An industrial-strength- audio search algorithm

Papers We Love, Montreal
July 26, 2017

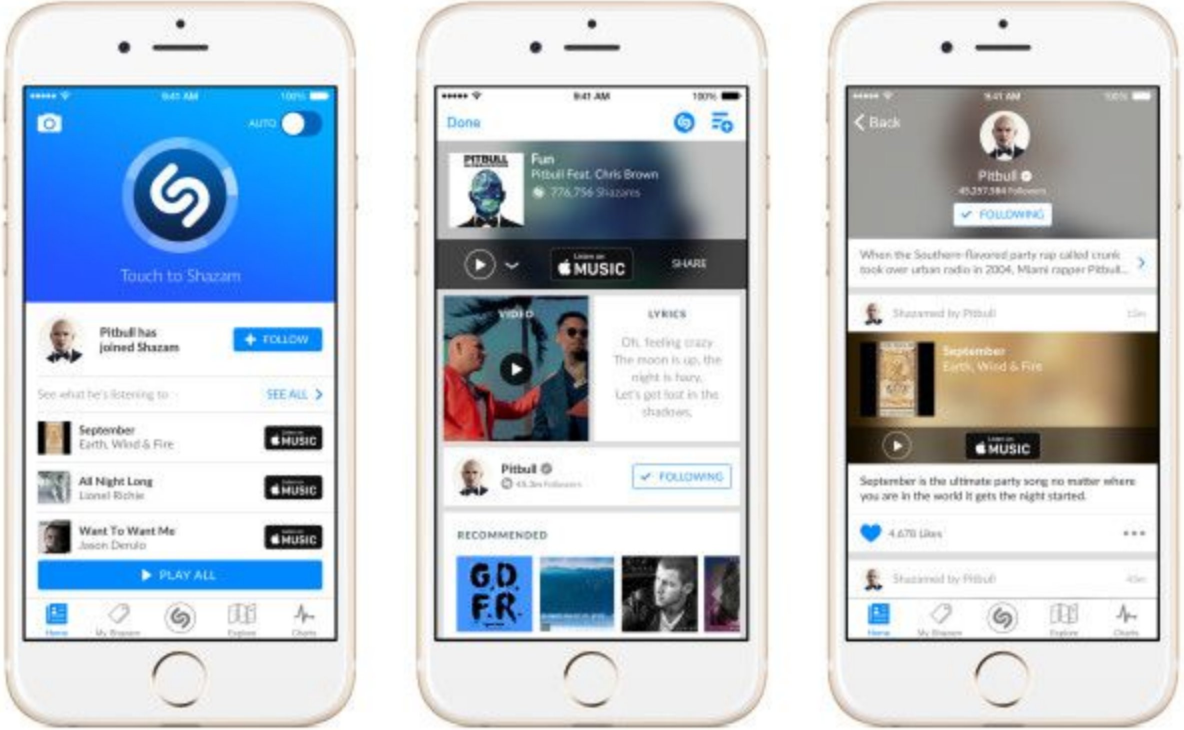
Hello, I’m Tom

wrnch

UFORA

SHAZAM


Listening...
shazam



9:41 AM
100%
AUTO
Touch to Shazam
Pitbull has joined Shazam
FOLLOW
See what he's listening to
SEE ALL
September
Earth, Wind & Fire
MUSIC
All Night Long
Lionel Richie
MUSIC
Want To Want Me
Jason Derulo
MUSIC
PLAY ALL
Done
Fun
Pitbull Feat. Chris Brown
776,756 Shazams
MUSIC
SHARE
VIDEO
LYRICS
Oh, feeling crazy
The moon is up, the
night is hazy.
Let's get lost in the
shadows.
Pitbull
FOLLOWING
RECOMMENDED
Back
Pitbull
FOLLOWING
When the Southern-flavored party rap called crunk took over urban radio in 2004, Miami rapper Pitbull...
Shazamed by Pitbull
September
Earth, Wind & Fire
MUSIC
September is the ultimate party song no matter where you are in the world it gets the night started.
Shazamed by Pitbull

What is Shazam?

- Identifies **exact tracks** of music.
- Only needs small samples (seconds)
- Robust to noise

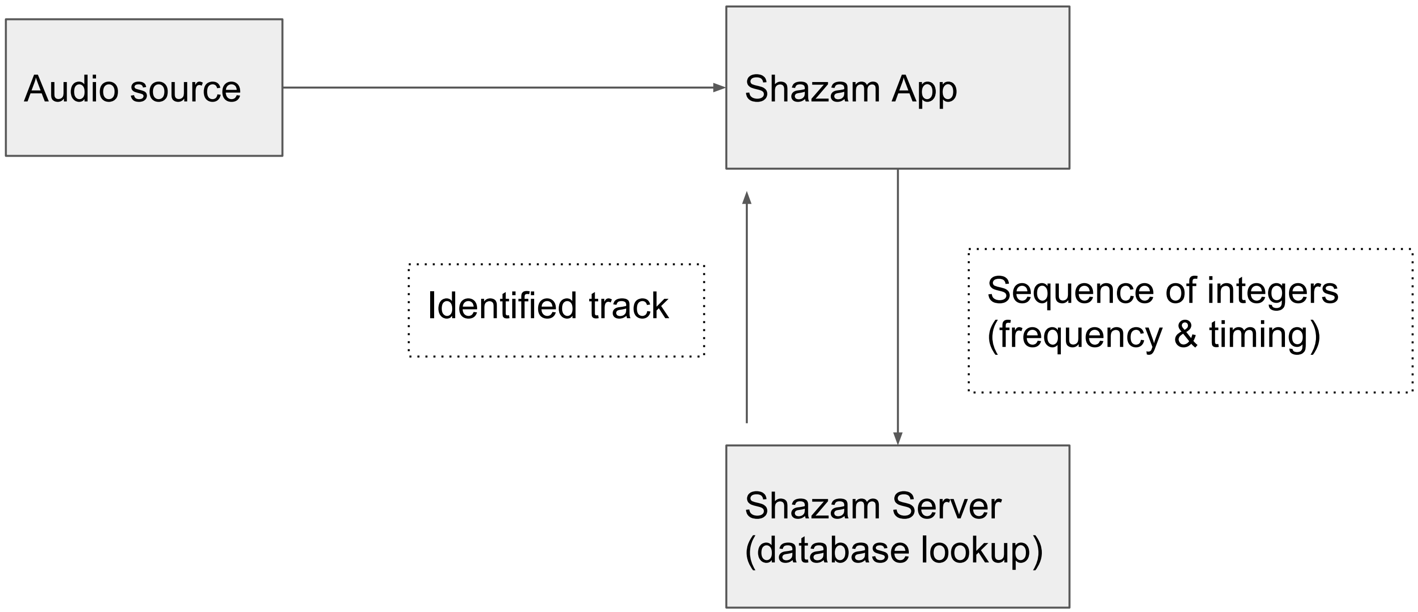
What isn’t Shazam

- Not designed to detect live recordings.

Shazam is **old**

Introduced in 1999!

Basic idea: audio fingerprinting



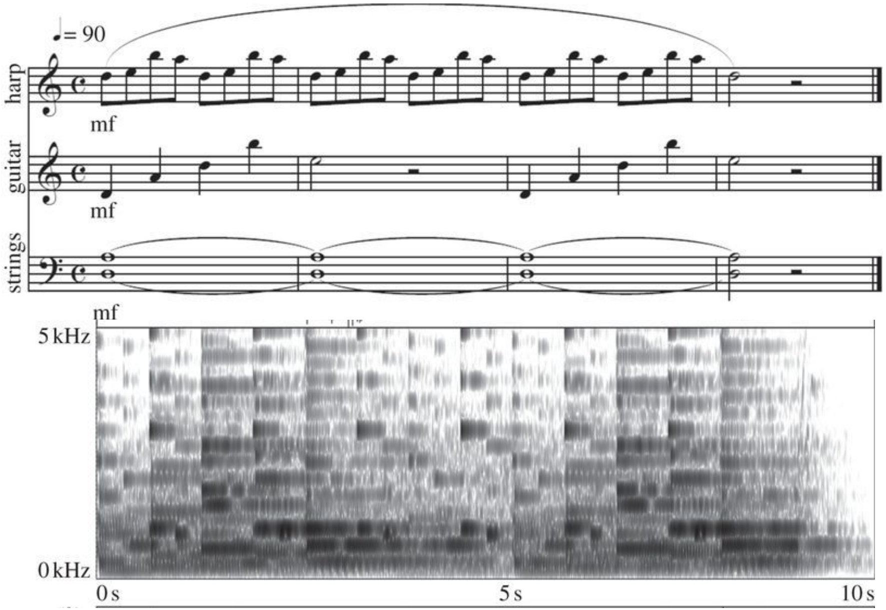
Two key pieces to Shazam:

1) Construction of “fingerprints”
   a) Contain frequency and timing information
2) Lookup of fingerprints

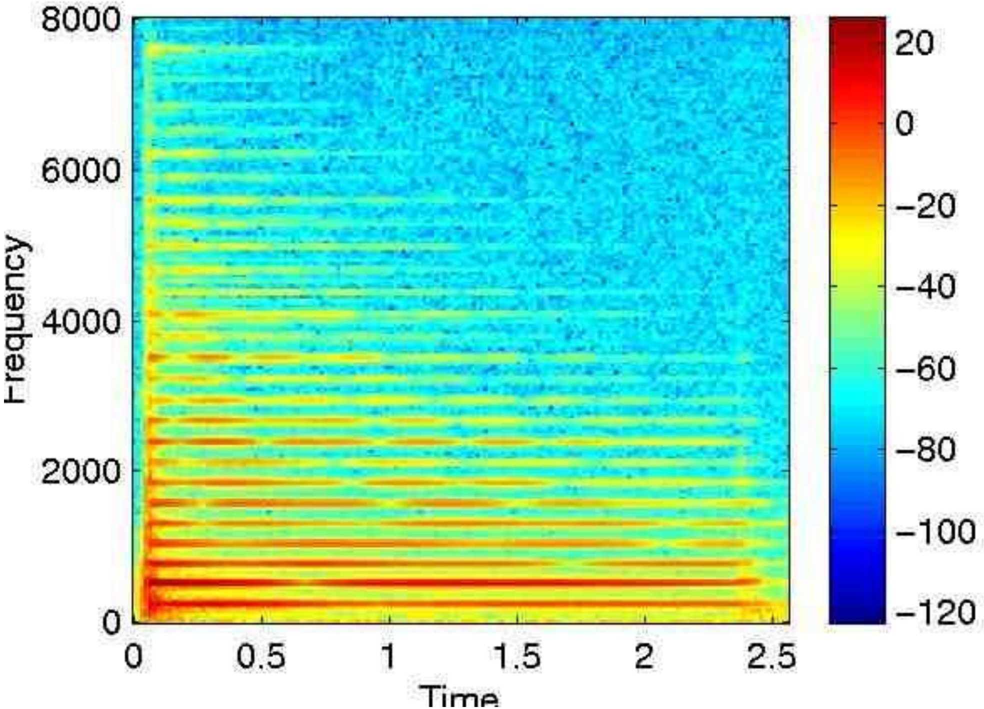
Part 1: Construction of the fingerprints

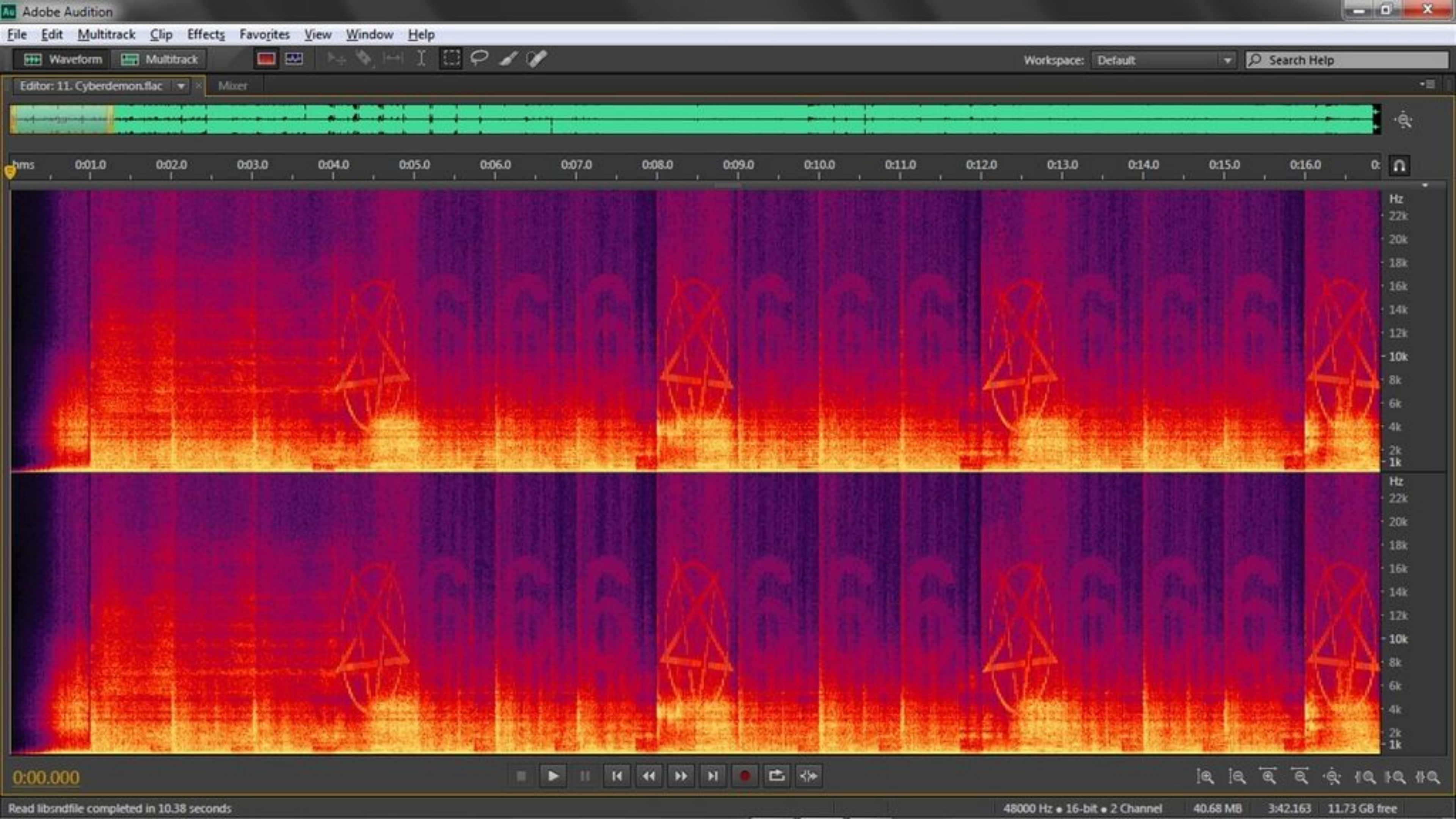
Spectrograms and Sheet music

Spectrograms and Sheet music

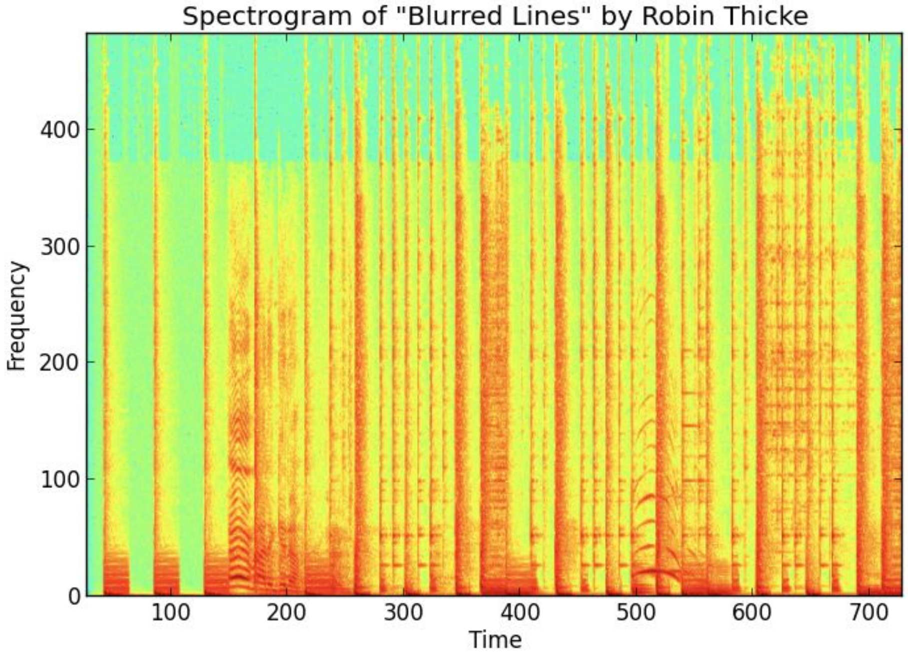


Piano C4 note (~260Hz fundamental freq)

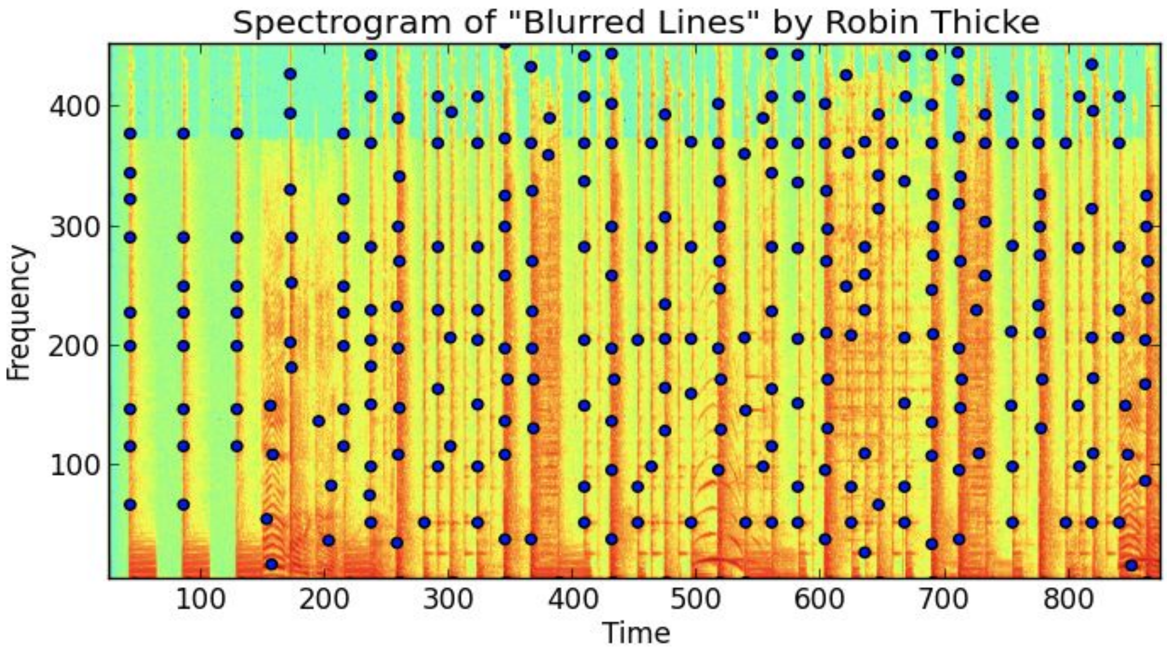



Adobe Audition
File Edit Multitrack Clip Effects Favorites View Window Help
Waveform
Multitrack
Workspace: Default
Search Help
Editor: 11. Cyberdemon.flac
Mixer
hms 0:01.0 0:02.0 0:03.0 0:04.0 0:05.0 0:06.0 0:07.0 0:08.0 0:09.0 0:10.0 0:11.0 0:12.0 0:13.0 0:14.0 0:15.0 0:16.0
Hz 22k 20k 18k 16k 14k 12k 10k 8k 6k 4k 2k 1k
Hz 22k 20k 18k 16k 14k 12k 10k 8k 6k 4k 2k 1k
0:00.000
Read libsndfile completed in 10.38 seconds
48000 Hz • 16-bit • 2 Channel
40.68 MB
3:42.163
11.73 GB free

How to select interesting frequencies?



Peak detection



- Extremely robust to noise
- Highly reproducible

Open source implementation!

https://github.com/worldveil/dejavu

Uses scipy's `maximum_filter` for peak detection.

Constructing the hashes 1: quantization

Frequencies are binned into 1024 values

=> We only need 10 bits to encode a quantized frequency.

Constructing the hashes 2: a wrong idea

What if we sent off the locations of the peaks?

In other words, send (quantized)

```
(time_offset, frequency)
```

What’s wrong with this?

Lookup on the database is the problem

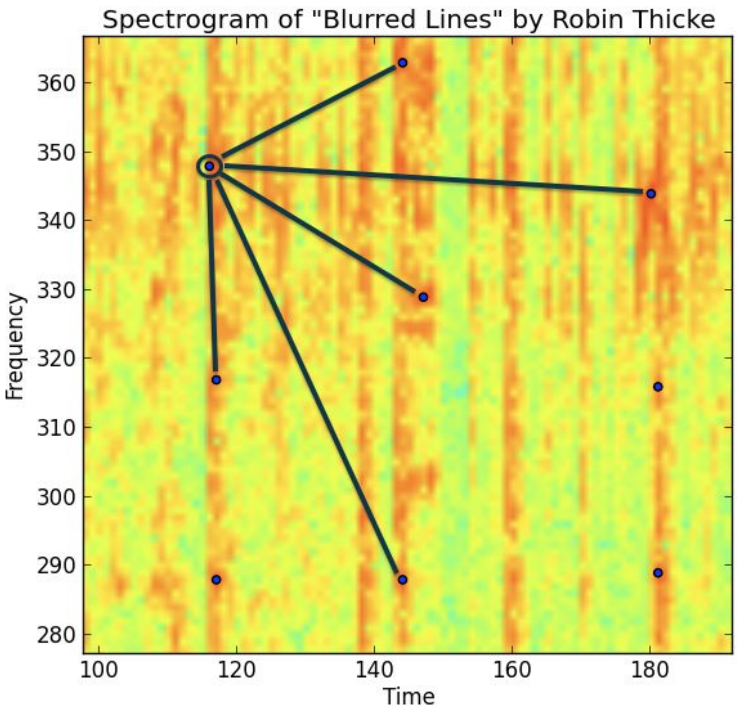
- We can’t key off the pair `(time_offset, frequency)`: database would be enormous, and processing would be terrible.
- Frequency alone leads to many prospective matches.

Shazam’s solution: look at frequency pairs

Anchor: `(t0, f0)`
Target: `(t1, f1)`

Hash is 32-bit integer of:

```
[10 bits f0,
10 bits f1,
10 bits (t1 - t0)]
```



Server side:

Only need a linear scan of each track to generate fingerprints

Part II: Looking up fingerprints

Server side: lookup

Incoming stream: h0:t0, h1:t1, h2:t2, …

(recall each hash = [freq0, freq1, time_delta])

Form buckets:

Song_xyz: h1:t1_xyz, h4:t4_xyz. h7:t7_xyz

Song_abc: h0:t0_abc, h1:t1_abc, h3:t3_abc, h5:t5_abc, h6:t6_abc

Song_123: h0:t0_123, h8:t8_123

Server side: computing correlations

Bad match: many key matches but not time-aligned

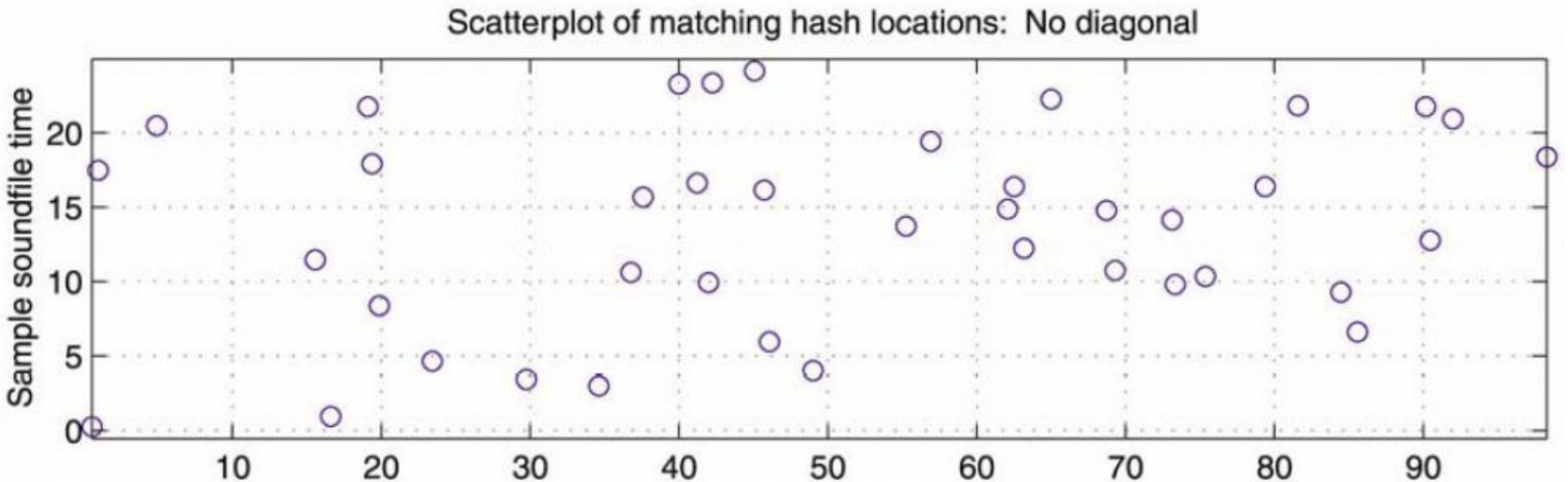


Fig. 2A

Server side: computing correlations

Good match: many are time-aligned

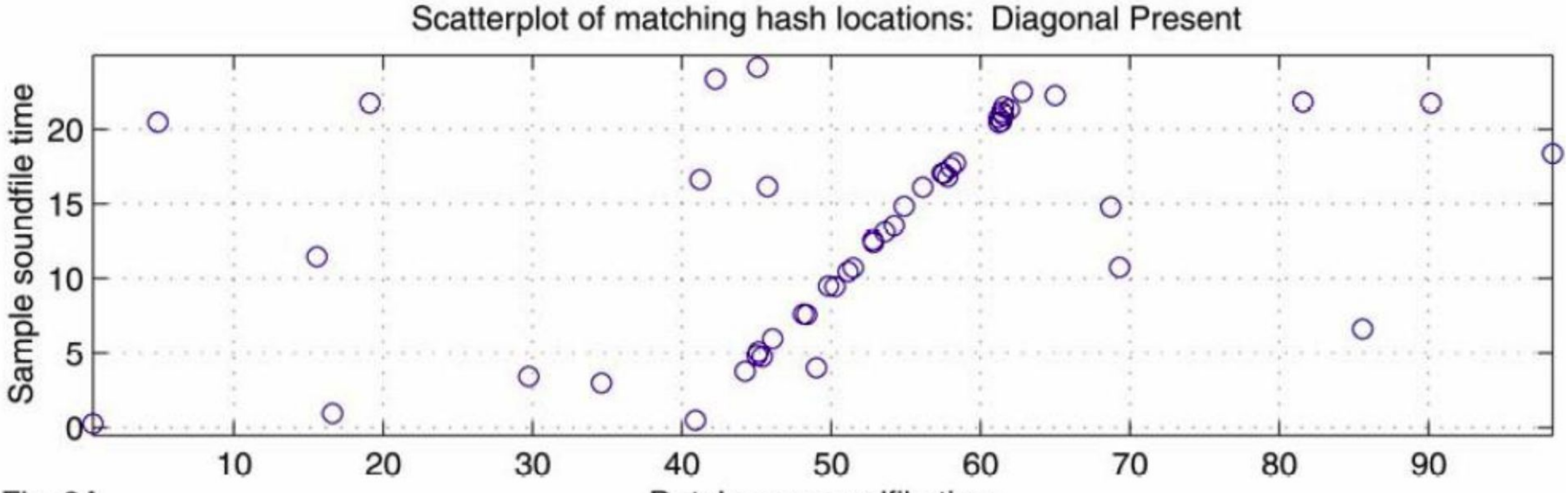


Fig. 3A

How does shazam measure correlations?

- Could use robust regression, R^2 or whatnot (time complexity anyone?)
- Much simpler approach: histograms (time complexity anyone?):
  - Denote $\{ t_i \}$ set of time offsets from sample, $\{ t'_i \}$ time offsets from database.
  - If from same song, $t_i = t'_i + c$ for some constant c.
  - Form histograms of $\{ t_i - t'_i \}$ and look for peaks.

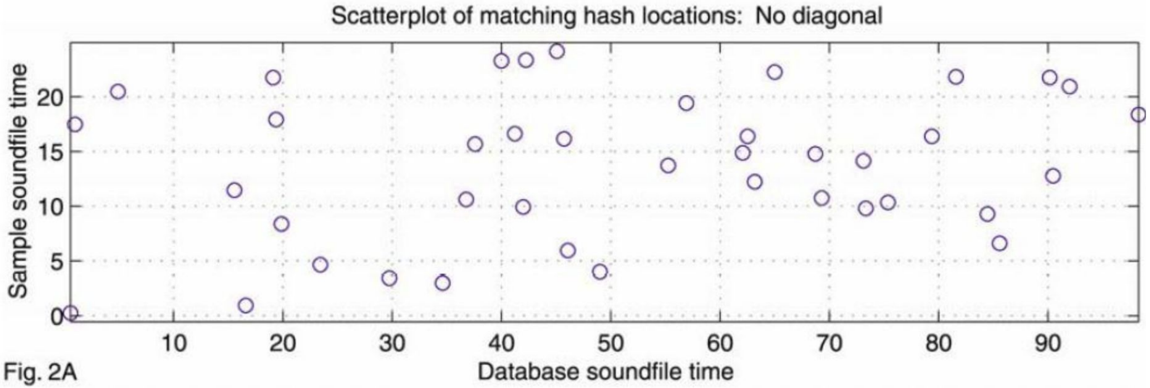

Scatterplot of matching hash locations: No diagonal
Sample soundfile time
Database soundfile time
0
5
10
15
20
10
20
30
40
50
60
70
80
90

Fig. 2A

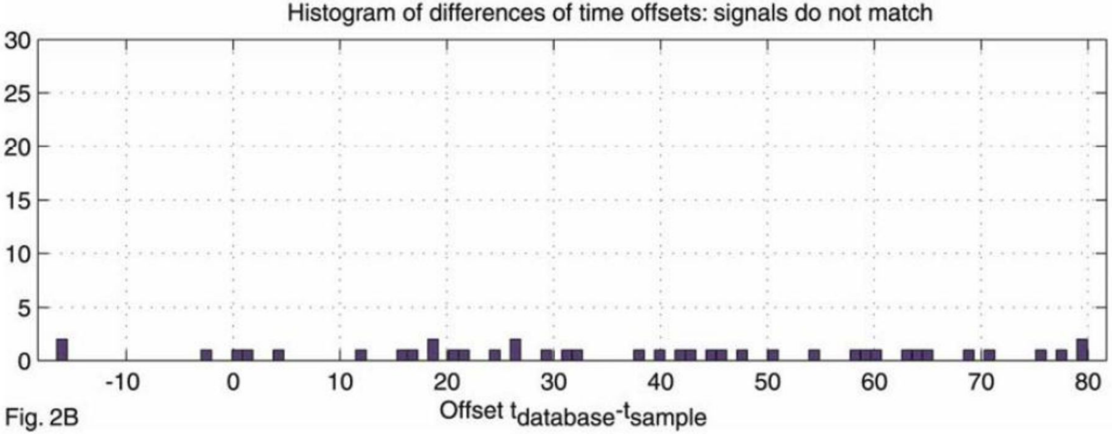

Histogram of differences of time offsets: signals do not match
0
5
10
15
20
25
30
-10
0
10
20
30
40
50
60
70
80
Offset $t_{database}$-t_{sample}

Fig. 2B

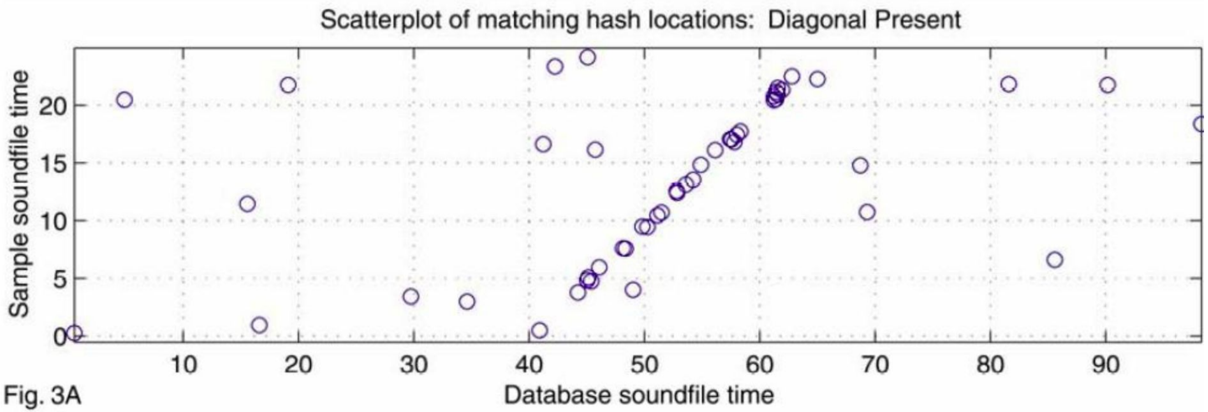

Scatterplot of matching hash locations: Diagonal Present
Sample soundfile time
Database soundfile time
0
5
10
15
20
10
20
30
40
50
60
70
80
90

Fig. 3A

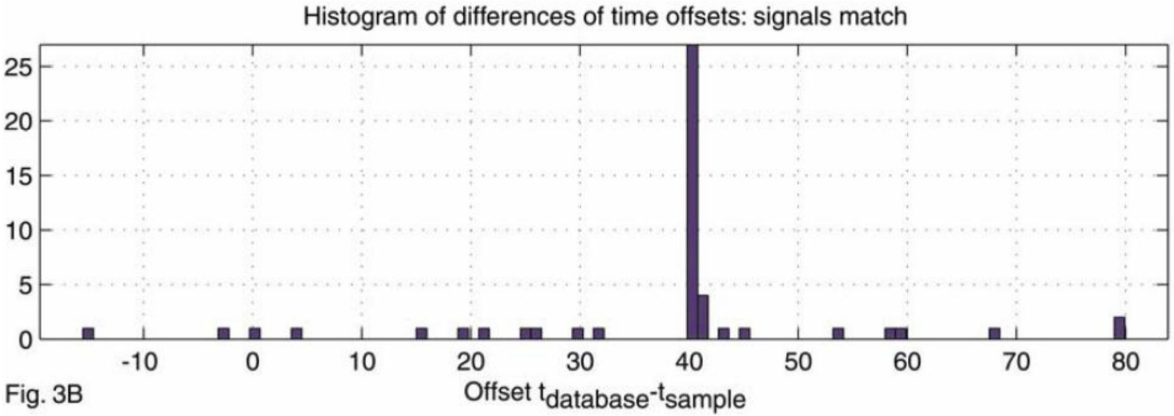

Histogram of differences of time offsets: signals match
0
5
10
15
20
25
-10
0
10
20
30
40
50
60
70
80
Offset $t_{database}$-t_{sample}

Fig. 3B

Questions? Thank you!

thomas.d.peters@gmail.com

- “An industrial-strength audio search algorithm”, by Avery Li-chun Wang. *Proceedings of the 4 th International Conference on Music Information Retrieval*
- https://github.com/worldveil/dejavu