ECS7006 Music Informatics

Week 10 - Audio Matching & Cover Song Detection

School of Electronic Engineering and Computer Science
Queen Mary University of London

prepared by Emmanouil Benetos
adapted from material by Meinard Müller

emmanouil.benetos@qmul.ac.uk

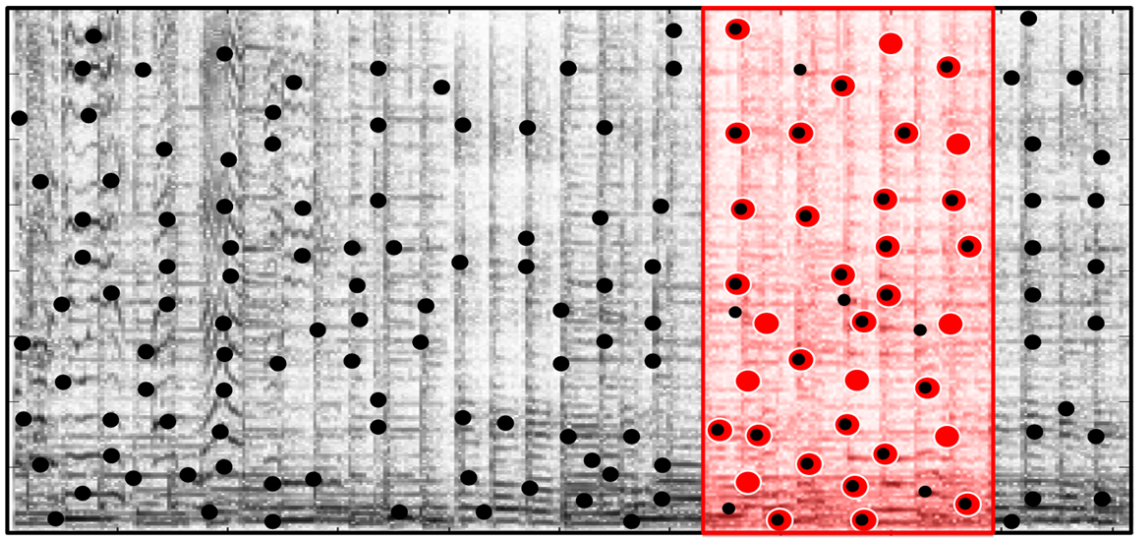
2023

Queen Mary University of London

Week 9 Recap

Audio Identification

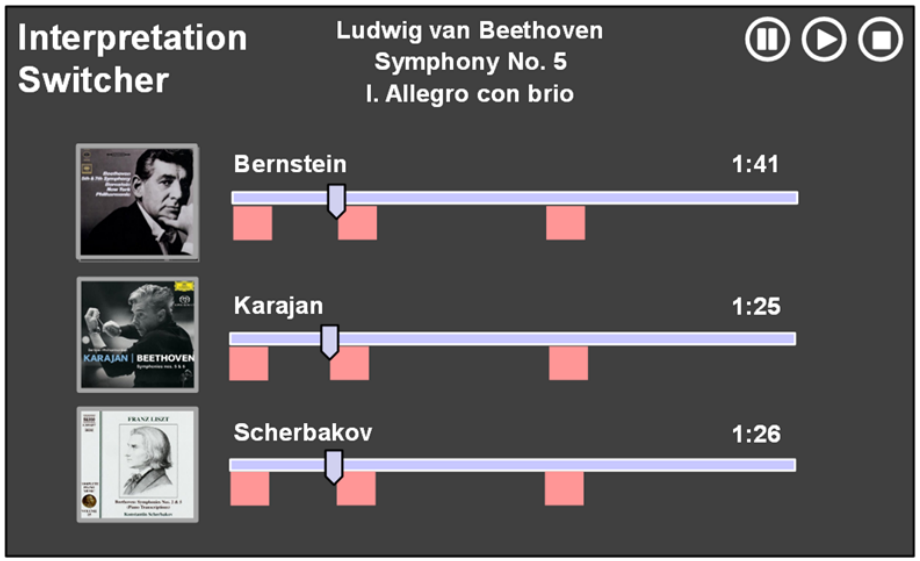
- Intro: content-based audio retrieval
- Audio identification
- Audio fingerprints
- Indexing techniques for audio identification
- Evaluation measures



This week's content

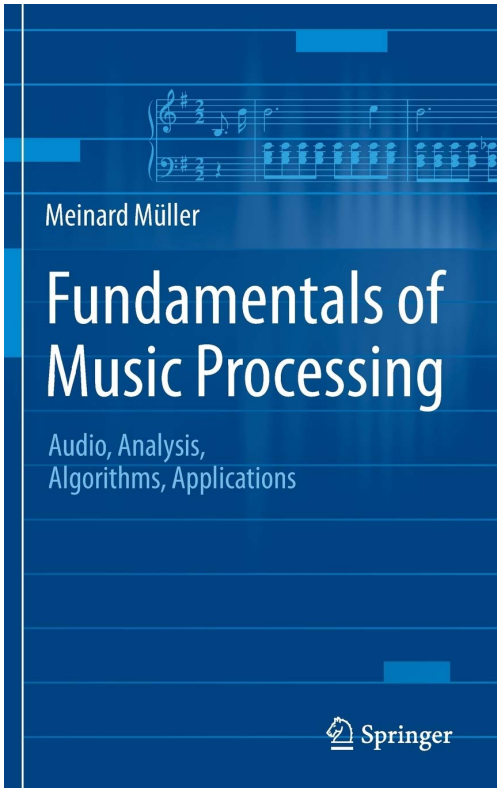
Audio Matching & Cover Song Detection

- Audio matching: requirements and feature design
- Diagonal matching
- DTW-based matching
- Cover song detection requirements
- Cover song detection procedure



Reading

Sections 7.2, 7.3.1, 7.3.2, and 7.4 of M. Müller, “Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications”, Springer, 2015.



Audio Matching: requirements and feature design

Audio Matching

Database:

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

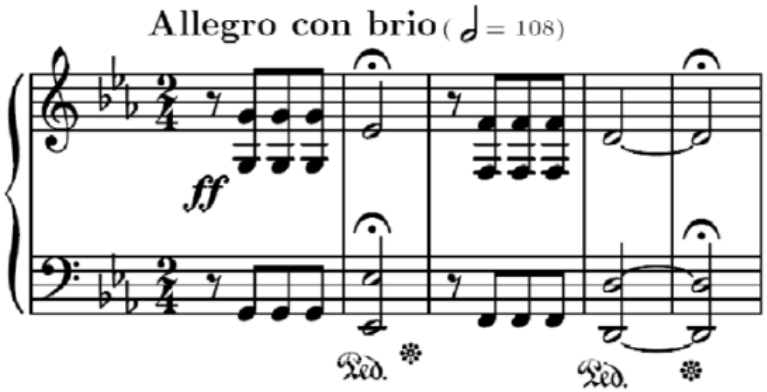
Goal: Given a short query audio fragment, find all corresponding audio fragments of similar musical content.

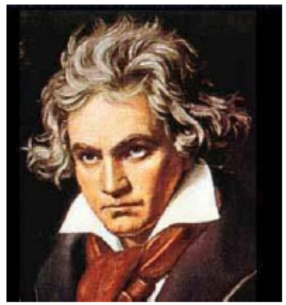
Notes:

- Instance of fragment-based retrieval
- Medium specificity
- A single document may contain several hits

Audio Matching

Example: Beethoven's Fifth





- Bernstein:
- Karajan:
- Gould (piano):
- Scherbakov (piano):

Audio Matching

Cross-modal retrieval in audio matching

Feature Design for Audio Matching

Audio fingerprints based on spectral peaks can characterise the local acoustic properties of a specific recording – but are not designed to handle musical variations.

Therefore, we need descriptors that capture musical characteristics of the underlying piece of music rather than acoustic characteristics of a specific recording.

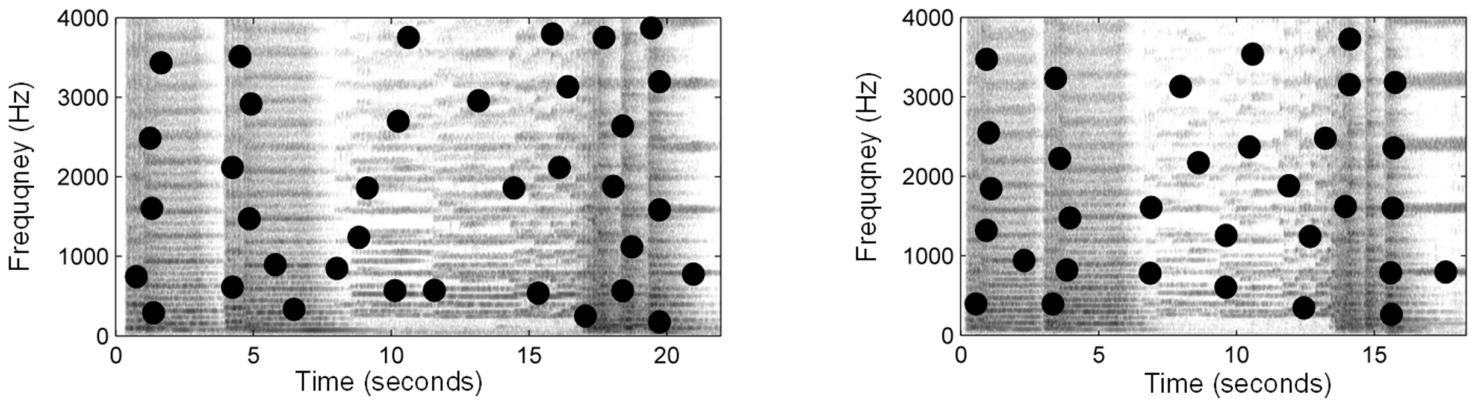


Figure: Spectral peaks for a Bernstein recording (left) and a Karajan recording (right).

Feature Design for Audio Matching

All these versions have roughly the same note material; and the same melodies are played within the same harmonic context.

⇒ chroma features are suitable representations for capturing this kind of information.

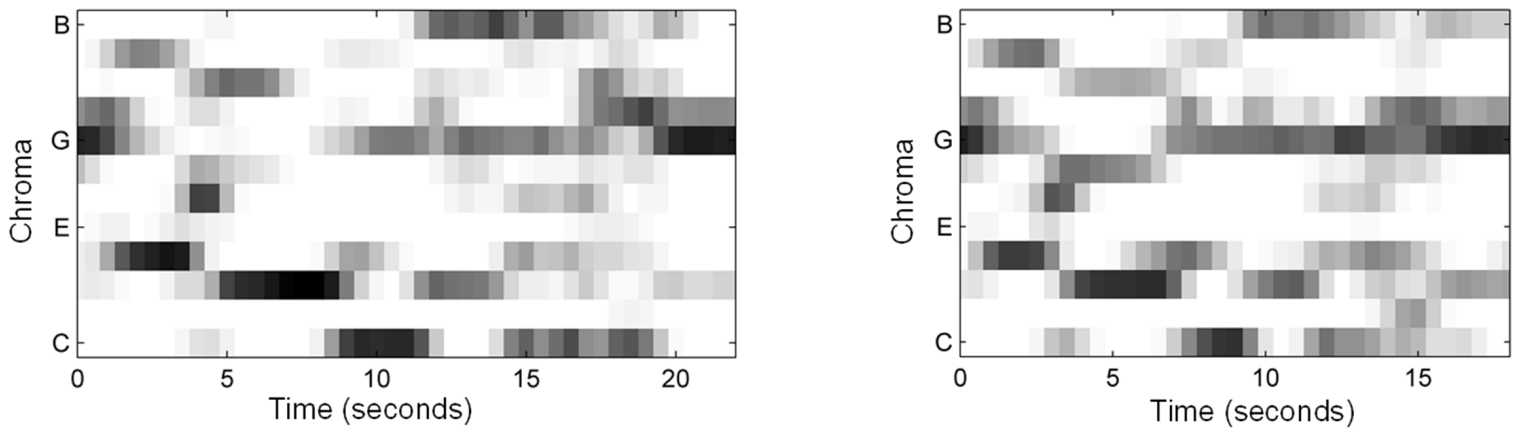


Figure: Chromagram for a Bernstein recording (left) and a Karajan recording (right).

Feature Design for Audio Matching

Which chroma variant is suitable for audio matching?

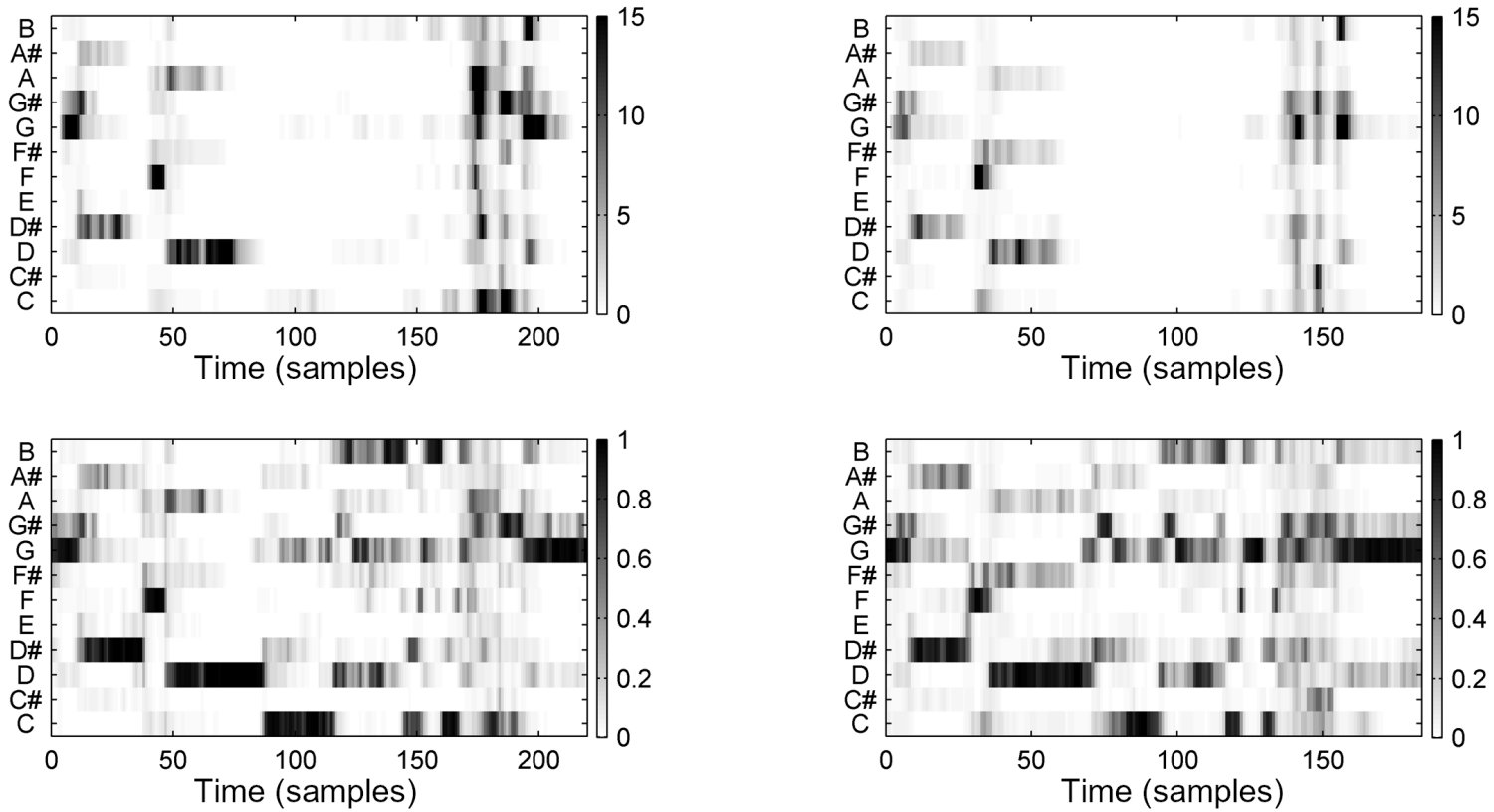


Figure: Top: basic chroma features for the Bernstein and Karajan recordings (10 Hz feature rate). Bottom: Normalised chroma features for the same recordings.

Feature Design for Audio Matching

We can apply additional quantisation and smoothing procedures to further reduce the effect of variations in local tempo, articulation, and note execution.

CENS features:

- Stand for Chroma Energy Normalised Statistics
- Main idea: taking statistics over large windows smooths local deviations in tempo, articulation, and execution of note groups (e.g. trills, arpeggios)

Feature Design for Audio Matching

CENS feature computation:

1. Obtain sequence of (Manhattan/L1) normalised chroma vectors $x_n \in [0,1]^{12}$, $n \in [1:N]$
2. Quantise x_n using quantisation function $Q : [0,1] \to \{0,1,2,3,4\}$:

$$Q(\alpha) = \begin{cases} 0, & 0 \leq \alpha < 0.05 \\ 1, & 0.05 \leq \alpha < 0.1 \\ 2, & 0.1 \leq \alpha < 0.2 \\ 3, & 0.2 \leq \alpha < 0.4 \\ 4, & 0.4 \leq \alpha \leq 1. \end{cases}$$

3. Smooth the quantised sequence for each chroma bin using a smoothing window (e.g. Hann) with length l.
4. Downsample sequence by a factor d and normalize sequence wrt the Euclidean norm.

Feature Design for Audio Matching

Notes on CENS features:

- Chroma components below a 5% threshold are set to zero, which introduces robustness to noise.
- Thresholds are chosen in a logarithmic fashion (based on the logarithmic perception of sound intensity).
- The smoothing and downsampling values (l and d) control the resulting features, which are more formally denoted as CENS_d^l.
- CENS features are a flexible and computationally inexpensive way to adjust feature specificity and resolution without performing any cost-intensive spectral audio decomposition (e.g. NMF)

Feature Design for Audio Matching

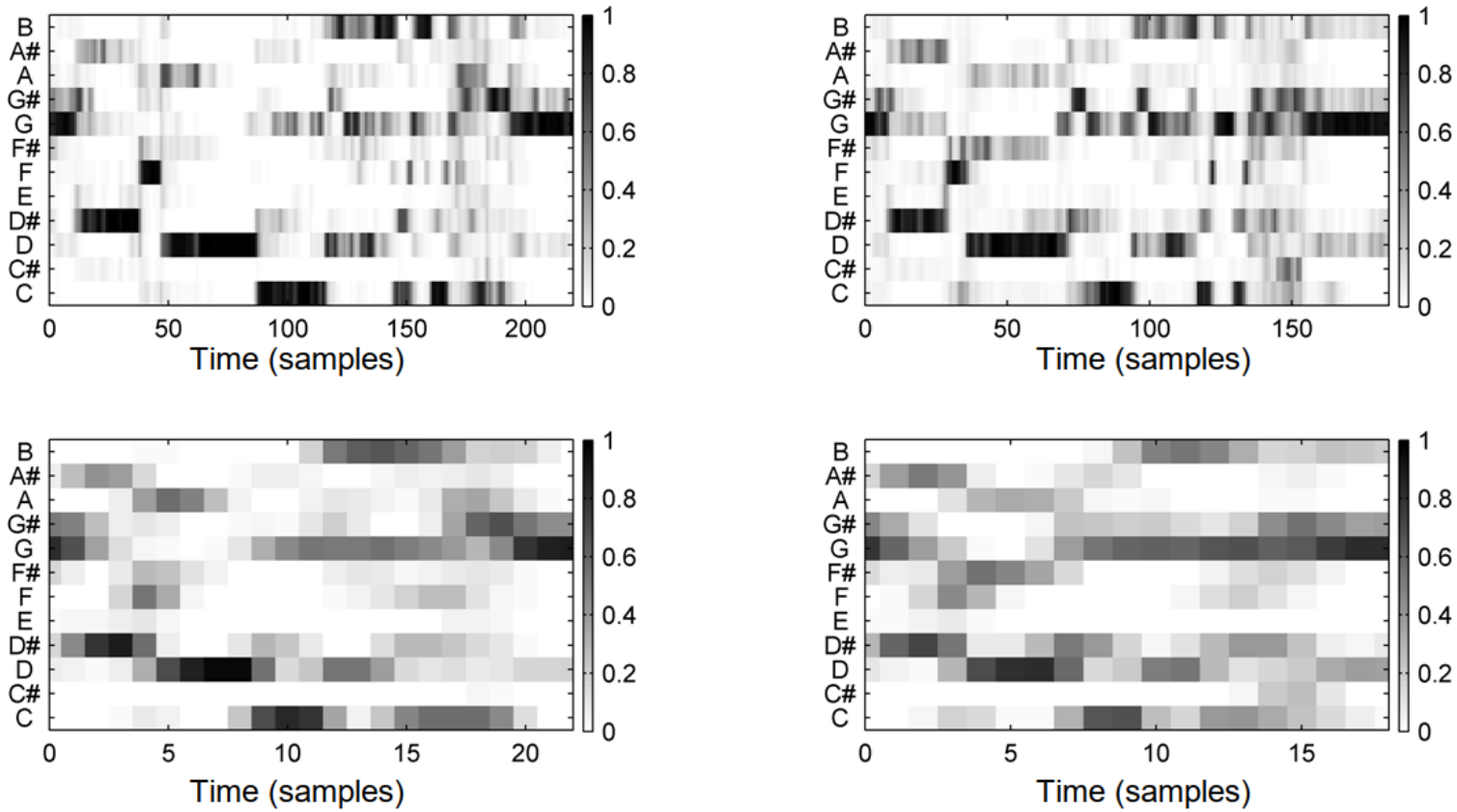


Figure: Top: normalised chroma features for the Bernstein and Karajan recordings. Bottom: CENS_{10}^{41} features for the same recordings.

Feature Design for Audio Matching

Conclusions:

- Strategy: absorb variations at feature level
- Chroma $\rightarrow$ invariance to timbre
- Normalization $\rightarrow$ invariance to dynamics
- Smoothing $\rightarrow$ invariance to local time deviations

Note: There is no standard chroma feature – variants can make a huge difference!

Diagonal Matching

Diagonal Matching

Problem statement:

- We are given a query audio fragment $\mathcal{Q}$ and a database recording $\mathcal{D}$.
- Let $X = (x_1, x_2, \ldots, x_N)$ and $Y = (y_1, y_2, \ldots, y_M)$ the chroma feature sequences for $\mathcal{Q}$ and $\mathcal{D}$, respectively.
- Typically $N \ll M$.

Intuitively, we can shift X over Y. Every subsequence of Y that is similar or has a small distance to X is considered a match for the query.

Diagonal Matching

We need a local cost or distance measure to compare X and Y, such as the cosine distance:

$$c(x, y) = 1 - \frac{\langle x|y\rangle}{\|x\| \cdot \|y\|}$$

where x and y are two vectors, $\langle x|y\rangle$ denotes the inner product of x and y, and $\|\cdot\|$ denotes the Euclidean norm.

When the chroma vectors are normalised wrt the Euclidean norm, the above distance is given by: $c(x, y) = 1 - \langle x|y\rangle$.

Diagonal Matching

Having a cost function, we can now compare the query sequence $X = (x_1, x_2, \ldots, x_N)$ with all subsequences $(y_{1+m}, \ldots, y_{N+m})$ of Y, where m denotes the shift index.

This procedure yields a matching function Δ_{diag} defined by:

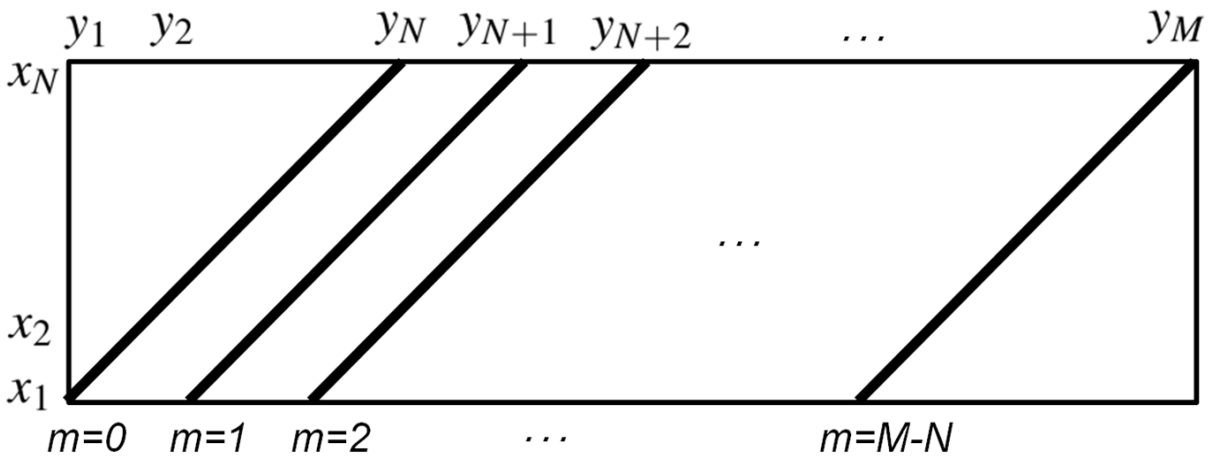
$$\Delta_{\text{diag}}(m) := \frac{1}{N} \sum_{n=1}^{N} c(x_n, y_{n+m}).$$

Diagonal Matching

Another way of interpreting the matching function is by considering a cost matrix $\mathbf{C} \in \mathbb{R}^{N \times M}$ as:

$$\mathbf{C}(n, m) = c(x_n, y_m)$$

Then, the value $\Delta_{\text{diag}}(m)$ is obtained by summing up diagonals of the matrix $\mathbf{C}$ (hence "diagonal matching").



Diagonal Matching

To determine the best match between $\mathcal{Q}$ and $\mathcal{D}$, we look for the index m^* that minimises $\Delta_{\text{diag}}(m)$:

$$m^* = \underset{m \in [0:M-N]}{\text{argmin}} \; \Delta_{\text{diag}}(m)$$

The best match is then given by the audio clip corresponding to the subsequence:

$$Y(1 + m^* : N + m^*) = (y_{1+m^*}, \ldots, y_{N+m^*})$$

Diagonal Matching

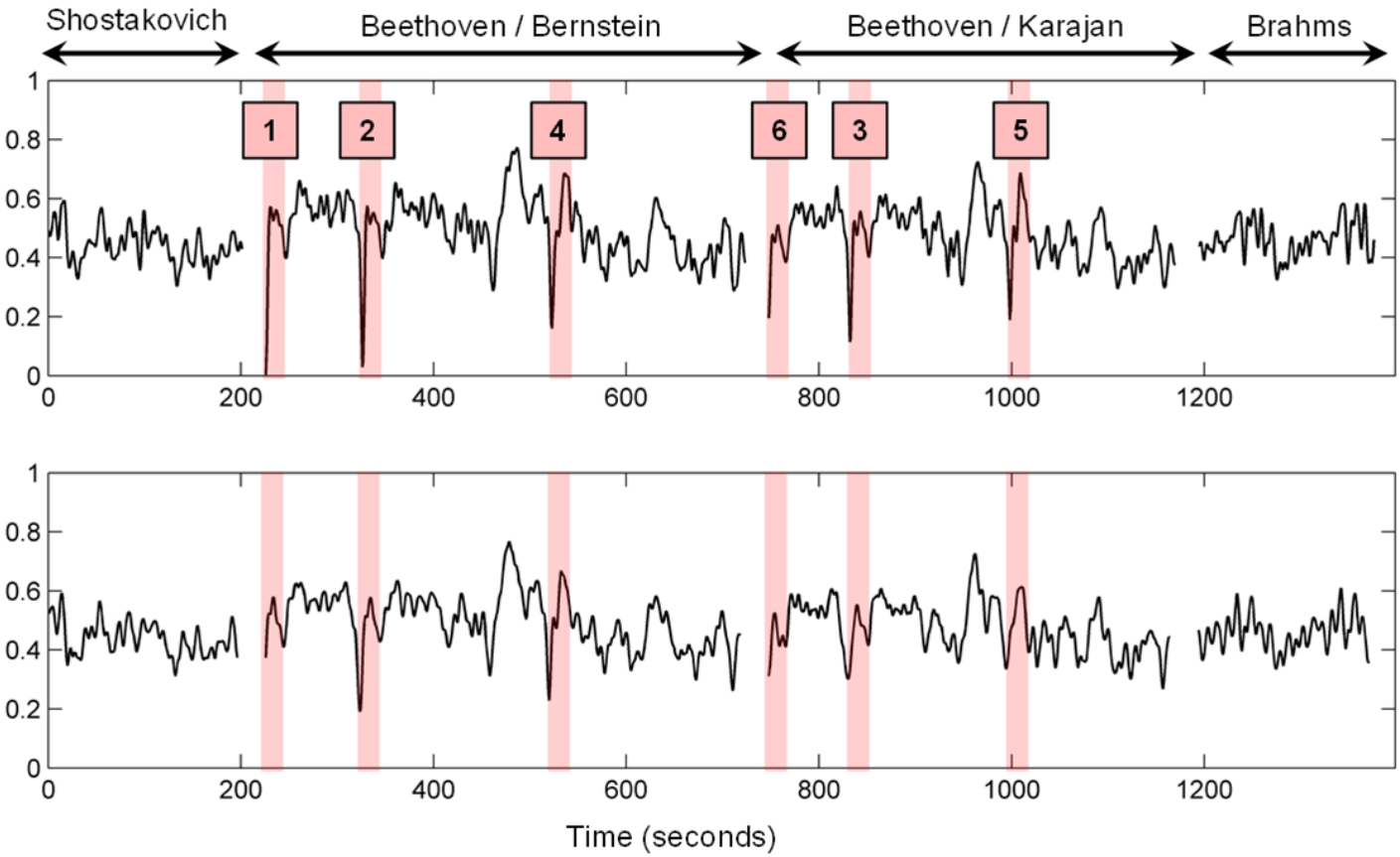


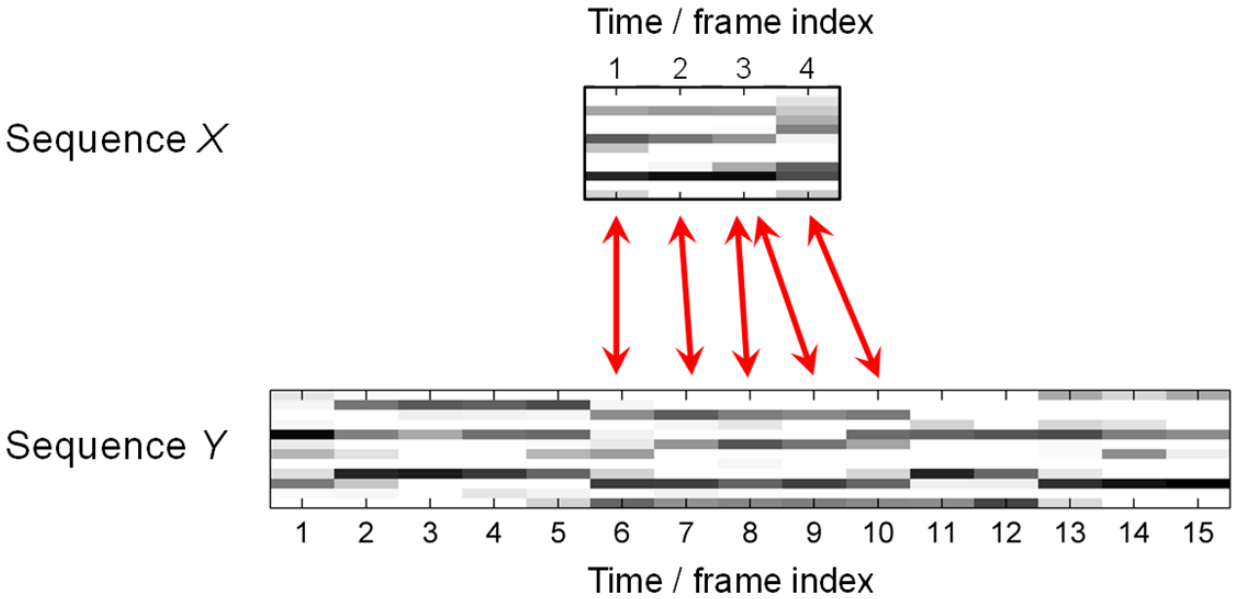
Figure: Matching functions for a database consisting of 4 recordings. Top: query consists of a segment of the Bernstein recording. Bottom: Query has the Bernstein recording with a tempo reduced by 25% (matching fails).

DTW-based Matching

DTW-based Matching

When the tempo difference between the query and a database section becomes too large, the diagonal matching procedure is doomed to fail.

Solution: finding optimal subsequences using Dynamic Time Warping.



DTW-based Matching

The matching problem can be formulated as the task of finding the subsequence of Y that minimises the DTW distance to X:

$$(a^*, b^*) = \underset{(a,b):1 \leq a \leq b \leq M}{\text{argmin}} \text{DTW}\big(X, Y(a:b)\big)$$

where $Y(a:b) = (y_a, y_{a+1}, \ldots, y_b)$.

A slight modification of the DTW algorithm is necessary in order to allow for omissions at the beginning and at the end of Y in the alignment with X.

DTW-based Matching

1. Initialize accumulated cost matrix $\mathbf{D} \in \mathbb{R}^{N \times M}$:

$$\mathbf{D}(n,1) = \sum_{k=1}^{n} \mathbf{C}(k,1), \qquad \mathbf{D}(1,m) = \mathbf{C}(1,m)$$

2. Recursion:

$$\mathbf{D}(n,m) = \mathbf{C}(n,m) + \min \begin{cases} \mathbf{D}(n-1,m-1) \\ \mathbf{D}(n-1,m) \\ \mathbf{D}(n,m-1) \end{cases}$$

3. Estimating a^* and b^*:

$$b^* = \underset{b \in [1:M]}{\operatorname{argmin}} \mathbf{D}(N,b)$$

a^* is estimated through the DTW backtracking procedure.

DTW-based Matching

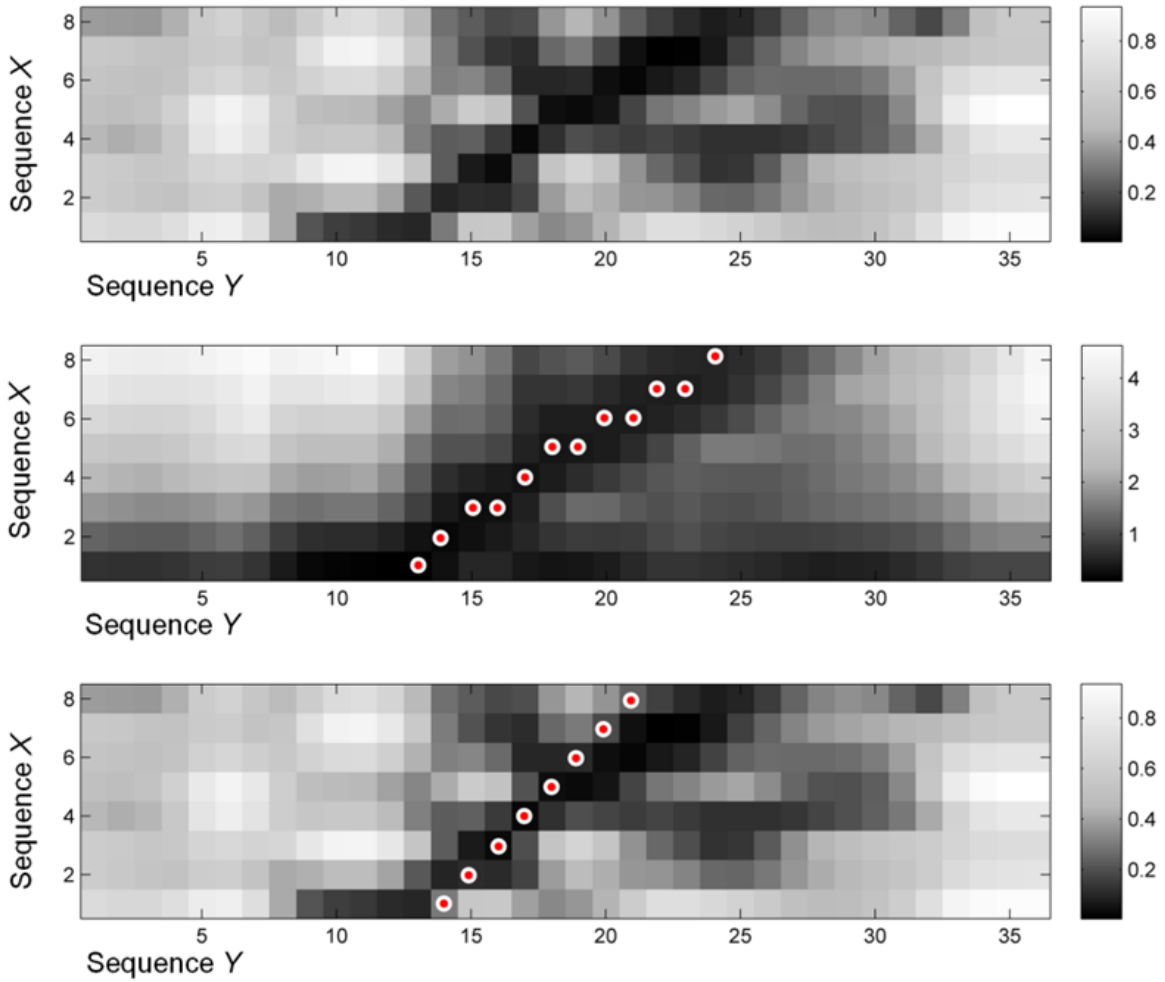


Figure: Top: cost matrix **C**. Middle: Accumulated cost matrix **D** and warping path. Bottom: Cost matrix **C** with optimal diagonal match.

DTW-based Matching

We can also define a matching function Δ_{DTW} by setting:

$$\Delta_{\text{DTW}}(m) = \frac{1}{N}\mathsf{D}(N, m)$$

which normalises the accumulated cost by the length N of the query.

Each local minimum b of Δ_{DTW} indicates the end position of a subsequence $Y(a : b)$ that has a small DTW distance to X.

DTW-based Matching

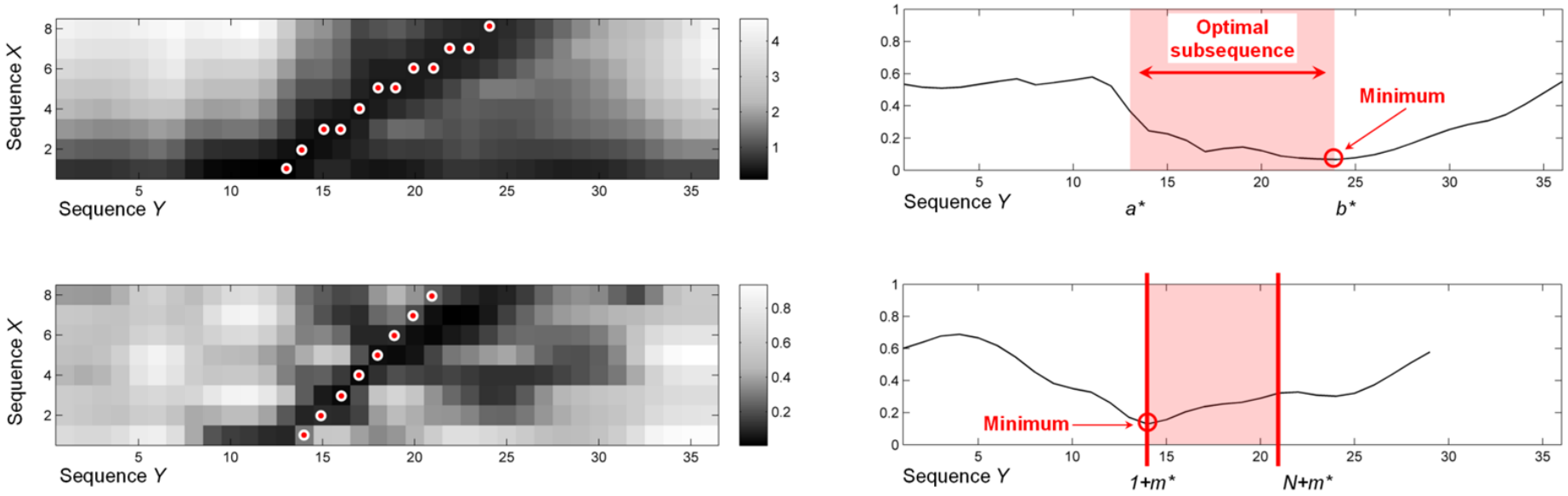


Figure: Top row: accumulated cost matrix **D** and matching function Δ_{DTW}. Bottom row: cost matrix **C** and matching function Δ_{diag}.

DTW-based Matching

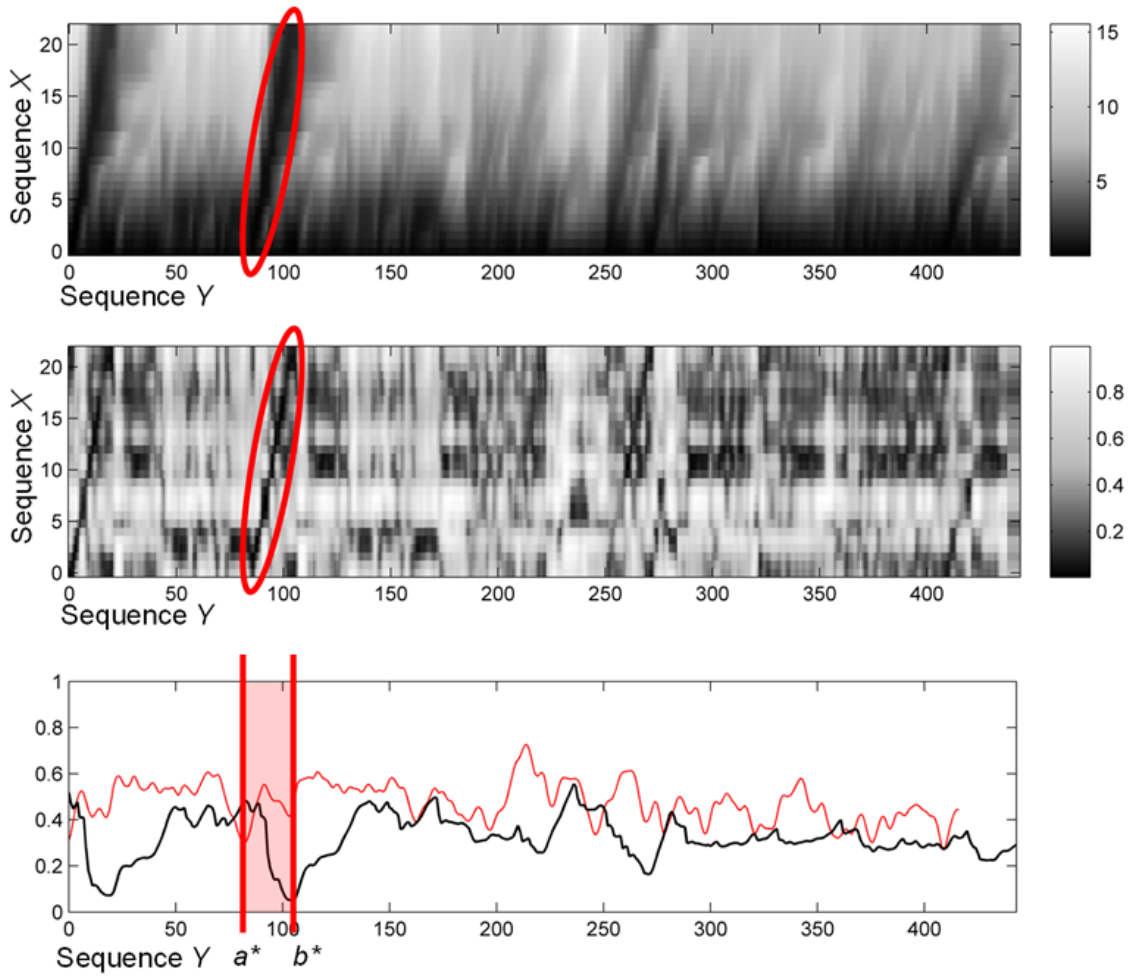


Figure: Matching tempo-modified Bernstein query and Karajan recording. Top: Cost matrix **C**. Middle: accumulated cost matrix **D**. Bottom: Δ_{diag} (red) and Δ_{DTW} (black).

DTW-based Matching

The subsequence variant of DTW can be modified in the same way as classical DTW:

- Using the set $\Sigma = \{(1,0),(0,1),(1,1)\}$ can lead to paths that are highly deteriorated, e.g. sequence X can be assigned to a single element of Y.
- Using $\Sigma = \{(1,1)\}$ leads to diagonal matching.
- The set $\Sigma = \{(2,1),(1,2),(1,1)\}$ yields a compromise between strict diagonal matching and DTW-based matching.

It is also possible to create a transposition-invariant matching function using DTW.

Cover Song Detection

Cover Song Detection

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!

“Day Tripper” (The Beatles):
“Day Tripper” (Ocean Colour Scene):

Cover Song Detection

Motivation

- Automated organization of music collections
  "Find me all covers of..."
- Musical rights management
- Learning about music itself
  "Understanding the essence of a song"

Versions in music

- Using the term "piece of music" or "original version" can be problematic
- Folk music $\rightarrow$ oral transmission, undergoing various changes over hundreds of years
- Jazz $\rightarrow$ improvisation and resulting variations are key elements of the music

Cover Song Detection

Types of versions

- Western classical music: the original version is given in the form of a musical score. All performances are considered original versions.
- Arrangement (also “transcription”): reworking of a piece of music so that it can be played by different instruments.
- Medley: piece composed by parts of existing pieces
- Sampling: taking portions of one recording and reusing them in a different piece
- Remix: a recording that uses original material, but has been edited or completely recreated to sound different from its original version
- A remix can be close to a remaster, or closer to a mashup
- Cover version / cover song: loosely refers to a new performance of a previously released song by someone other than the original artist

Cover Song Detection

Types of modifications

- Tempo, e.g. slow vs. fast
- Timing, i.e. subtle temporal fluctuations
- Timbre, e.g. instrumentation, playing styles, room acoustics, equalisation
- Key, e.g. pitch range might adapted to a different singer or instrumentation
- Harmonization, e.g. new chordal accompaniment in jazz
- Lyrics, e.g. translation to another language
- Structure, e.g. intro may be skipped, instrumental solo section may be added

Cover Song Detection

Relevant literature

- Gómez/Herrera (ISMIR 2006)
- Casey/Slaney (ISMIR 2006)
- Serrà (ISMIR 2007)
- Ellis/Poliner (ICASSP 2007)
- Serrà/Gómez/Herrera/Serra (IEEE TASLP 2008)
- Du et al (ICASSP 2021)
- O'Hanlon et al (MLSP 2021)

(plus section 7.3 in the FMP book)

Cover Song Detection - Procedure

When assessing global similarity between a query document and a database document, in cover song detection we look for local concurrences with regard to certain musical properties $\rightarrow$ global comparison is done on a local basis.

In the following, we restrict to the scenario where versions share a similar melodic or harmonic progression.

Cover Song Detection - Procedure

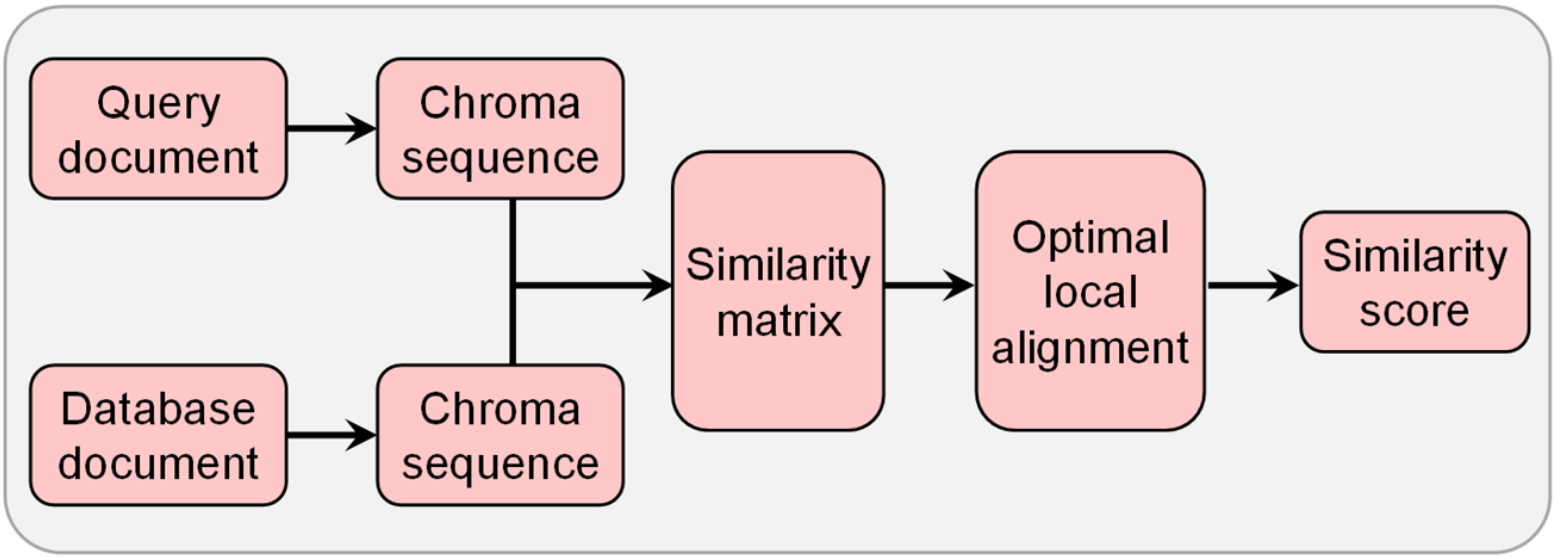


Figure: Overview of the pipeline for a typical cover song detection system.

Cover Song Detection - Procedure

Assumption:

Two songs are considered as similar if they contain possibly long subsegments that possess a similar harmonic progression. Temporal deformations should also be supported.

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.

Tradeoff:

Minimising the overall matching cost on the one hand and maximising the lengths of the subsequences on the other hand.

Cover Song Detection - Procedure

Similarity matrix

We compute a similarity matrix $\mathbf{S} \in \mathbb{R}^{N \times M}$ by setting:

$$\mathbf{S}(n, m) = s(x_n, y_m)$$

where $s(x, y) = |\langle x|y\rangle|$. The similarity matrix can also be enhanced (e.g. by thresholding).

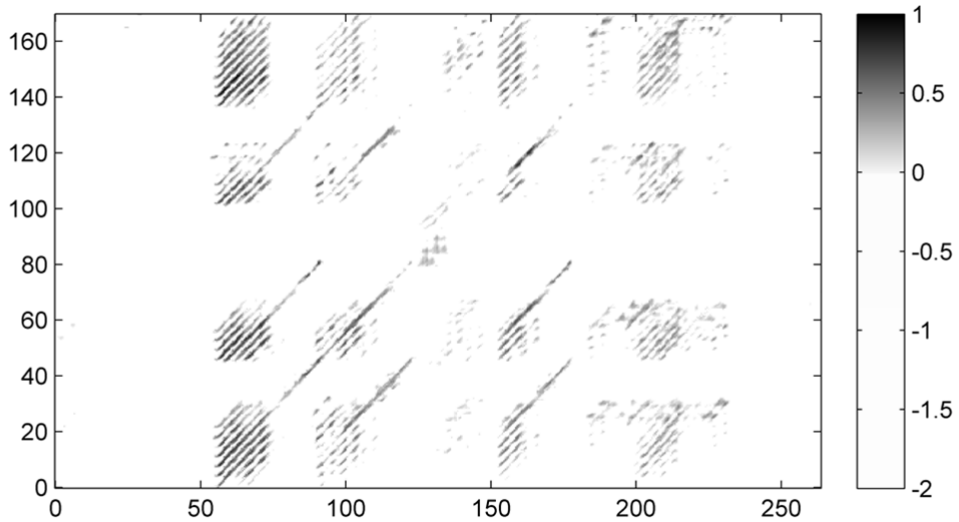


Figure: Enhanced similarity matrix for the “Day Tripper” song (vertical axis: Beatles; horizontal axis: Ocean Colour Scene).

Cover Song Detection - Procedure

Local Alignment

The Smith-Waterman algorithm is a well-known algorithm for performing local sequence alignment in bioinformatics, used for determining similar regions between two nucleotide or protein sequences.

Strategy:
Using a variant of the Smith-Waterman algorithm applied to **S**. The algorithm constructs an accumulated scoring matrix **D** similar to DTW, and obtains a score-maximising path P^*.

The similarity score $\gamma(\mathcal{Q}, \mathcal{D})$ between query $\mathcal{Q}$ and database document $\mathcal{D}$ is given by the maximal entry of **D**.

Cover Song Detection - Procedure

More formally:

- A path of length L is a sequence $P = \big((n_1, m_1), \ldots, (n_L, m_L)\big)$.
- The score $\sigma(P)$ of P is defined as:

$$\sigma(P) = \sum_{l=1}^{L} \mathsf{S}(n_l, m_l)$$

- The optimisation task finds the score-maximising path:

$$P^* = \arg\max_{P} \sigma(P)$$

- The score-maximising path is computed via a dynamic programming method similar to the DTW algorithm, which computes D.
- The similarity between the query and document is given by:

$$\gamma(\mathcal{Q}, \mathcal{D}) = \sigma(P^*) = \max_{(n,m) \in [1:N] \times [1:M]} \mathsf{D}(n, m)$$

Cover Song Detection - Procedure

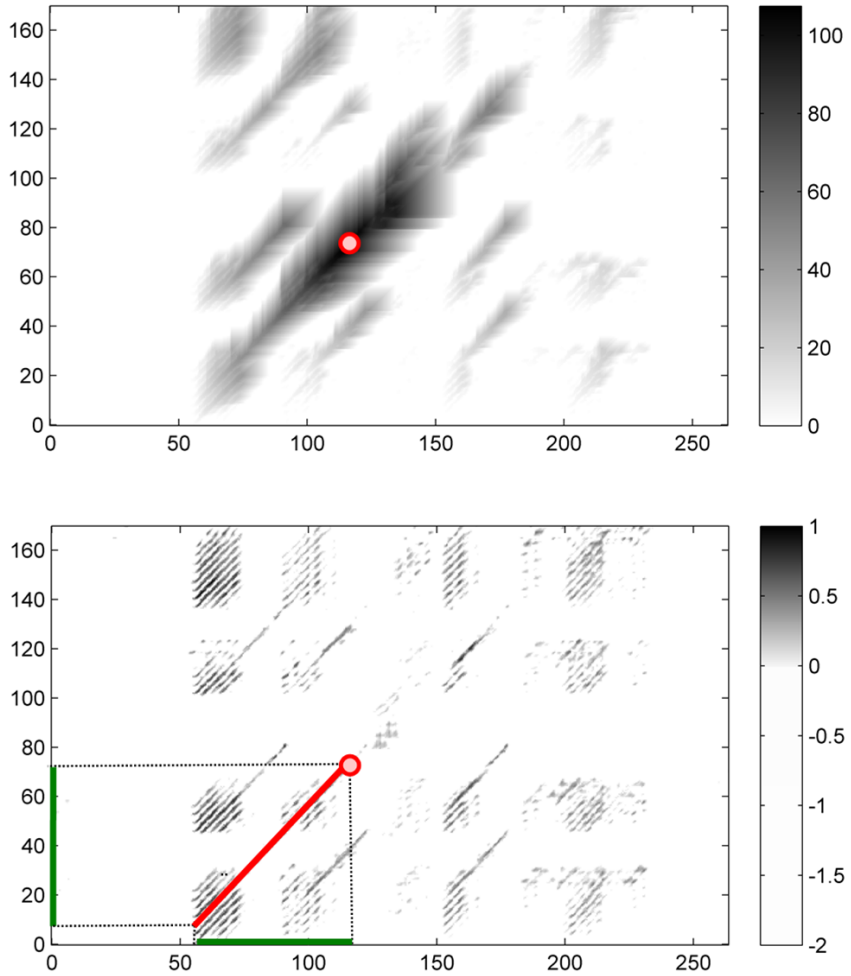


Figure: Top: Accumulated score matrix **D** for the “Day Tripper” example. Bottom: **S** with score-maximising path.

Cover Song Detection - Procedure

In a cover song detection system, a given query document is compared with all database documents. The database documents are then ranked according to the computed similarity score $\gamma(\mathcal{Q}, \mathcal{D}_i)$.

Every song has to be compared with any other
$\rightarrow$ method does not scale to large music collections

Current research in cover song detection:

- Focusing on harmony/melody; beat-synchronous chroma features are commonly used
- Dynamic programming algorithms are used for local alignment
- Common evaluation metric is the precision at rank

Audio Retrieval - Summary

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)