Advanced Course Computer Science

Music Processing

Summer Term 2010

Meinard Müller

Saarland University and MPI Informatik
meinard@mpi-inf.mpg.de

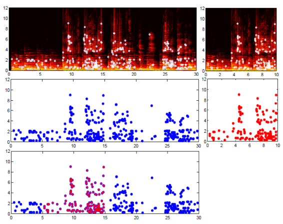
Audio Retrieval

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

- Allamanche et al. (AES 2001)
- Cano et al. (*IEEE MMSP* 2002)
- Kurth/Clausen/Ribbrock (AES 2002)
- Wang (ISMIR 2003)
- Shrestha/Kalker (ISMIR 2004)

Audio Identification

Shazam application scenario

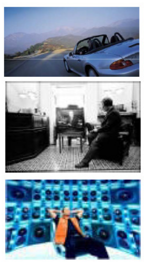
- User hears music playing in the environment
- User records music fragment (5-15 seconds) with mobile phone
- Audio fingerprints are extracted from recording and sent to a service
- Server identifies audio recording based on fingerprints
- Server sends back metadata (track title, artist) to user

[Wang, ISMIR 2003]

Audio Identification

Shazam application scenario

"THE MOMENT"



Radio - Car, Home, Work

TV and Cinema

Clubs and Bars

Cafes, Shops, Restaurants

[Wang, ISMIR 2003]

Audio Identification

Shazam application scenario: Target audience

Core target:
Music mobile
- 18-25 years old
- Struggle to keep up with *LATEST RELEASES*
- Enjoy new technologies

Music 'Experts'

Early Youth
- 14-17 years old
- Identify next purchase quickly
- Enjoy practical services

Music Community

More Mature
- 26-40 years old
- Identify classic hits as well as new music
- Need advice on what to buy

Music Confidence

[Wang, ISMIR 2003]

Audio Identification

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

Audio Identification

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, high entropy)*
- *Low probability of false positives*
- *Recorded query excerpt (only a few seconds)*
- *Large audio collection on the server side (millions of songs)*

Audio Identification

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 Identification

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 Identification

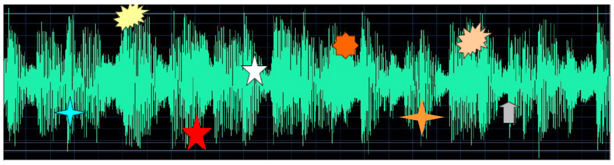
An audio fingerprint is a content-based compact signature that summarizes a piece of audio content

Requirements:

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

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

Matching Fingerprints (Shazam)



- For each database document (audio file), generate reproducible landmarks
- Each landmark occurs at a time position
- For each landmark, generate a “fingerprint” that characterizes its location
- Do same for query fragment

[Wang, ISMIR 2003]

Matching Fingerprints (Shazam)

- Generate list of matching fingerprints (matches between query and database document)
- Each match is represented by a pair $(t_{database}, t_{query})$ of time positions
- Matching segment is characterized by set M of pairs each having the same time difference

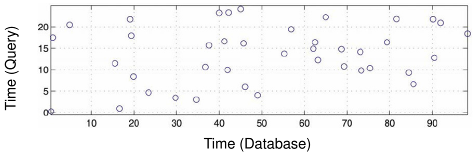
$$t_{database} - t_{query} = constant \quad \text{for} \quad (t_{database}, t_{query}) \in M$$

- Set of false positives have random time differences
- Filter out cruft by doing a histogram on time differences
- Score is size of largest histogram peak

[Wang, ISMIR 2003]

Matching Fingerprints (Shazam)

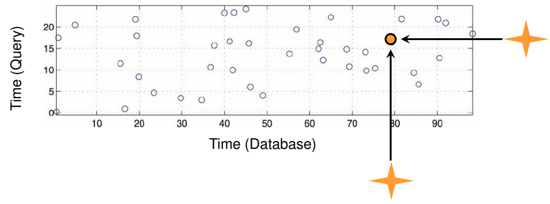
Scatter plot of matching hash locations



[Wang, ISMIR 2003]

Matching Fingerprints (Shazam)

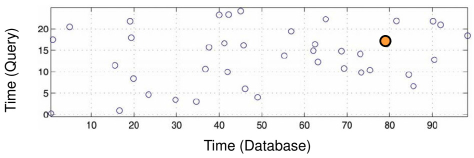
Scatter plot of matching hash locations



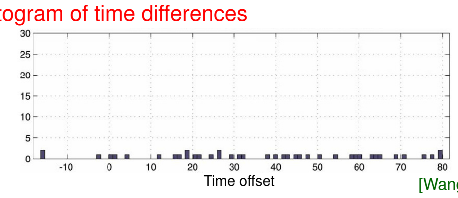
[Wang, ISMIR 2003]

Matching Fingerprints (Shazam)

Scatter plot of matching hash locations



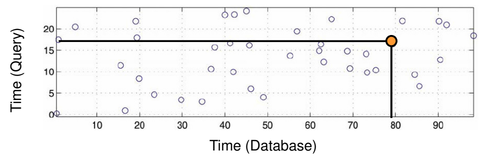
Histogram of time differences



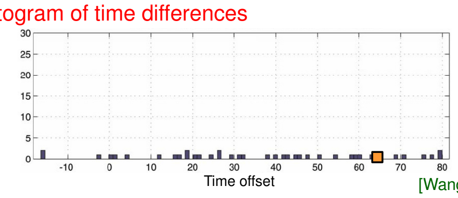
[Wang, ISMIR 2003]

Matching Fingerprints (Shazam)

Scatter plot of matching hash locations



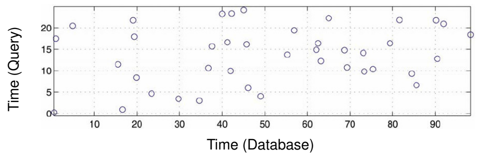
Histogram of time differences



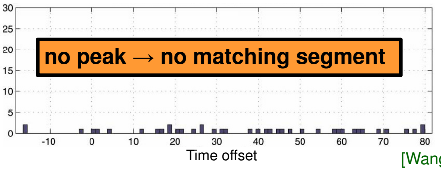
[Wang, ISMIR 2003]

Matching Fingerprints (Shazam)

Scatter plot of matching hash locations



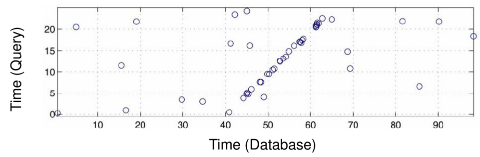
Histogram of time differences



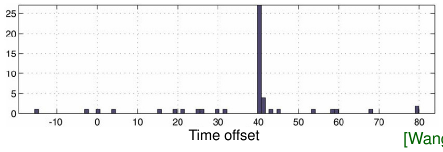
[Wang, ISMIR 2003]

Matching Fingerprints (Shazam)

Scatter plot of matching hash locations



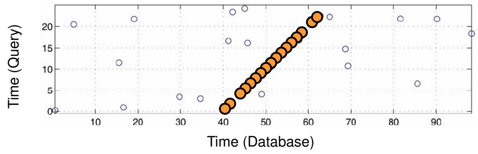
Histogram of time differences



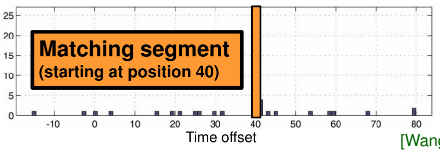
[Wang, ISMIR 2003]

Matching Fingerprints (Shazam)

Scatter plot of matching hash locations

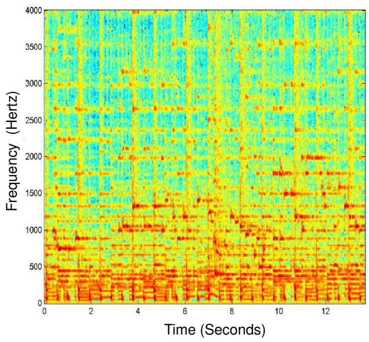


Histogram of time differences



[Wang, ISMIR 2003]

Fingerprints (Shazam)



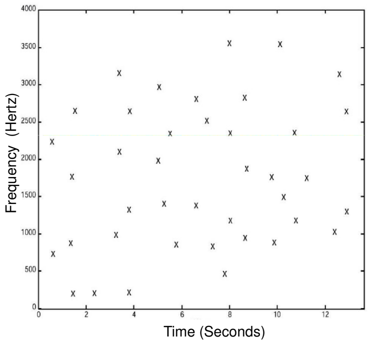
Steps:

1. Spectrogram

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

[Wang, ISMIR 2003]

Fingerprints (Shazam)



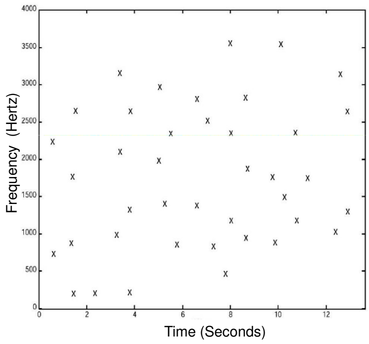
Steps:

1. Spectrogram
2. Peaks

- *``Constellation map´´*
- *Robust to noise, reverb, room acoustics*
- *Tend to survive through voice codec*

[Wang, ISMIR 2003]

Fingerprints (Shazam)



Steps:

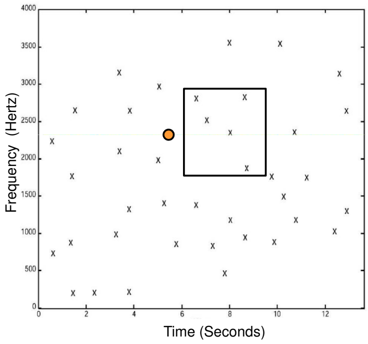
1. Spectrogram
2. Peaks

Problem:

- Individual peaks have low entropy
- Not suitable for indexing

[Wang, ISMIR 2003]

Fingerprints (Shazam)



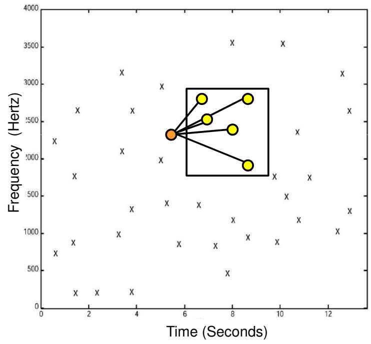
Steps:

1. Spectrogram
2. Peaks
3. Target zone
4. Pairs of peaks

- *Fix anchor point*
- *Define target zone*
- *Use pairs of points*
- *Use every point as anchor point*

[Wang, ISMIR 2003]

Fingerprints (Shazam)



Steps:

1. Spectrogram
2. Peaks
3. Target zone
4. Pairs of peaks

- *Fix anchor point*
- *Define target zone*
- *Use pairs of points*
- *Use every point as anchor point*

[Wang, ISMIR 2003]

Indexing (Shazam)

- Hash is formed between anchor point and each point in target zone using frequency values and 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 traget zone.
- Using more complex hashes increases specificity (leading to much smaller hash bucktes) and speed (making the retrieval much faster).

[Wang, ISMIR 2003]

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 of at least on hash survives per anchor point

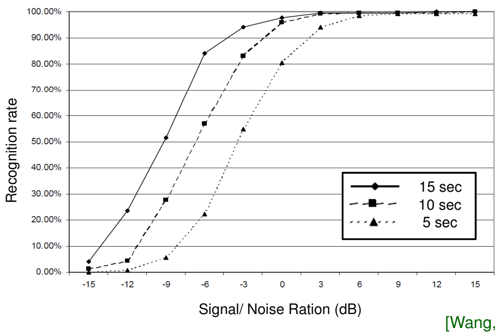
Example: $F = 10$ and $B = 10$

- 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)

[Wang, ISMIR 2003]

Results (Shazam)

Test dataset of 10000 tracks
Search time: 5 to 500 milliseconds



[Wang, ISMIR 2003]

Conclusions (Shazam)

Many parameters to choose:

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

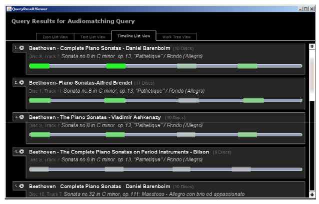
[Wang, ISMIR 2003]

Conclusions (Audio Identification)

- Identifies audio recording (**not** piece of music)
- Highly robust to noise, artifacts, deformations
- May even work to handle superimposed recordings
- Does not allow to identify studio recordings by query taken from live recordings
- Does not generalize to identify different interpretations of the same piece of music

Overview (Audio Retrieval)

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



Audio Matching

- Pickens et al. (ISMIR 2002)
- Müller/Kurth/Clausen (ISMIR 2005)
- Suyoto et al. (IEEE TASLP 2008)
- Kurth/Müller (IEEE TASLP 2008)

Audio Matching

Various interpretations – Beethoven's Fifth

Bernstein	▶
Karajan	▶
Scherbakov (piano)	▶
MIDI (piano)	▶

Audio Matching

Given: Large music database containing several

- recordings of the same piece of music
- interpretations by various musicians
- arrangements in different instrumentations

Goal: Given a short query audio clip, identify all corresponding audio clips of similar musical content

- irrespective of the specific interpretation and instrumentation
- automatically and efficiently

Query-by-Example paradigm

[Müller et al., ISMIR 2005]

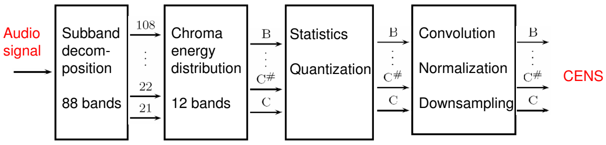
Audio Matching

General strategy

- Normalized and smoothed chroma features
  - correlate to harmonic progression
  - robust to variations in dynamics, timbre, articulation, local tempo
- Robust matching procedure
  - efficient
  - robust to global tempo variations
  - scalable using index structure

[Müller et al., ISMIR 2005]

Feature Design



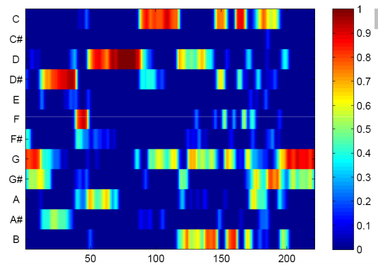
Two stages:

Stage 1: Local chroma energy distribution features
Stage 2: Normalized short-time statistics

⇝ CENS = Chroma Energy Normalized Statistics

Feature Design

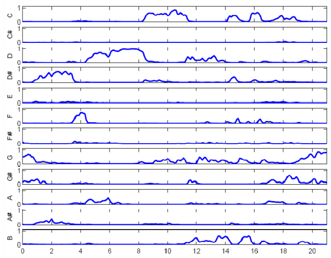
Beethoven's Fifth: Bernstein



Resolution: 10 features/second
Feature window size: 200 milliseconds

Feature Design

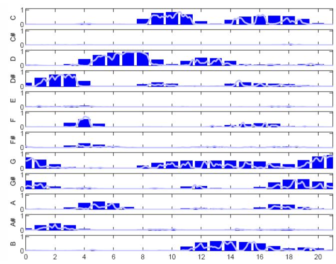
Beethoven's Fifth: Bernstein



Resolution: 10 features/second
Feature window size: 200 milliseconds

Feature Design

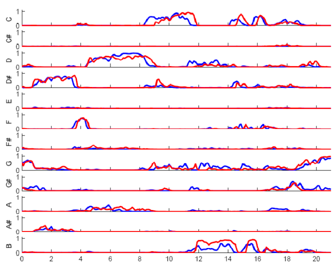
Beethoven's Fifth: Bernstein



Resolution: 1 features/second
Feature window size: 4000 milliseconds

Feature Design

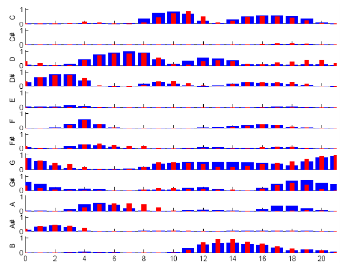
Beethoven's Fifth: Bernstein vs. Sawallisch



Resolution: 10 features/second
Feature window size: 200 milliseconds

Feature Design

Beethoven's Fifth: Bernstein vs. Sawallisch

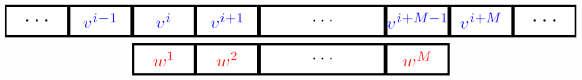


Resolution: 1 features/second
Feature window size: 4000 milliseconds

Matching Procedure

Compute CENS feature sequences

- Database $D \rightsquigarrow F[D] = (v^1, v^2, \ldots, v^N)$
- Query $Q \rightsquigarrow F[Q] = (w^1, w^2, \ldots, w^M)$
- $N \approx 500000,\ M \approx 20$

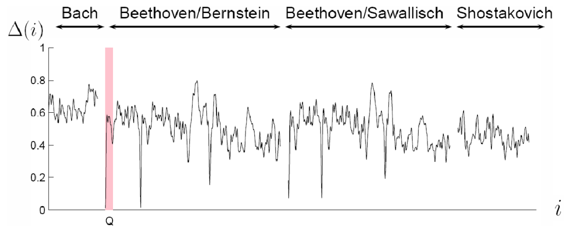


$\Delta(i) :=$ local distance($(v^i, v^{i+1} \ldots, v^{i+M-1})$,$(w^1, w^2, \ldots, w^M)$)

$\rightsquigarrow$ Global distance function $\Delta : [1:N] \to [0,1]$

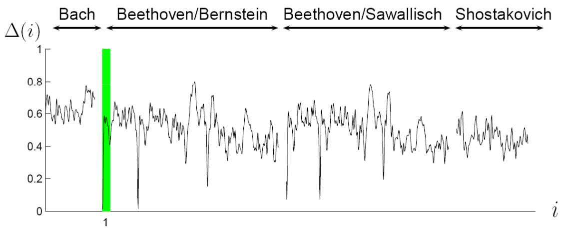
Matching Procedure

Query: Beethoven's Fifth / Bernstein, first 20 seconds



Matching Procedure

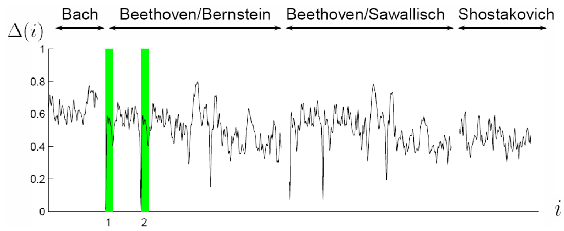
Query: Beethoven's Fifth / Bernstein, first 20 seconds



Best audio matches: 1

Matching Procedure

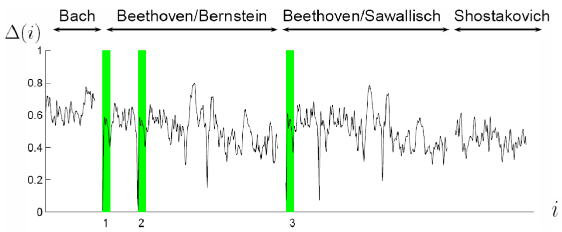
Query: Beethoven's Fifth / Bernstein, first 20 seconds



Best audio matches: 2

Matching Procedure

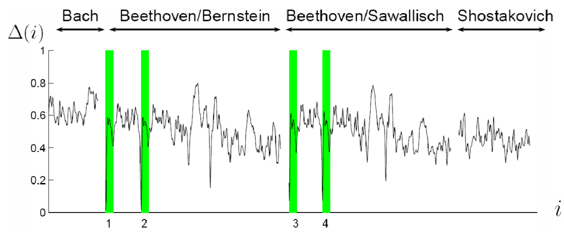
Query: Beethoven's Fifth / Bernstein, first 20 seconds



Best audio matches: 3

Matching Procedure

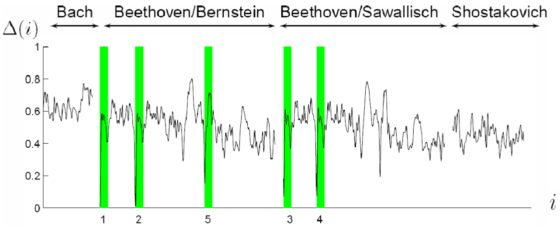
Query: Beethoven's Fifth / Bernstein, first 20 seconds



Best audio matches: 4

Matching Procedure

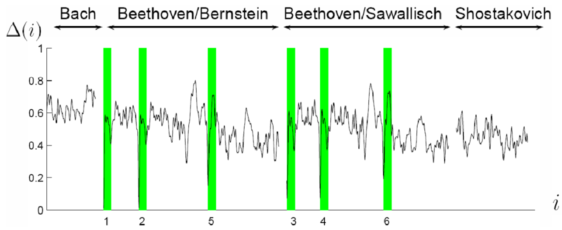
Query: Beethoven‘s Fifth / Bernstein, first 20 seconds



Best audio matches: 5

Matching Procedure

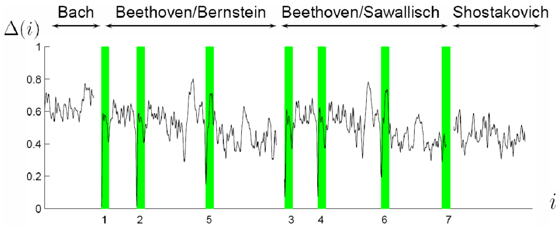
Query: Beethoven‘s Fifth / Bernstein, first 20 seconds



Best audio matches: 6

Matching Procedure

Query: Beethoven‘s Fifth / Bernstein, first 20 seconds



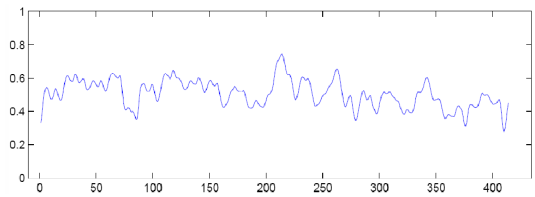
Best audio matches: 7

Global Tempo Variations

Query: Beethoven‘s Fifth / Bernstein, first 20 seconds

Problem: Karajan is much faster $\rightsquigarrow$ useless Δ

Solution?

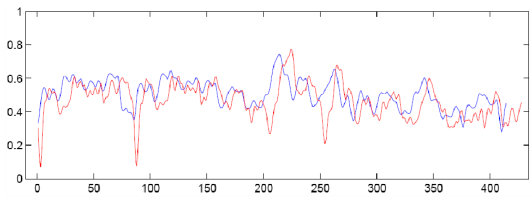


Global Tempo Variations

Query: Beethoven‘s Fifth / Bernstein, first 20 seconds

Problem: Karajan is much faster $\rightsquigarrow$ useless Δ

Solution: Make Bernstein query faster and comute new Δ

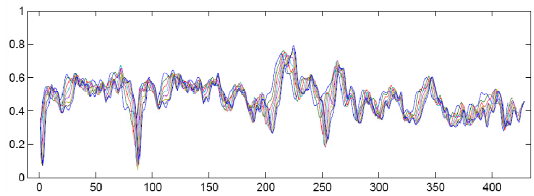


Global Tempo Variations

Query: Beethoven‘s Fifth / Bernstein, first 20 seconds

Problem: Karajan is much faster $\rightsquigarrow$ useless Δ

Solution: Compute Δ for various tempi

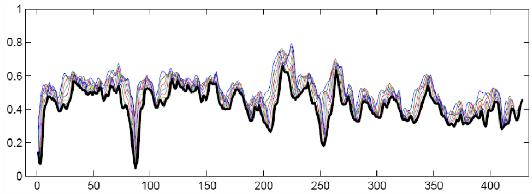


Global Tempo Variations

Query: Beethoven‘s Fifth / Bernstein, first 20 seconds

Problem: Karajan is much faster $\rightsquigarrow$ useless Δ

Solution: Minimize over all resulting Δ's $\rightsquigarrow$ $\Delta^{\min}$

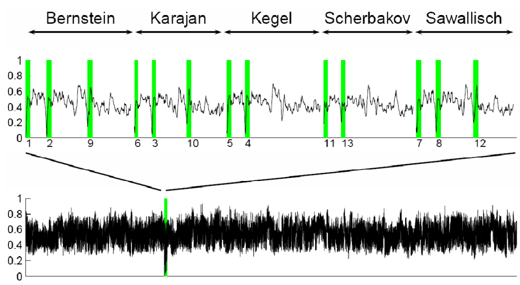


Experiments

- Audio database > 110 hours, 16.5 GB
- Preprocessing $\rightsquigarrow$ CENS features, 40.3 MB
- Query clip $\approx$ 20 seconds
- Query response time < 10 seconds

Experiments

Query: Beethoven‘s Fifth / Bernstein, first 20 seconds



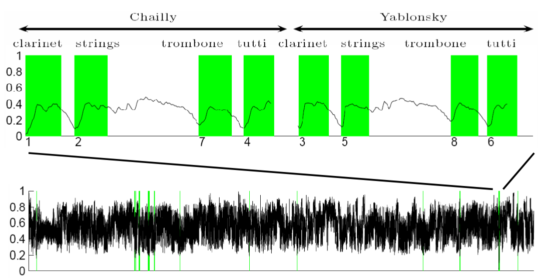
Experiments

Query: Beethoven‘s Fifth / Bernstein, first 20 seconds

Rank	$\Delta^{\min}$	Piece	Position
1	0.0114	Beethoven's Fifth/Bernstein	0 - 21
2	0.0150	Beethoven's Fifth/Bernstein	101 - 122
3	0.0438	Beethoven's Fifth/Karajan	86 - 103
⋮	⋮	⋮	⋮
10	0.1796	Beethoven's Fifth/Karajan	252 - 271
11	0.1827	Beethoven (Liszt) Fifth/Scherbakov	0 - 19
12	0.1945	Beethoven's Fifth/Sawallisch	275 - 296
13	0.1970	Beethoven's Fifth (Liszt)/Scherbakov	86 - 103
14	0.2169	Schumann op 97,1/Levine	28 - 43
⋮	⋮	⋮	⋮

Experiments

Query: Shostakovich, Waltz/Chailly, first 27 seconds



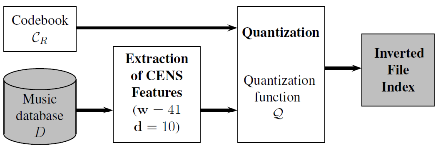
Experiments

Query: Shostakovich, Waltz/Chailly, first 21 seconds

Rank	$\Delta^{\min}$	Piece	Position
1	0.0172	Shostakovich/Chailly	0 - 21
2	0.0505	Shostakovich/Chailly	41 - 60
3	0.0983	Shostakovich/Chailly	180 - 198
4	0.1044	Shostakovich/Yablonsky	1 - 19
5	0.1090	Shostakovich/Yablonsky	36 - 52
6	0.1401	Shostakovich/Yablonsky	156 - 174
7	0.1476	Shostakovich/Chailly	144 - 162
8	0.1626	Bach BWV 582/Chorzempa	358 - 373
9	0.1668	Beethoven op 37,1/Toscanini	12 - 28
10	0.1729	Beethoven op 37,1/Pollini	202 - 218
⋮	⋮	⋮	⋮

Index-based Matching

Indexing stage



- Convert database into feature sequence (chroma/CENS)
- Quantize features with respect to a fixed codebook
- Create an inverted file index
  - contains for each codebook vector an inverted list
  - each list contains feature indices in ascending order

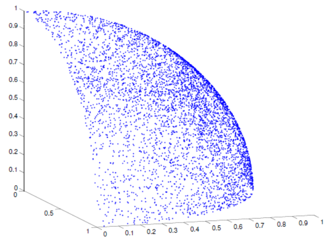
[Kurth/Müller, IEEE-TASLP 2008]

Index-based Matching

Quantization

- Feature space $\mathcal{F} := \{v \in [0,1]^{12} \mid \|v\|_2 = 1\}$

Visualization (3D)



Index-based Matching

Quantization

- Feature space $\mathcal{F} := \{v \in [0,1]^{12} \mid \|v\|_2 = 1\}$
- Codebook selection of suitable size *R* $\{c_1, \ldots, c_R\} \subset \mathcal{F}$
- Quantization using nearest neighbors

$$\mathcal{Q}[v] := \text{argmin}_{r\in[1:R]} \arccos(\langle v, c_r \rangle)$$

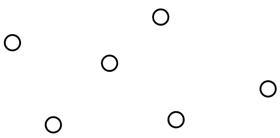
Index-based Matching

How to derive a good codebook?

- Codebook selection by unsupervised learning
  - Linde–Buzo–Gray (LBG) algorithm
  - similar to k-means
  - adjust algorithm to spheres
- Codebook selection based on musical knowledge

Index-based Matching

LBG algorithm

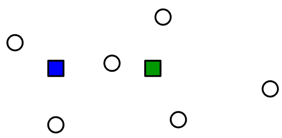


Steps:

1. Initialization of codebook vectors
2. Assignment
3. Recalculation
4. Iteration (back to 2.)

Index-based Matching

LBG algorithm

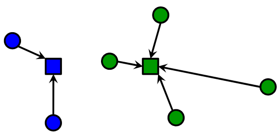


Steps:

1. Initialization of codebook vectors
2. Assignment
3. Recalculation
4. Iteration (back to 2.)

Index-based Matching

LBG algorithm

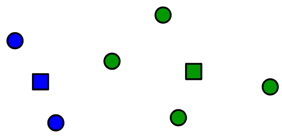


Steps:

1. Initialization of codebook vectors
2. Assignment
3. Recalculation
4. Iteration (back to 2.)

Index-based Matching

LBG algorithm

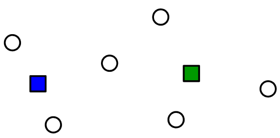


Steps:

1. Initialization of codebook vectors
2. Assignment
3. Recalculation
4. Iteration (back to 2.)

Index-based Matching

LBG algorithm

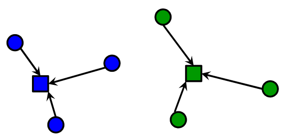


Steps:

1. Initialization of codebook vectors
2. Assignment
3. Recalculation
4. Iteration (back to 2.)

Index-based Matching

LBG algorithm

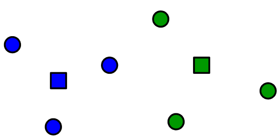


Steps:

1. Initialization of codebook vectors
2. Assignment
3. Recalculation
4. Iteration (back to 2.)

Index-based Matching

LBG algorithm

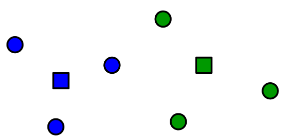


Steps:

1. Initialization of codebook vectors
2. Assignment
3. Recalculation
4. Iteration (back to 2.)

Index-based Matching

LBG algorithm



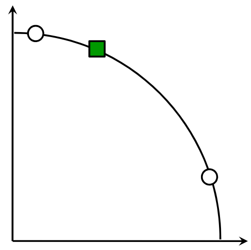
Steps:

1. Initialization of codebook vectors
2. Assignment
3. Recalculation
4. Iteration (back to 2.)

Until convergence

Index-based Matching

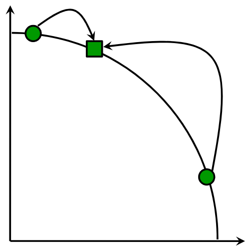
LBG algorithm for spheres



- Example: 2D
- Assignment
- Recalculation
- Projection

Index-based Matching

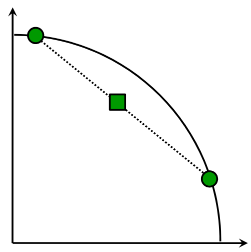
LBG algorithm for spheres



- Example: 2D
- Assignment
- Recalculation
- Projection

Index-based Matching

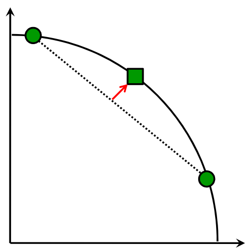
LBG algorithm for spheres



- Example: 2D
- Assignment
- Recalculation
- Projection

Index-based Matching

LBG algorithm for spheres



- Example: 2D
- Assignment
- Recalculation
- Projection

Index-based Matching

Codebook using musical knowledge

- Observation: Chroma features capture harmonic information
- Example: C-Major $\frac{1}{\sqrt{3}}(1,0,0,0,1,0,0,1,0,0,0,0)$
- Example: C#-Major $\frac{1}{\sqrt{3}}(0,1,0,0,0,1,0,0,1,0,0,0)$
- Experiments: For more then 95% of all chroma features >50% of energy lies in at most 4 components

Index-based Matching

Codebook using musical knowledge

- C-Major $\frac{1}{\sqrt{3}}(1,0,0,0,1,0,0,1,0,0,0,0) = \frac{1}{\sqrt{3}}(\delta_1+\delta_5+\delta_8)$
- C#-Major $\frac{1}{\sqrt{3}}(0,1,0,0,0,1,0,0,1,0,0,0) = \frac{1}{\sqrt{3}}(\delta_2+\delta_6+\delta_9)$
- Choose codebook to contain *n*-chords for *n=1,2,3,4*

n	*1*	*2*	*3*	*4*	
template	δj	$\frac{1}{\sqrt{2}}(\delta_{k_1}+\delta_{k_2})$	$\frac{1}{\sqrt{3}}(\delta_{r_1}+\delta_{r_2}+\delta_{r_3})$	$\frac{1}{\sqrt{4}}(\delta_{n_1}+\delta_{n_2}+\delta_{n_3}+\delta_{n_4})$	
#	12	66	220	495	793

Index-based Matching

Codebook using musical knowledge

Additional consideration of harmonics in chord templates

Example: 1-chord C

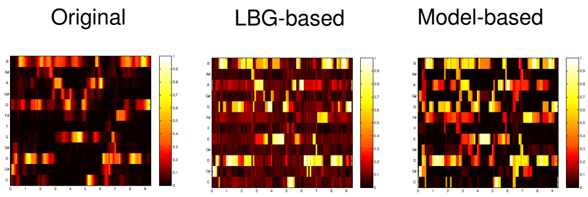
Harmonics	1	2	3	4	5	6
Pitch	C3	C4	G4	C5	E5	G5
Frequency	131	262	392	523	654	785
Chroma	C	C	G	C	E	C

Replace δ_1 by $w_1\delta_1 + w_2\delta_1 + w_3\delta_8 + w_4\delta_1 + w_5\delta_5 + w_6\delta_8$
with suitable weights for the harmonics

Index-based Matching

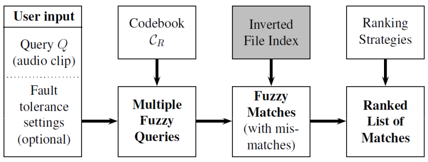
Quantization

Orignal chromagram and projections on codebooks



Index-based Matching

Query and retrieval stage



- Query consists of a short audio clip (10-40 seconds)
- Specification of fault tolerance setting
  - fuzzyness of query
  - number of admissable mismatches
  - tolerance to tempo variations
  - tolerance to modulations

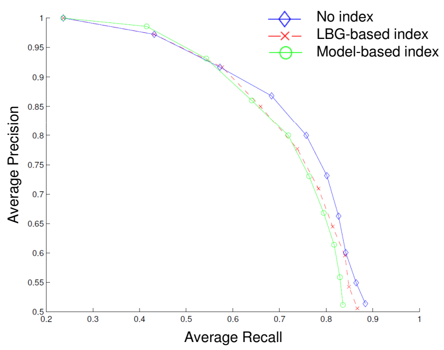
Index-based Matching

Retrieval results

- Medium sized database
  - 500 pieces
  - 112 hours of audio
  - mostly classical music
- Selection of various queries
  - 36 queries
  - duration between 10 and 40 seconds
  - hand-labelled matches in database
- Indexing leads to speed-up factor between 15 and 20 (depending on query length)
- Only small degradation in precision and recall

Index-based Matching

Retrieval results



Conclusions (Index-based Matching)

- Described method suitable for medium-sized databases
  - index is assumed to be in main memory
  - inverted lists may be long
- Goal was to find all meaningful matches
  - high-degree of fault-tolerance required (fuzzyness, mismatches)
  - number of intersections and unions may explode
- What to do when dealing with millions of songs?
- Can the quantization be avoided?
- Better indexing and retrieval methods needed!
  - kd-trees
  - locality sensitive hashing
  - …

Conclusions (Audio Matching)

Strategy: Absorb variations at feature level

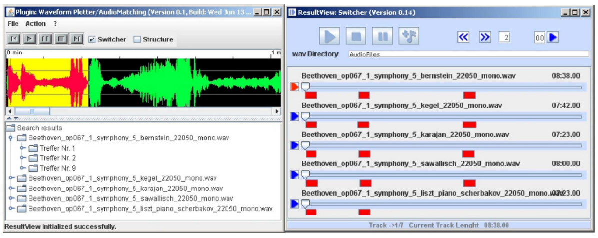
- Chroma $\rightsquigarrow$ invariance to timbre
- Normalization $\rightsquigarrow$ invariance to dynamics
- Smoothing $\rightsquigarrow$ invariance to local time deviations

Conclusions (Audio Matching)

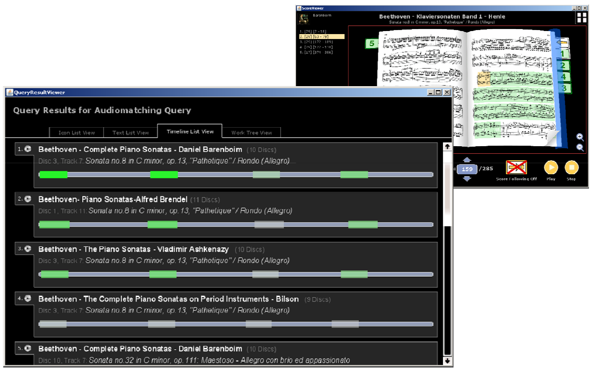
Global matching procedure

- Strategy: Exact matching and multiple scaled queries
  - simulate tempo variations by feature resampling
  - different queries correspond to different tempi
  - indexing possible
- Strategy: Dynamic time warping
  - subsequence variant
  - more flexible (in particular for longer queries)
  - indexing hard

Application: Audio Matching

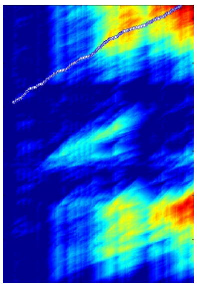


Application: Audio Matching



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

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	

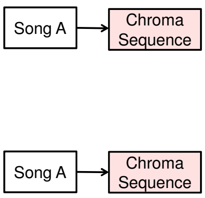
Cover Song Identification

How to compare two different songs?

[Serrà et al., IEEE-TASLP 2009]

Cover Song Identification

How to compare two different songs?

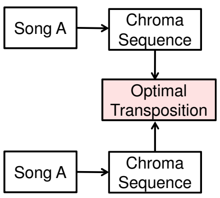


- Feature computation

[Serrà et al., IEEE-TASLP 2009]

Cover Song Identification

How to compare two different songs?

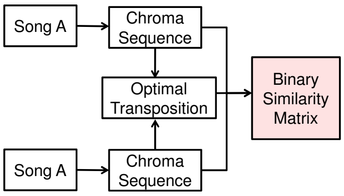


- Feature computation
- Dealing with different keys

[Serrà et al., IEEE-TASLP 2009]

Cover Song Identification

How to compare two different songs?

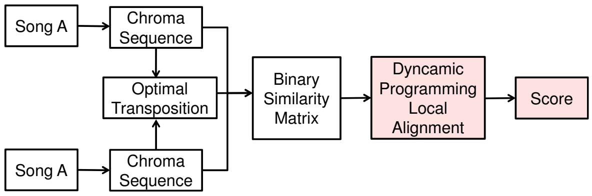


- Feature computation
- Dealing with different keys
- Local similarity measure

[Serrà et al., IEEE-TASLP 2009]

Cover Song Identification

How to compare two different songs?

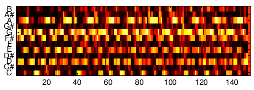


- Feature computation
- Dealing with different keys
- Local similarity measure
- Global similarity measure

[Serrà et al., IEEE-TASLP 2009]

Cover Song Identification

Feature computation



- Chroma features
  - correlates to harmonic progression
  - robust to changes in timbre and instrumentation
  - normalization introduces invariance to dynamics
- Enhancement strategies
  - model for considering harmonics
  - compensation of tuning differences
  - finer resolution (1, 1/2, 1/3 semitone resolution) → 12/24/36 dimensional chroma features

[Gómez, PhD 2006]

Cover Song Identification

Dealing with different keys

Bob Dylan – Knockin' on Heaven's Door
Avril Lavigne – Knockin' on Heaven's Door

- Compute average chroma vectors for each song
- Consider cyclic shifts of the chroma vectors to simulate transpositions
- Determine optimal shift indices so that the shifted chroma vectors are matched with minimal cost
- Transpose the songs accordingly

Cyclic Chroma Shifts

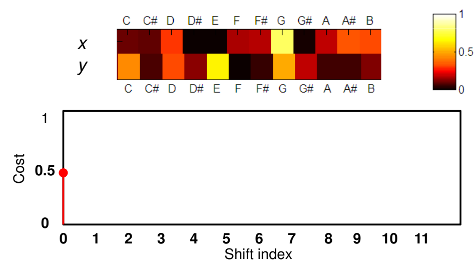
- Feature space: $\mathcal{F} = \mathbb{R}^{12}$
- Chroma vector: $x := (x(1), \ldots, x(12))^{\mathrm{T}} \in \mathcal{F}$
- Cyclic shift operator: $\sigma : \mathcal{F} \to \mathcal{F}$

$$\sigma((x(1), \ldots, x(12))^{\mathrm{T}}) := (x(12), x(1) \ldots, x(11))^{\mathrm{T}}$$

- Composition of shifts: $\sigma^i(x) = \sigma(\sigma^{i-1}(x)), \ \ i \in \mathbb{Z}$
- Note: $\sigma^{12} = \sigma^0$

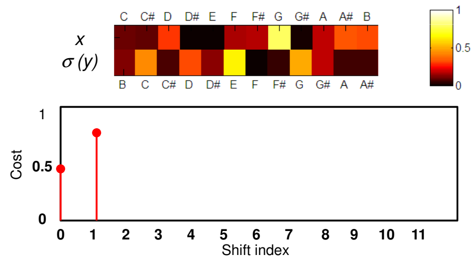
Cyclic Chroma Shifts

- Given chroma vectors $x, y \in \mathcal{F}$
- Fix a local cost measure $c : \mathcal{F} \times \mathcal{F} \to \mathbb{R}$
- Compute cost between *x* and shifted *y*



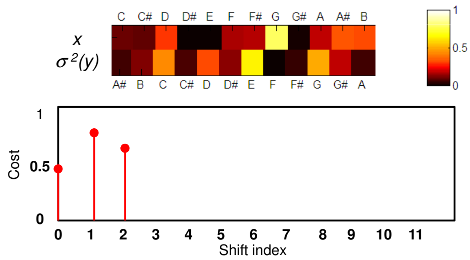
Cyclic Chroma Shifts

- Given chroma vectors $x, y \in \mathcal{F}$
- Fix a local cost measure $c : \mathcal{F} \times \mathcal{F} \to \mathbb{R}$
- Compute cost between *x* and shifted *y*



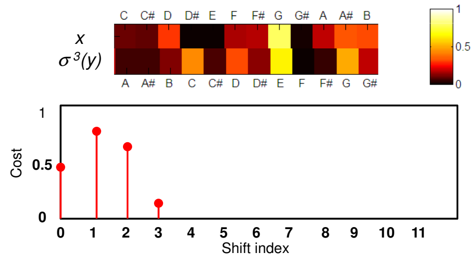
Cyclic Chroma Shifts

- Given chroma vectors $x, y \in \mathcal{F}$
- Fix a local cost measure $c : \mathcal{F} \times \mathcal{F} \to \mathbb{R}$
- Compute cost between *x* and shifted *y*



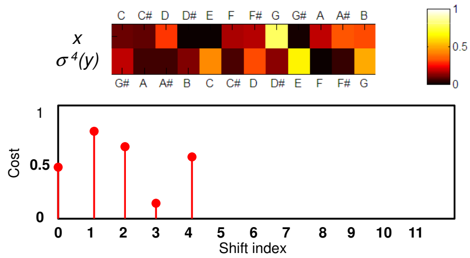
Cyclic Chroma Shifts

- Given chroma vectors $x, y \in \mathcal{F}$
- Fix a local cost measure $c : \mathcal{F} \times \mathcal{F} \to \mathbb{R}$
- Compute cost between *x* and shifted *y*



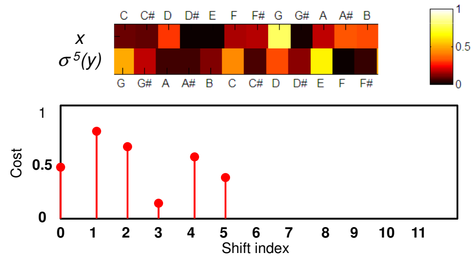
Cyclic Chroma Shifts

- Given chroma vectors $x, y \in \mathcal{F}$
- Fix a local cost measure $c : \mathcal{F} \times \mathcal{F} \to \mathbb{R}$
- Compute cost between *x* and shifted *y*



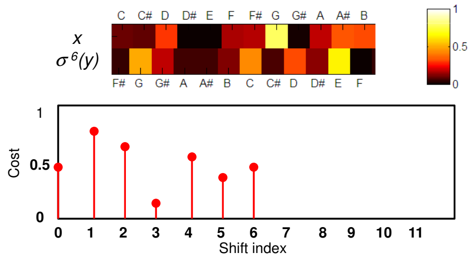
Cyclic Chroma Shifts

- Given chroma vectors $x, y \in \mathcal{F}$
- Fix a local cost measure $c : \mathcal{F} \times \mathcal{F} \to \mathbb{R}$
- Compute cost between *x* and shifted *y*



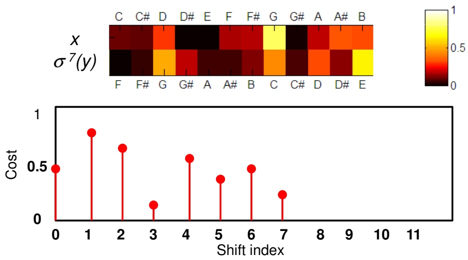
Cyclic Chroma Shifts

- Given chroma vectors $x, y \in \mathcal{F}$
- Fix a local cost measure $c : \mathcal{F} \times \mathcal{F} \to \mathbb{R}$
- Compute cost between *x* and shifted *y*



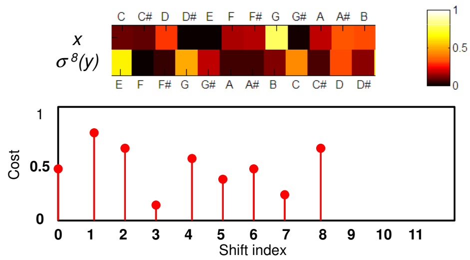
Cyclic Chroma Shifts

- Given chroma vectors $x, y \in \mathcal{F}$
- Fix a local cost measure $c : \mathcal{F} \times \mathcal{F} \to \mathbb{R}$
- Compute cost between *x* and shifted *y*



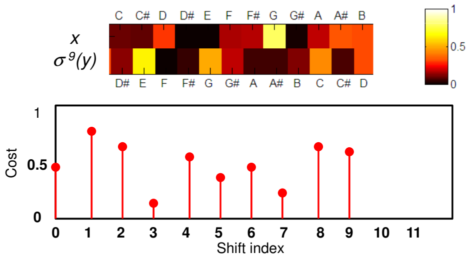
Cyclic Chroma Shifts

- Given chroma vectors $x, y \in \mathcal{F}$
- Fix a local cost measure $c : \mathcal{F} \times \mathcal{F} \to \mathbb{R}$
- Compute cost between *x* and shifted *y*



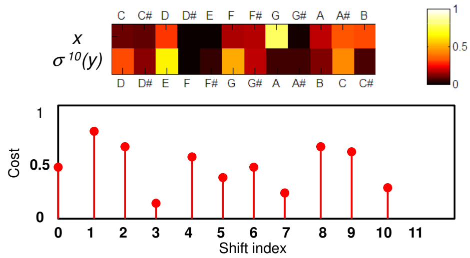
Cyclic Chroma Shifts

- Given chroma vectors $x, y \in \mathcal{F}$
- Fix a local cost measure $c : \mathcal{F} \times \mathcal{F} \to \mathbb{R}$
- Compute cost between *x* and shifted *y*



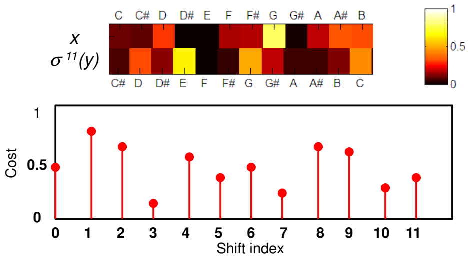
Cyclic Chroma Shifts

- Given chroma vectors $x, y \in \mathcal{F}$
- Fix a local cost measure $c : \mathcal{F} \times \mathcal{F} \to \mathbb{R}$
- Compute cost between *x* and shifted *y*



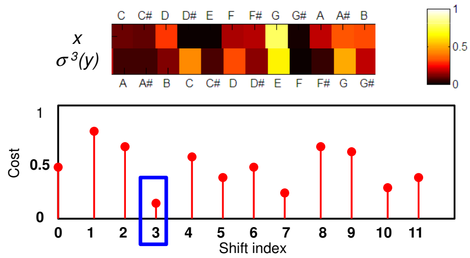
Cyclic Chroma Shifts

- Given chroma vectors $x, y \in \mathcal{F}$
- Fix a local cost measure $c : \mathcal{F} \times \mathcal{F} \to \mathbb{R}$
- Compute cost between *x* and shifted *y*



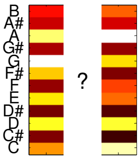
Cyclic Chroma Shifts

- Given chroma vectors $x, y \in \mathcal{F}$
- Fix a local cost measure $c : \mathcal{F} \times \mathcal{F} \to \mathbb{R}$
- Compute cost between *x* and shifted *y*
- Minimizing shift index: 3



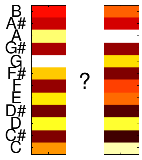
Cyclic Chroma Shifts

- What is a good local cost meaure for chroma space?



Cyclic Chroma Shifts

- What is a good local cost meaure for chroma space?
  Euclidean? Cosine distance?



- Is the chroma space Euclidean?
  Probably not!
  For example, C is musically closer to G than C#
- Idea: Usage of very coarse binary cost measure that indicates the same tonal root

[Serrà et al., IEEE-TASLP 2009]

Cyclic Chroma Shifts

- Original local cost measure $c : \mathcal{F} \times \mathcal{F} \to \mathbb{R}$
- Binary cost measure $c_{\mathrm{b}} : \mathcal{F} \times \mathcal{F} \to \{0, 1\}$

$$\mu(x, y) := \mathrm{argmin}_{i \in [0:11]} \left(c(x, \sigma^i(y)) \right)$$

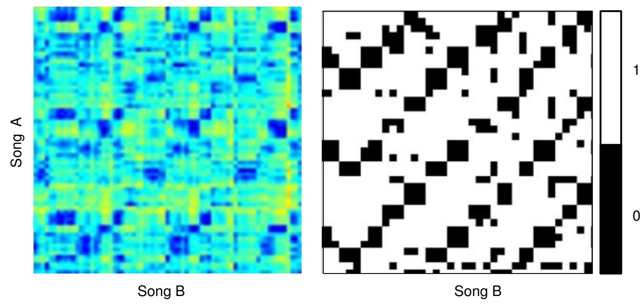
$$c_{\mathrm{b}}(x, y) := \begin{cases} 0 & \text{for } \mu(x, y) = 0 \\ 1 & \text{otherwise} \end{cases}$$

for $x, y \in \mathcal{F}$

[Serrà et al., IEEE-TASLP 2009]

Cyclic Chroma Shifts

Cost matrix based on *c* Binary cost matrix based on c_b

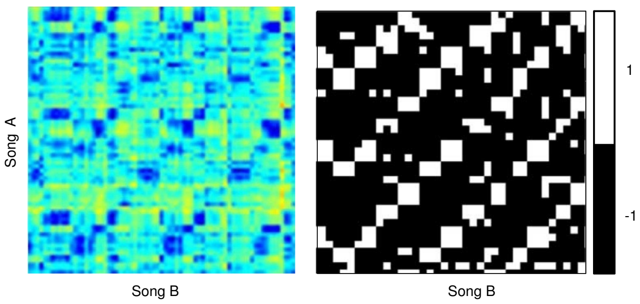


[Serrà et al., IEEE-TASLP 2009]

Cyclic Chroma Shifts

Think positive!

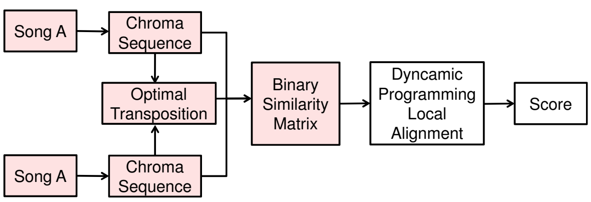
Cost matrix based on *c* Binary similarity matrix



[Serrà et al., IEEE-TASLP 2009]

Cover Song Identification

How to compare two different songs?

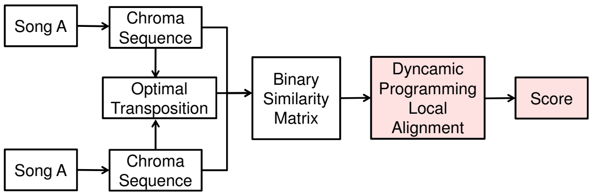


- Feature computation
- Dealing with different keys
- Local similarity measure
- Global similarity measure

[Serrà et al., IEEE-TASLP 2009]

Cover Song Identification

How to compare two different songs?



- Feature computation
- Dealing with different keys
- Local similarity measure
- Global similarity measure

[Serrà et al., IEEE-TASLP 2009]

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.

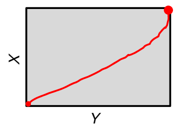
Local Alignment

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.

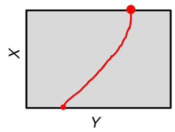
Strategy:
We use a variant of the Smith-Waterman algorithm.

Local Alignment

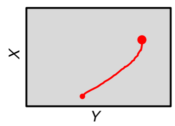
- Classical DTW
  Global correspondence between X and Y



- Subsequence DTW
  Subsequence of Y corresponds to X



- Local Alignment
  Subsequence of Y corresponds to subequence of X



Local Alignment

Computation of accumulated score matrix D
from given binary similarity (score) matrix S

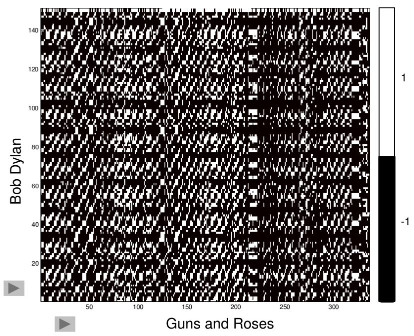
$$D(n,0) = D(0,m) = 0, \quad n \in [0:N], m \in [0:M]$$

$$D(n,m) = \max \begin{cases} 0 \\ D(n-1,m) - g \\ D(n,m-1) - g \\ D(n-1,m-1) + S(n,m) \end{cases}, \quad n,m > 0$$

- Zero-entry allows for jumping to any cell without penalty
- g *penalizes* ``inserts´´ and ``delets´´ in alignment
- Best local alignment score is the highest value in D
- Best local alignment ends at cell of highest value
- Start is obtained by backtracking to first cell of value zero

Local Alignment

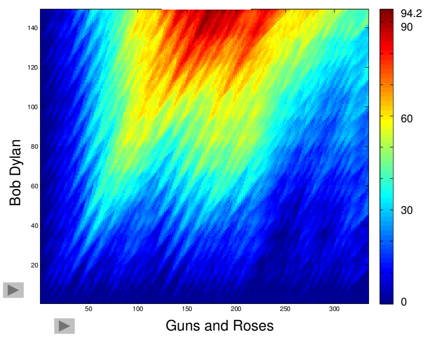
Example: Knockin' on Heaven's Door



Binary similarity matrix

Local Alignment

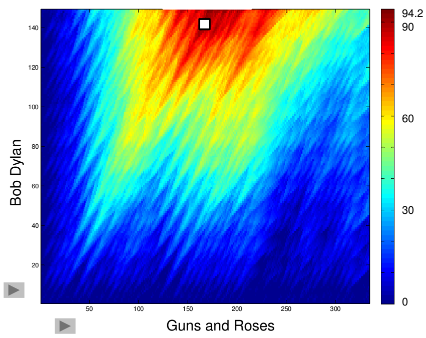
Example: Knockin' on Heaven's Door



Accumulated score matrix

Local Alignment

Example: Knockin' on Heaven's Door

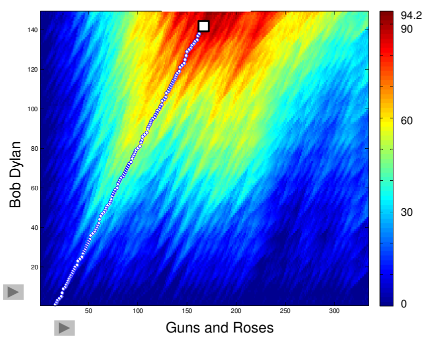


Accumulated score matrix

Cell with max. score = 94.2

Local Alignment

Example: Knockin' on Heaven's Door



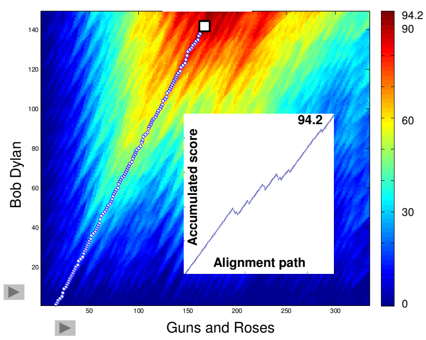
Accumulated score matrix

Cell with max. score = 94.2

Alignment path of maximal score

Local Alignment

Example: Knockin' on Heaven's Door



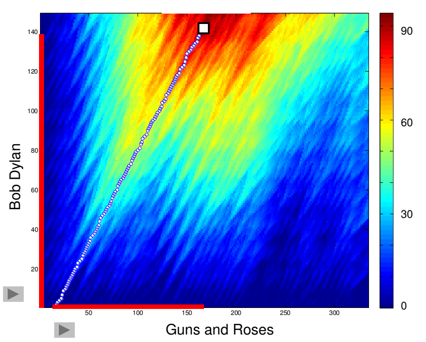
Accumulated score matrix

Cell with max. score = 94.2

Alignment path of maximal score

Local Alignment

Example: Knockin' on Heaven's Door



Accumulated score matrix

Cell with max. score = 94.2

Alignment path of maximal score

Matching subsequences

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
- Binary cost measure a good trade-off between robustness and expressiveness
- 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	Cover song identification
Identification	Concrete audio recording	Different interpretations	Different versions
Query	Short fragment (5-10 seconds)	Audio clip (10-40 seconds)	Entire song
Retrieval level	Subsequence	Subsequence	Document
Features	Spectral peaks (abstract)	Chroma (harmony)	Chroma (harmony)
Indexing	Hashing	Inverted lists	No indexing