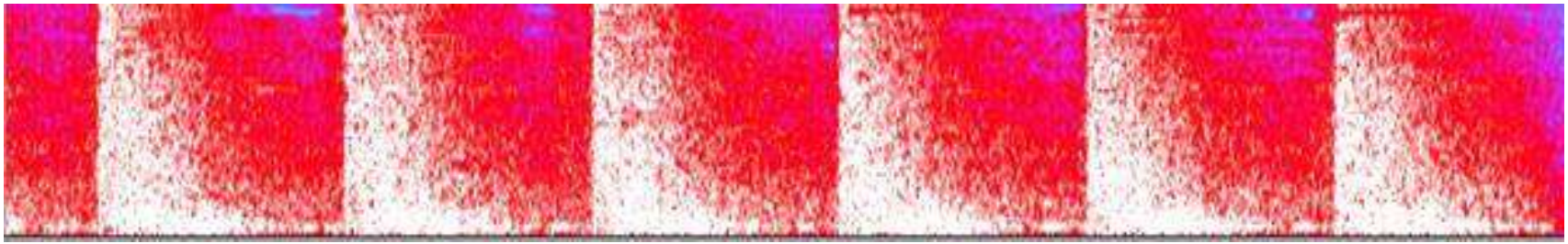
x = time, y = frequency, z = A

Spectrogram

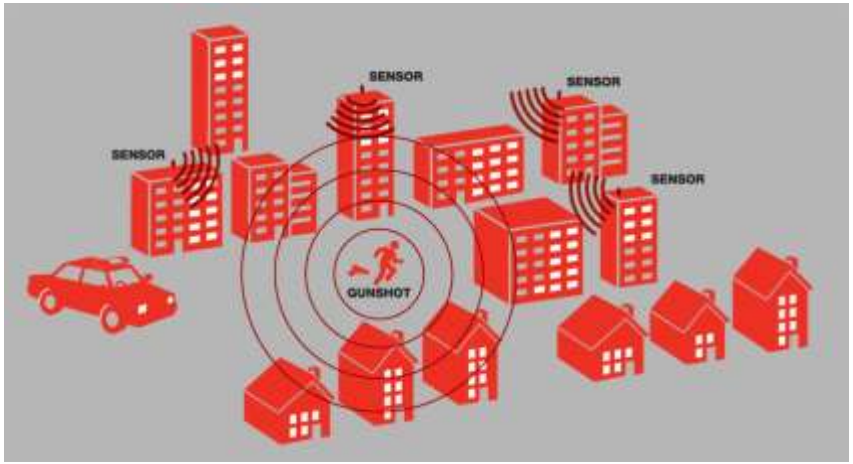


x = time, y = frequency, z = A

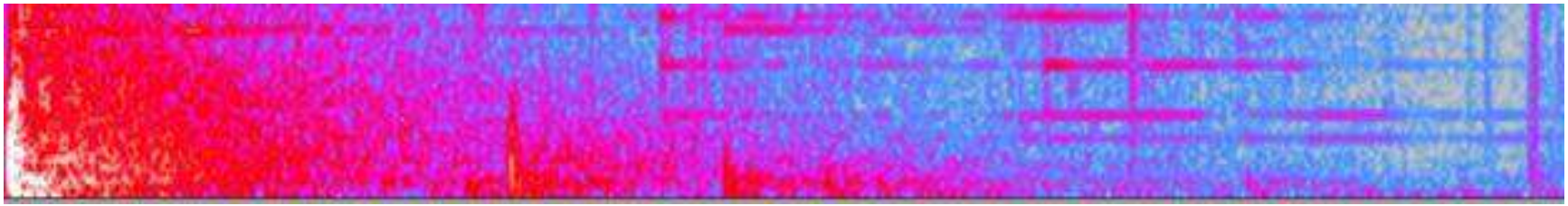
Spectrogram of **shots set**



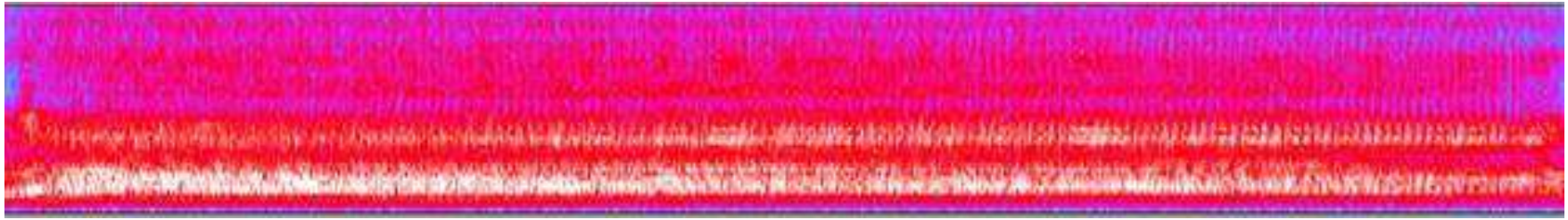
http://www.shotspotter.com/



Spectrogram of **broken window**

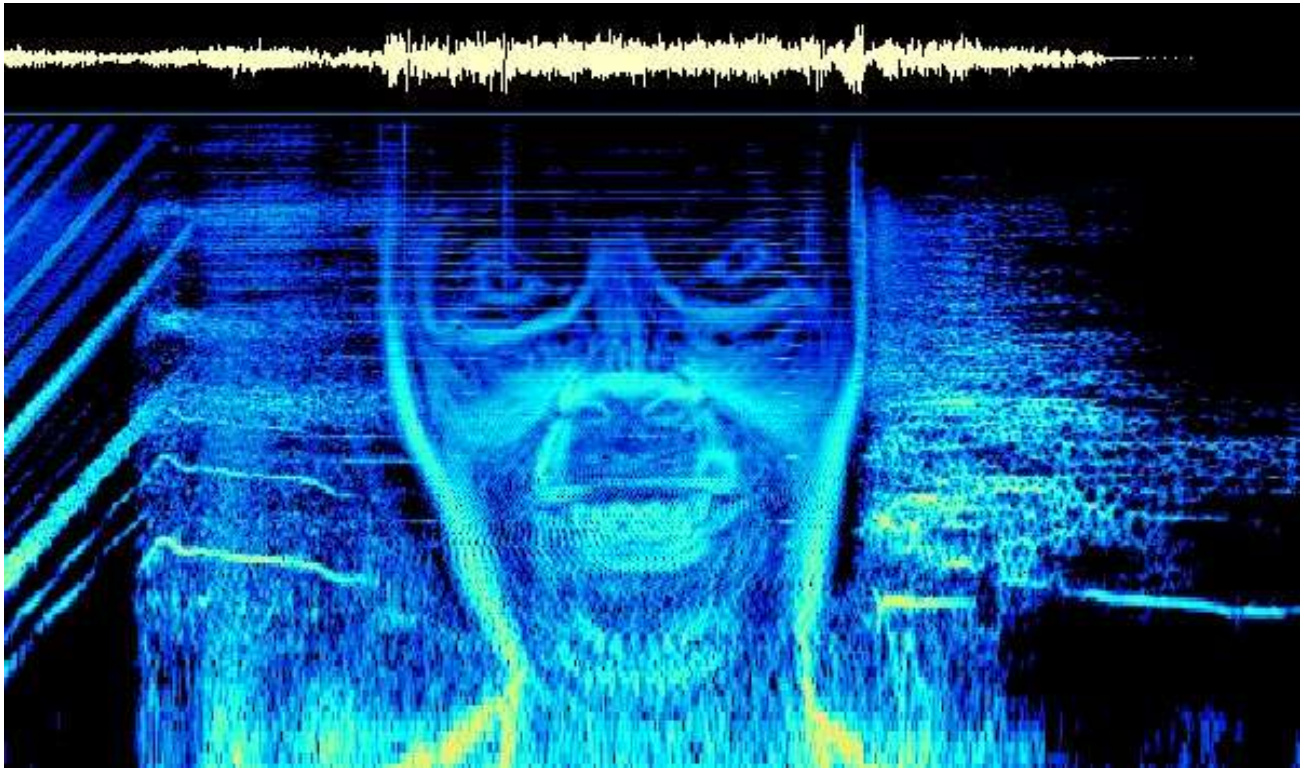


Spectrogram of **shout (cry)**



Spectrogram of **detecting shots & window in real-time**

Spectrogram in **songs**

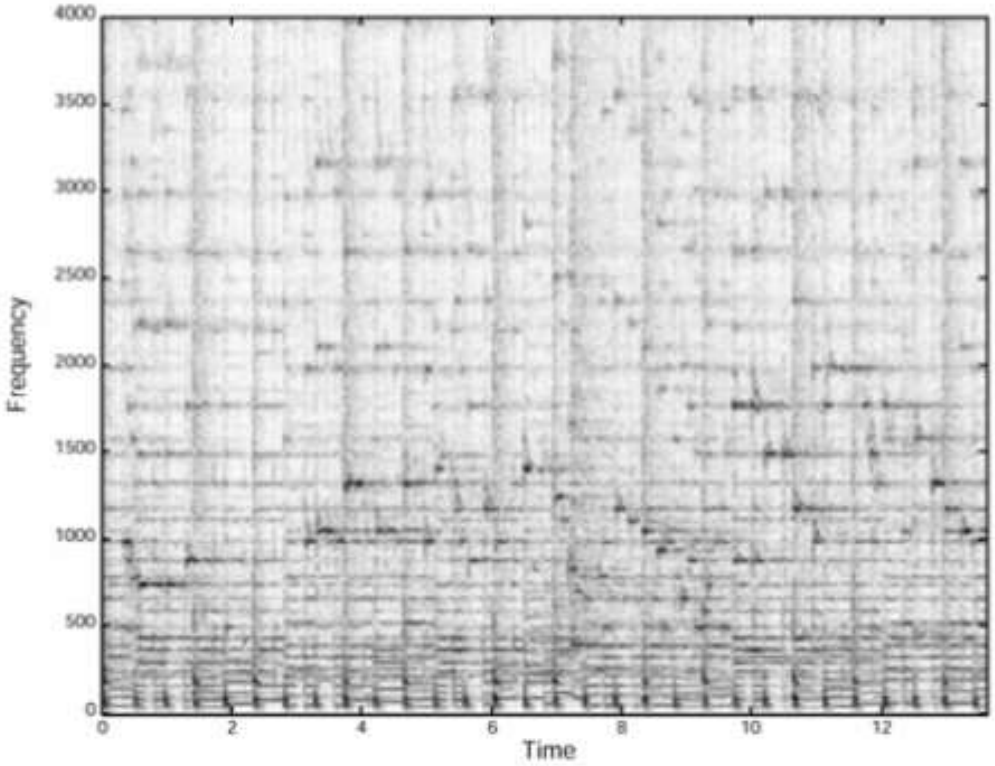


Aphex Twin
“Windowlicker” album

5:27

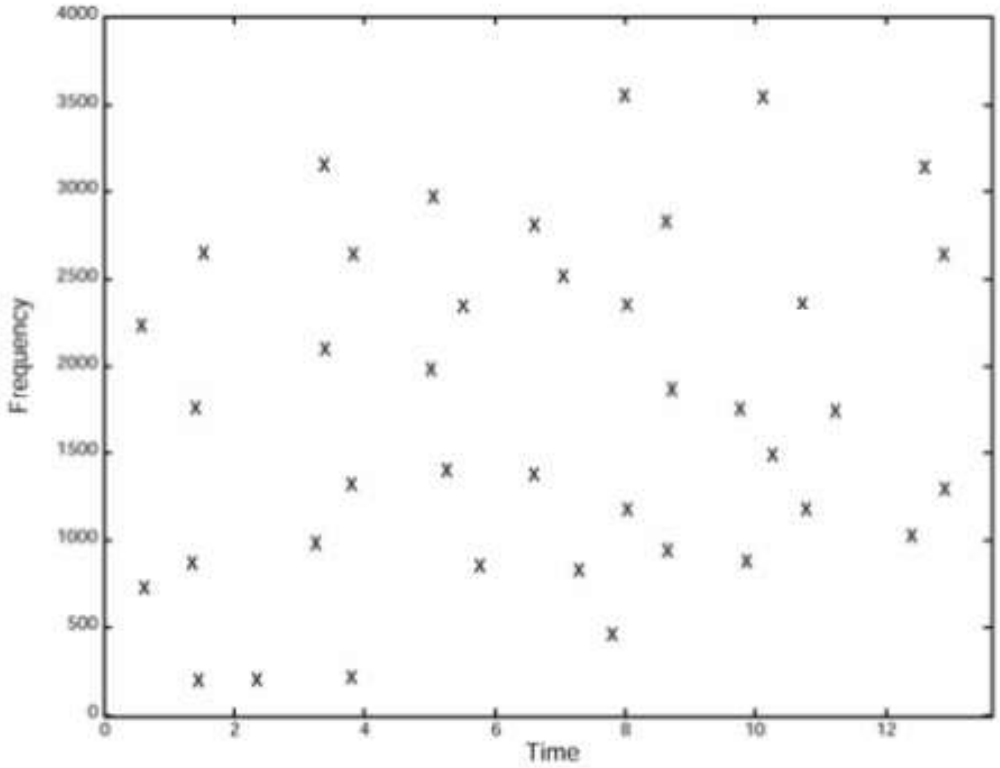
http://bit.ly/2bWucab

Spectrogram **of a piece of music**

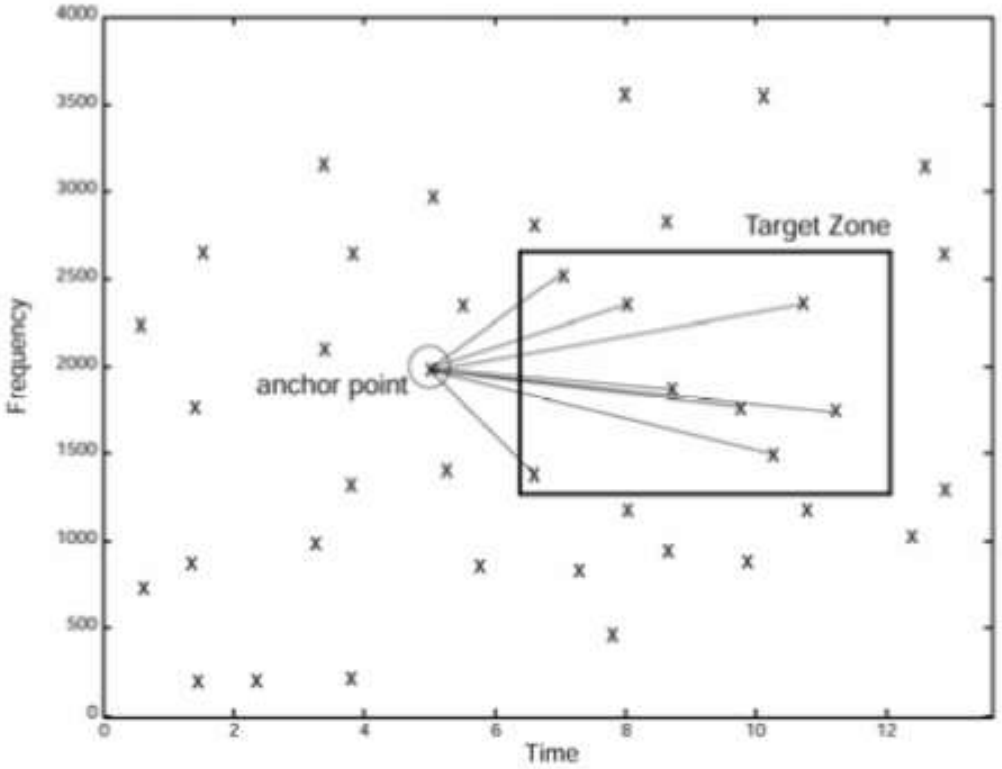


(4s, 1300Hz)
(7s, 1400Hz)
(13s, 1700Hz)

Acoustic fingerprint (**peak frequencies**)

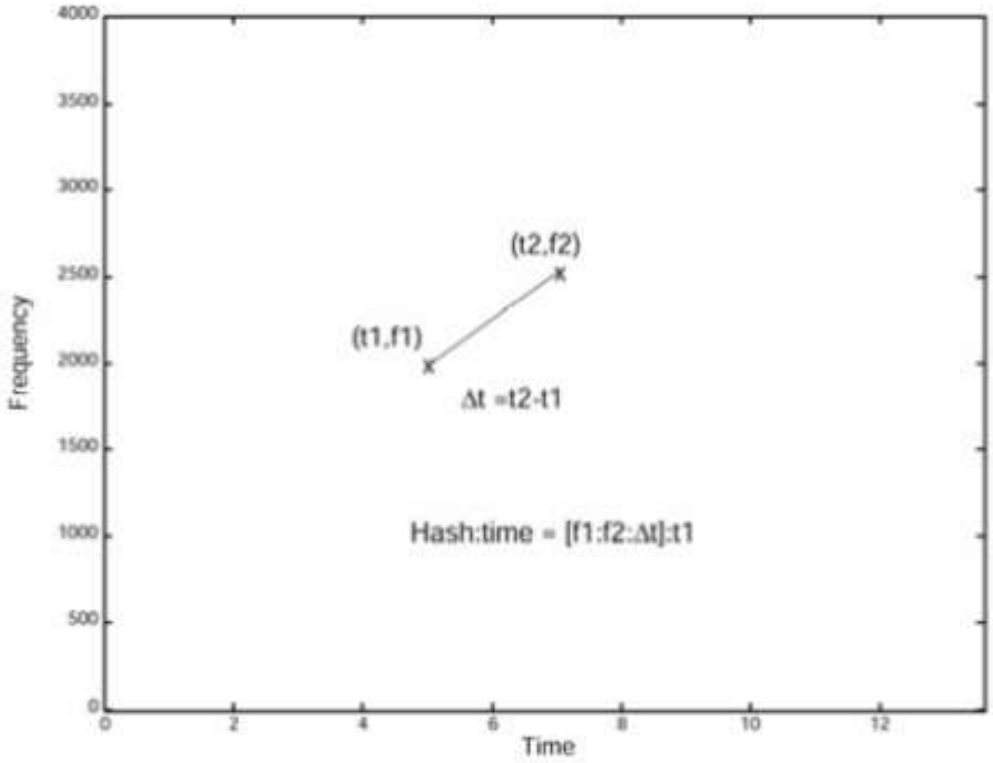


Time-Invariant Hashes



Combinatorial Hash Generation

Time-Invariant Hashes



Time-Invariant Hashes

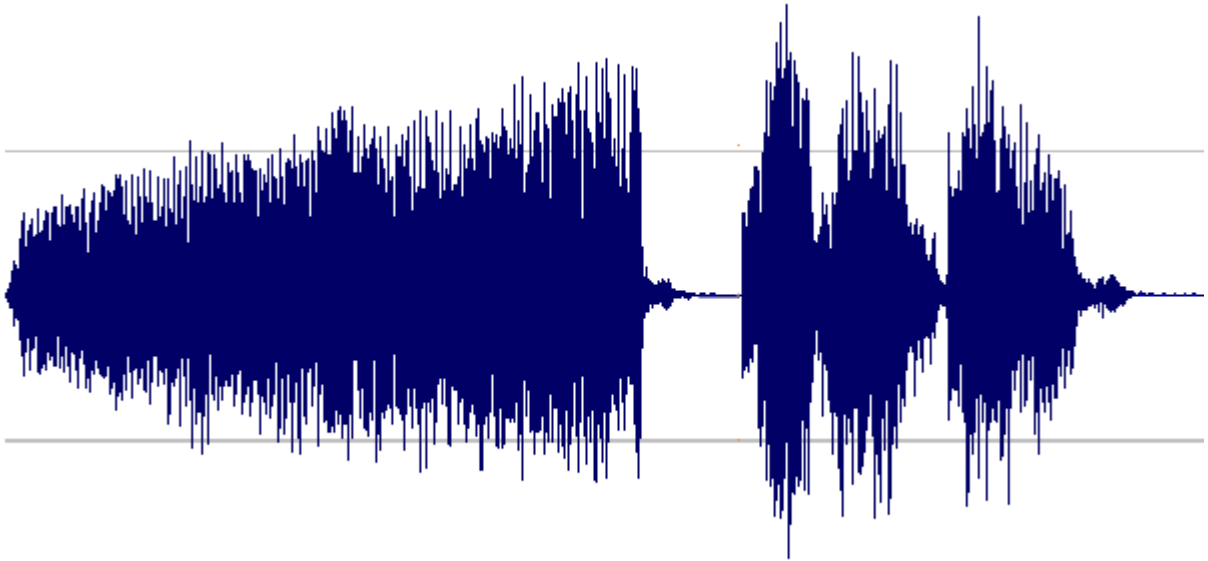
db = [

{freq1, freq2, Δtime}, => **sha1(freq1, freq2, Δtime)**

hash, hash, hash, hash, hash, … => **substr(sha1(freq1, freq2, Δtime), 0, 8)**

]

Time-Invariant Hashes



=> [261209922, 928719572, 927571829, 756562712, 875626731, 726187626, 817592192, 8217646272, 9960192815, 987125921, 972857192, 81266852, 98172975, 91729852, 7579812752, 987219872, 965876125, 918729875, 1982798712, 981729871, 716287652, …)

Time-Invariant Hashes

=> [261209922, 928719572, 927571829, 756562712, 875626731, 726187626, 817592192, 8217646272, 9960192815, 987125921, **972857192, 81266852, 98172975, 91729852, 7579812752**, 987219872, 965876125, 918729875, 1982798712, 981729871, 716287652, …)

SHAZAM

generate **hash**
find **hash** matches

fingerprints db

Store & find matches

fingerprint

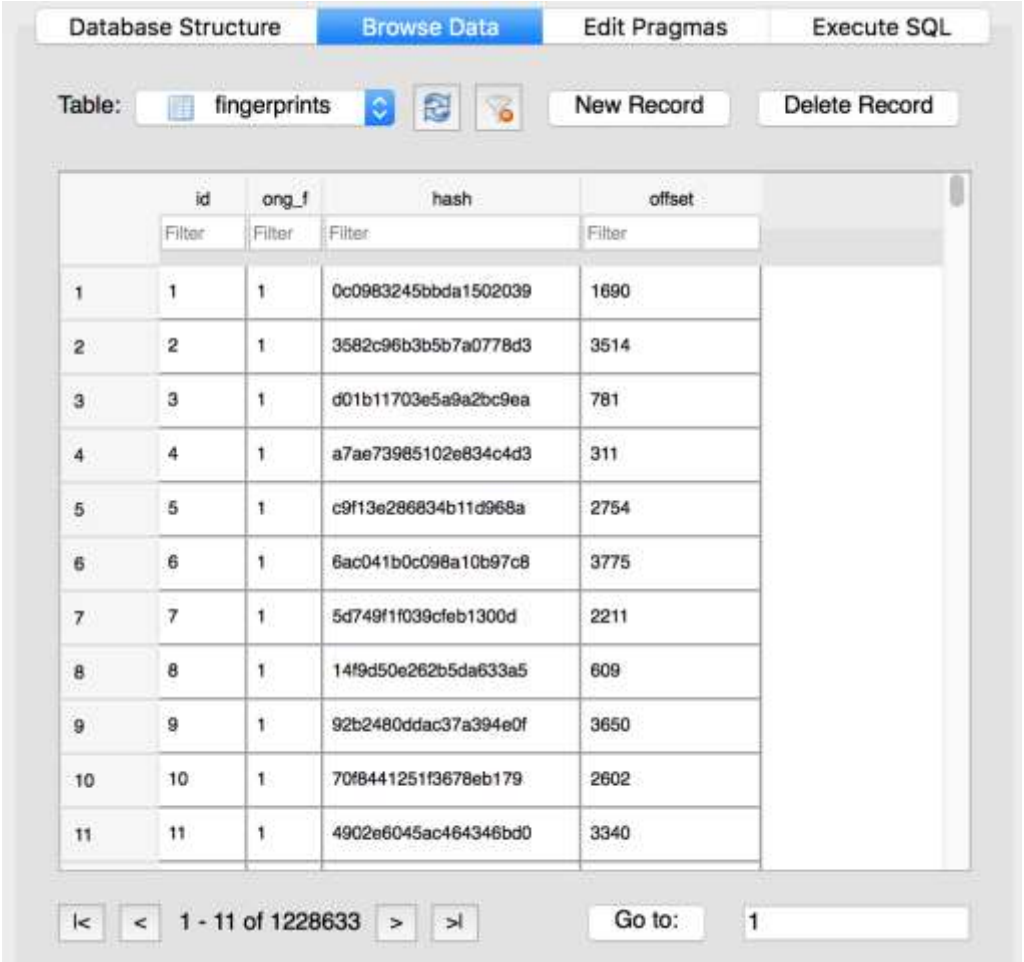
find matches

labeled music

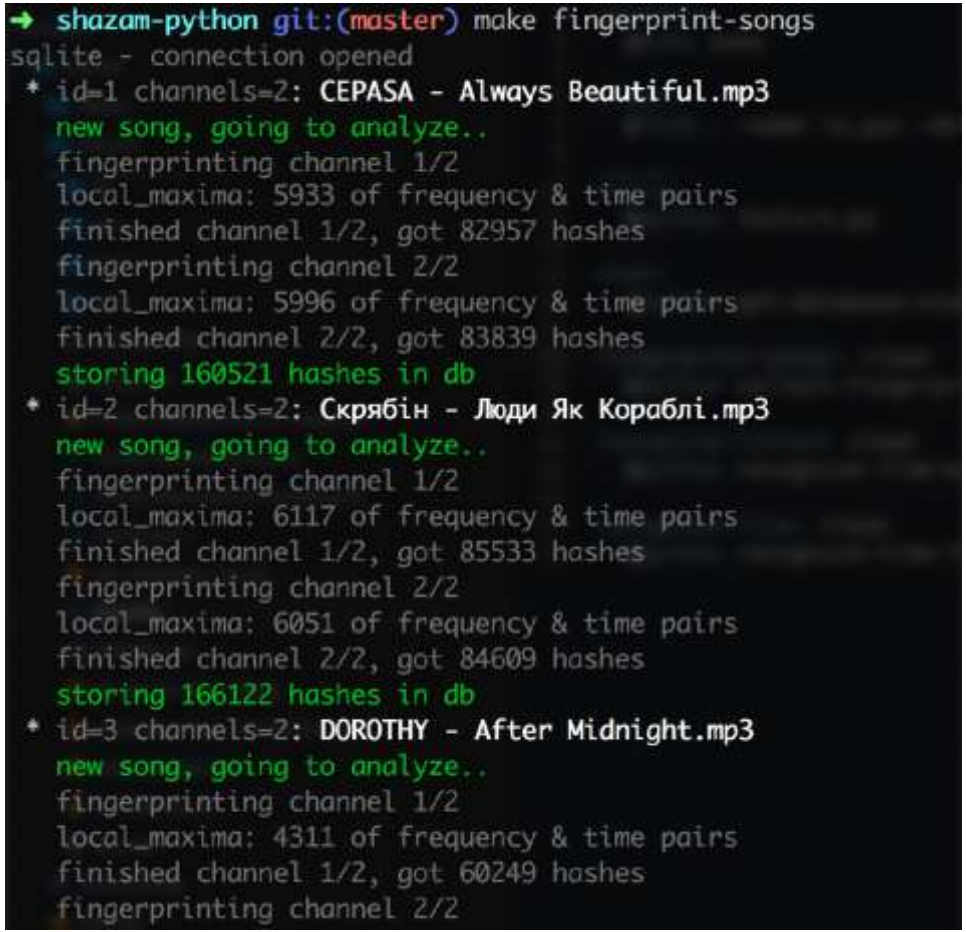
```
db.songs

int      id
varchar  title
varchar  filehash
```

audio-samples

```
db.fingerprints

int      id
int      song_fk
varchar  hash
int      offset
```



Time-Invariant Hashes



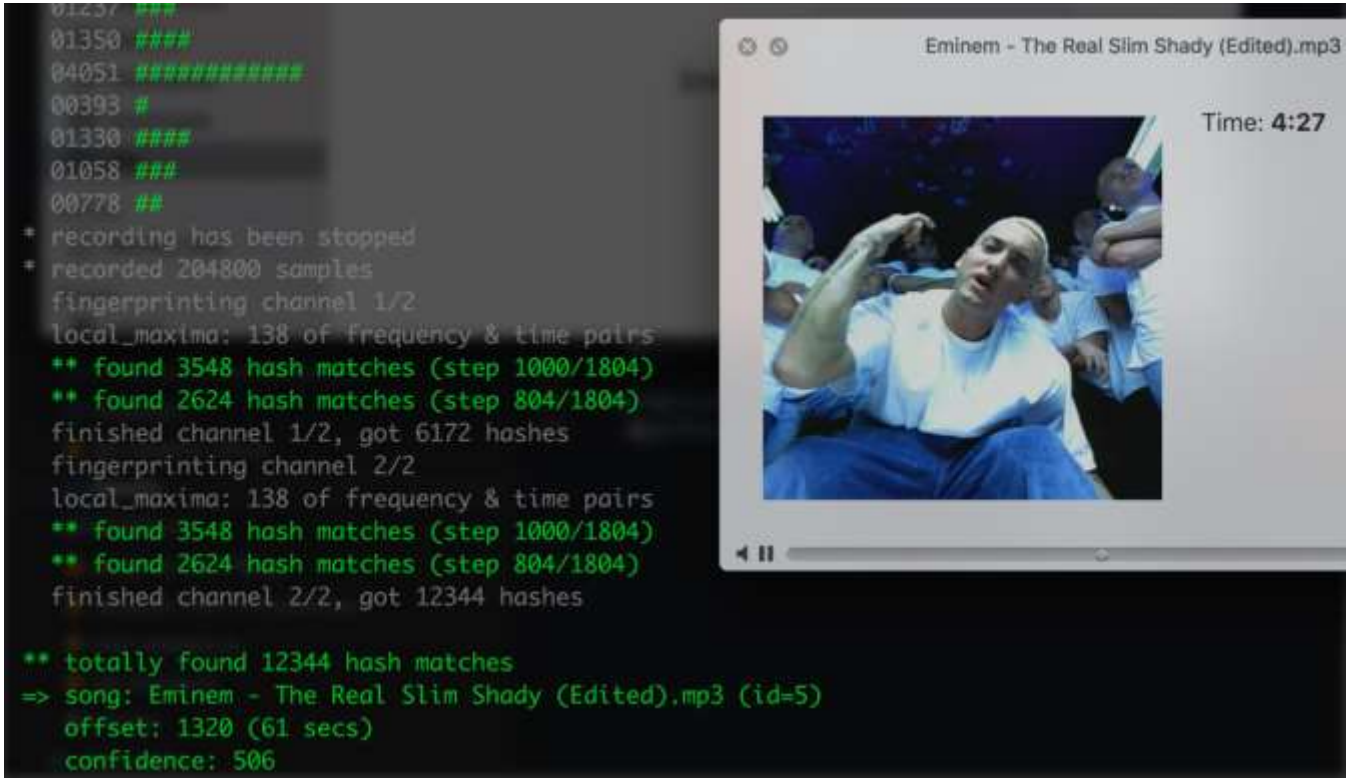
Demo

fingerprint songs

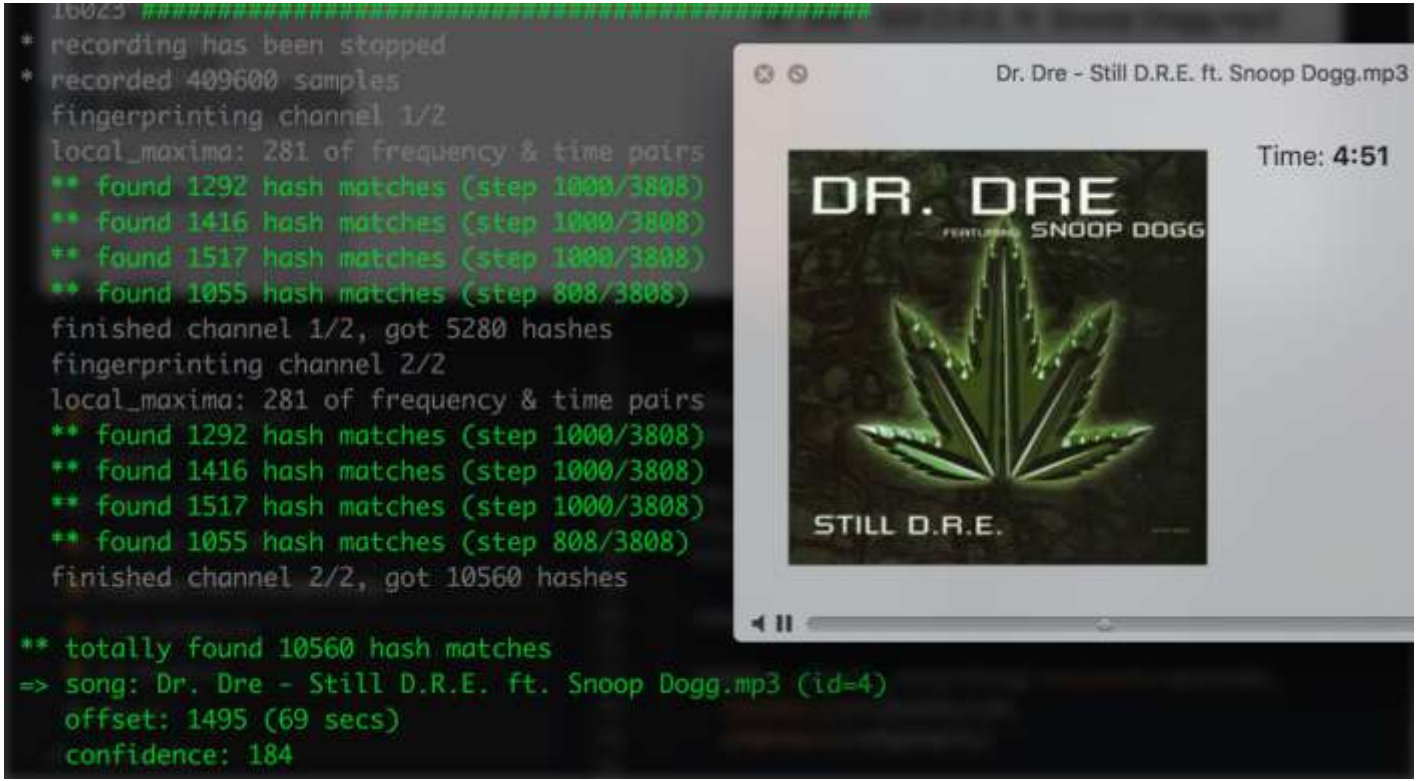
Demo **check the db stat**

```
➜  shazam-python git:(master) make stat
sqlite - connection opened

 * total: 7 song(s) (1057186 fingerprint(s))
   ** id=5 Eminem - The Real Slim Shady (Edited).mp3: 213270 hashes
   ** id=4 Dr. Dre - Still D.R.E. ft. Snoop Dogg.mp3: 196910 hashes
   ** id=2 Скрябін - Люди Як Кораблі.mp3: 166122 hashes
   ** id=1 CEPASA - Always Beautiful.mp3: 160521 hashes
   ** id=6 Скрябин - Бультер'єр - Skryabin.mp3: 141264 hashes
   ** id=7 ХЛЕБ - Чай Сахар.mp3: 96217 hashes
   ** id=3 DOROTHY - After Midnight.mp3: 82882 hashes

 * duplications: 0 song(s)

 * colissions: 859754 hash(es)
```

Demo **record audio & identify the song (5s)**



Demo **listen audio in & identify the song (10s)**



Questions? **Thanks!**

http://**bit.ly**/pacemaker-shazam-source
= https://**github**.com/itspoma/audio-fingerprint-identifying-python

http://**bit.ly**/pacemaker-shazam-slides
= http://**slideshare**.net/rodomansky/ok-shazam-la-la-laaa