The Basics of Audio Fingerprinting

Stephen Shum

October 24, 2011

Take Home Questions

- What is audio fingerprinting?
- Why fingerprint?
- How does one fingerprint effectively?

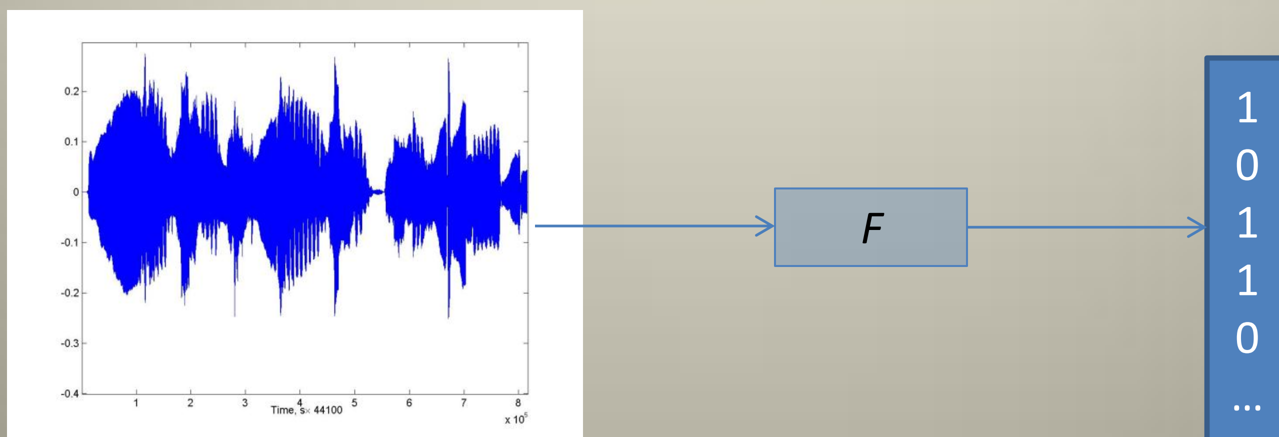
- Along the way,
  - Examples of existing systems and technologies
    - Philips, Shazam, Echonest
  - Brief overview of my summer at Google

Disclaimer

- My exposure to this topic is <u>at most</u> 4 papers ahead of anyone in this group who is seeing this material for the first time.
  - Questions, interruptions, and speaker berating will be tolerated

What is an audio fingerprint?

- Short summary of an audio object using a limited number of bits.



  - Example: i-vectors

Fingerprinting → Hashing

- Hash functions allow comparison of two large objects, X and Y, by just comparing their respective hash values $H(X)$ and $H(Y)$.
  – For a properly designed fingerprint function F, there should be a threshold T such that…
  – If X and Y are similar, then $||F(X) – F(Y)|| < T$ with very high probability,
  – And $||F(X) – F(Y)|| > T$ if X and Y are dissimilar.

- More on this later…

Why fingerprint?

- Efficient mechanism to establish the perceptual equality of two audio objects.

- Advantages
  - Reduced memory/storage requirements
  - Efficient comparison
    - Perceptual irrelevancies removed
  - Efficient search

Applications of Audio Fingerprinting

- Broadcast monitoring
  – Identifying what’s played on public broadcasts
- Audio/song identification
  – Mobile phone recordings severely degraded.
- Filtering technology for file sharing
- Automatic music library organization
  – Correct meta-data inconsistencies

Qualities of Effective Fingerprints

- Discriminative power
- Distortion invariance
- Compactness
- Computational simplicity

- Granularity (application dependent)
  - How many seconds of audio is needed to identify an audio clip?

The Generic Framework (I)

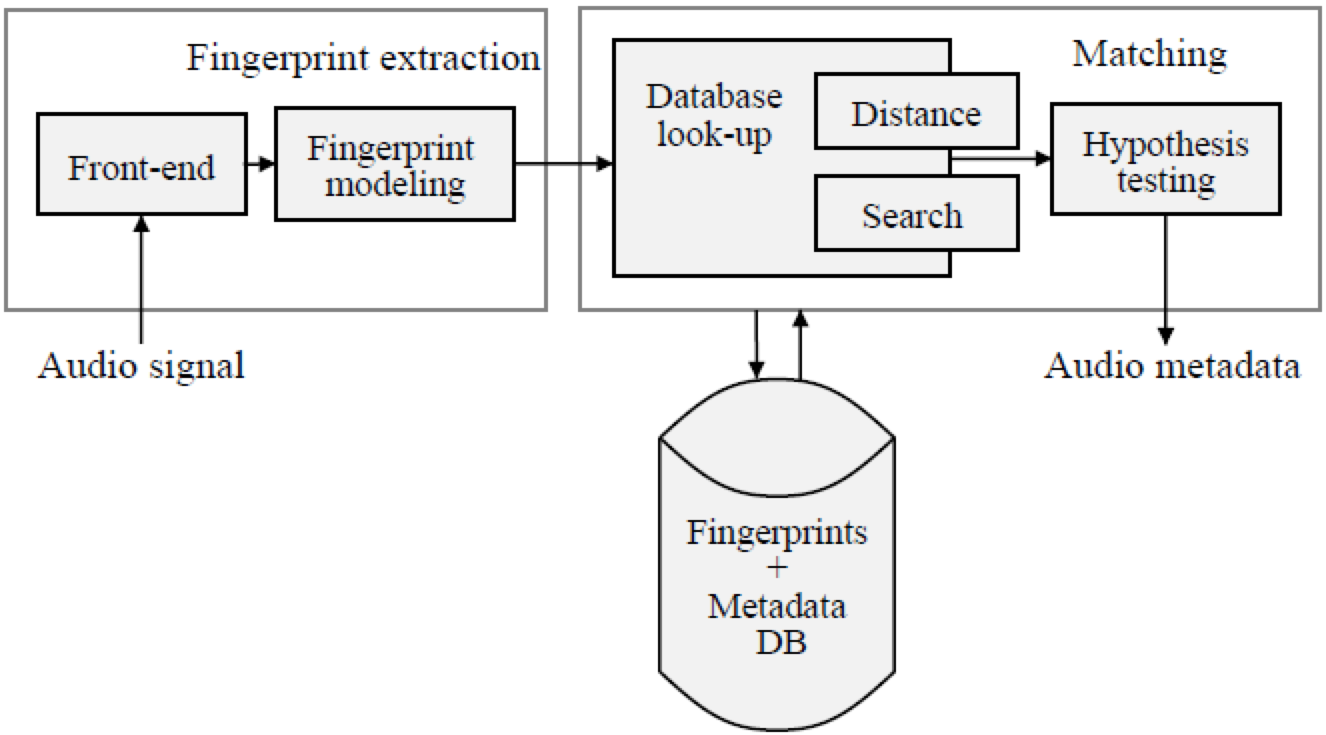


Fig. 1. Content-based Audio Identification Framework.

The Generic Framework (II)

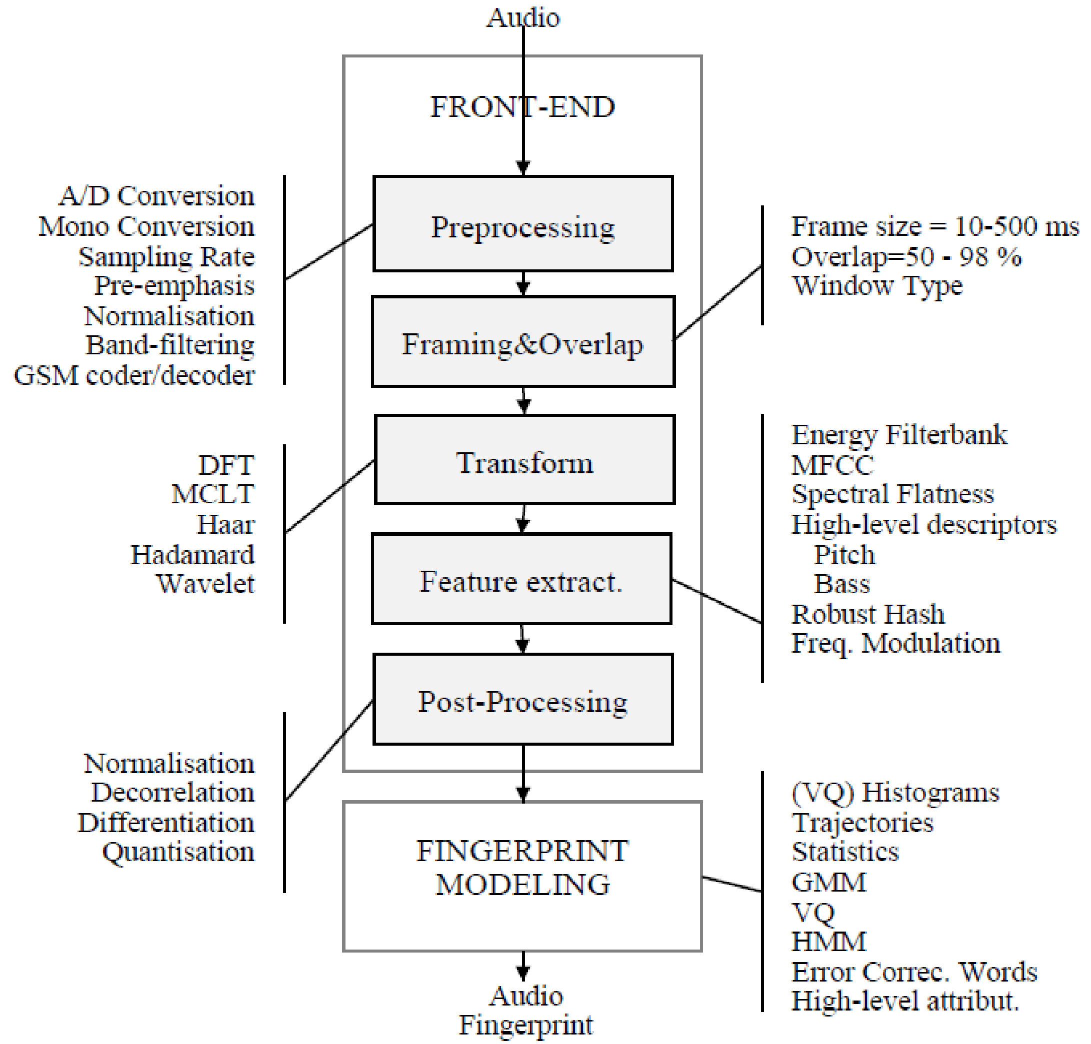


Fig. 2. Fingerprint Extraction Framework: Front-end (top) and Fingerprint modeling (bottom).

The Generic Framework (III)

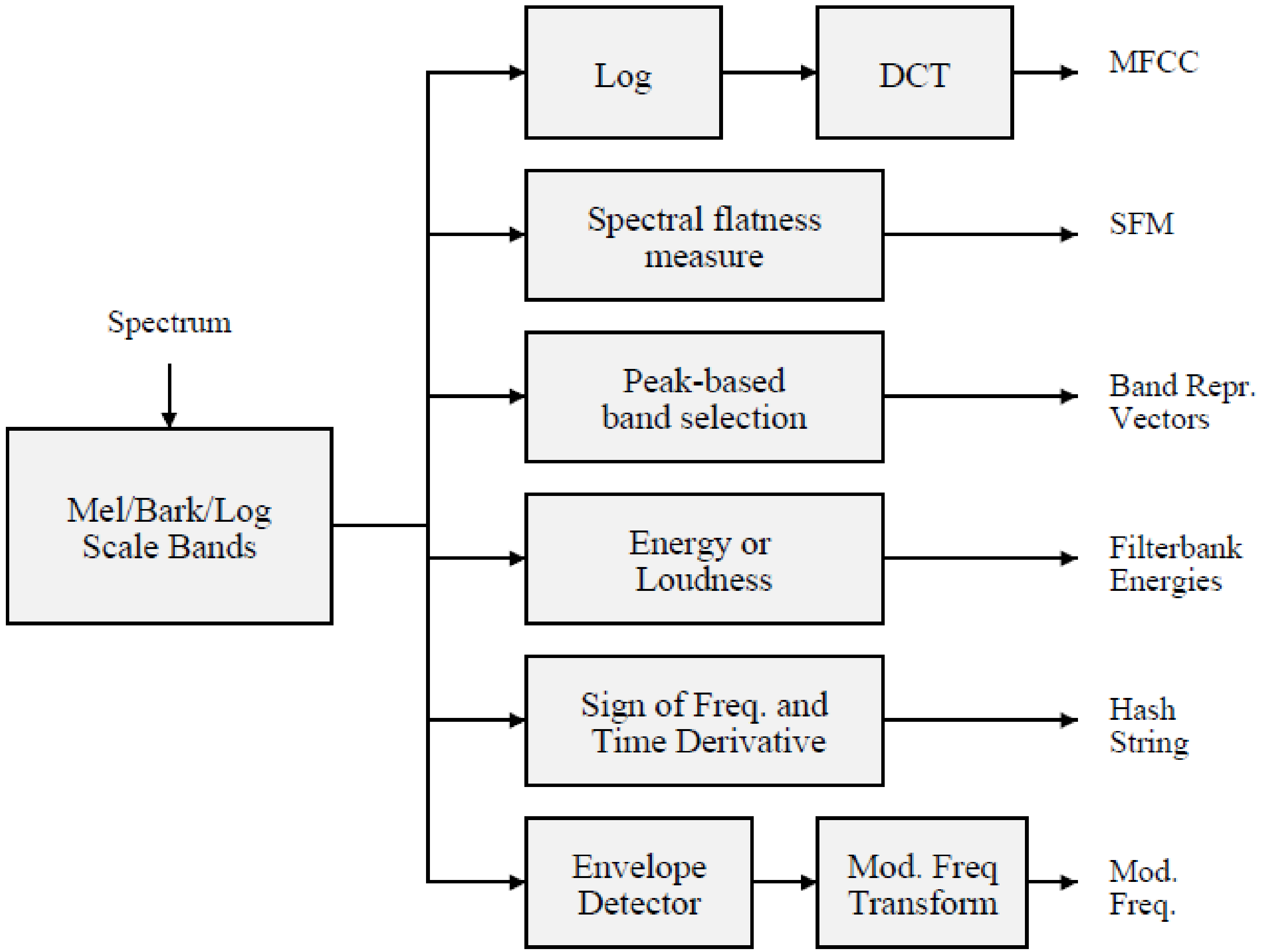


Fig. 3. Feature Extraction Examples

Quick Clarification

- An audio fingerprint can be…
  - A single vector that summarizes the entire file
    - i-vector
  - A stream of *sub-fingerprints*
    - Shazam, etc.

Roadmap

- Introduction
  - Audio Fingerprinting Basics
- Example Systems
  - Shazam
  - Google
- Locality Sensitive Hashing (LSH)
  - Winner Take All (WTA) Hash
  - MinHash
- Recap

Roadmap

- Introduction
  - Audio Fingerprinting Basics
- Example Systems
  - **Shazam**
  - Google
- Locality Sensitive Hashing (LSH)
  - Winner Take All (WTA) Hash
  - MinHash
- Recap

Let’s talk Shazam

- Recognize a song from a short snippet of audio recorded on a mobile phone.
  - Database of nearly 2 million tracks
  - Recorded snippet up to 15 seconds in length
- “Combinatorically hashed time-frequency constellation analysis”

Shazam's Guiding Principles

- Temporally localized
  - Calculate sub-fingerprints using audio samples near a corresponding point in time.
- Translation-invariant
  - Recorded snippet can start anywhere in the song.
- Robust
  - Dealing with severely degraded audio.
- Sufficiently (but not overly) entropic

Entropy???

- Insufficient entropy leads to excessive and spurious matches.

- Too much entropy leads to fragility and non-reproducibility of fingerprint tokens in the presence of noise and distortion.

“Combinatorically hashed time-frequency constellation analysis”

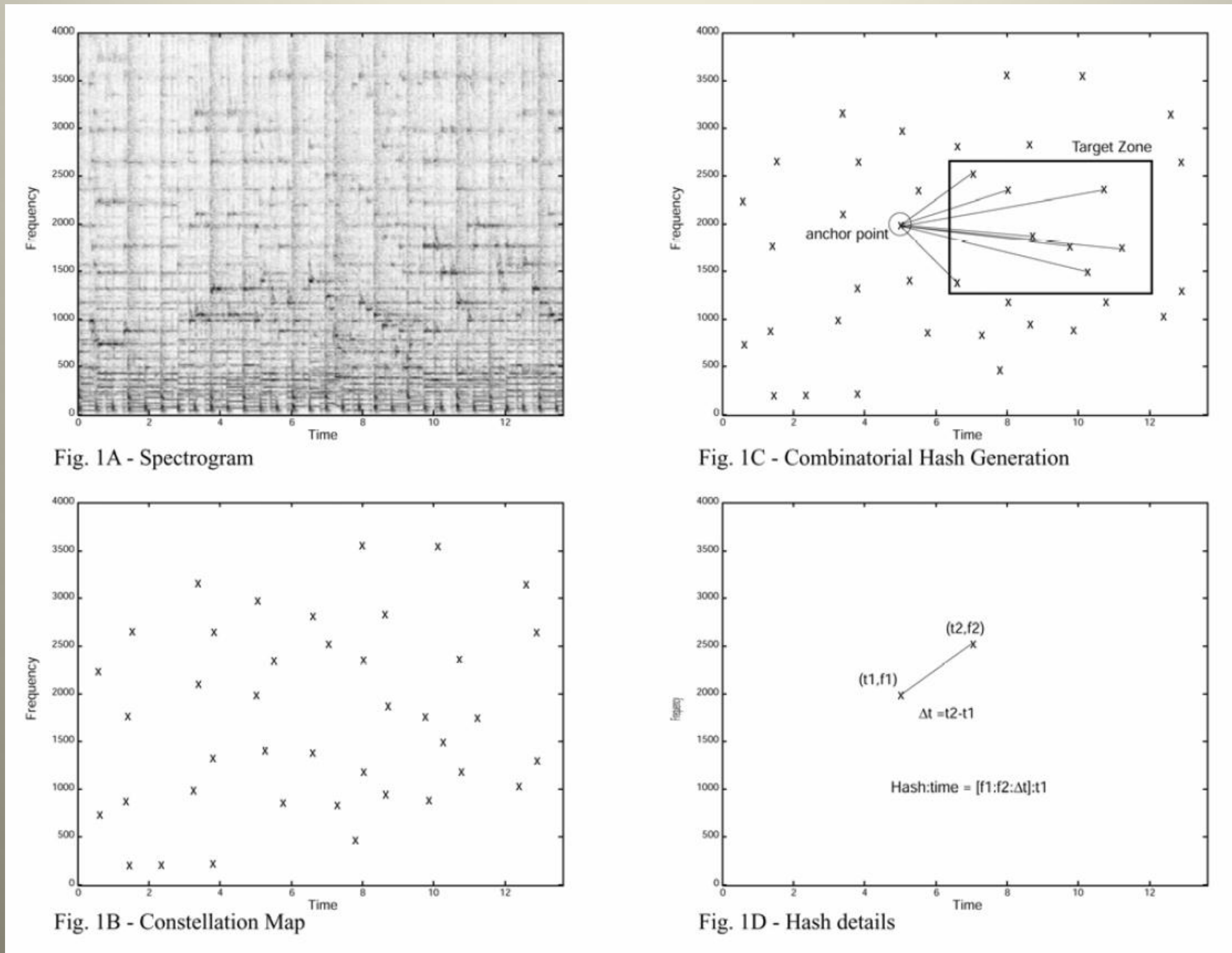


Fig. 1A - Spectrogram

Fig. 1C - Combinatorial Hash Generation

Fig. 1B - Constellation Map

Fig. 1D - Hash details

Hash Generation

- 1024 frequency bins
  - f1 → 10bits
  - f2 → 10bits
- t2-t1 → 10bits

= 30 bits of entropy

→ 32-bit unsigned integer

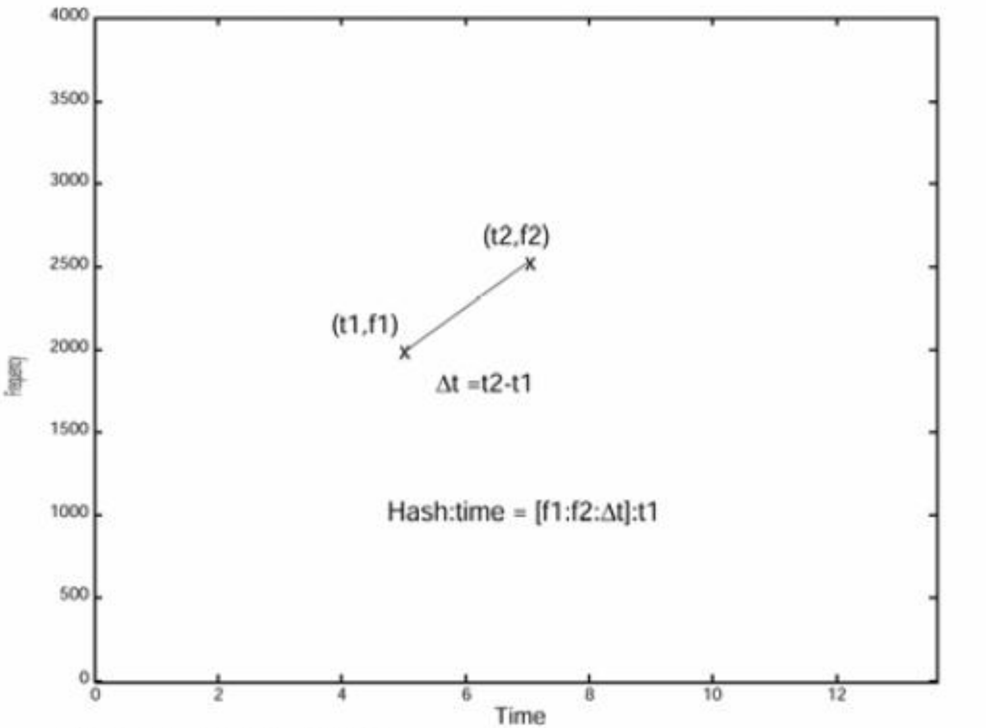


Fig. 1D - Hash details

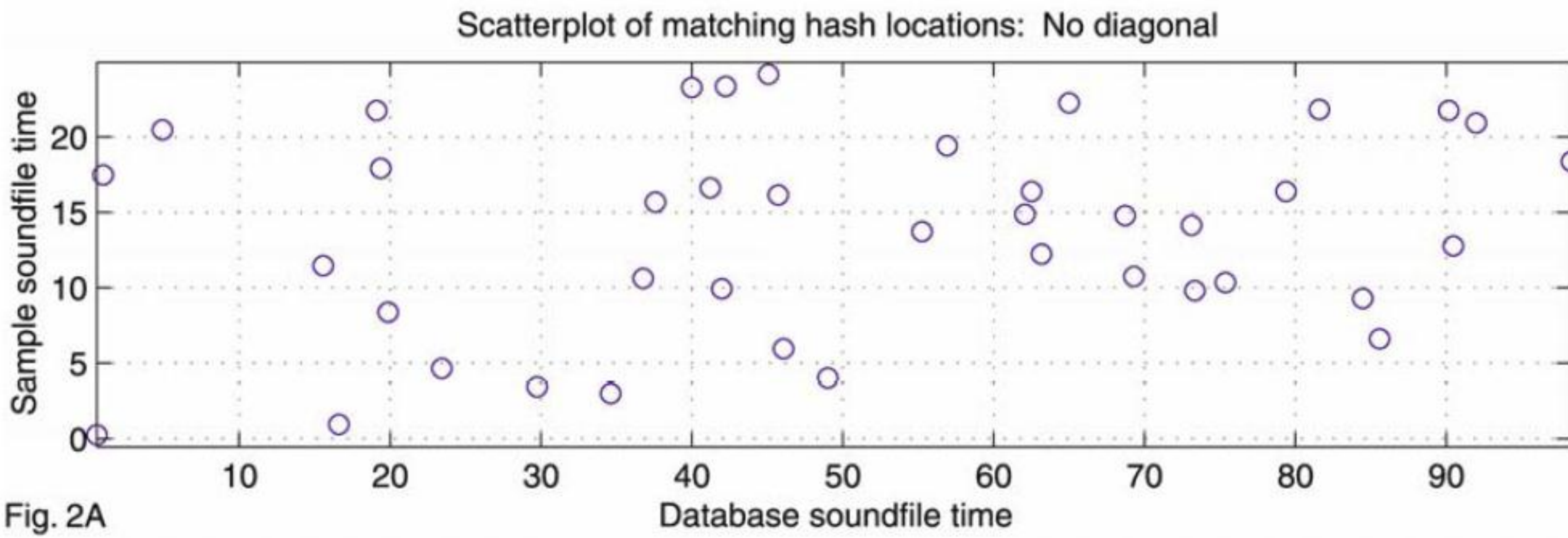

Scatterplot of matching hash locations: No diagonal
Sample soundfile time
0
5
10
15
20
10
20
30
40
50
60
70
80
90
Database soundfile time

Fig. 2A

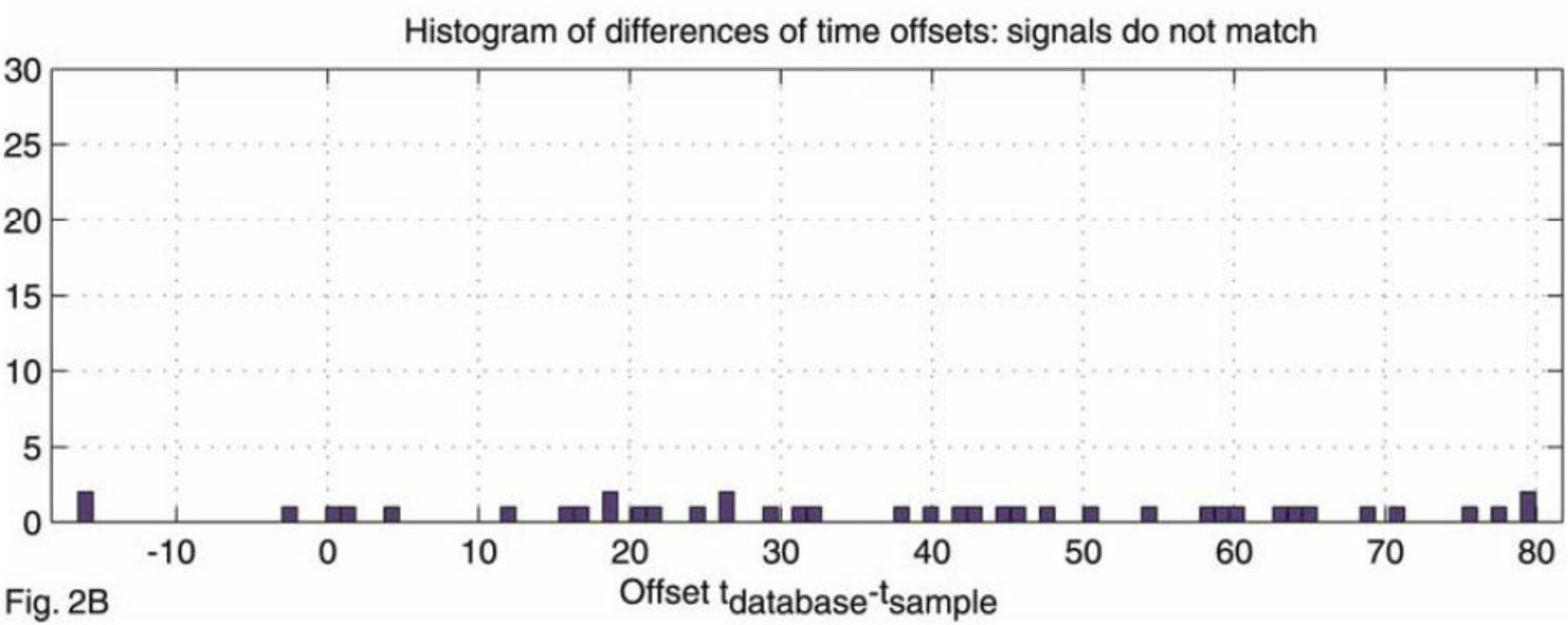

Histogram of differences of time offsets: signals do not match
0
5
10
15
20
25
30
-10
0
10
20
30
40
50
60
70
80
Offset $t_{database}-t_{sample}$

Fig. 2B

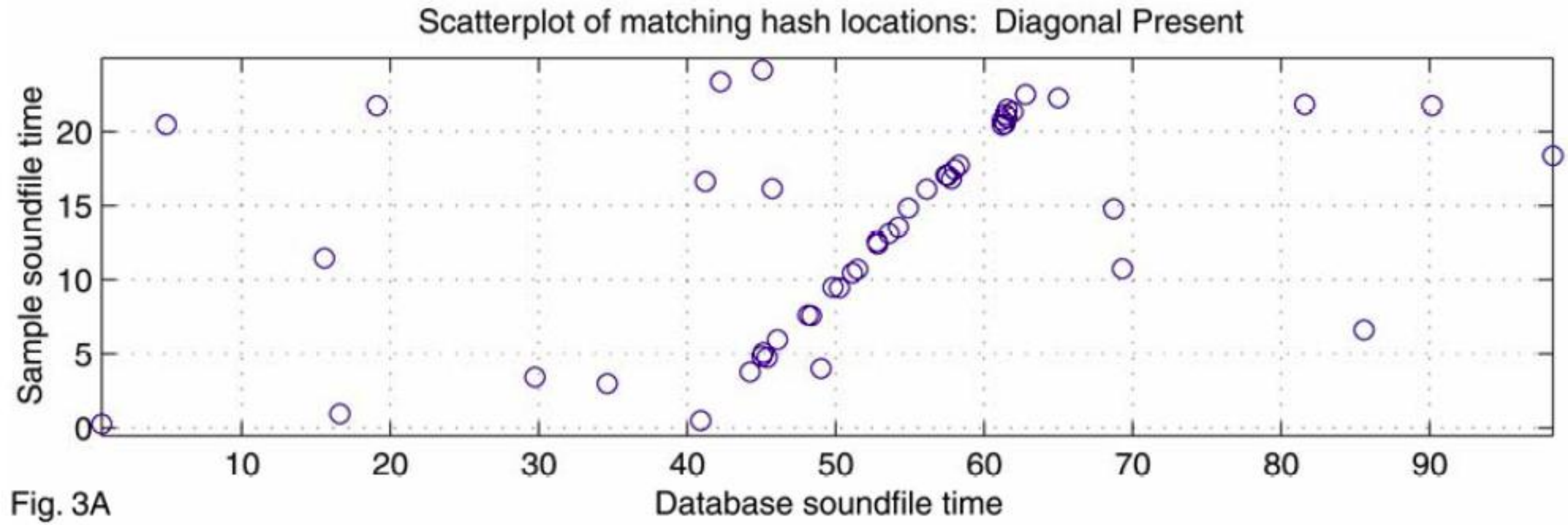

Scatterplot of matching hash locations: Diagonal Present
Sample soundfile time
Database soundfile time
0
5
10
15
20
30
40
50
60
70
80
90

Fig. 3A

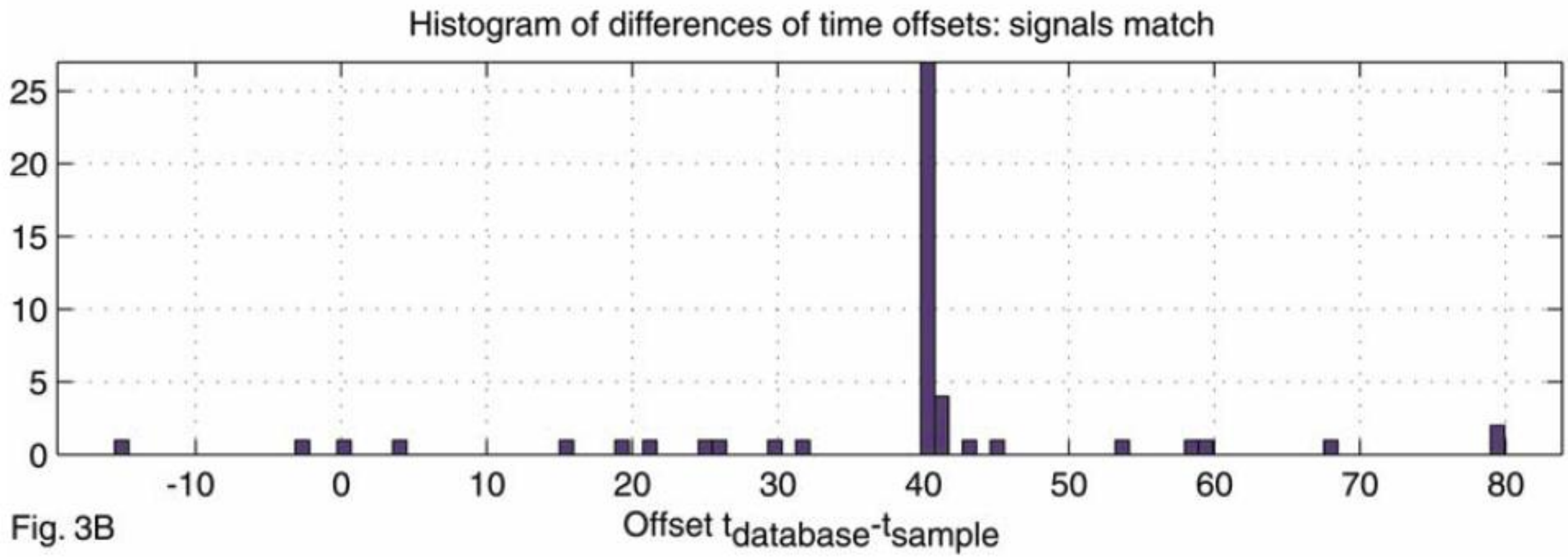

Histogram of differences of time offsets: signals match
0
5
10
15
20
25
-10
30
40
50
60
70
80
Offset $t_{database}$-t_{sample}

Fig. 3B

Roadmap

- Introduction
  - Audio Fingerprinting Basics
- Example Systems
  - Shazam
  - **Google**
- Locality Sensitive Hashing (LSH)
  - Winner Take All (WTA) Hash
  - MinHash
- Recap

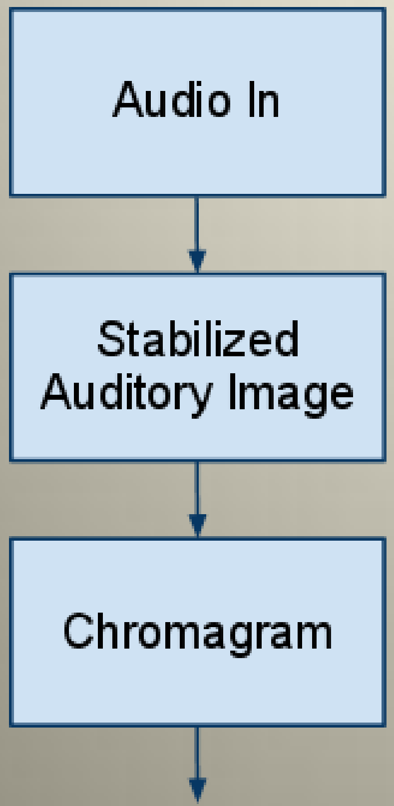
My Google Challenge

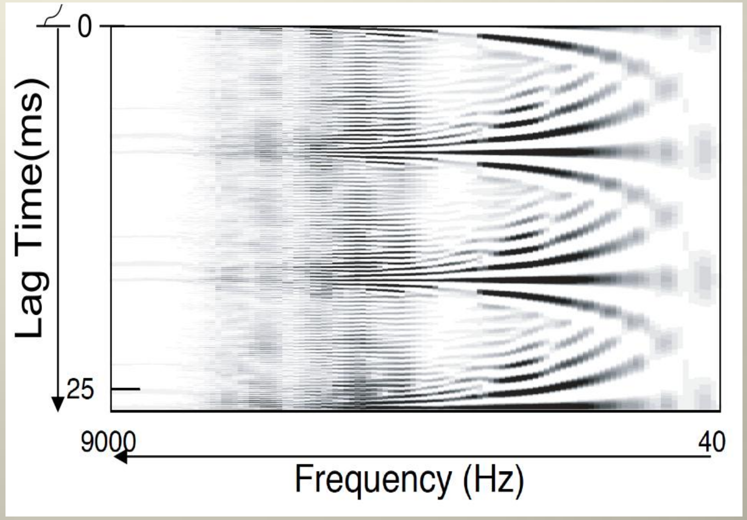
- Cover song detection
  - Also known as “version identification”
  - To identify a common musical work that might have been highly transformed by two different musicians.

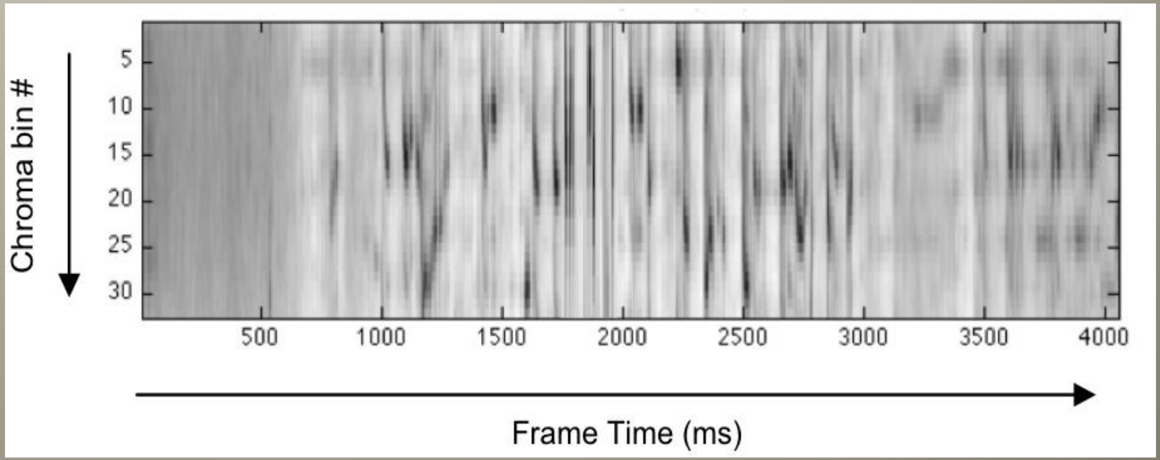
Motivation

- Commercial reasons
  - Detection of copyright infringement
  - Content filtering on YouTube
- Academic reasons
  - “Finding and understanding human transformations of a musical piece force us to develop intelligence audio algorithms that recognize common patterns among musical excerpts.”
  - Most existing algorithms were not designed for datasets of Google proportions.

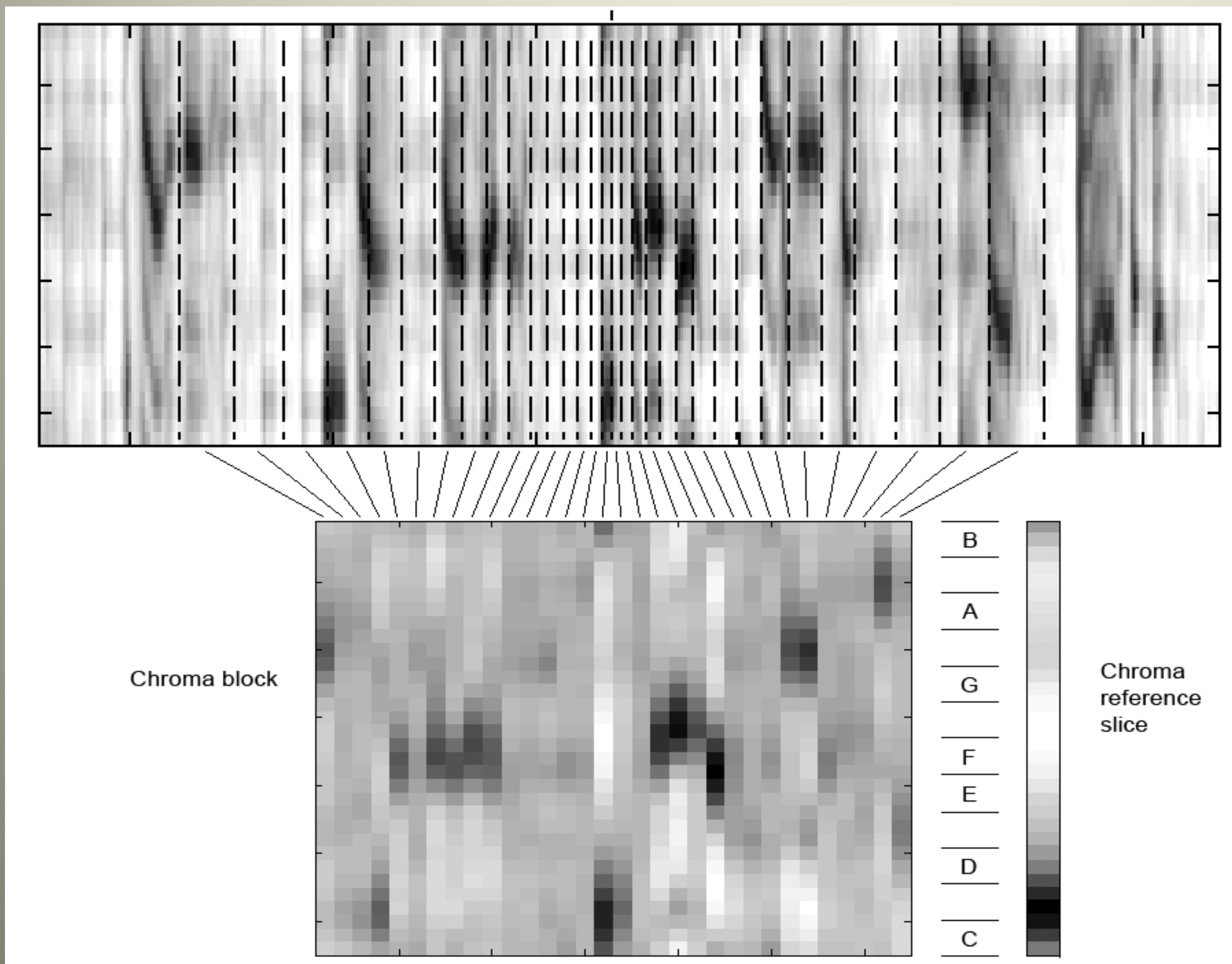
Existing System – Audio Features

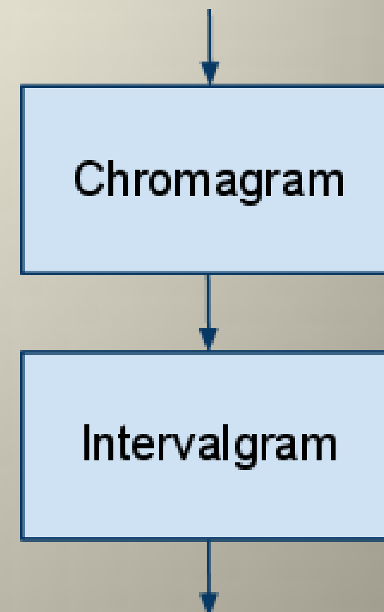




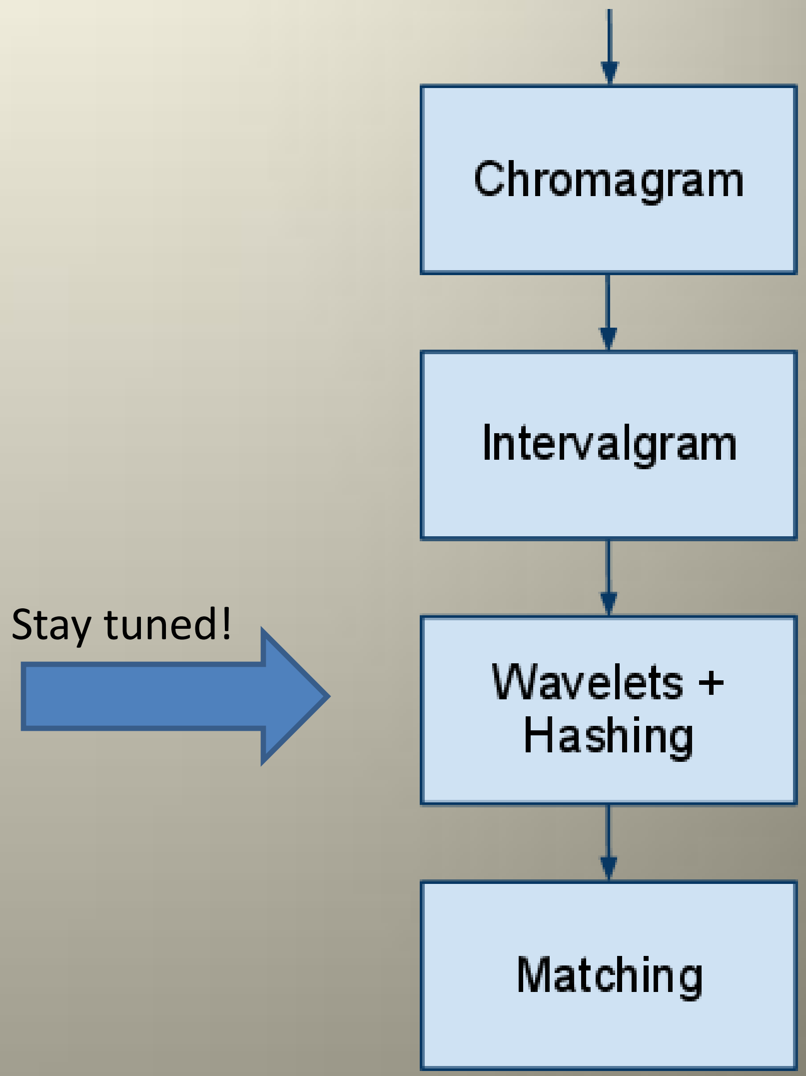


Current Pipeline - Melody Features



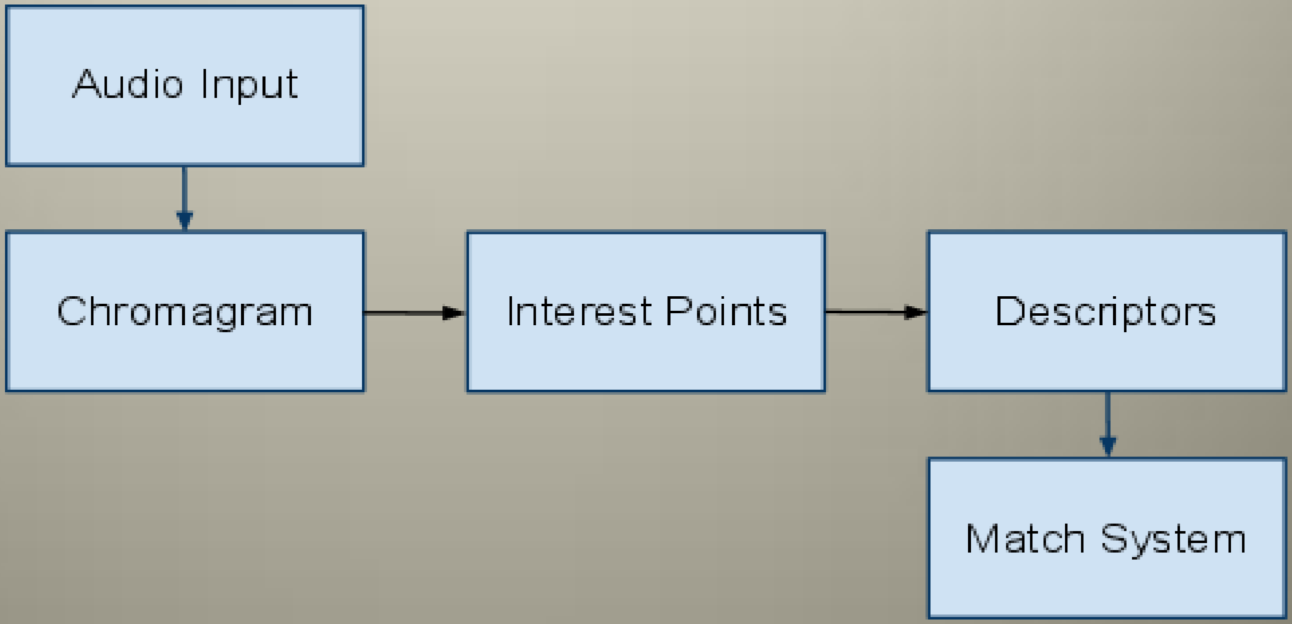


Current Pipeline - Overview



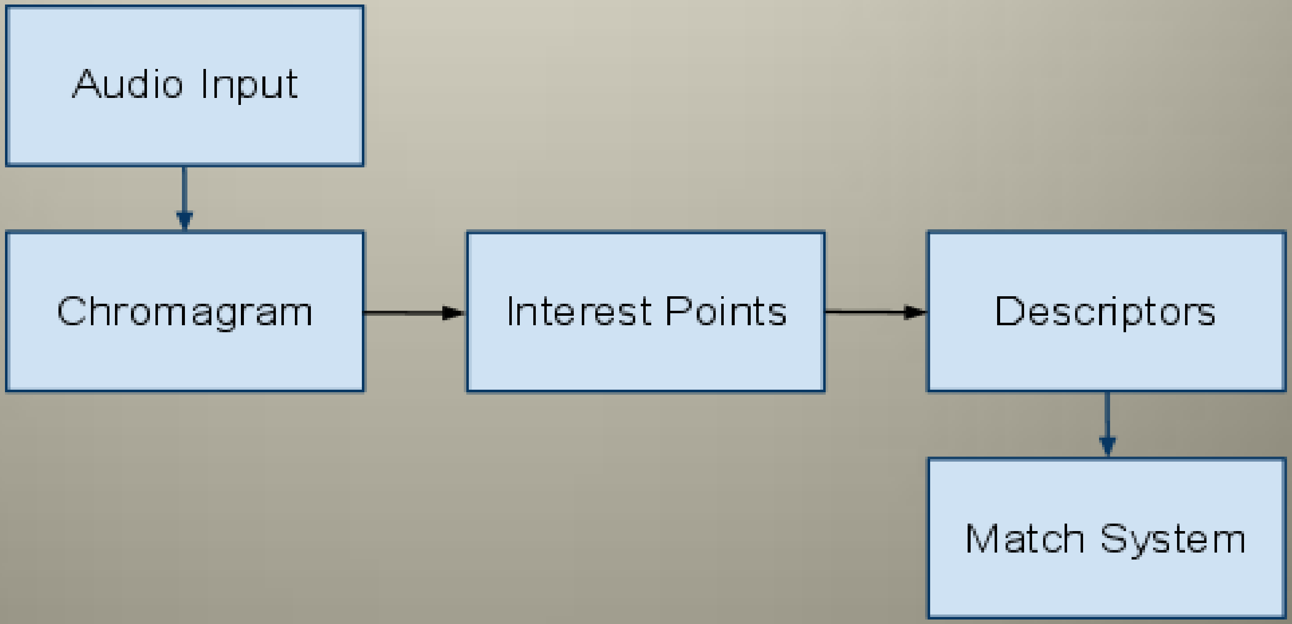
Interest Point-based Approach

- Shazam-inspired
  - "Constellation" → Interest Points
  - "Combinatorial hash" → Descriptors



Key Challenge

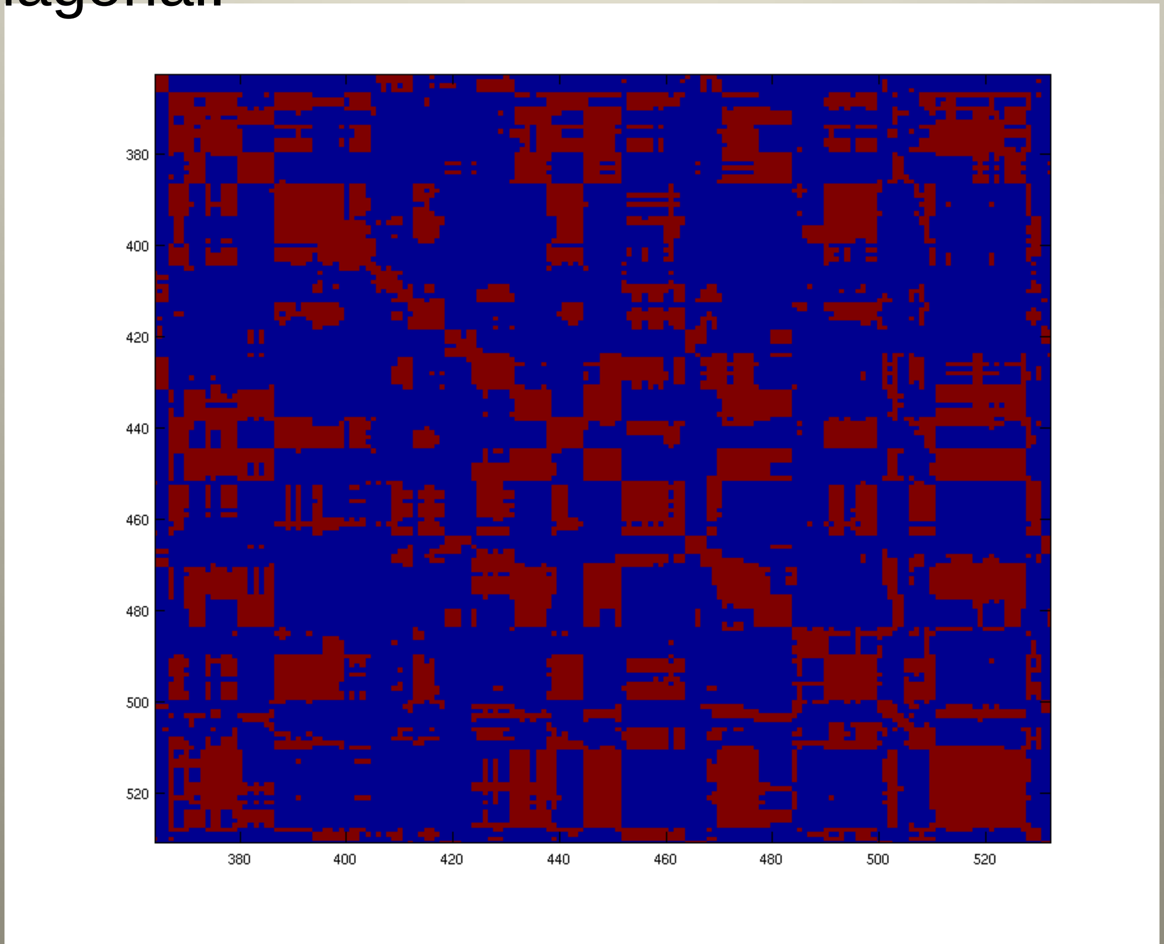
- Shazam is about exact-match audio.
  - Descriptors must match exactly for hit to occur.
- Cover song detection is all about *fuzziness*.



Interest Point Detection - Approach I

<u>Binarized Self-Similarity Matrix</u>

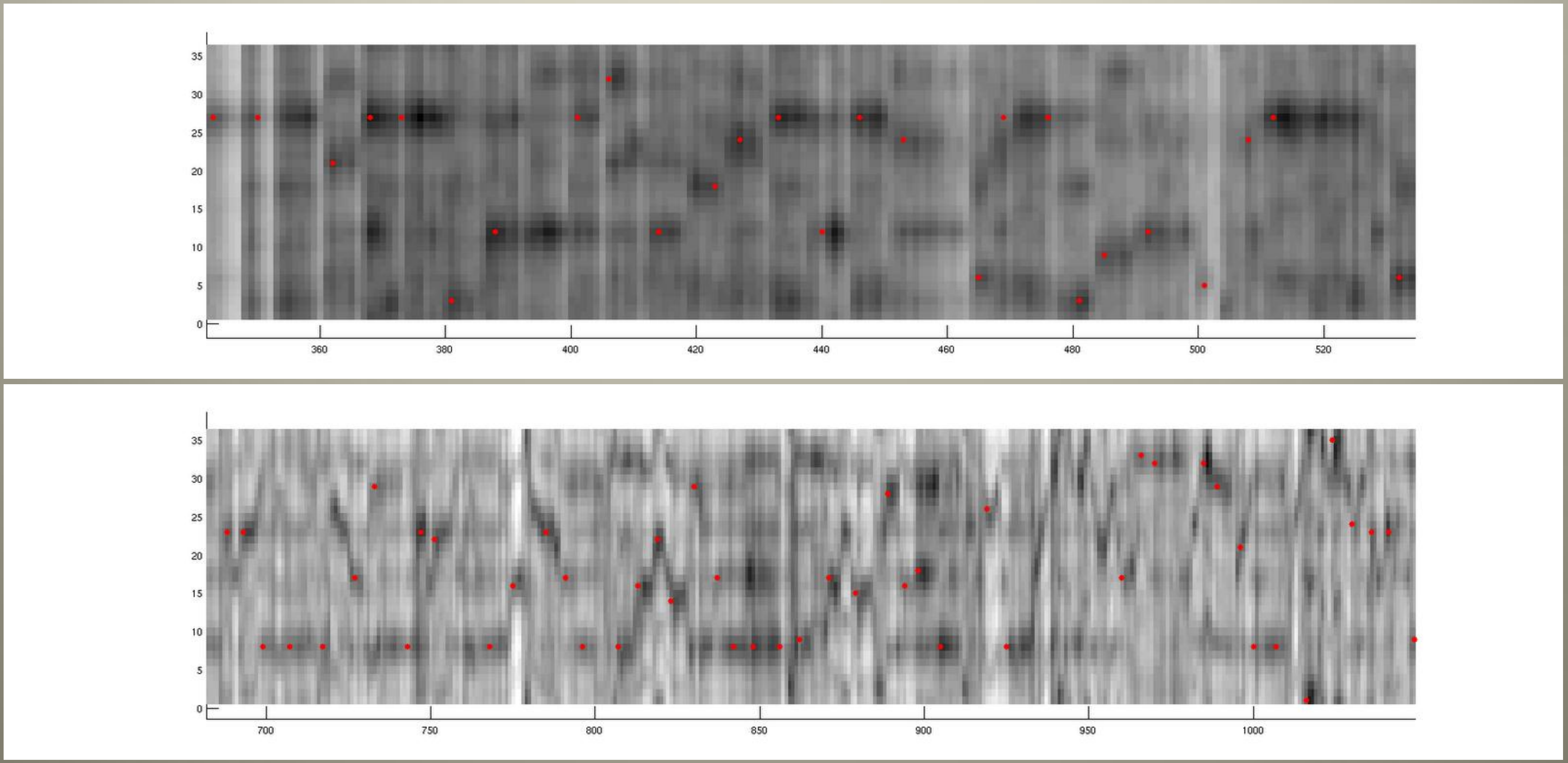
Onset of sustained notes create locally self-similar squares along the diagonal.



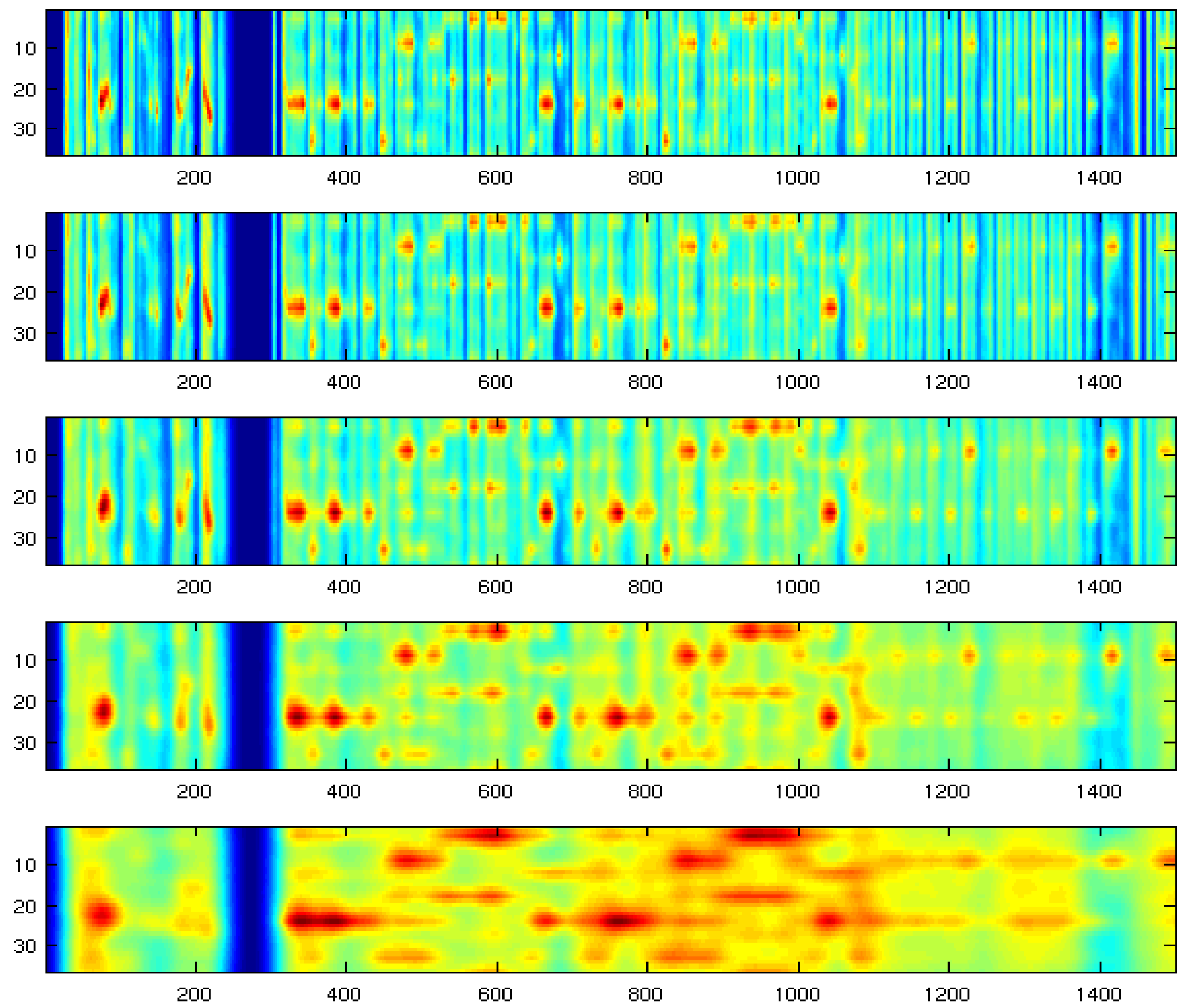
Interest Point Detection - Approach I

<u>Binarized Self-Similarity Matrix</u>

Onset of sustained notes create locally self-similar squares along the diagonal.

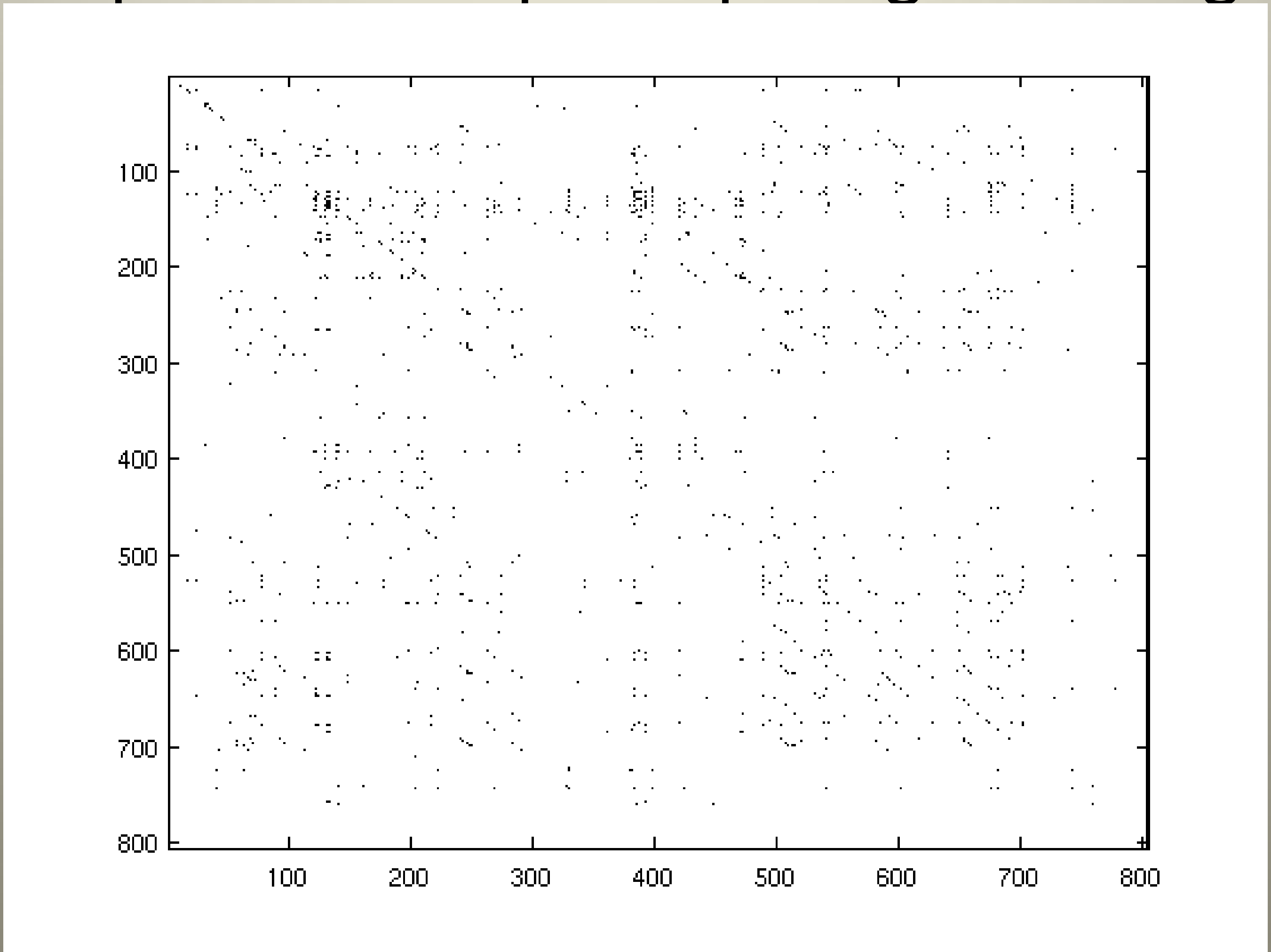


Approach II - Scale Space



Sample Result – Fail!

- Descriptor heatmap comparing two songs



Consolation

T. Bertin-Mahieux and D. Ellis, “Large-Scale Cover Song Recognition Using Hashed Chroma Landmarks,” in *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2011.

Roadmap

- Introduction
  - Audio Fingerprinting Basics
- Example Systems
  - Shazam
  - Google
- Locality Sensitive Hashing (LSH)
  - Winner Take All (WTA) Hash
  - MinHash
- Recap

Back to the Big Picture

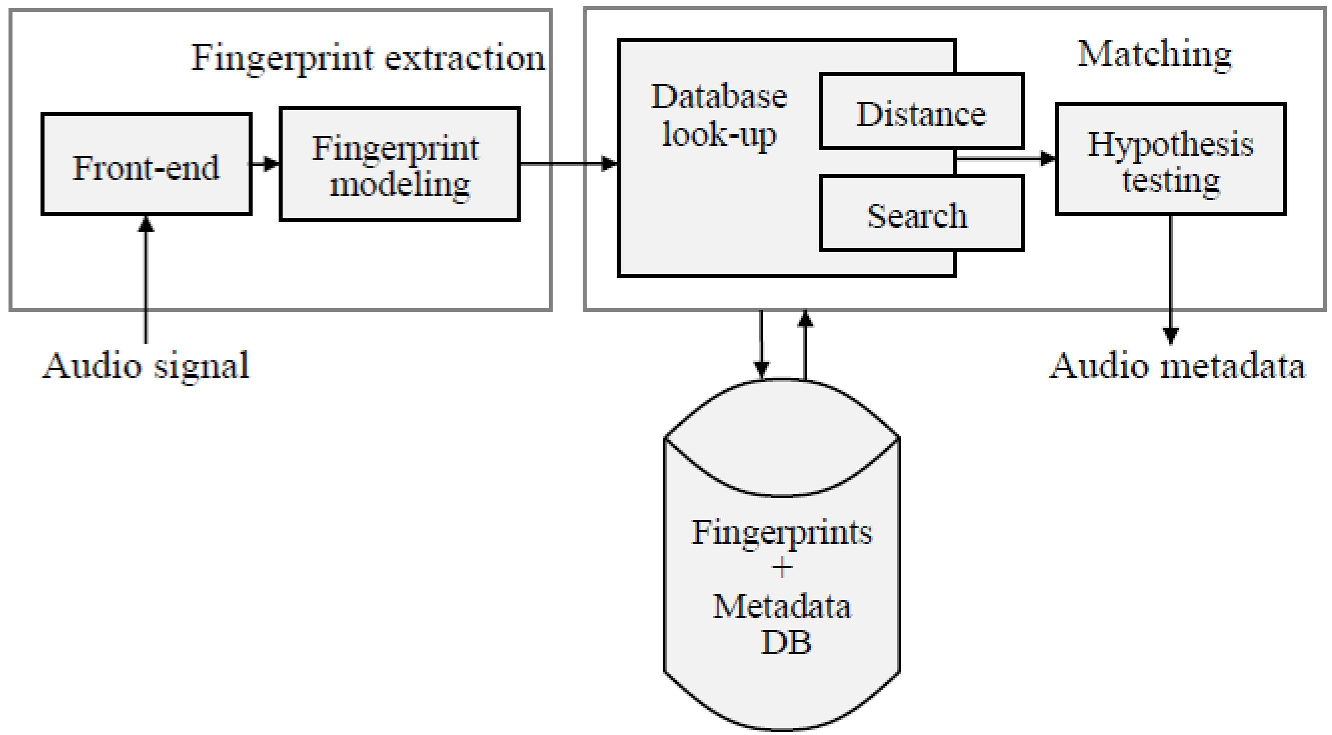


Fig. 1. Content-based Audio Identification Framework.

Algorithms for Hashing

- Locality Sensitive Hash (LSH)
  - For some distance metric $d(\cdot)$ and threshold $R > 0$,
  - An LSH family $\boldsymbol{F}$ is a family of functions where
  - For any two points p, q
  - And function $h(\cdot)$ chosen uniformly at random from $\boldsymbol{F}$,
  - If $d(p,q) < R$
    - Then $h(p) = h(q)$ with probability at least P_1 (i.e. collide)
  - And if $d(p,q) > R$
    - Then $h(p) = h(q)$ with probability at most P_2.

Details

- $h(\cdot)$ is typically a hash function
  - Bit sampling of binary input vectors
    - $h_i(\boldsymbol{x}) = x_i \in \{0, 1\}$
  - Random projection on some normal unit vector r
    - $h_r(v) = sgn(v \cdot r) \in \{+1, -1\}$
  - MinHash

- Can create more complex hash functions
  - $g(\cdot)=[h_1(\cdot), \ldots, h_k(\cdot)]$

Algorithm Preprocessing
Input A set of points P,
l (number of hash tables),
Output Hash tables $\mathcal{T}_i$, $i = 1, \ldots, l$
Foreach $i = 1, \ldots, l$
Initialize hash table $\mathcal{T}_i$ by generating
a random hash function $g_i(\cdot)$
Foreach $i = 1, \ldots, l$
Foreach $j = 1, \ldots, n$
Store point p_j on bucket $g_i(p_j)$ of hash table $\mathcal{T}_i$

Algorithm Approximate Nearest Neighbor Query
Input A query point q,
K (number of appr. nearest neighbors)
Access To hash tables $\mathcal{T}_i$, $i = 1, \ldots, l$
generated by the preprocessing algorithm
Output K (or less) appr. nearest neighbors
$S \leftarrow \emptyset$
Foreach $i = 1, \ldots, l$
$S \leftarrow S \cup \{$points found in $g_i(q)$ bucket of table $\mathcal{T}_i\}$
Return the K nearest neighbors of q found in set S
/* Can be found by main memory linear search */

Parameter Choices

- k is the width parameter
  - i.e. how many hash functions h to concatenate together to obtain g
- l is the number of hash tables

- Theoretical analysis in
  - A. Gionis, P. Indyk, R. Motwani, “Similarity Search in High Dimensions via Hashing,” in *Proceedings of VLDB*, 1999.

Winner Take All (WTA) Hash

theta = randperm(n);

[max_val, c(i)] = max(X(i, theta(1:K)));

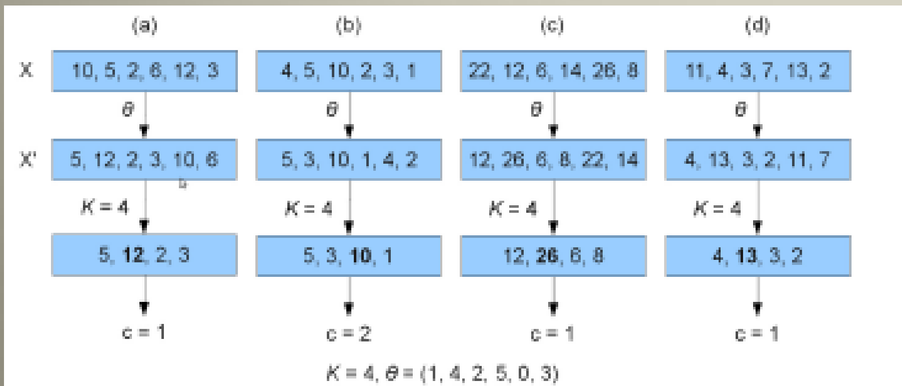


Figure 1: An example with 6-dimensional input vectors, K = 4, and $\theta = (1, 4, 2, 5, 0, 3)$. X in (a) and (b) are unrelated and result in different output codes, 1 and 2 respectively. X in (c) is a scaled and offset version of (a) and results in the same code as (a). X in (d) has each element perturbed by 1 which results in a different ranking of the elements, but the maximum of the first K elements is the same, again resulting in the same code.

Algorithm 1 WTA Hash

Input: A set of m Permutations Θ, window size K, input vector X.

Output: Sparse vector of codes C_X.

1. For each permutation θ_i in Θ.
   (a) Permute elements of X according to θ_i to get X'.
   (b) Initialize i^{th} sparse code c_{x_i} to 0.
   (c) Set c_{x_i} to the index of the maximum value in $X'(1...K)$
      i. For $j = 0$ to $K - 1$
         A. If $X'(j) > X'(c_{x_i})$ then $c_{x_i} = j$.
2. $C_X = [c_{x0}, c_{x1}, ..., c_{xm-1}]$, C contains m codes, each taking a value between 0 and $K - 1$.

MinHash

- Encodes the index of the first 1 under random permutations of binary vectors.
- Hash collision rate corresponds to the Jaccard similarity between binary vectors:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

  – Popular for large-scale clustering (document similarity, etc.)
- Special case of WTA Hash
  – $K = n$, so as to avoid case of having all 0's.

Roadmap

- Introduction
  - Audio Fingerprinting Basics
- Example Systems
  - Shazam
  - Google
- Locality Sensitive Hashing (LSH)
  - Winner Take All (WTA) Hash
  - MinHash
- Recap

References

- J. Haitsma, “A Highly Robust Audio Fingerprinting System,” in *Proceedings of ISMIR*, 2002.
- P. Cano, E. Batlle, T. Kalker, and J.Haitsma, “A Review of Algorithms for Audio Fingerprinting,” in *Workshop on Multimedia Signal Processing*, 2002.
- A. Wang, “An Industrial-Strength Audio Search Algorithm,” in *Proceedings of ISMIR*, 2003.
- V. Chandrasekhar, M. Sharifi, and D. Ross, “Survey and Evaluation of Audio Fingerprinting Schemes for Mobile Query-By-Example Applications,” in *Proceedings of ISMIR*, 2011.
- T. Bertin-Mahieux and D. Ellis, “Large-scale Cover Song Recognition Using Hashed Chroma Landmarks,” in *Proceedings of WASPAA*, 2011.
- A. Gionis, P. Indyk, R. Motwani, “Similarity Search in High Dimensions via Hashing,” in *Proceedings of VLDB*, 1999.
- J. Yagnik, D. Strelow, D. Ross, R. Lin, “The Power of Comparative Reasoning,” in *Proceedings of ICCV*, 2011.