INTERNATIONAL AUDIO LABORATORIES ERLANGEN

AUDIO LABS

Lecture

Music Processing

Audio Retrieval

Meinard Müller

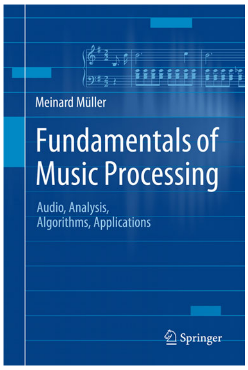
International Audio Laboratories Erlangen

meinard.mueller@audiolabs-erlangen.de

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

Fraunhofer IIS

Book: Fundamentals of Music Processing



Meinard Müller
Fundamentals of Music Processing
Audio, Analysis, Algorithms, Applications
483 p., 249 illus., hardcover
ISBN: 978-3-319-21944-8
Springer, 2015

Accompanying website:
www.music-processing.de

Book: Fundamentals of Music Processing

Chapter		Music Processing Scenario
1		Music Representations
2		Fourier Analysis of Signals
3		Music Synchronization
4		Music Structure Analysis
5		Chord Recognition
6		Tempo and Beat Tracking
7		Content-Based Audio Retrieval
8		Musically Informed Audio Decomposition

Meinard Müller
Fundamentals of Music Processing
Audio, Analysis, Algorithms, Applications
483 p., 249 illus., hardcover
ISBN: 978-3-319-21944-8
Springer, 2015

Accompanying website:
www.music-processing.de

Book: Fundamentals of Music Processing

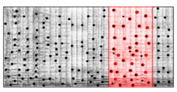
Chapter		Music Processing Scenario
1		Music Representations
2		Fourier Analysis of Signals
3		Music Synchronization
4		Music Structure Analysis
5		Chord Recognition
6		Tempo and Beat Tracking
7		Content-Based Audio Retrieval
8		Musically Informed Audio Decomposition

Meinard Müller
Fundamentals of Music Processing
Audio, Analysis, Algorithms, Applications
483 p., 249 illus., hardcover
ISBN: 978-3-319-21944-8
Springer, 2015

Accompanying website:
www.music-processing.de

Chapter 7: Content-Based Audio Retrieval

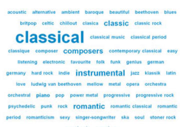
7.1 Audio Identification
7.2 Audio Matching
7.3 Version Identification
7.4 Further Notes

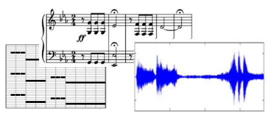


One important topic in information retrieval is concerned with the development of search engines that enable users to explore music collections in a flexible and intuitive way. In Chapter 7, we discuss audio retrieval strategies that follow the query-by-example paradigm: given an audio query, the task is to retrieve all documents that are somehow similar or related to the query. Starting with audio identification, a technique used in many commercial applications such as Shazam, we study various retrieval strategies to handle different degrees of similarity. Furthermore, considering efficiency issues, we discuss fundamental indexing techniques based on inverted lists—a concept originally used in text retrieval.

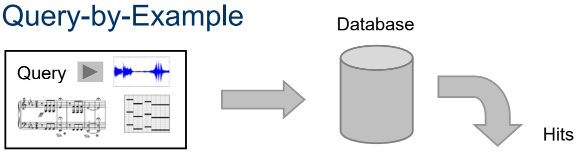
Music Retrieval

- Textual metadata
  - Traditional retrieval
  - Searching for artist, title, …
- Rich and expressive metadata
  - Generated by experts
  - Crowd tagging, social networks
- Content-based retrieval
  - Automatic generation of tags
  - Query-by-example





Query-by-Example



Retrieval tasks:

Audio identification

Audio matching

Version identification

Category-based music retrieval

Bernstein (1962)
Beethoven, Symphony No. 5

Beethoven, Symphony No. 5:
- Bernstein (1962)
- Karajan (1982)
- Gould (1992)

- Beethoven, Symphony No. 9
- Beethoven, Symphony No. 3
- Haydn Symphony No. 94

Query-by-Example

Taxonomy

Retrieval tasks:

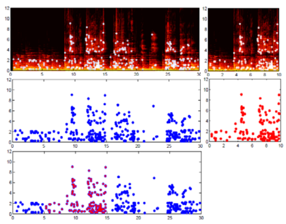
	Specificity level	Granularity level
Audio identification	High specificity	Fragment-based retrieval
Audio matching	↕	↕
Version identification		
Category-based music retrieval	Low specificity	Document-based retrieval

Overview (Audio Retrieval)

- Audio identification (audio fingerprinting)
- Audio matching
- Cover song identification

Overview (Audio Retrieval)

- Audio identification (audio fingerprinting)
- Audio matching
- Cover song identification



Audio Identification

Database: Huge collection consisting of all audio recordings (feature representations) to be potentially identified.

Goal: Given a short query audio fragment, identify the original audio recording the query is taken from.

Notes:
- Instance of fragment-based retrieval
- High specificity
- Not the piece of music is identified but a specific rendition of the piece

Application Scenario

- User hears music playing in the environment
- User records music fragment (5-15 seconds) with mobile phone
- Audio fingerprints are extracted from the recording and sent to an audio identification service
- Service identifies audio recording based on fingerprints
- Service sends back metadata (track title, artist) to user

Audio Fingerprints

An audio fingerprint is a content-based compact signature that summarizes some specific audio content.

Requirements:

- Discriminative power
- Invariance to distortions
- Compactness
- Computational simplicity

Audio Fingerprints

An audio fingerprint is a content-based compact signature that summarizes a piece of audio content

Requirements:

- Discriminative power
- Invariance to distortions
- Compactness
- Computational simplicity

- *Ability to accurately identify an item within a huge number of other items (informative, characteristic)*
- *Low probability of false positives*
- *Recorded query excerpt only a few seconds*
- *Large audio collection on the server side (millions of songs)*

Audio Fingerprints

An audio fingerprint is a content-based compact signature that summarizes a piece of audio content

Requirements:

- Discriminative power
- Invariance to distortions
- Compactness
- Computational simplicity

- *Recorded query may be distorted and superimposed with other audio sources*
- *Background noise*
- *Pitching (audio played faster or slower)*
- *Equalization*
- *Compression artifacts*
- *Cropping, framing*
- *…*

Audio Fingerprints

An audio fingerprint is a content-based compact signature that summarizes a piece of audio content

Requirements:

- Discriminative power
- Invariance to distortions
- Compactness
- Computational simplicity

- *Reduction of complex multimedia objects*
- *Reduction of dimensionality*
- *Making indexing feasible*
- *Allowing for fast search*

Audio Fingerprints

An audio fingerprint is a content-based compact signature that summarizes a piece of audio content

Requirements:

- Discriminative power
- Invariance to distortions
- Compactness
- Computational simplicity

- *Computational efficiency*
- *Extraction of fingerprint should be simple*
- *Size of fingerprints should be small*

Literature (Audio Identification)

- Allamanche et al. (AES 2001)
- Cano et al. (AES 2002)
- Haitsma/Kalker (ISMIR 2002)
- Kurth/Clausen/Ribbrock (AES 2002)
- Wang (ISMIR 2003)

⋮

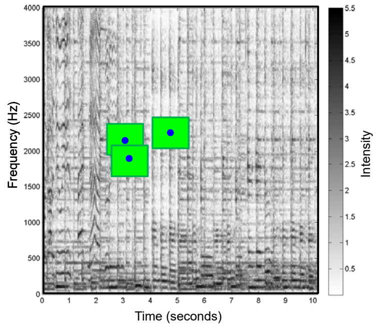
- Dupraz/Richard (ICASSP 2010)
- Ramona/Peeters (ICASSP 2011)

Literature (Audio Identification)

- Allamanche et al. (AES 2001)
- Cano et al. (AES 2002)
- Haitsma/Kalker (ISMIR 2002)
- Kurth/Clausen/Ribbrock (AES 2002)
- Wang (ISMIR 2003)

  ⋮
- Dupraz/Richard (ICASSP 2010)
- Ramona/Peeters (ICASSP 2011)

Fingerprints (Shazam)

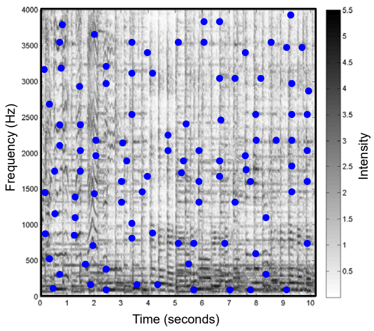


Steps:

1. Spectrogram
2. Peaks
   (local maxima)

- *Efficiently computable*
- *Standard transform*
- *Robust*

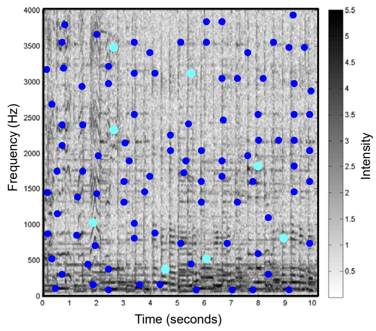
Fingerprints (Shazam)



Steps:

1. Spectrogram
2. Peaks

Fingerprints (Shazam)



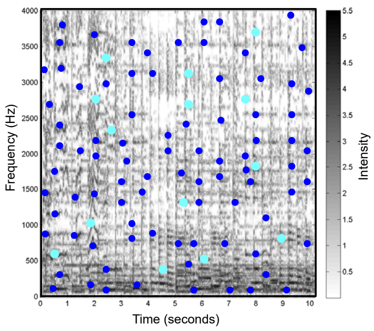
Steps:

1. Spectrogram
2. Peaks / differing peaks

Robustness:

- Noise, reverb, room acoustics, equalization

Fingerprints (Shazam)



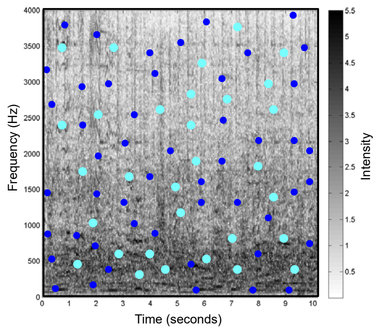
Steps:

1. Spectrogram
2. Peaks / differing peaks

Robustness:

- Noise, reverb, room acoustics, equalization
- Audio codec

Fingerprints (Shazam)



Steps:

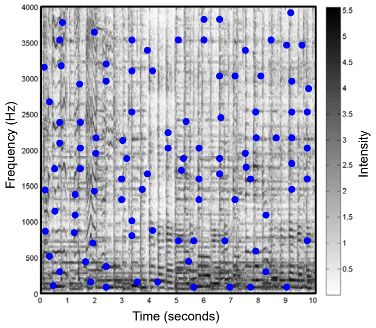
1. Spectrogram
2. Peaks / differing peaks

Robustness:

- Noise, reverb, room acoustics, equalization
- Audio codec
- Superposition of other audio sources

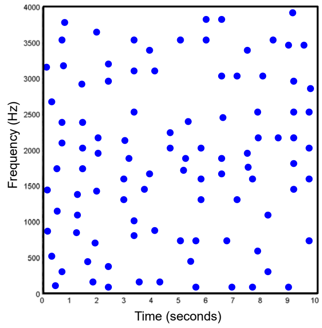
Matching Fingerprints (Shazam)

Database document



Matching Fingerprints (Shazam)

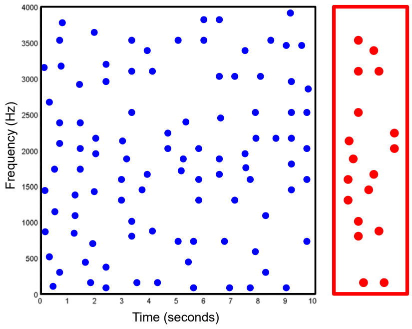
Database document
(constellation map)



Matching Fingerprints (Shazam)

Database document
(constellation map)

Query document
(constellation map)

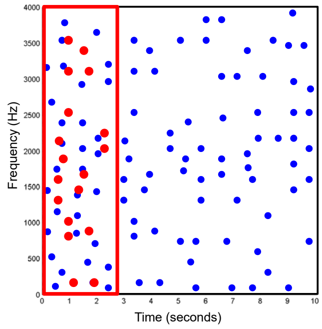


Matching Fingerprints (Shazam)

Database document
(constellation map)

Query document
(constellation map)

1. Shift query across database document
2. Count matching peaks

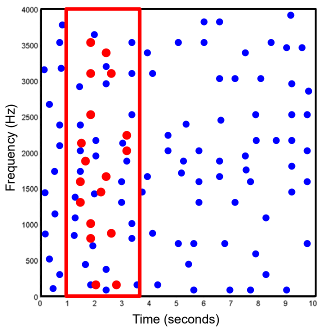


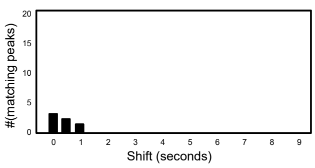
Matching Fingerprints (Shazam)

Database document
(constellation map)

Query document
(constellation map)

1. Shift query across database document
2. Count matching peaks



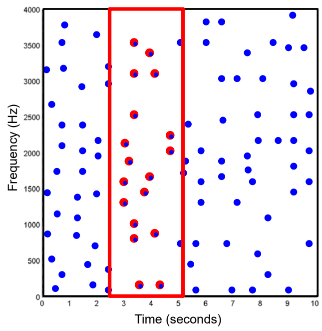


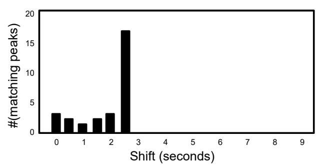
Matching Fingerprints (Shazam)

Database document
(constellation map)

Query document
(constellation map)

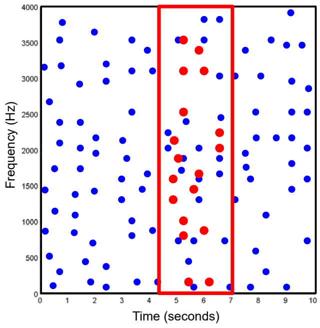
1. Shift query across database document
2. Count matching peaks





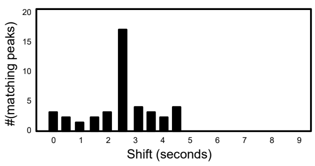
Matching Fingerprints (Shazam)

Database document
(constellation map)



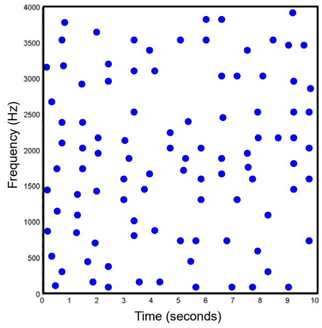
Query document
(constellation map)

1. Shift query across database document
2. Count matching peaks



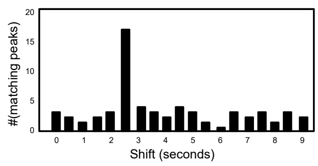
Matching Fingerprints (Shazam)

Database document
(constellation map)



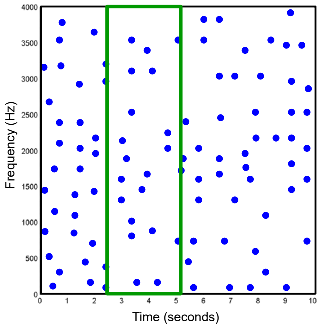
Query document
(constellation map)

1. Shift query across database document
2. Count matching peaks



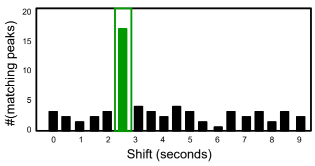
Matching Fingerprints (Shazam)

Database document
(constellation map)

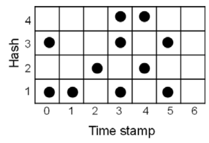


Query document
(constellation map)

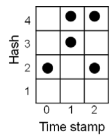
1. Shift query across database document
2. Count matching peaks
3. High count indicates a hit (document ID & position)



Indexing



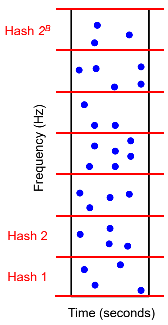
L(4) = (3,4)
L(3) = (0,3,5)
L(2) = (2,4)
L(1) = (0,1,3,5)



Query (*n*,*h*)	*L*(*h*) - *n*	Indicator functions									
		...	-1	0	1	2	3	4	5	6	...
(0,2)	(2,4)	0	0	0	0	1	0	1	0	0	0
(1,3)	(-1,2,4)	0	1	0	0	1	0	1	0	0	0
(1,4)	(2,3)	0	0	0	0	1	1	0	0	0	0
(2,2)	(0,2)	0	0	1	0	1	0	0	0	0	0
(2,4)	(1,2)	0	0	0	1	1	0	0	0	0	0
Matching function		**0**	**1**	**1**	**1**	**5**	**1**	**2**	**0**	**0**	**0**

Indexing (Shazam)

- Index the fingerprints using hash lists
- Hashes correspond to (quantized) frequencies



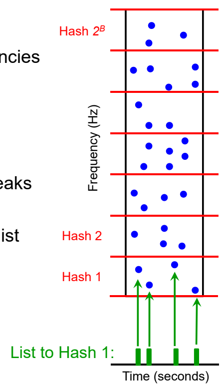
Indexing (Shazam)

- Index the fingerprints using hash lists
- Hashes correspond to (quantized) frequencies
- Hash list consists of time positions
  (and document IDs)

- N = number of spectral peaks
- B = #(bits) used to encode spectral peaks
- 2^B = number of hash lists
- $N / 2^B$ = average number of elements per list

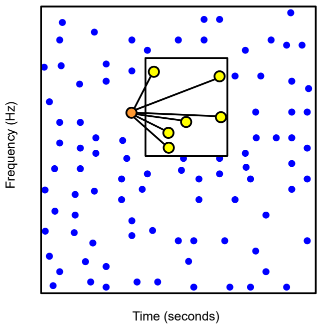
Problem:

- Individual peaks are not characteristic
- Hash lists may be very long
- Not suitable for indexing



Indexing (Shazam)

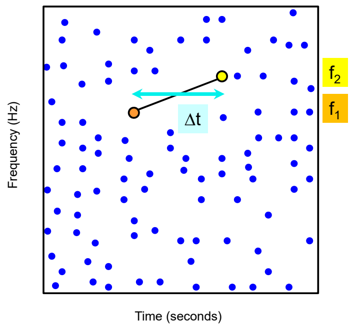
Idea: Use pairs of peaks to increase specificity of hashes



1. Peaks
2. Fix anchor point
3. Define target zone
4. Use paris of points
5. Use every point as anchor point

Indexing (Shazam)

Idea: Use pairs of peaks to increase specificity of hashes



1. Peaks
2. Fix anchor point
3. Define target zone
4. Use paris of points
5. Use every point as anchor point

New hash:

Consists of two frequency values and a time difference:

$(f_1 , f_2 , \Delta t)$

Indexing (Shazam)

- A hash is formed between an anchor point and each point in the target zone using two frequency values and a time difference.
- Fan-out (taking pairs of peaks) may cause a combinatorial explosion in the number of tokens. However, this can be controlled by the size of the target zone.
- Using more complex hashes increases specificity (leading to much smaller hash lists) and speed (making the retrieval much faster).

Indexing (Shazam)

Definitions:

- N = number of spectral peaks
- p = probability that a spectral peak can be found in (noisy and distorted) query
- F = fan-out of target zone, e. g. $F = 10$
- B = #(bits) used to encode spectral peaks and time difference

Consequences:

- $F \cdot N$ = #(tokens) to be indexed
- 2^{B+B} = increase of specifity (2^{B+B+B} instead of 2^B)
- p^2 = propability of a hash to survive
- $p \cdot (1-(1-p)^F)$ = probability that, at least, on hash survives per anchor point

Example: $F = 10$ and $B = 10$

- Memory requirements: $F \cdot N = 10 \cdot N$
- Speedup factor: $2^{B+B} / F^2 \sim 10^6 / 10^2 = 10000$
  (F times as many tokens in query and database, respectively)

Conclusions (Shazam)

Many parameters to choose:

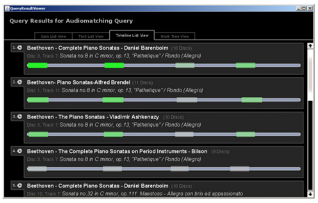
- Temporal and spectral resolution in spectrogram
- Peak picking strategy
- Target zone and fan-out parameter
- Hash function
- …

Conclusions (Audio Identification)

- Many more ways to define robust audio fingerprints
- Delicate trade-off between specificity, robustness, and efficiency
- Audio recording is identified (**not** a piece of music)
- Does not allow for identifying studio recording using a query taken from live recordings
- Does not generalize to identify different interpretations or versions of the same piece of music

Overview (Audio Retrieval)

- Audio identification (audio fingerprinting)
- Audio matching
- Cover song identification



Audio Matching

Database: Audio collection containing:

- Several recordings of the same piece of music
- Different interpretations by various musicians
- Arrangements in different instrumentations

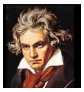
Goal: Given a short query audio fragment, find all corresponding audio fragments of similar musical content.

Notes:

- Instance of fragment-based retrieval
- Medium specificity
- A single document may contain several hits
- Cross-modal retrieval also feasible

Audio Matching

Beethoven's Fifth

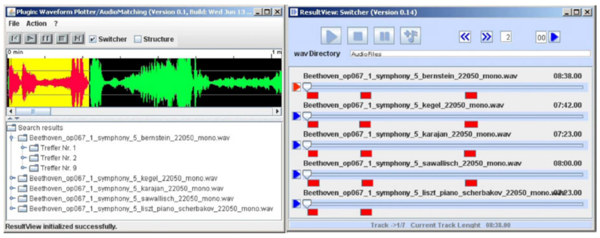


Various interpretations

Bernstein	
Karajan	
Scherbakov (piano)	
MIDI (piano)	

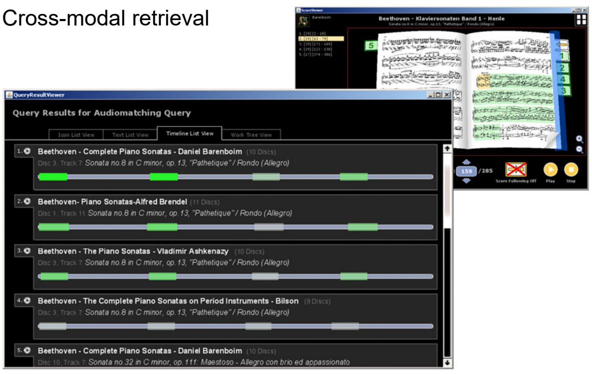
Application Scenario

Content-based retrieval



Application Scenario

Cross-modal retrieval



Audio Matching

Two main ingredients:

1.) Audio features

- Robust but discriminating
- Chroma-based features
- Correlate to harmonic progression
- Robust to variations in dynamics, timbre, articulation, local tempo

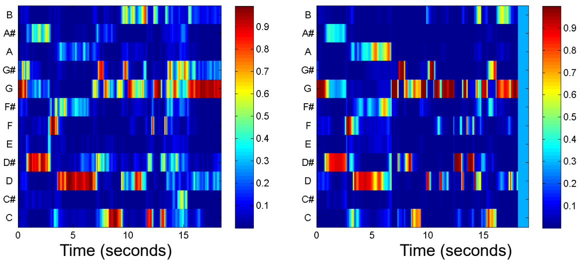
2.) Matching procedure

- Efficient
- Robust to local and global tempo variations
- Scalable using index structure

Audio Features

Example: Beethoven's Fifth
Chroma representation (normalized, 10 Hz)

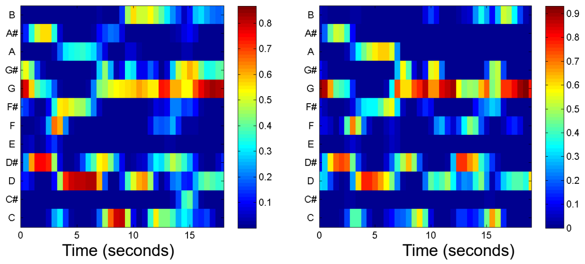
Karajan   Scherbakov



Audio Features

Example: Beethoven's Fifth
Chroma representation (normalized, 2 Hz)
Smoothing (2 seconds) + downsampling (factor 5)

Karajan   Scherbakov



Matching Procedure

Compute chroma feature sequences

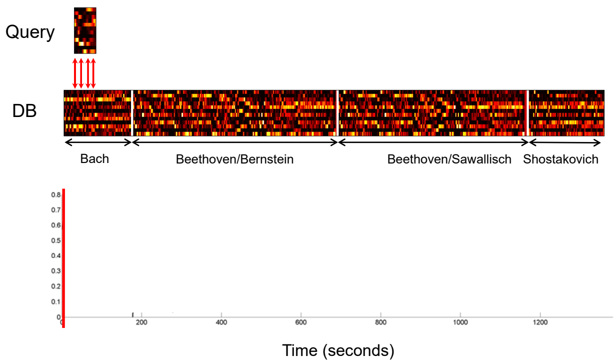
- Database $D \rightsquigarrow F[D] = (v^1, v^2, \ldots, v^N)$
- Query $Q \rightsquigarrow F[Q] = (w^1, w^2, \ldots, w^M)$
- N very large (database size), M small (query size)

$\cdots$	v^{i-1}	v^i	v^{i+1}	$\cdots$	v^{i+M-1}	v^{i+M}	$\cdots$
		w^1	w^2	$\cdots$	w^M		

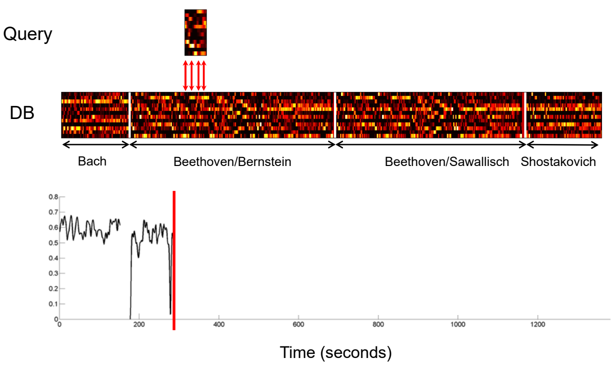
$\Delta(i) :=$ local distance$((v^i, v^{i+1} \ldots, v^{i+M-1}),(w^1, w^2, \ldots, w^M))$

$\rightsquigarrow$ Matching curve $\Delta : [1 : N] \to [0, 1]$

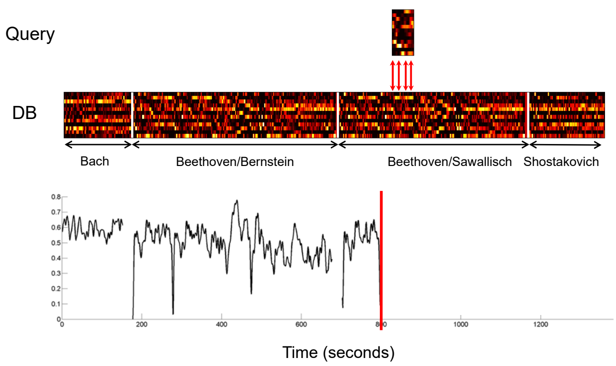
Matching Procedure



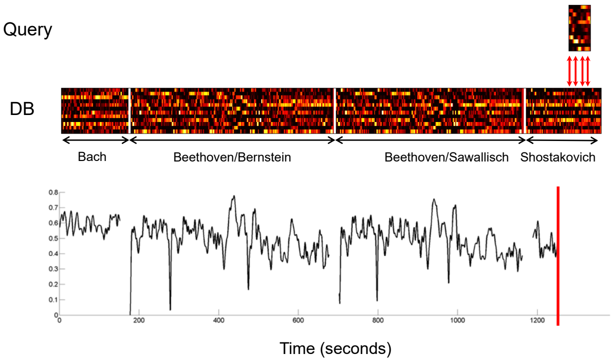
Matching Procedure



Matching Procedure



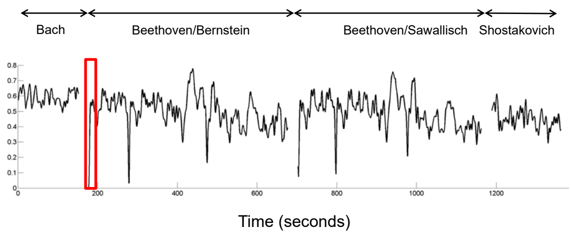
Matching Procedure



Matching Procedure

Matching curve

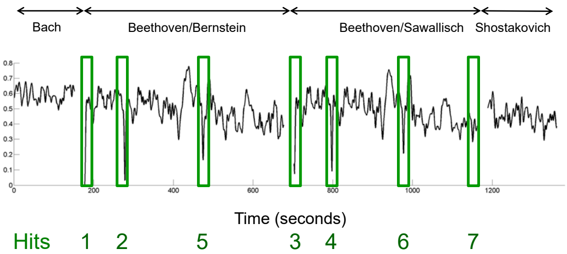
Query: Beethoven's Fifth / Bernstein (first 20 seconds)



Matching Procedure

Matching curve

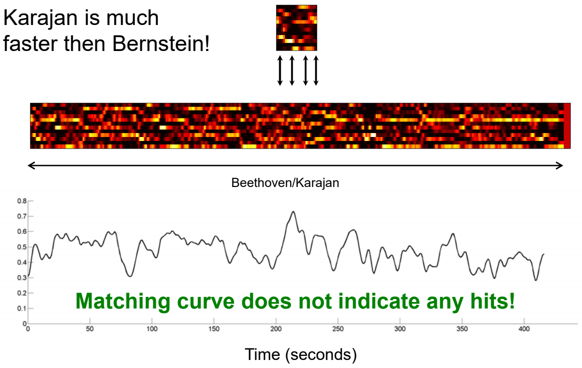
Query: Beethoven's Fifth / Bernstein (first 20 seconds)



Matching Procedure

Problem: How to deal with tempo differences?

Karajan is much
faster then Bernstein!

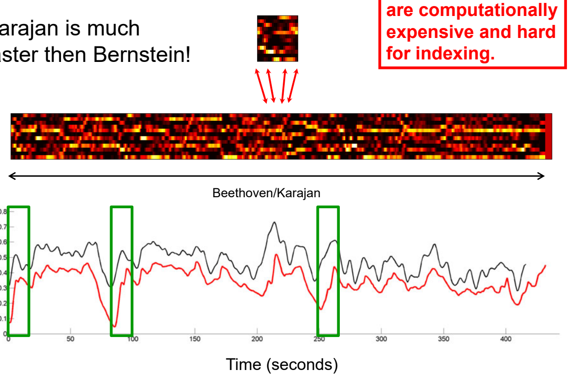


Matching Procedure

1. Strategy: Usage of local warping

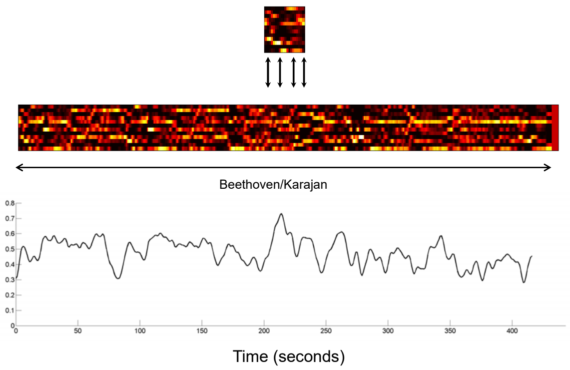
Karajan is much
faster then Bernstein!

Warping strategies are computationally expensive and hard for indexing.



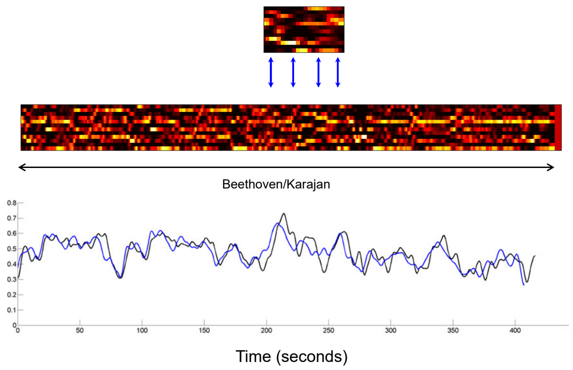
Matching Procedure

2. Strategy: Usage of multiple scaling



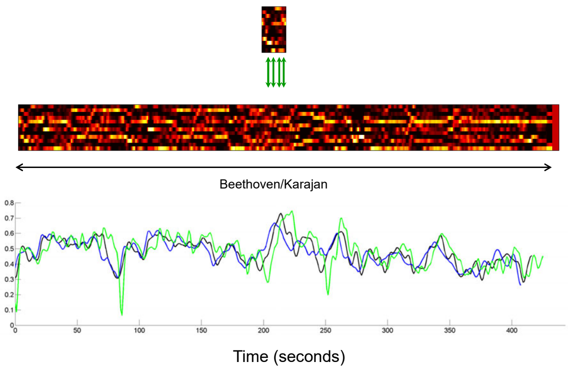
Matching Procedure

2. Strategy: Usage of multiple scaling



Matching Procedure

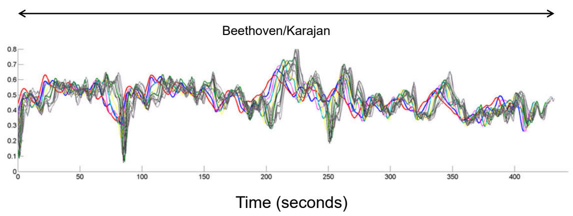
2. Strategy: Usage of multiple scaling



Matching Procedure

2. Strategy: Usage of multiple scaling

Query resampling simulates tempo changes

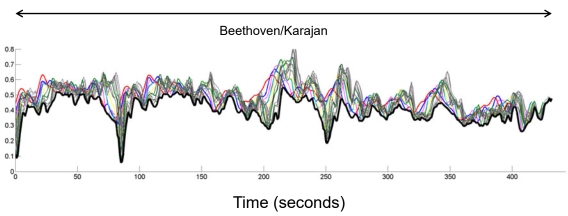


Matching Procedure

2. Strategy: Usage of multiple scaling

Query resampling simulates tempo changes

Minimize over all curves



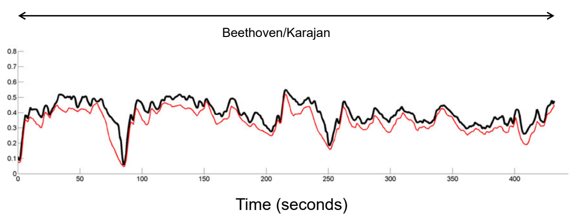
Matching Procedure

2. Strategy: Usage of multiple scaling

Query resampling simulates tempo changes

Minimize over all curves

Resulting curve is similar warping curve



Experiments

- Audio database ≈ 110 hours, 16.5 GB
- Preprocessing → chroma features, 40.3 MB
- Query clip ≈ 20 seconds
- Retrieval time ≈ 10 seconds (using MATLAB)

Experiments

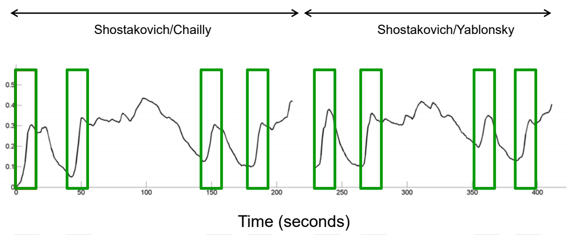
Query: Beethoven's Fifth / Bernstein (first 20 seconds)

Rank	Piece	Position
1	Beethoven's Fifth/Bernstein	0 - 21
2	Beethoven's Fifth/Bernstein	101- 122
3	Beethoven's Fifth/Karajan	86 - 103
⋮	⋮	⋮
⋮	⋮	⋮
10	Beethoven's Fifth/Karajan	252 - 271
11	Beethoven (Liszt) Fifth/Scherbakov	0 - 19
12	Beethoven's Fifth/Sawallisch	275 - 296
13	Beethoven (Liszt) Fifth/Scherbakov	86 - 103
14	Schumann Op. 97,1/Levine	28 - 43

Experiments

Query: Shostakovich, Waltz / Chailly (first 21 seconds)

Expected hits



Experiments

Query: Shostakovich, Waltz / Chailly (first 21 seconds)

Rank	Piece	Position
1	Shostakovich/Chailly	0 - 21
2	Shostakovich/Chailly	41- 60
3	Shostakovich/Chailly	180 - 198
4	Shostakovich/Yablonsky	1 - 19
5	Shostakovich/Yablonsky	36 - 52
6	Shostakovich/Yablonsky	156 - 174
7	Shostakovich/Chailly	144 - 162
8	Bach BWV 582/Chorzempa	358 - 373
9	Beethoven Op. 37,1/Toscanini	12 - 28
10	Beethoven Op. 37,1/Pollini	202 - 218

Conclusions (Audio Matching)

Audio Features

Strategy: Absorb variations already at feature level

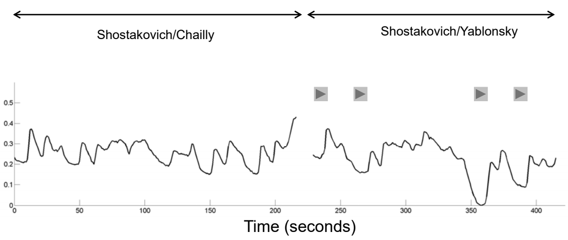
- Chroma → invariance to timbre
- Normalization → invariance to dynamics
- Smoothing → invariance to local time deviations

Message: There is no standard chroma feature!
Variants can make a huge difference!

Quality: Audio Matching

Query: Shostakovich, Waltz / Yablonsky (3. occurrence)

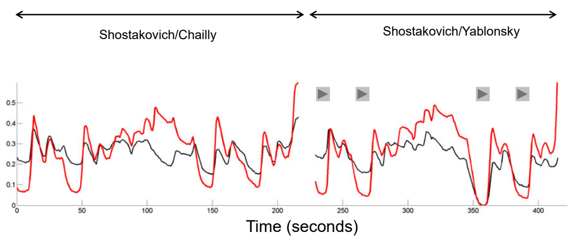
Standard Chroma (Chroma Pitch)



Quality: Audio Matching

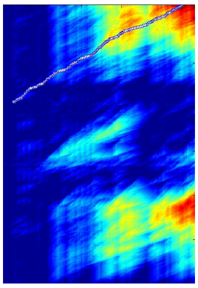
Query: Shostakovich, Waltz / Yablonsky (3. occurrence)

Standard Chroma (Chroma Pitch)
CRP(55)



Overview (Audio Retrieval)

- Audio identification
  (audio fingerprinting)
- Audio matching
- Cover song identification



Cover Song Identification

- Gómez/Herrera (ISMIR 2006)
- Casey/Slaney (ISMIR 2006)
- Serrà (ISMIR 2007)
- Ellis/Polioner (ICASSP 2007)
- Serrà/Gómez/Herrera/Serra (IEEE TASLP 2008)

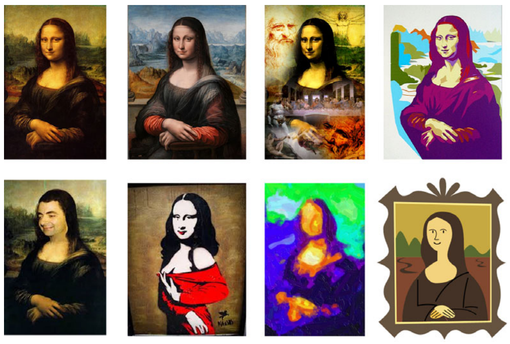
Cover Song Identification

Goal: Given a music recording of a song or piece of music, find all corresponding music recordings within a huge collection that can be regarded as a kind of version, interpretation, or cover song.

- Live versions
- Versions adapted to particular country/region/language
- Contemporary versions of an old song
- Radically different interpretations of a musical piece
- …

Instance of document-based retrieval!

Cover Song Identification



Cover Song Identification

Motivation

- Automated organization of music collections

  "Find me all covers of …"

- Musical rights management
- Learning about music itself

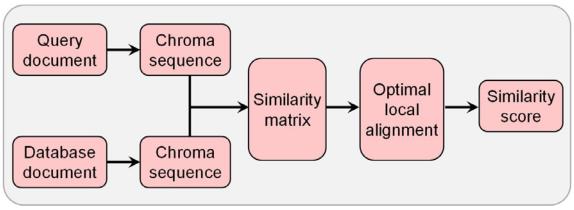
  "Understanding the essence of a song"

Cover Song Identification

Nearly anything can change! But something doesn't change.
Often this is chord progression and/or melody

Bob Dylan Knockin' on Heaven's Door	key	Avril Lavigne Knockin' on Heaven's Door
Metallica Enter Sandman	timbre	Apocalyptica Enter Sandman
Nirvana Poly [Incesticide Album]	tempo	Nirvana Poly [Unplugged]
Black Sabbath Paranoid	lyrics	Cindy & Bert Der Hund Der Baskerville
AC/DC High Voltage	recording conditions	AC/DC High Voltage [live]
	song structure	

Cover Song Identification



Local Alignment

Assumption:
Two songs are considered as similar if they contain possibly long subsegments that possess a similar harmonic progression

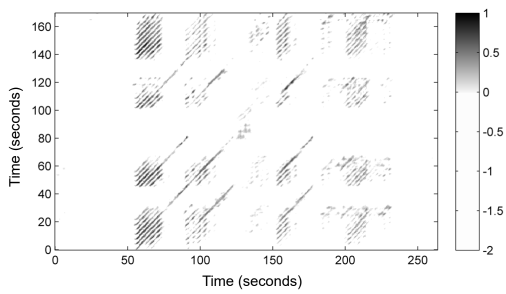
Task:
Let $X=(x_1,\ldots,x_N)$ and $Y=(y_1,\ldots,y_M)$ be the two chroma sequences of the two given songs, and let S be the resulting similarity matrix. Then find the maximum similarity of a subsequence of X and a subsequence of Y.

Local Alignment

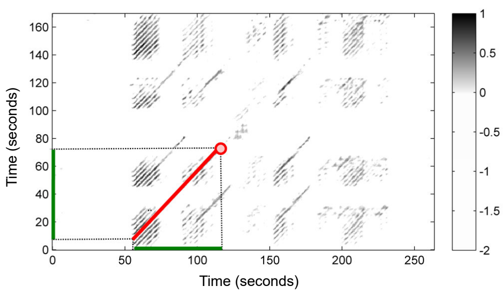
Note:
This problem is also known from bioinformatics.
The Smith-Waterman algorithm is a well-known algorithm for performing local sequence alignment; that is, for determining similar regions between two nucleotide or protein sequences.

Strategy:
We use a variant of the Smith-Waterman algorithm.

Local Alignment



Local Alignment



Cover Song Identification

Query: Bob Dylan – Knockin' on Heaven's Door
Retrieval result:

Rank	Recording	Score
1.	Guns and Roses: Knockin' On Heaven's Door	94.2
2.	Avril Lavigne: Knockin' On Heaven's Door	86.6
3.	Wyclef Jean: Knockin' On Heaven's Door	83.8
4.	Bob Dylan: Not For You	65.4
5.	Guns and Roses: Patience	61.8
6.	Bob Dylan: Like A Rolling Stone	57.2
7.-14.	…	

Cover Song Identification

Query: AC/DC – Highway To Hell

Retrieval result:

Rank	Recording	Score
1.	AC/DC: Hard As a Rock	79.2
2.	Hayseed Dixie: Dirty Deeds Done Dirt Cheap	72.9
3.	AC/DC: Let There Be Rock	69.6
4.	AC/DC: TNT (Live)	65.0
5.-11.	…	
12.	Hayseed Dixie: Highway To Hell	30.4
13.	AC/DC: Highway To Hell Live (live)	21.0
14.	…	

Conclusions (Cover Song Identification)

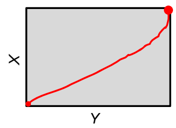
- Harmony-based approach
- Measure is suitable for document retrieval, but seems to be too coarse for audio matching applications
- Every song has to be compared with any other
  → method does not scale to large data collection
- What are suitable indexing methods?

Conclusions (Audio Retrieval)

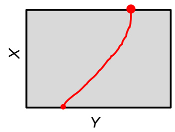
Retrieval task	Audio identification	Audio matching	Version identification
Identification	Specific audio recording	Different interpretations	Different versions
Query	Short fragment (5–10 seconds)	Audio clip (10–40 seconds)	Entire recording
Retrieval level	Fragment	Fragment	Document
Specificity	High	Medium	Medium / low
Features	Spectral peaks (abstract)	Chroma (harmony)	Chroma (harmony)

Conclusions (Alignment Strategies)

- Classical DTW
  Global correspondence between X and Y



- Subsequence DTW
  Subsequence of Y corresponds to X



- Local Alignment
  Subsequence of Y corresponds to subequence of X