



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

@itspoma





how shazam wo...

All

Images



Saved

Safe Search



I think it's Prodigy...



581 x 575 - thepoke.co.uk



How Shazam works



How Shazam works



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/rodomansky>

<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 patents** <https://patents.google.com/?q=music+identification...>
 - how to collect, parse, identify, query, etc & etc

Shazam uses a smartphone or computer's built-in microphone to gather a brief sample of audio being played. It creates an acoustic fingerprint based on the sample and compares it against a central database for a match. If it finds a match, it sends information such as the artist, song title, and album back to the user. May 29, 2015

How does Shazam work? What is the logic behind Shazam tracing out ...
<https://www.quora.com/How-does-Shazam-work-What-is-the-logic-behind-Shazam-traci...>

<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

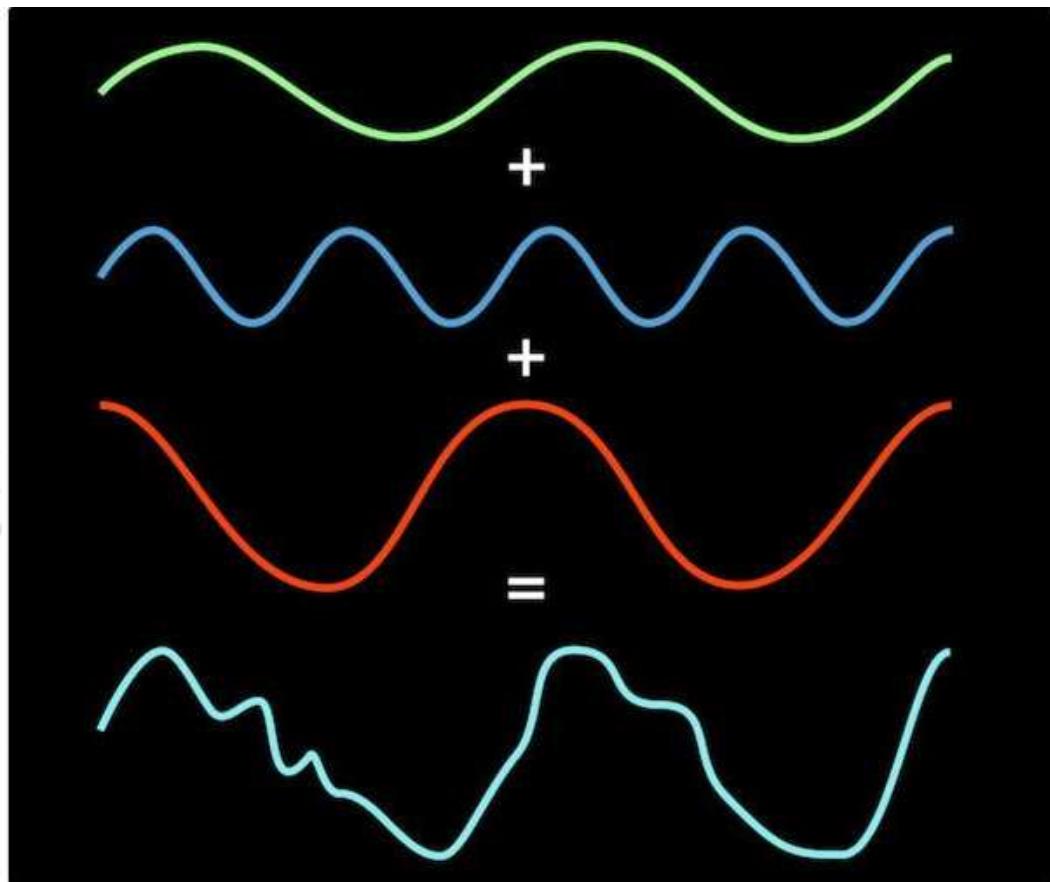
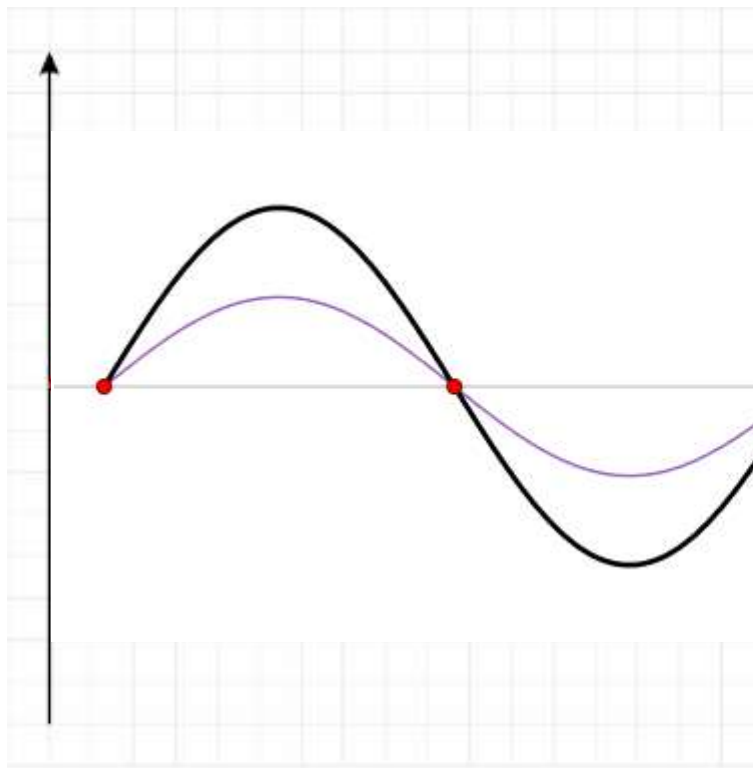
30-second clip of the song to a friend. Other services, such as purchasing an MP3 download may become available soon.

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]. Licensing the algorithm from Relatable Neuros has included a

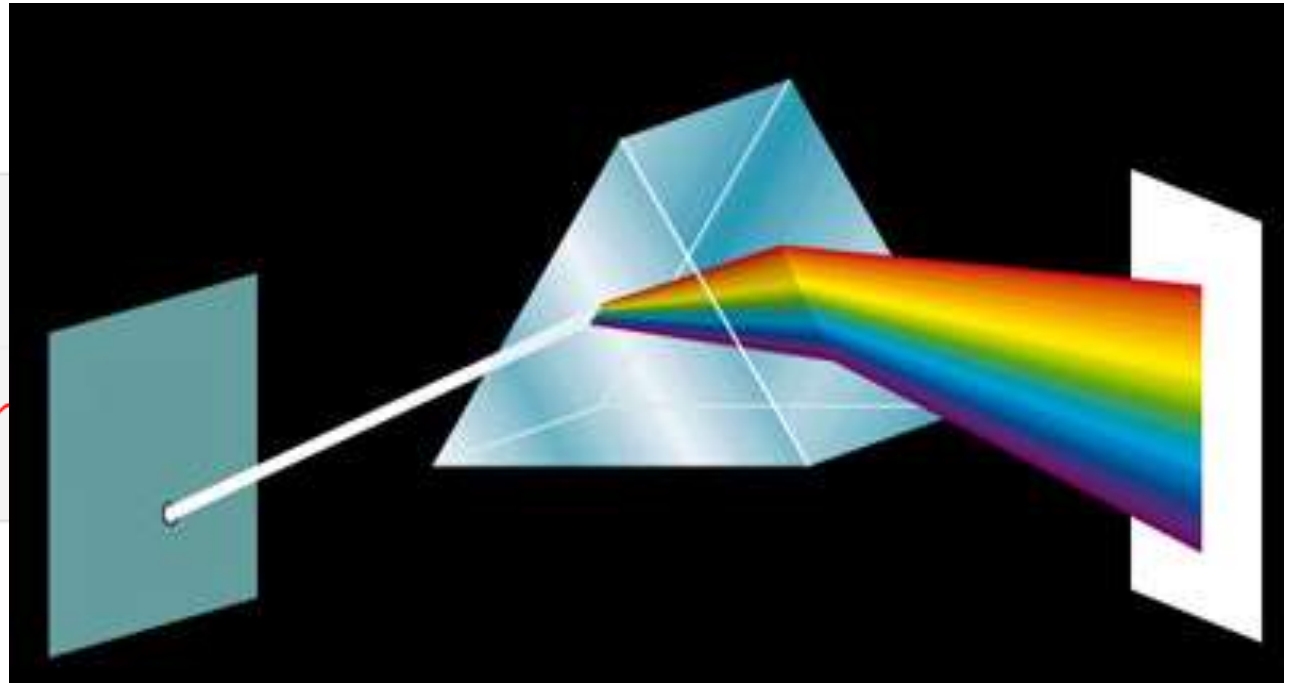
<https://www.ee.columbia.edu/~dpwe/papers/Wang03-shazam.pdf>

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.

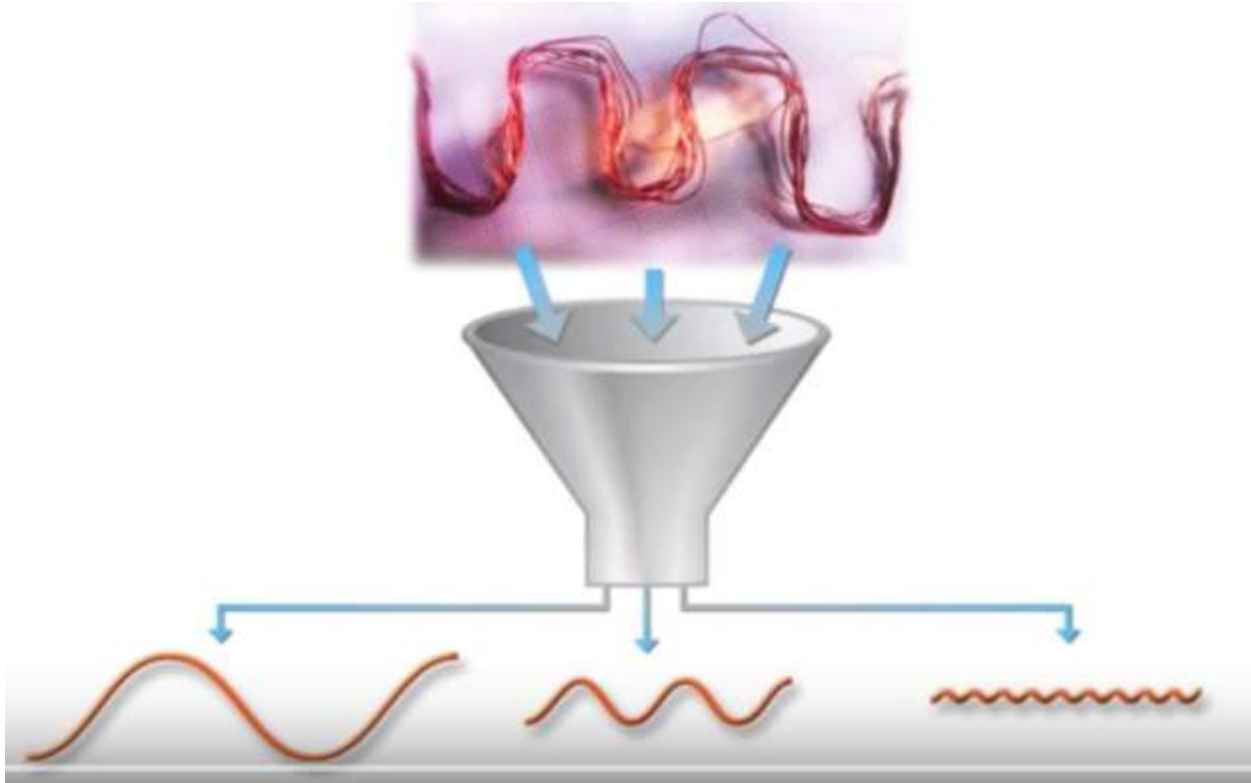
Audio Signal



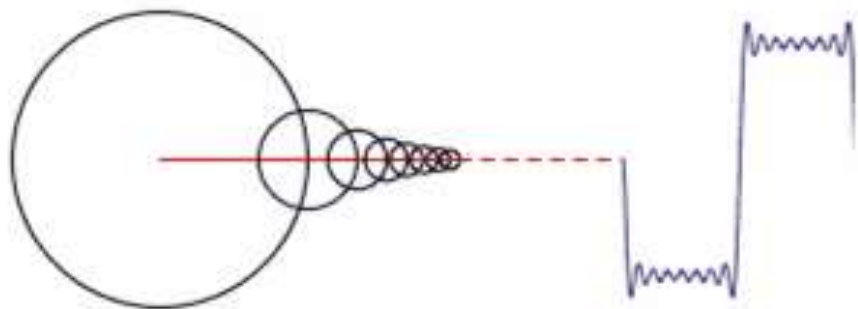
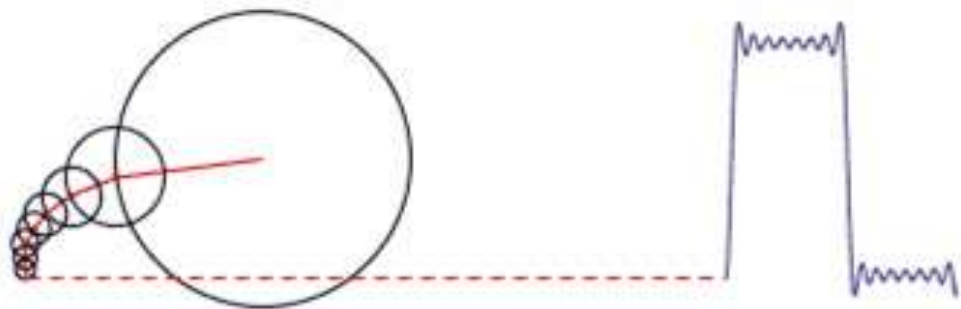
Fourier transform \mathcal{F}



Fourier transform \mathcal{F}



Fourier transform \mathcal{F}



Fourier transform in real life

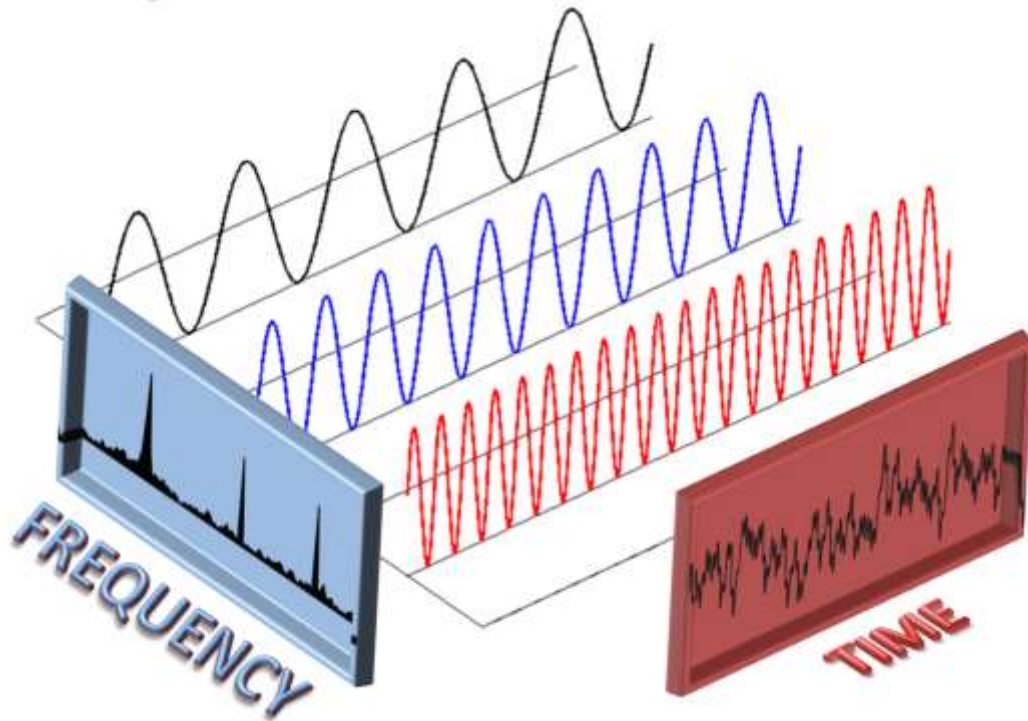


Discrete Fourier Transform (DFT)

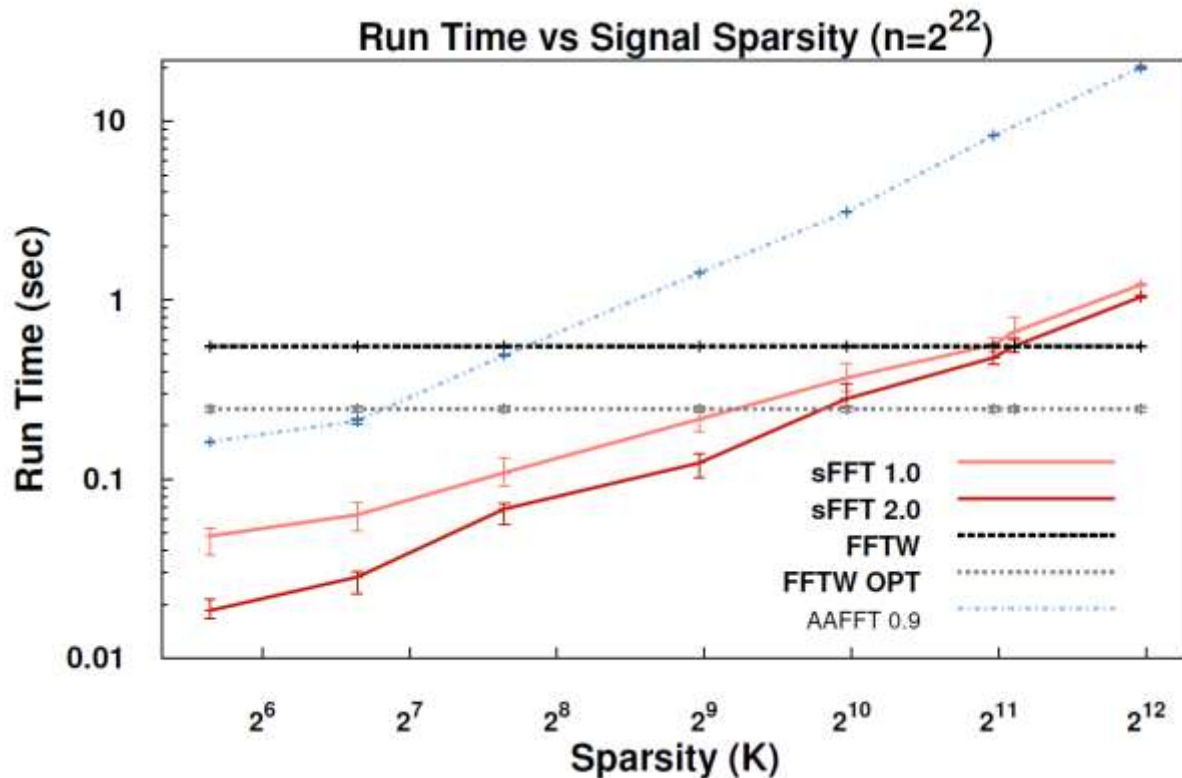
$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N}kn}, k = 0 \dots N-1$$

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

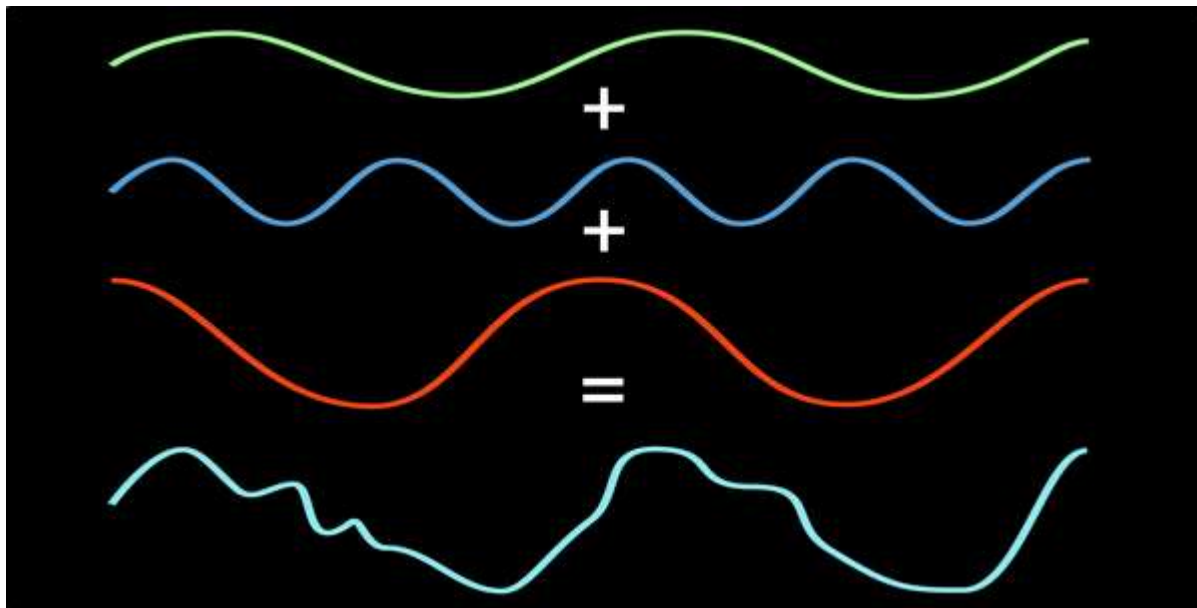
Fast Fourier transform



Fastest Fourier Transform in the West (FFTW)

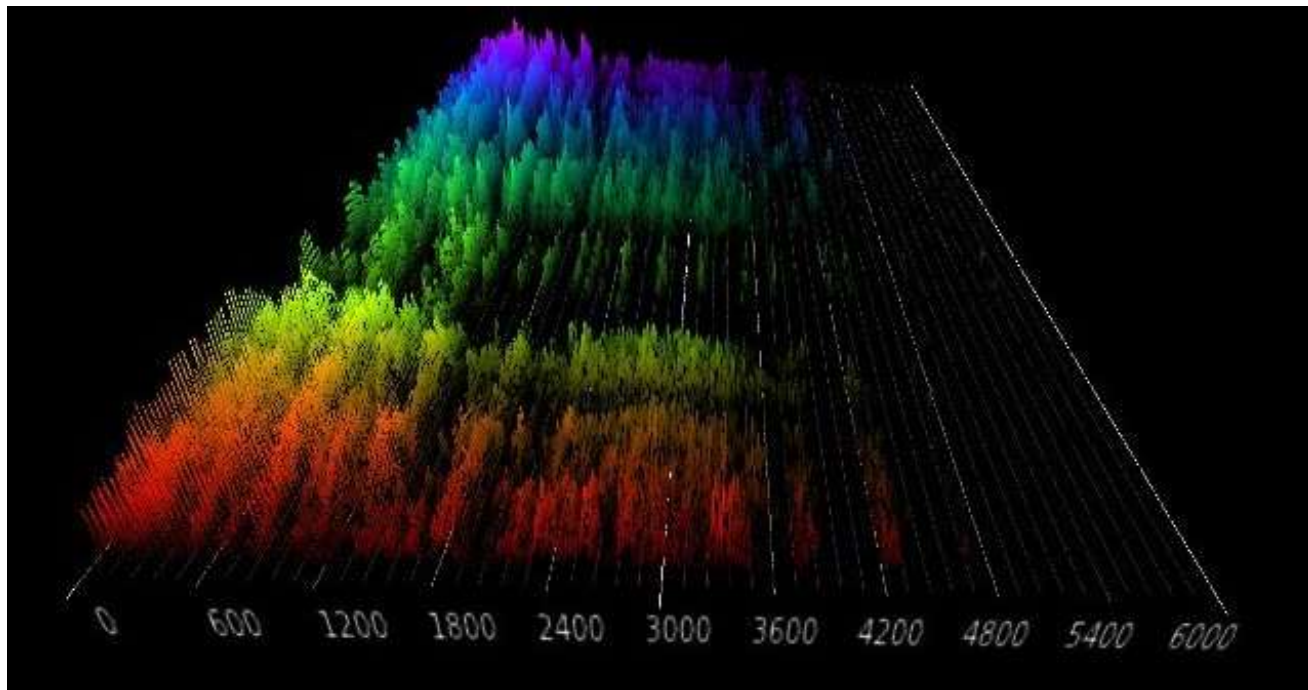


Sparse Fast Fourier Transform (sFFT)



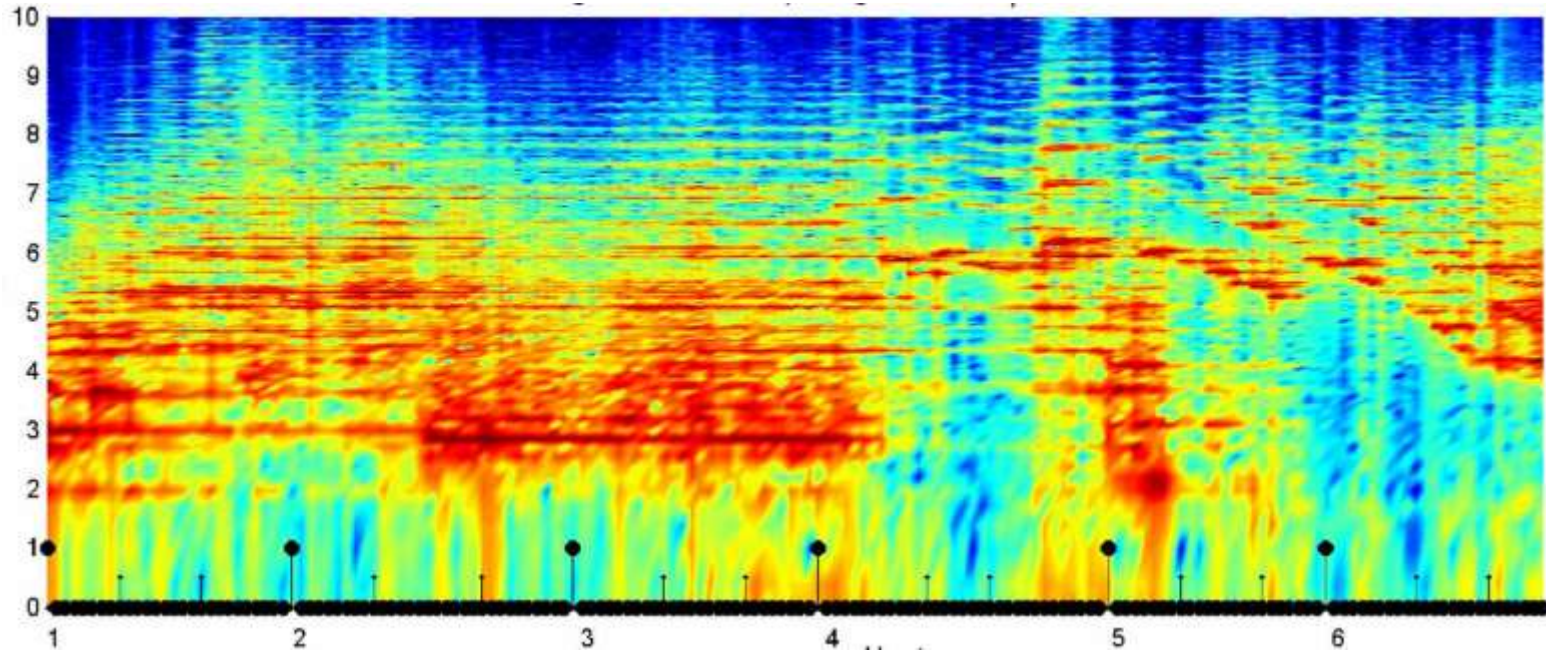
$O(k \log n)$
 $O(k \log k \log(n/k))$

Spectrogram



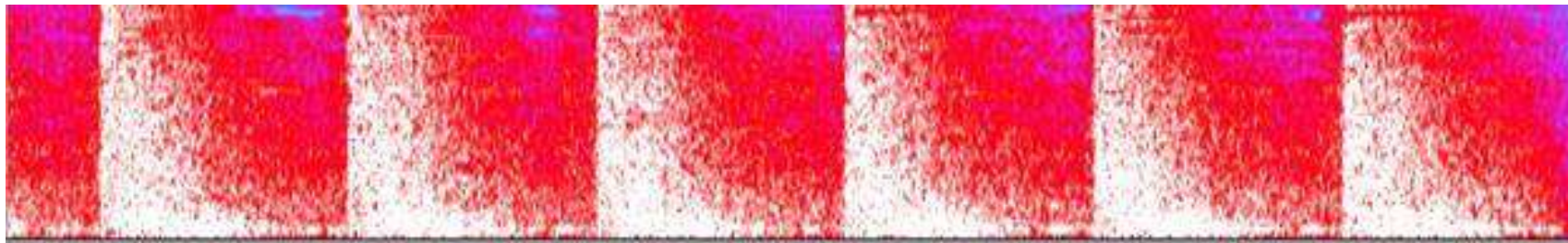
x = time, y = frequency, z = A

Spectrogram



$x = \text{time}, y = \text{frequency}, z = A$

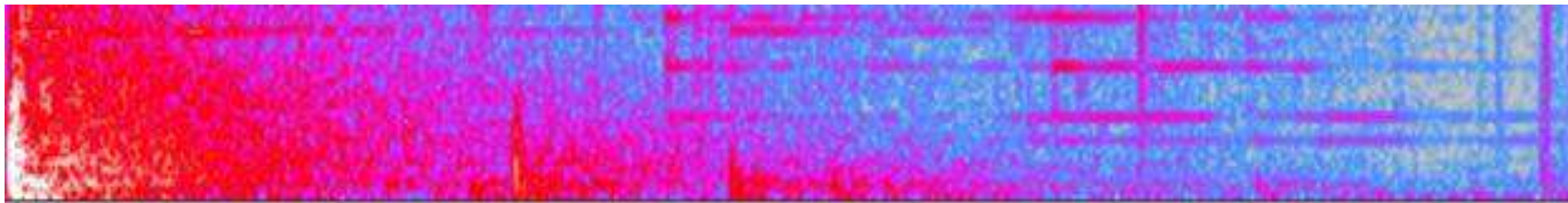
Spectrogram of **shots set**



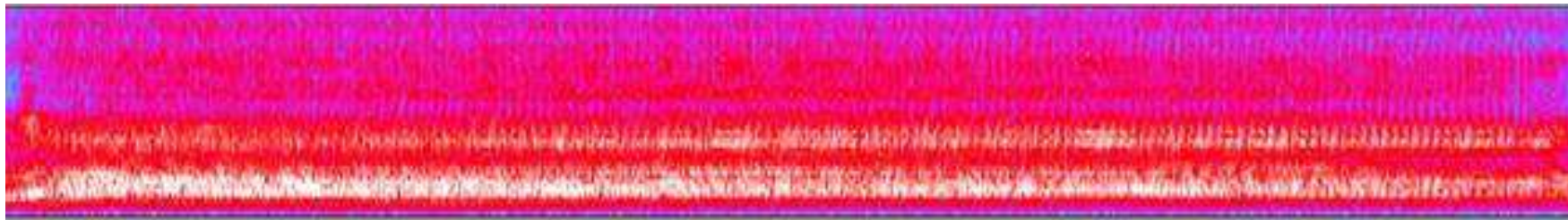
<http://www.shotspotter.com/>



Spectrogram of **broken window**

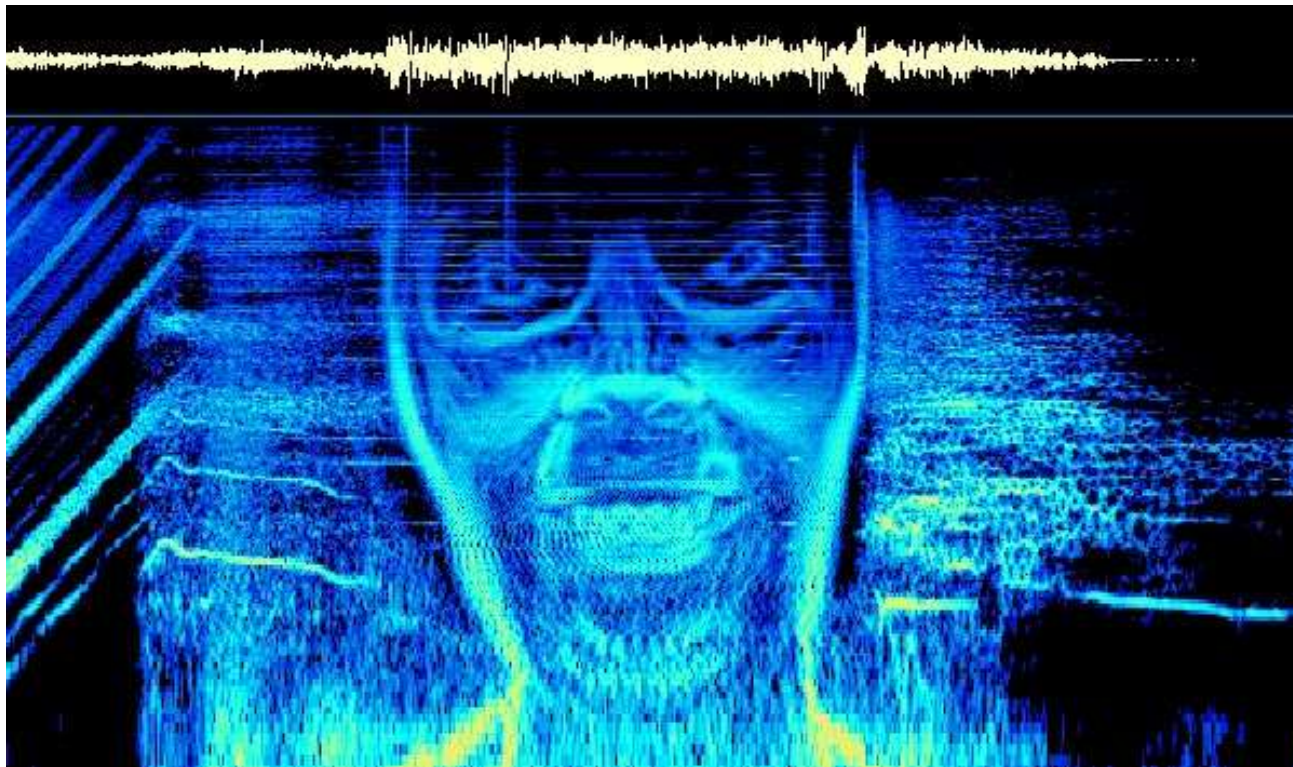


Spectrogram of **shout (cry)**



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

Spectrogram in **songs**



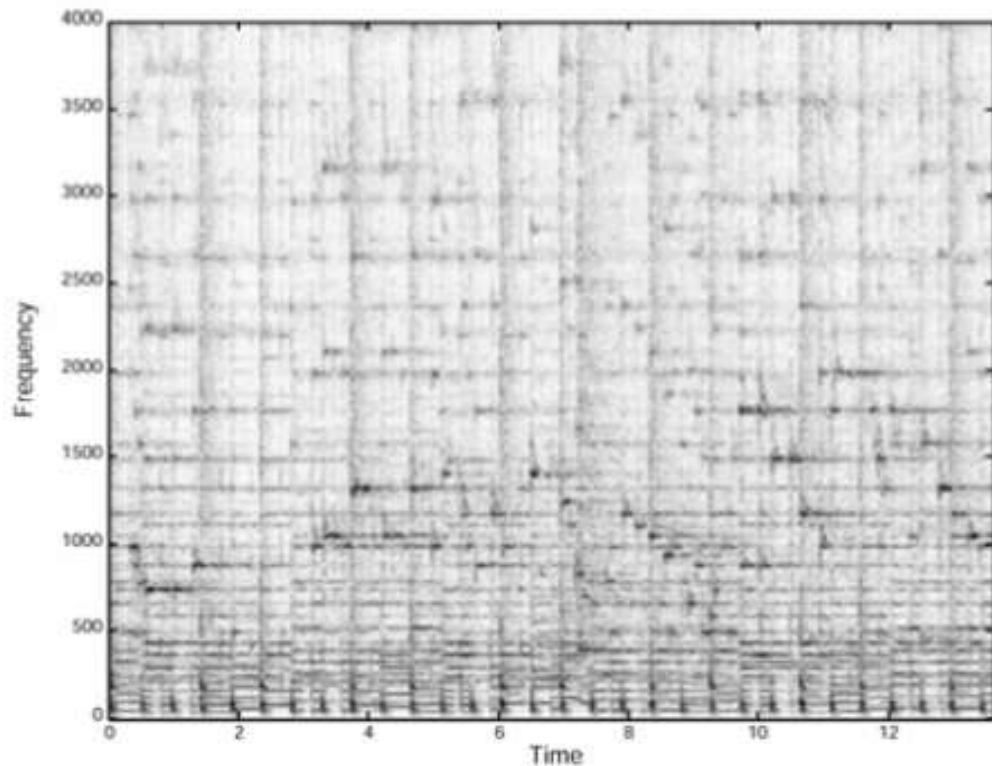
Aphex Twin

“Windowlicker” album

5:27

<http://bit.ly/2bWucab>

Spectrogram of a piece of music

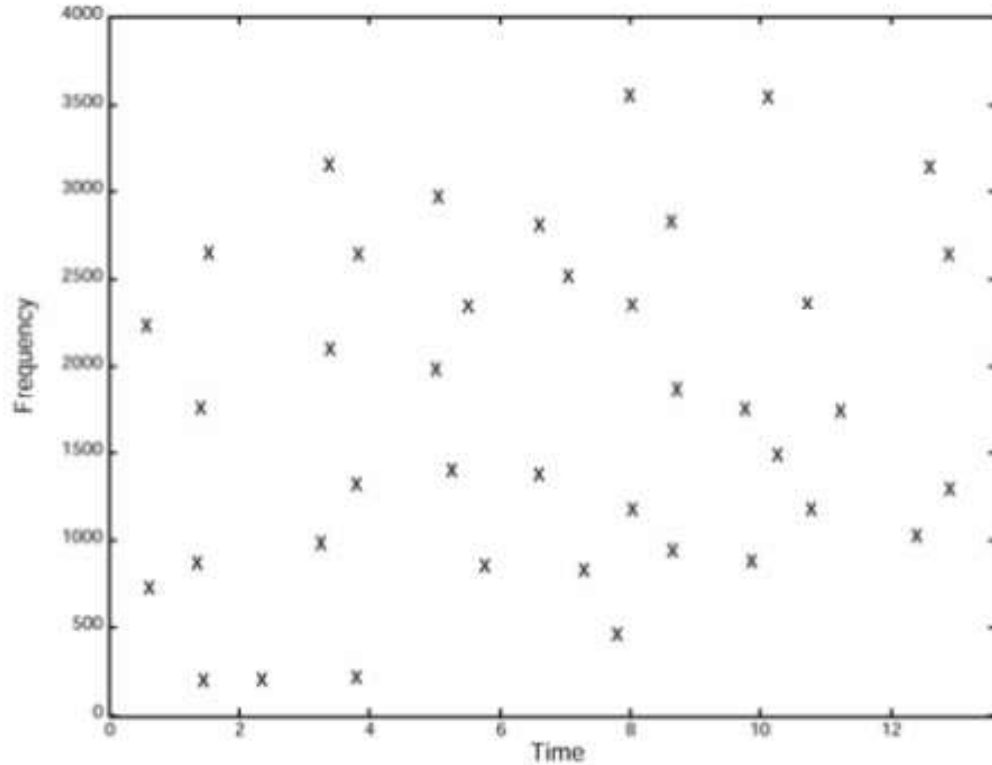


(4s, 1300Hz)

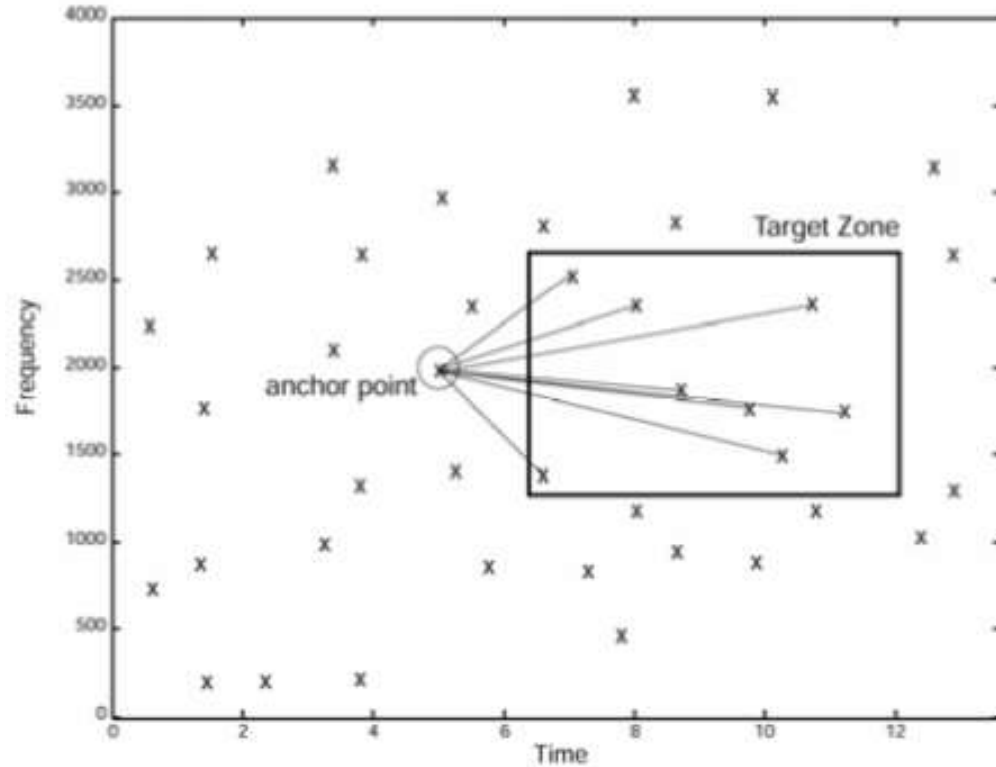
(7s, 1400Hz)

(13s, 1700Hz)

Acoustic fingerprint (**peak frequencies**)

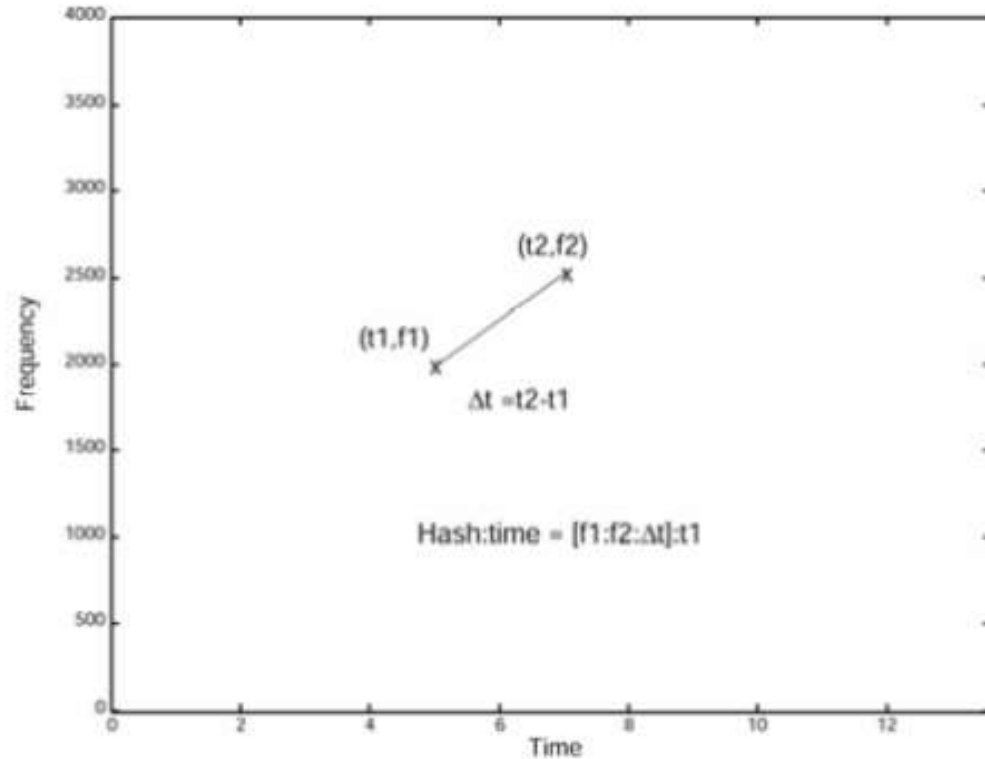


Time-Invariant Hashes



Combinatorial Hash Generation

Time-Invariant Hashes



Time-Invariant Hashes

db = [

{freq1, freq2, Δ time},

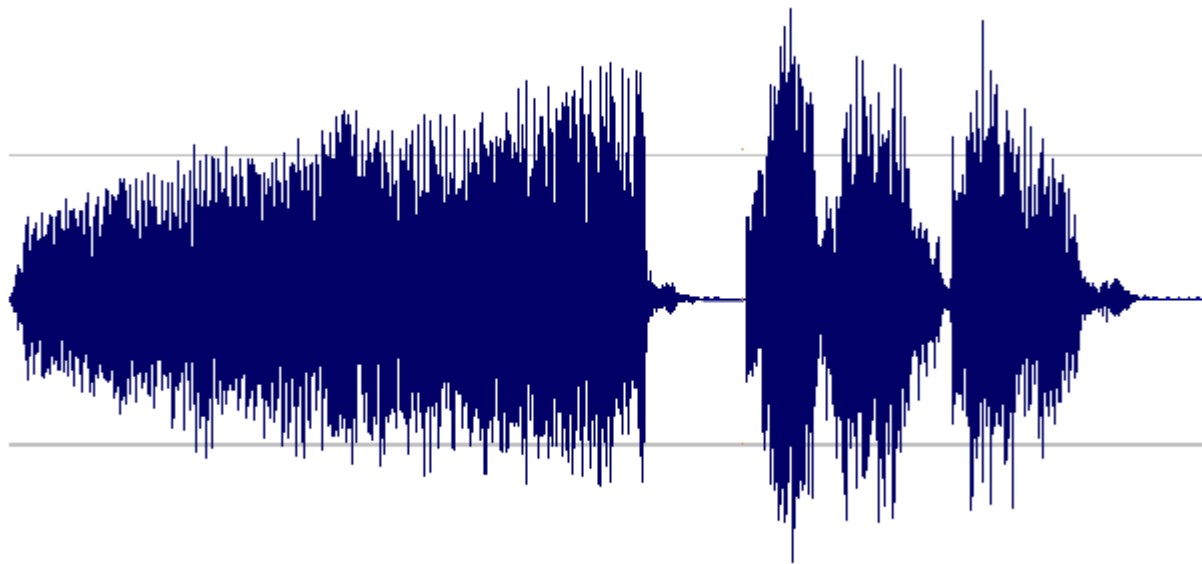
hash, hash, hash, hash, hash, ...

]

$\Rightarrow \text{sha1}(\text{freq1}, \text{freq2}, \Delta\text{time})$

$\Rightarrow \text{substr}(\text{sha1}(\text{freq1}, \text{freq2}, \Delta\text{time}), 0, 8)$

Time-Invariant Hashes

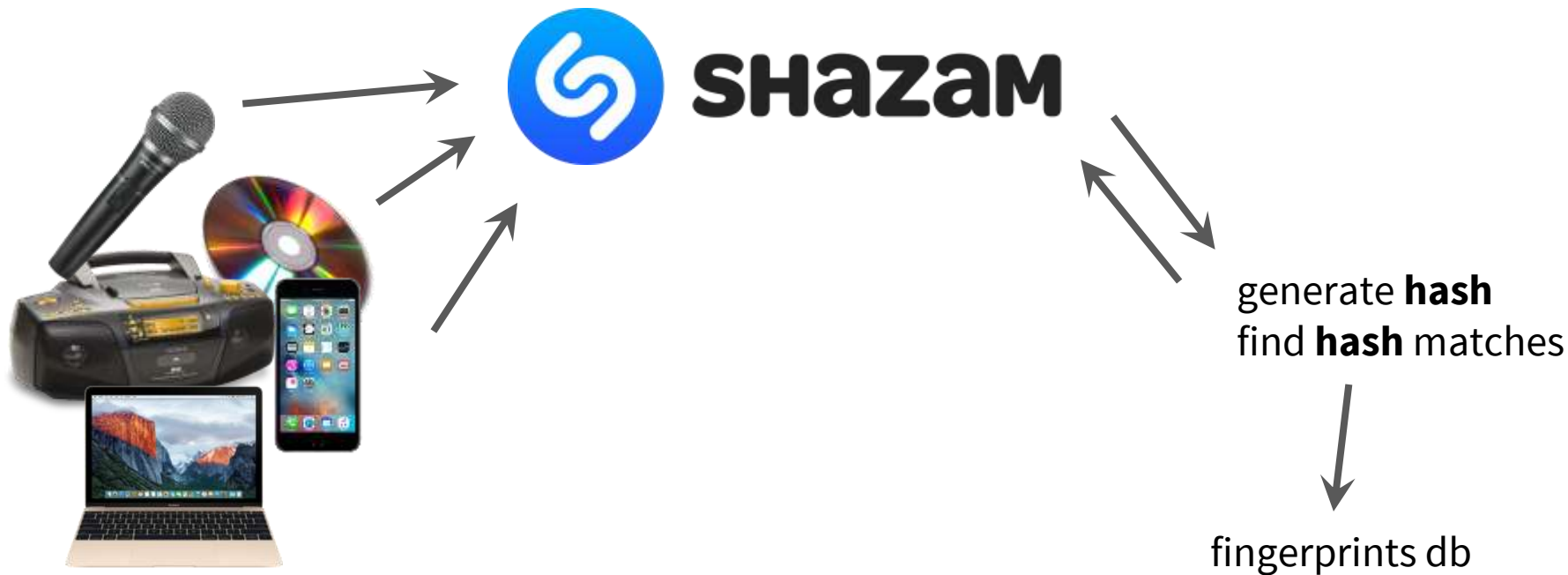


=> [261209922, 928719572, 927571829, 756562712, 875626731, 726187626, 817592192, 8217646272, 9960192815, 987125921, 972857192, 81266852, 98172975, 91729852, 7579812752, 987219872, 965876125, 918729875, 1982798712, 981729871, 716287652, ...)

Time-Invariant Hashes



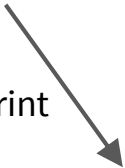
=> [261209922, 928719572, 927571829, 756562712, 875626731, 726187626, 817592192, 8217646272,
9960192815, 987125921, **972857192, 81266852, 98172975, 91729852, 7579812752**, 987219872,
965876125, 918729875, 1982798712, 981729871, 716287652, ...)



Store & find matches



fingerprint



SHAZAM

labeled music

db.songs	
int	id
varchar	title
varchar	filehash

audio-samples

db.fingerprints	
int	id
int	song_fk
varchar	hash
int	offset

find matches



Database Structure

Browse Data

Edit Pragmas

Execute SQL

Table: fingerprints

New Record

Delete Record

	id	ong_f	hash	offset
	Filter	Filter	Filter	Filter
1	1	1	0c0983245bbda1502039	1690
2	2	1	3582c96b3b5b7a0778d3	3514
3	3	1	d01b11703e5a9a2bc9ea	781
4	4	1	a7ae73985102e834c4d3	311
5	5	1	c9f13e286834b11d968a	2754
6	6	1	6ac041b0c098a10b97c8	3775
7	7	1	5d749f1f039cfef1300d	2211
8	8	1	14f9d50e262b5da633a5	609
9	9	1	92b2480ddac37a394e0f	3650
10	10	1	70f8441251f3678eb179	2602
11	11	1	4902e6045ac464346bd0	3340

<<1 - 11 of 1228633>>

Go to: 1

```
→ shazam-python git:(master) make fingerprint-songs
sqlite - connection opened
* id=1 channels=2: CEPASA - Always Beautiful.mp3
  new song, going to analyze..
  fingerprinting channel 1/2
  local_maxima: 5933 of frequency & time pairs
  finished channel 1/2, got 82957 hashes
  fingerprinting channel 2/2
  local_maxima: 5996 of frequency & time pairs
  finished channel 2/2, got 83839 hashes
  storing 160521 hashes in db
* id=2 channels=2: Скрябін - Люди Як Кораблі.mp3
  new song, going to analyze..
  fingerprinting channel 1/2
  local_maxima: 6117 of frequency & time pairs
  finished channel 1/2, got 85533 hashes
  fingerprinting channel 2/2
  local_maxima: 6051 of frequency & time pairs
  finished channel 2/2, got 84609 hashes
  storing 166122 hashes in db
* id=3 channels=2: DOROTHY - After Midnight.mp3
  new song, going to analyze..
  fingerprinting channel 1/2
  local_maxima: 4311 of frequency & time pairs
  finished channel 1/2, got 60249 hashes
  fingerprinting channel 2/2
```

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 СЕРАСА - 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)**

```
01237 ###
01350 ####
04051 #####
00393 #
01330 ####
01058 ###
00778 ##
* recording has been stopped
* recorded 204800 samples
fingerprinting channel 1/2
local_maxima: 138 of frequency & time pairs
** found 3548 hash matches (step 1000/1804)
** found 2624 hash matches (step 804/1804)
finished channel 1/2, got 6172 hashes
fingerprinting channel 2/2
local_maxima: 138 of frequency & time pairs
** found 3548 hash matches (step 1000/1804)
** found 2624 hash matches (step 804/1804)
finished channel 2/2, got 12344 hashes

** totally found 12344 hash matches
=> song: Eminem - The Real Slim Shady (Edited).mp3 (id=5)
offset: 1320 (61 secs)
confidence: 506
```



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

```
10023 *****
* recording has been stopped
* recorded 409600 samples
fingerprinting channel 1/2
local_maxima: 281 of frequency & time pairs
** found 1292 hash matches (step 1000/3808)
** found 1416 hash matches (step 1000/3808)
** found 1517 hash matches (step 1000/3808)
** found 1055 hash matches (step 808/3808)
finished channel 1/2, got 5280 hashes
fingerprinting channel 2/2
local_maxima: 281 of frequency & time pairs
** found 1292 hash matches (step 1000/3808)
** found 1416 hash matches (step 1000/3808)
** found 1517 hash matches (step 1000/3808)
** found 1055 hash matches (step 808/3808)
finished channel 2/2, got 10560 hashes

** totally found 10560 hash matches
=> song: Dr. Dre - Still D.R.E. ft. Snoop Dogg.mp3 (id=4)
    offset: 1495 (69 secs)
    confidence: 184
```



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>

