Music Retrieval Scenarios

PCPs used in classification, key/chord estimation, melody retrieval, and **cover song retrieval**, i.e., finding songs that are based on the same melody/tune, independent of instrumentation (timbre)

Another scenario is to find different songs that nevertheless **"sound similar"**

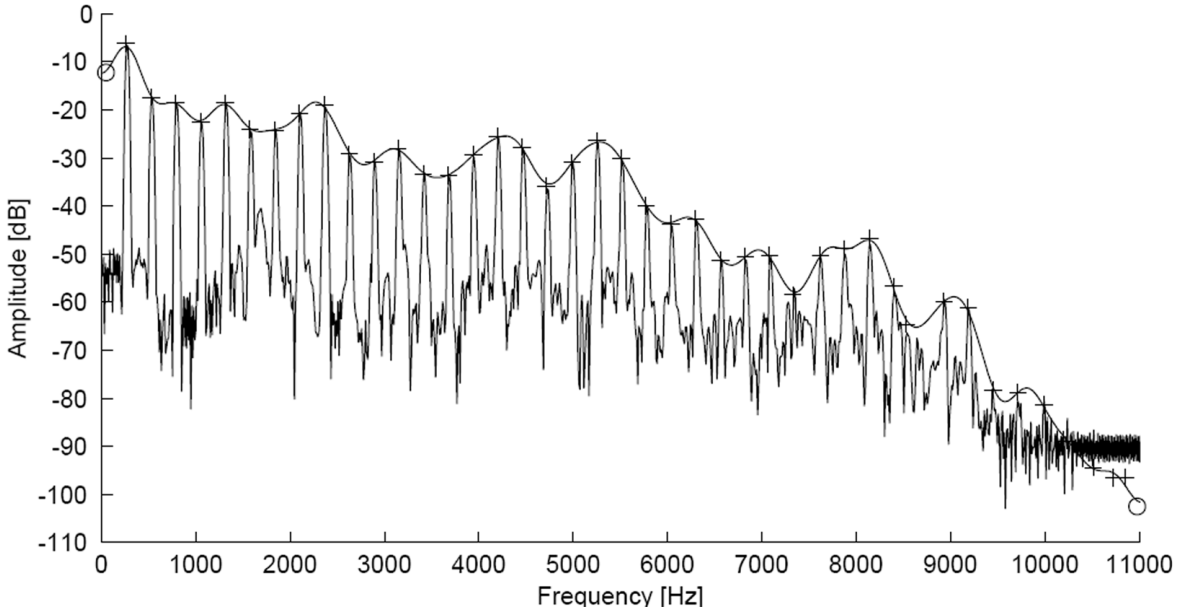
This is most often and predominantly related to timbre aspects (although it is more complex than that – see Lecture I)

MFCCs have shown to be better descriptors for this task

MFCCs

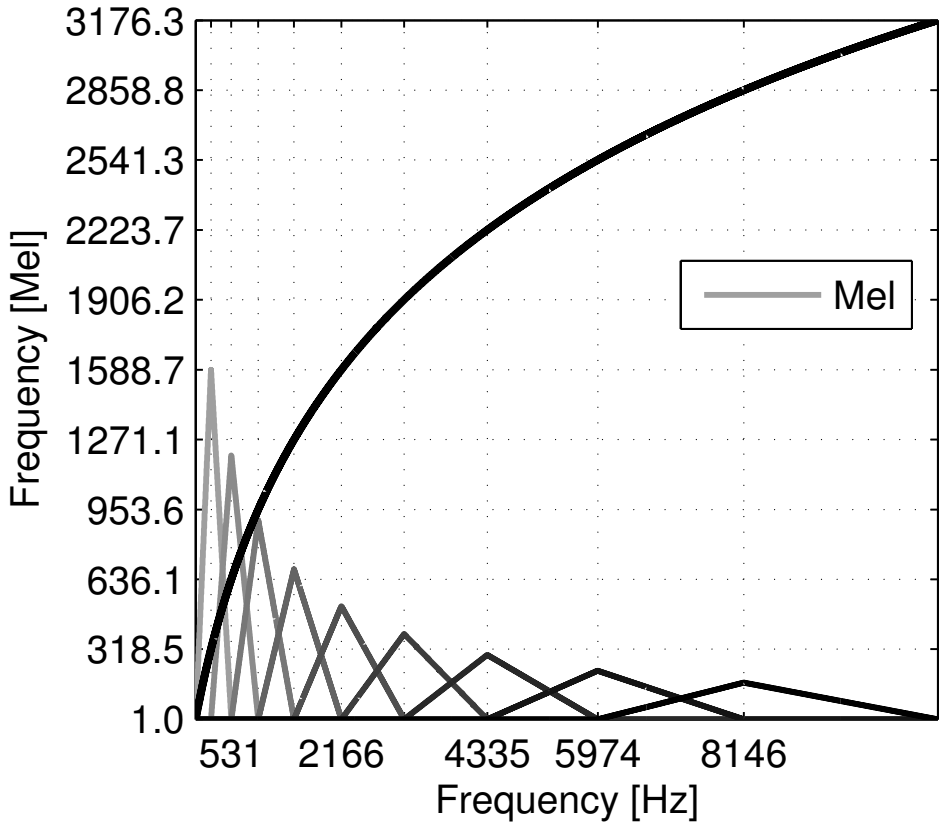
Mel Frequency Cepstral Coefficients (MFCCs) have their roots in speech recognition and are a way to represent the *envelope of the power spectrum* of an audio frame

- the spectral envelope captures perceptually important information about the corresponding sound excerpt (*timbral aspects*)
- most important for music similarity: sounds with similar spectral envelopes are generally perceived as similar.



Department of Computational Perception

The Mel Scale



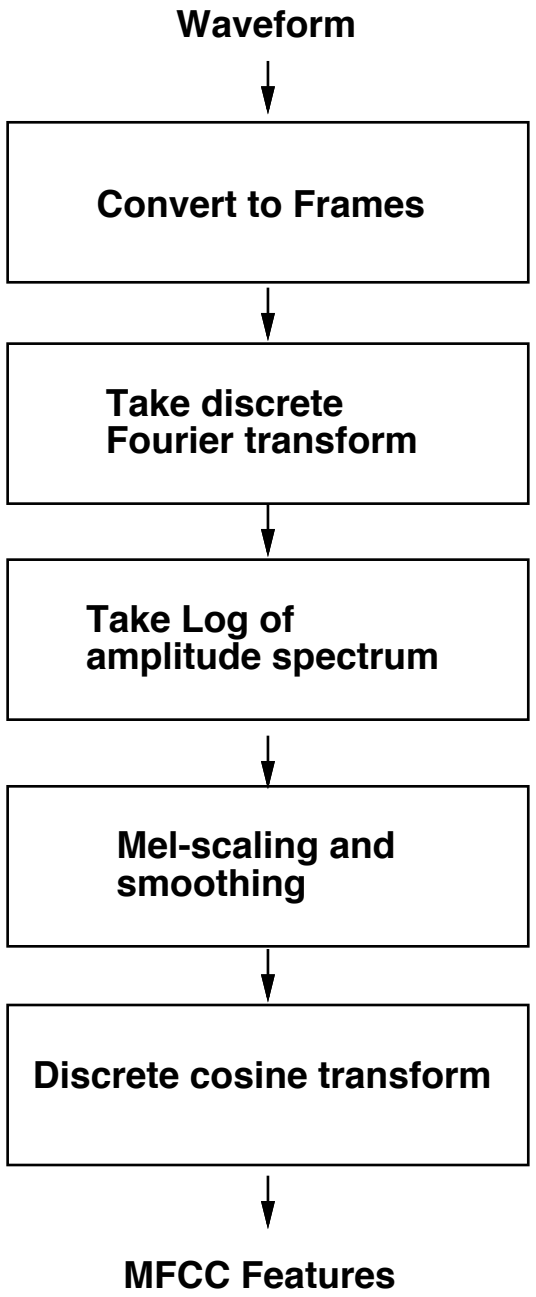
Perceptual scale of pitches judged by listeners to be equal in distance from one another

Given Frequency *f* in Hertz, the corresponding pitch in Mel can be computed by

$$m = 2595 \log_{10}\left(1 + \frac{f}{700}\right)$$

Normally around 40 bins equally spaced on the Mel scale are used

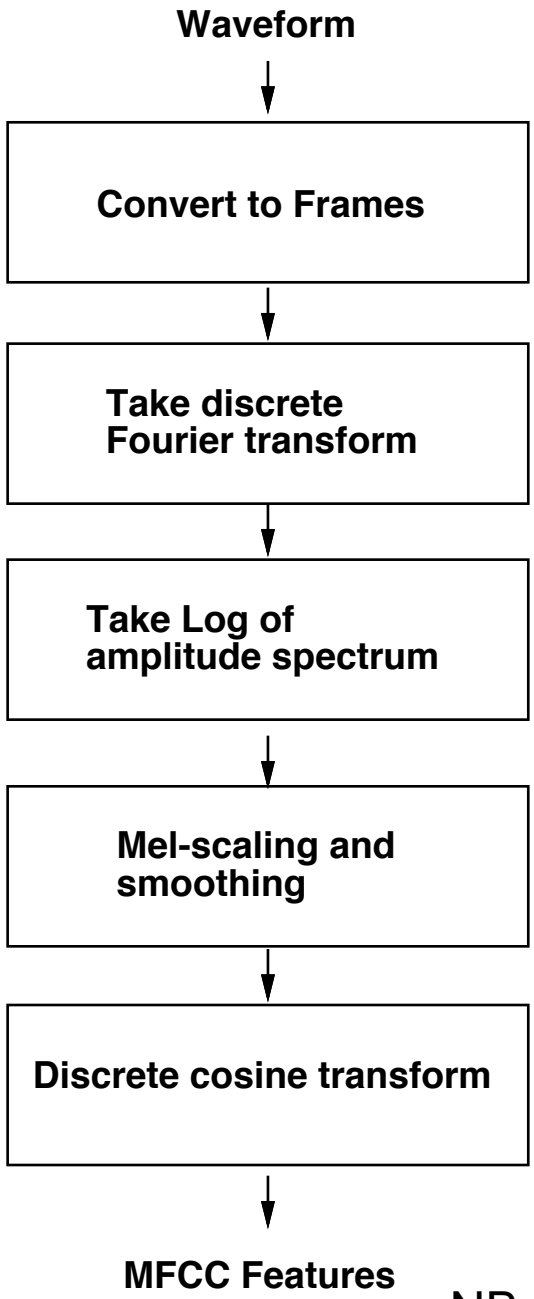
Department of Computational Perception



MFCCs

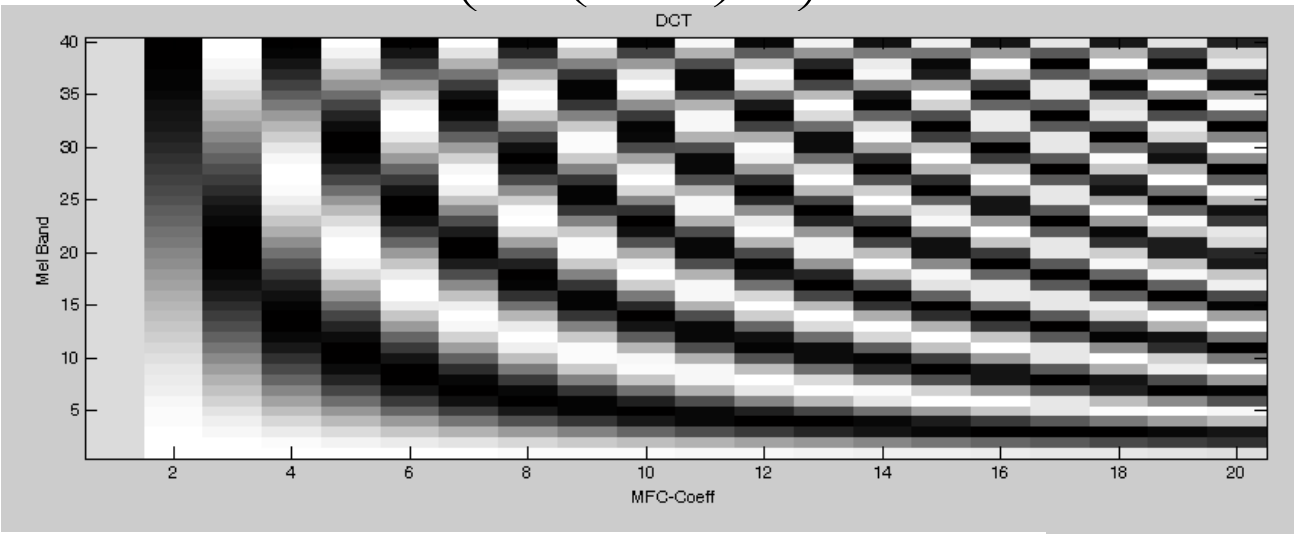
MFCCs are computed per frame

1. STFT: short-time Fourier transform
2. the logarithm of the amplitude spectrum is taken (motivated by the way we humans perceive loudness)
3. mapping of the amplitude spectrum to the Mel scale
4. quantize (e.g., 40 bins) and make linear (DCT doesn't operate on log scale)



5. perform Discrete Cosine Transform to de-correlate the Mel-spectral vectors
   - similar to FFT; only real-valued components
   - describes a sequence of finitely many data points as sum of cosine functions oscillating at different frequencies
   - results in n coefficients (e.g., $n = 20$)

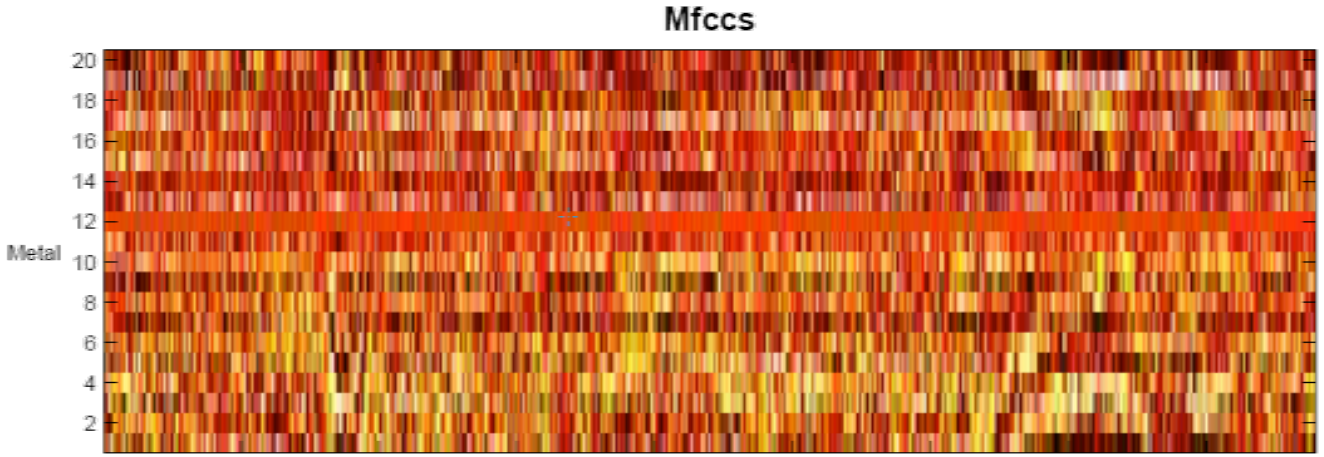
$$X_k = \sum_{n=0}^{N-1} x_n \cdot \cos\left(\frac{\pi}{N} \cdot \left(n + \frac{1}{2} \right) \cdot k \right) \qquad k = 0,\ldots,N-1$$



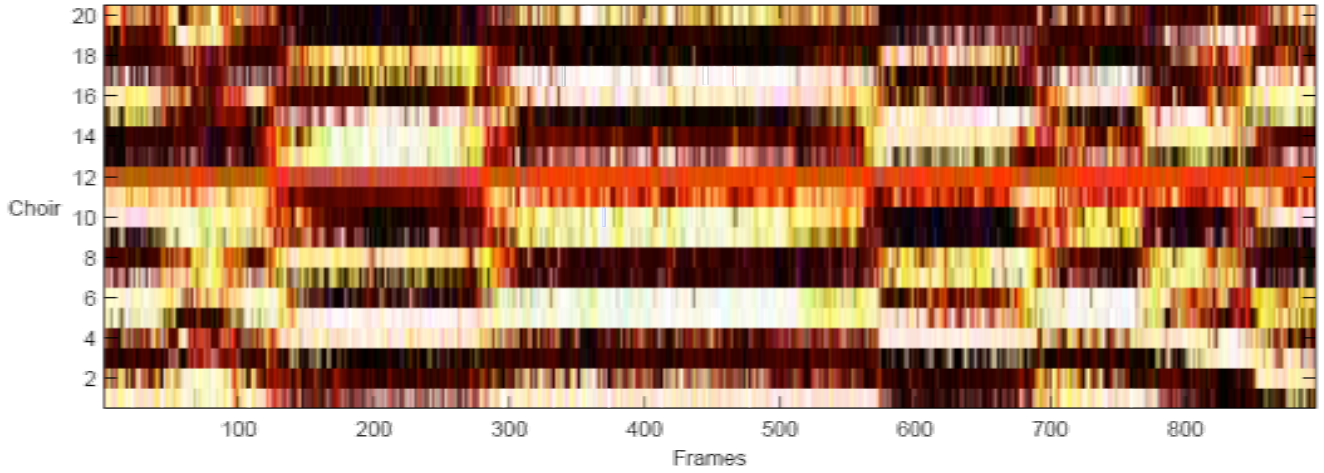
NB: performing (inverse) FT or similar on log representation of spectrum => “cepstrum” (anagram!)

MFCC Examples

Metal



Choir



Department of Computational Perception

“Bag-of-frames” Modeling

Full music piece is now a set of MFCC vectors; number of frames depends on length of piece

Need summary/aggregation/modeling of this set

- Average over all frames? Sum?

Most common approach: Statistically model the distribution of all these local features

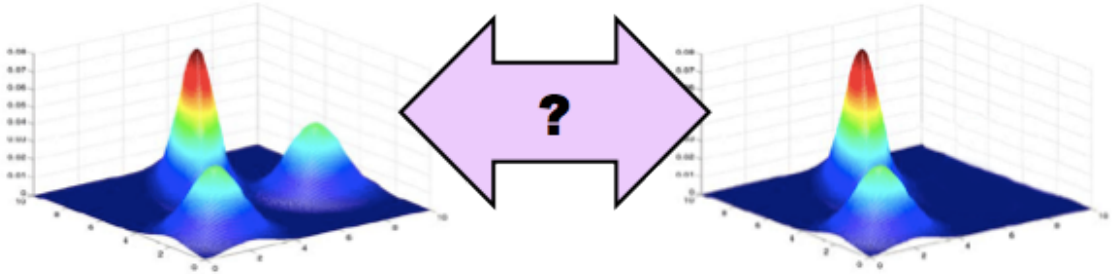
- memory requirements, runtime and also the recommendation quality depend on this step

Learn model that explains the data best

- State-of-the-art until 2005: learn a Gaussian Mixture Model (GMM)
- a GMM estimates a probability density as the weighted sum of M simpler Gaussian densities, called components of the mixture
- each song is modeled with a GMM
- the parameters of the GMM are learned with the classic Expectation-Maximization (EM) algorithm
  - this can be considered a shortcoming of this approach as this step is very time consuming

“Bag-of-frames” Modeling

Comparing two GMMs is non-trivial and expensive



- The Kullback-Leibler divergence can be used (approximated)

$$D_{KL}(P||Q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} \, dx$$

- Basically, this requires to (Monte-Carlo) sample one GMM and calculate the likelihood of these observations under the other model and vice versa (non-deterministic, slow)

State-of-the-Art since 2005: Single Gaussian Model

Single Gaussian "Bag-of-frames" model

Describe the frames using the mean vector and a full covariance matrix

For single Gaussian distributions, a closed form of the KL-divergence exists (not a metric!)

$$D_{\mathrm{KL}}(\mathcal{N}_0 \| \mathcal{N}_1) = \frac{1}{2}\left(\mathrm{tr}\left(\Sigma_1^{-1}\Sigma_0\right) + (\mu_1 - \mu_0)^\top \Sigma_1^{-1}(\mu_1 - \mu_0) - \ln\left(\frac{\det \Sigma_0}{\det \Sigma_1}\right) - k \right).$$

- μ ... mean, Σ ... cov. mat., *tr* ... trace, k ... dimensionality
- asymmetric, symmetrize by averaging

Alternatively, calculate Jenson-Shannon Divergence

$$JSD(P \parallel Q) = \frac{1}{2}D(P \parallel M) + \frac{1}{2}D(Q \parallel M) \qquad M = \frac{1}{2}(P + Q) \quad (\mathrm{D} = \mathrm{D}_{\mathrm{KL}})$$

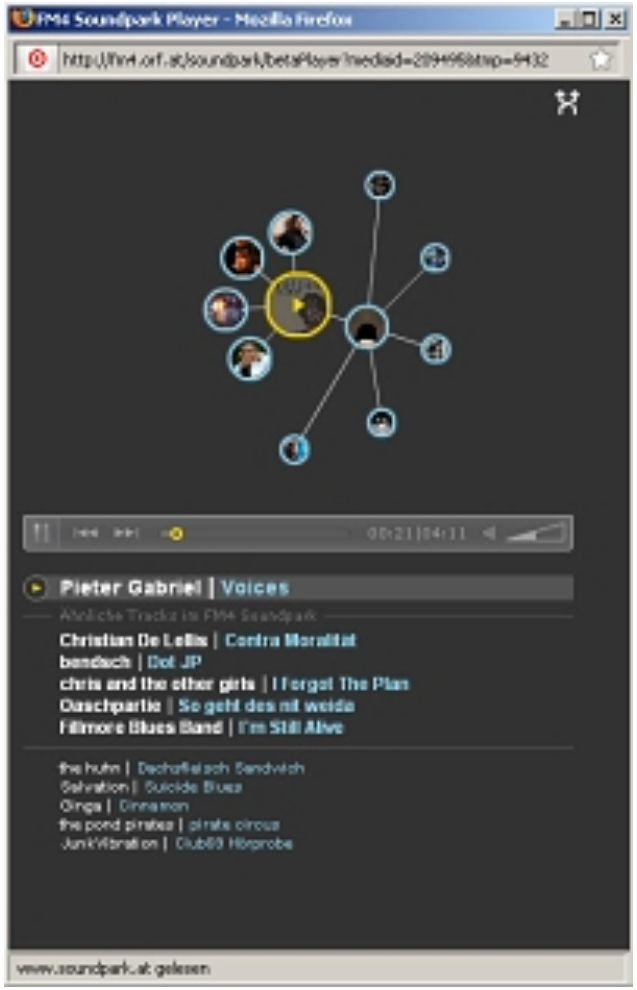
- symmetric, square root is a metric!

Efficient (instantaneous retrieval of 10Ks of pieces)

Department of Computational Perception

Query-by-Example in the Real World

- Single Gaussian MFCC music similarity measure used in
  FM4 Soundpark Player
- For each played song, 5 similar sounding songs are recommended
- Retrieval in real-time
  - full database ~20K songs (?)
  - played song model compared to all whenever played
  - no caching necessary



http://fm4.orf.at/soundpark/

Department of Computational Perception

Limitations of Bag-of-Frames Approaches

Loss of Temporal Information:

- temporal ordering of the MFCC vectors is completely lost because of the distribution model (bag-of-frames)
- possible approach: calculate delta-MFCCs to preserve difference between subsequent frames

Hub Problem ("Always Similar Problem")

- depending on the used features and similarity measure, some songs will yield high similarities with many other songs without actually sounding similar (requires post-processing to prevent, e.g., recommendation for too many songs)
- general problem in high-dimensional feature spaces

Wrapping up MFCCs and BoF

Similarity model applicable to real-world tasks

Satisfactory results ("world's best similarity measure" for several years)

Extensions make it applicable to search within millions of songs in real-time

- approximate searching in lower-dimensional projection

Possible Alternatives to BoF:

- Hidden Markov Models
- Vector Quantization Models ("Codebook")
- …

Block-Level Features

Instead of processing single frames, compute features on larger blocks of frames

- blocks are defined as consecutive sequences of audio frames
- thus features are (to some extent) able to capture local temporal information

Afterwards the blocks are summarized to form a generalized description of the piece of music

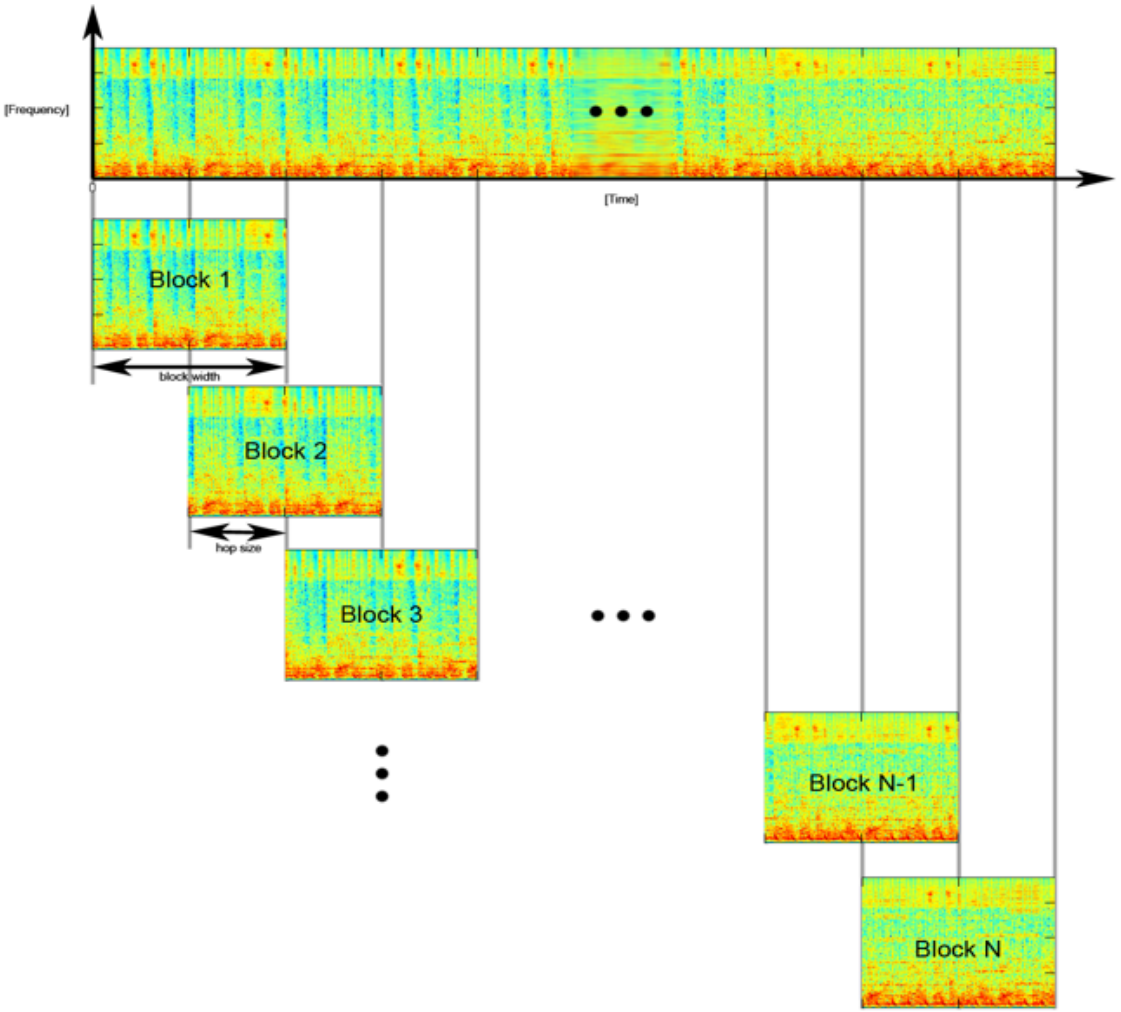
Features considered in the following:

- Fluctuation Patterns (Pampalk; 2001)

From Block Level Framework (BLF) (Seyerlehner; 2010)

- Correlation Pattern
- Spectral Contrast Pattern

Block Processing



$$\text{block} = \begin{bmatrix} b_{H,1} & \cdots & b_{H,W} \\ \vdots & \ddots & \vdots \\ b_{1,1} & \cdots & b_{1,W} \end{bmatrix}$$

- The whole spectrum is processed in terms of blocks
- Each block consists of a fixed number of frames (block size W)
- Number of rows H is defined by the frequency resolution
- Blocks may overlap (hop size)
- Main advantage of processing in blocks:
  - blocks allow to perform some (local) temporal processing

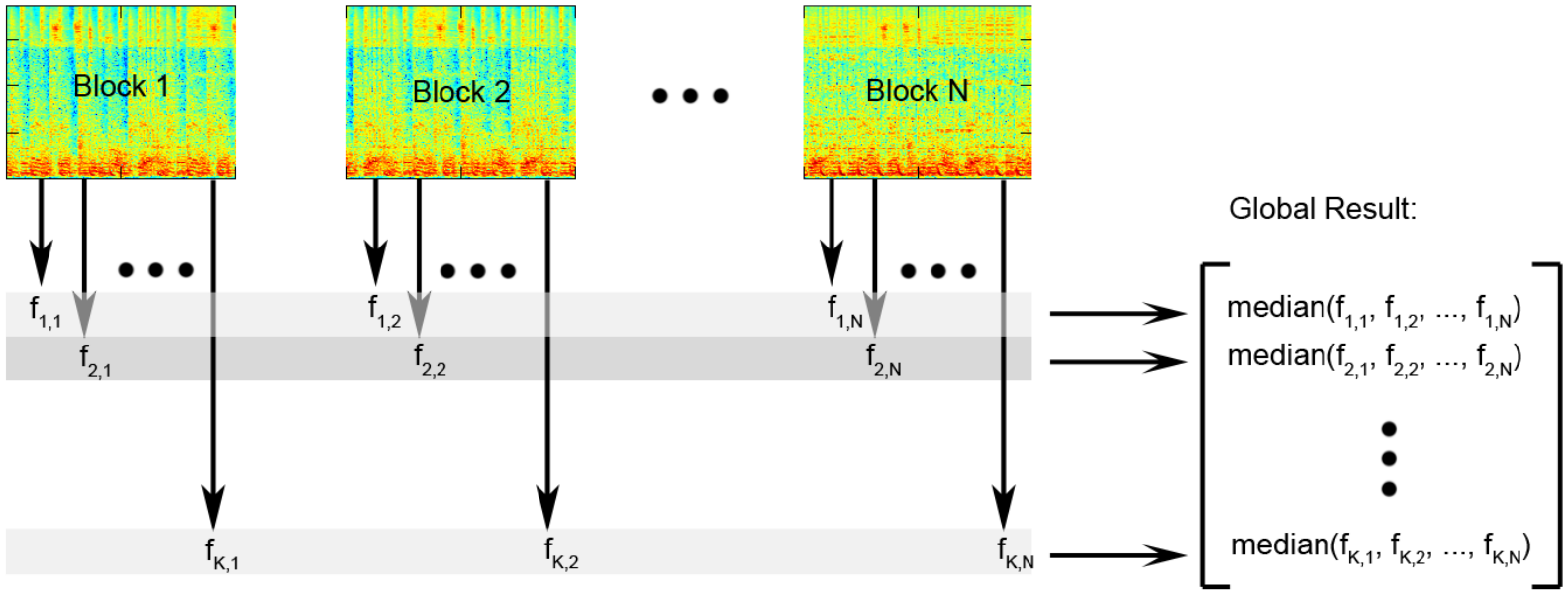
Department of Computational Perception

Generalization

To come up with a global feature vector per song, the local feature vectors must be combined into a single representation

This is done by a summarization function (e.g., mean, median, certain percentiles, variance, …)

The features in the upcoming slides will be matrices, however in these cases the summarization function simply is applied component by component



Fluctuation Patterns (FPs)

Idea: measure how strong and fast beats are played within certain perceptually adjusted frequency bands

Aims at capturing periodicities in the signal (“rhythmic properties“)

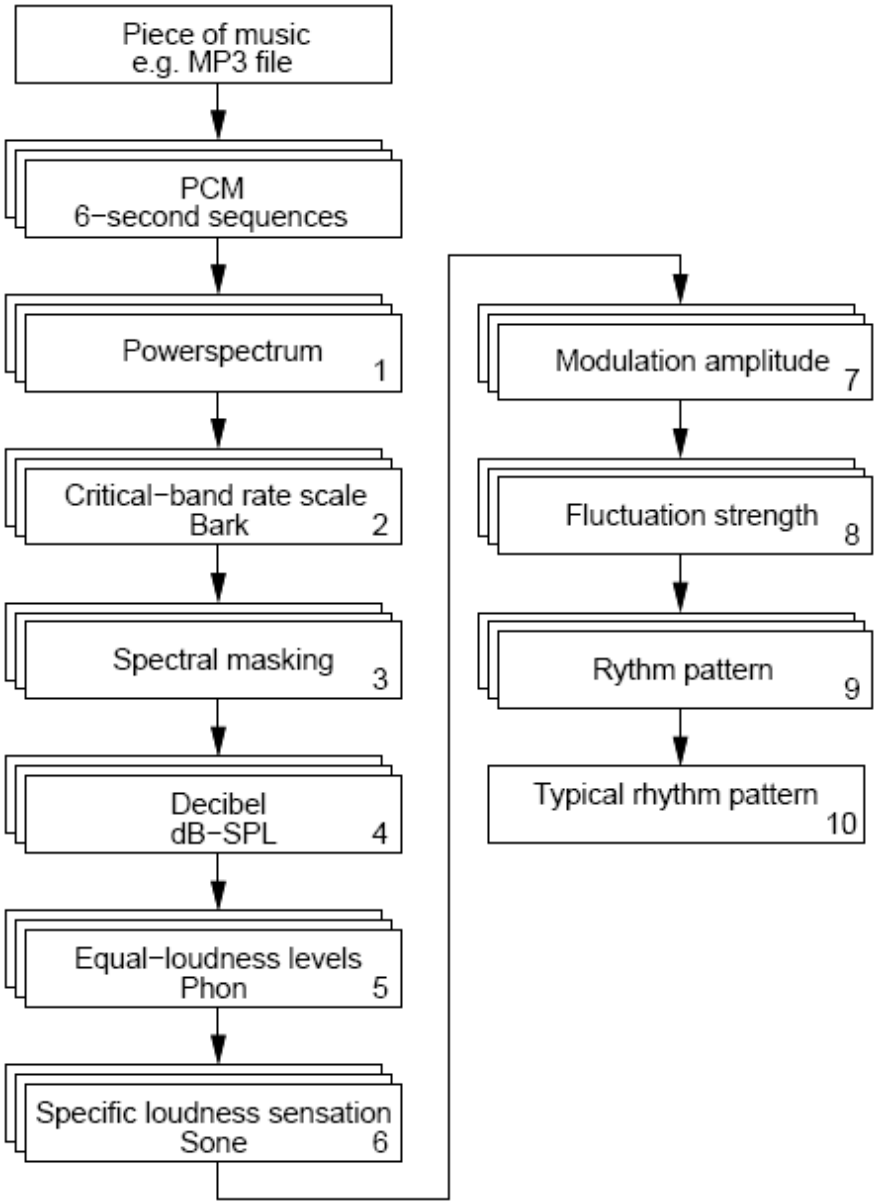
Incorporates several psychoacoustic transformations

- Logarithmic perception of frequencies (Bark scale)
- Loudness
- Periodicities

Results in a vector description for each music piece

- Vector Space Model
- Favorable for subsequent processing steps and applications: classification, clustering, etc.

Fluctuation Patterns



Extract 6 sec blocks

- discard beginning and end

In each block:

FFT on Hanning-windowed frames (256 samples)

Convert spectrum to **20 critical bands** according to *Bark scale*

Calculate Spectral Masking effects

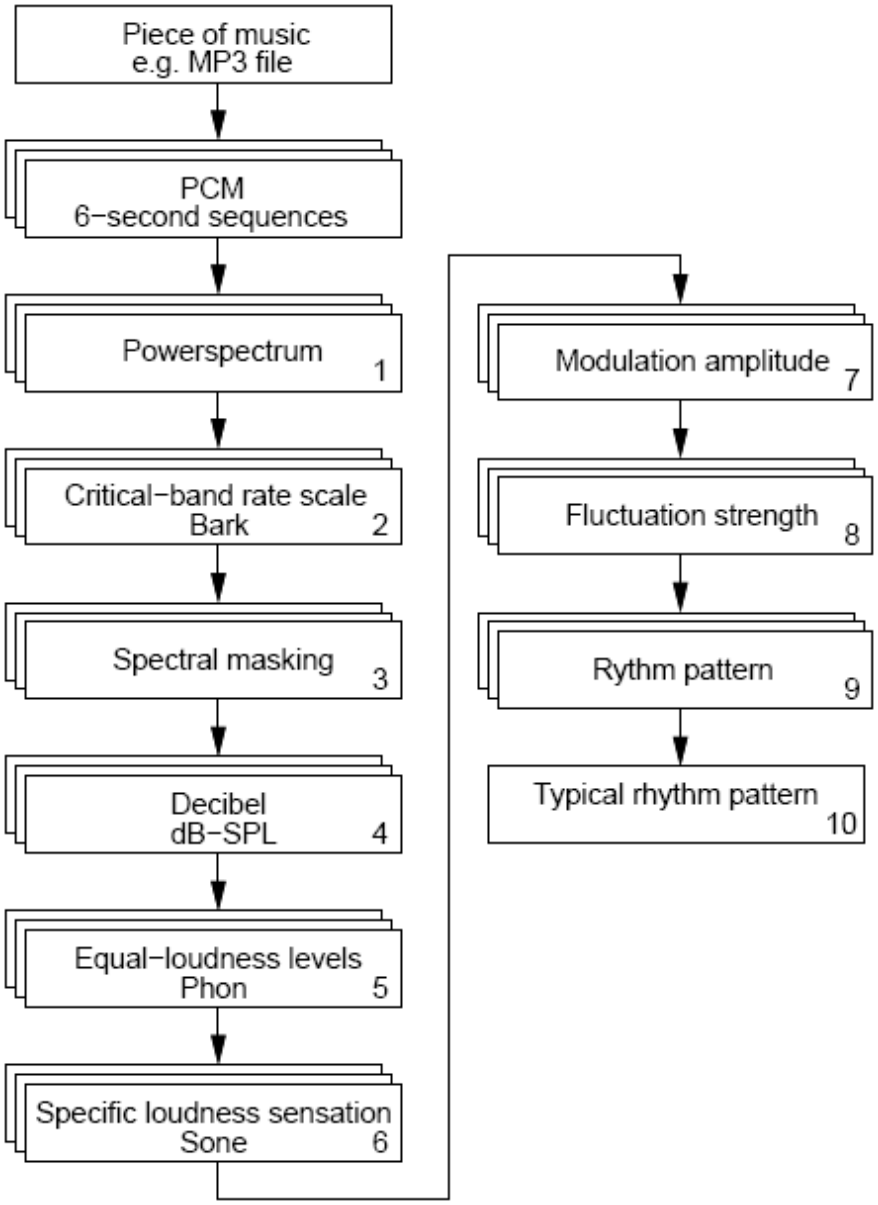
- (i.e. occlusion of a quiet sound when a loud sound is played simultaneously)

Several loudness transformations:

1. to dB (sound intensity)
2. to phon (human sensation: log)
3. to sone (back to linear)

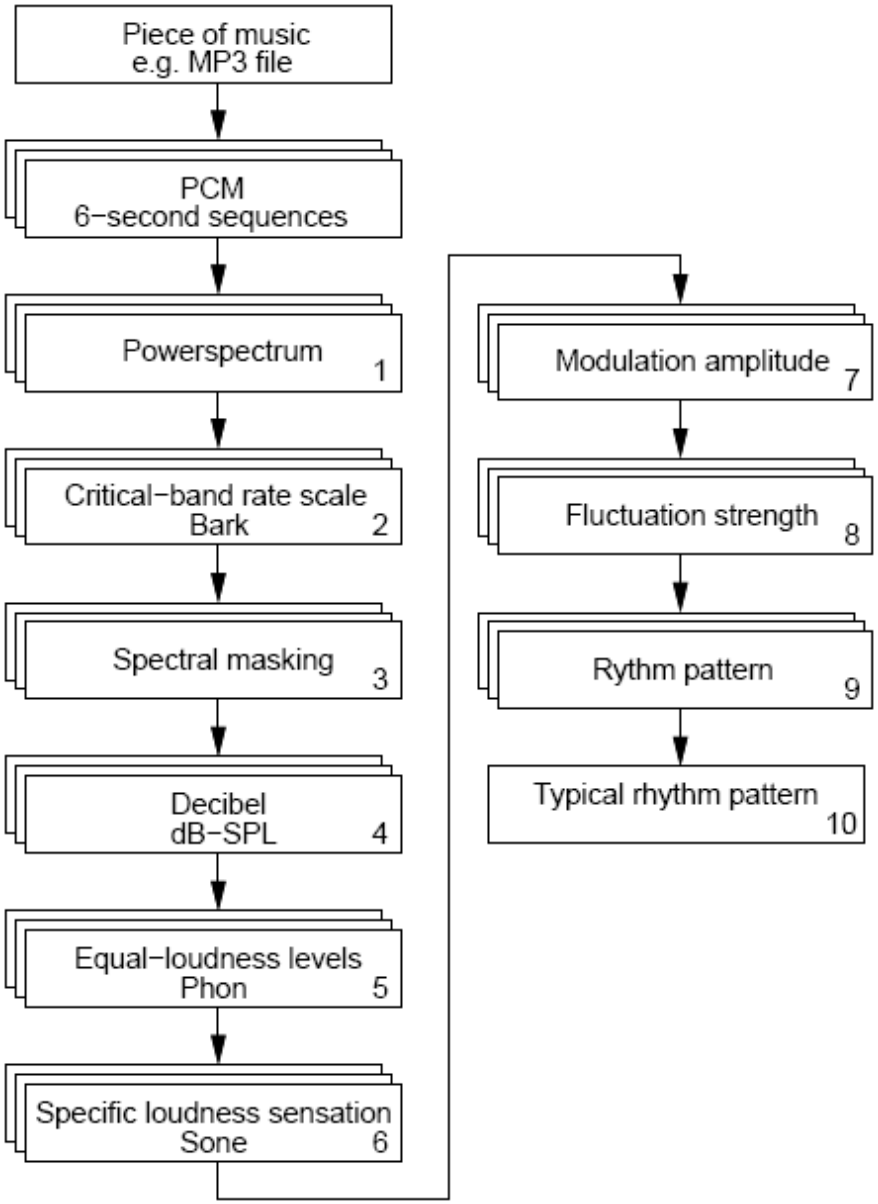
Department of Computational Perception

Fluctuation Patterns



- **Second FFT** reveals information about amplitude modulation, called *fluctuations*.
  - Fluctuations show how often frequencies reoccur at certain intervals within the 6-sec-segment
  - "frequencies of the frequencies"
- Psychoacoustic model of fluctuation strength
  - perception of fluctuations depends on their periodicities
  - reoccurring beats at 4Hz perceived most intensely
  - 60 levels of modulation (per band) (ranging from 0 to 600bpm)
- Emphasize distinctive beats

Fluctuation Patterns



Each block is now respresented as a matrix of fluctuation strengths with 1,200 entries (20 critical bands x 60 levels of modulation)

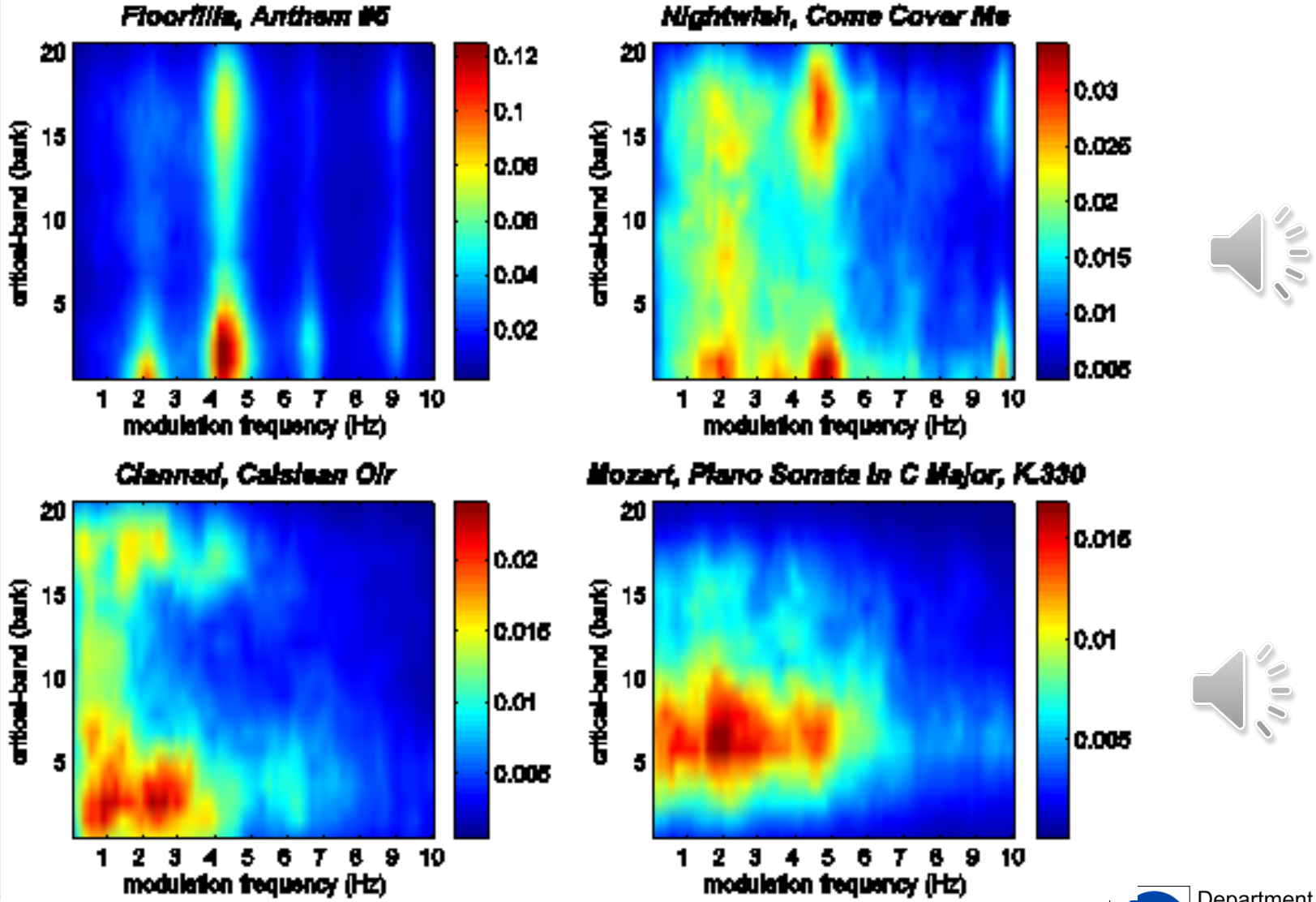
Aggregation of all blocks by taking *median* of each component

This results in a **1,200 dimensional feature vector** for each music piece

Comparison of two music pieces is done by calculating the *Euclidean distance* between their feature vectors

Department of Computational Perception

Examples



Department of Computational Perception

Wrapping up FPs and VSM

(Some) temporal dependencies are modeled within segments of 6 second length

Properties:

- \+ Vector Space Model: The whole mathematical toolbox of vector spaces is available.
- \+ easy to use in classification
- \+ song models can be visualized
- − high dimensional feature space (often a PCA is applied to reduce dim.)

More comprehensive block-level features by (Seyerlehner; 2010) currently best performing similarity measure according to MIREX:

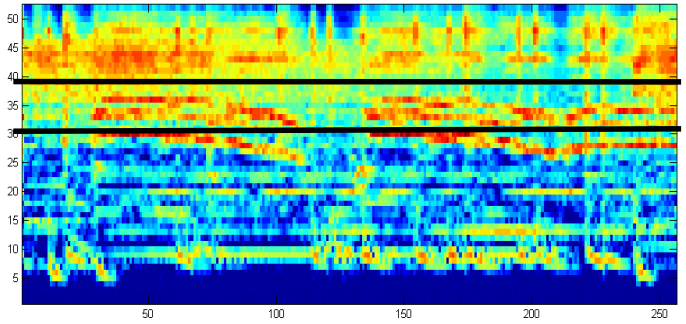
- Spectral Pattern (SP): frequency content
- Delta-Spectral Pattern (DSP): SP on delta frames
- Variance Delta-Spectral Pattern (VDSP): *variance* used to aggregate DSP
- Logarithmic Fluctuation Pattern (LFP): more tempo invariant
- **Correlation Pattern (CP)**: temporal relation of frequency bands
- **Spectral Contrast Pattern (SCP)**: estimate “tone-ness”
- Block **aggregation** via *percentiles*; **similarity** via *Manhattan distance*

Correlation Pattern (CP)

- Reduce the Cent spectrum to 52 frequency bands
- Captures the temporal relation of the frequency bands
- Compute the pairwise linear correlation between each frequency band.

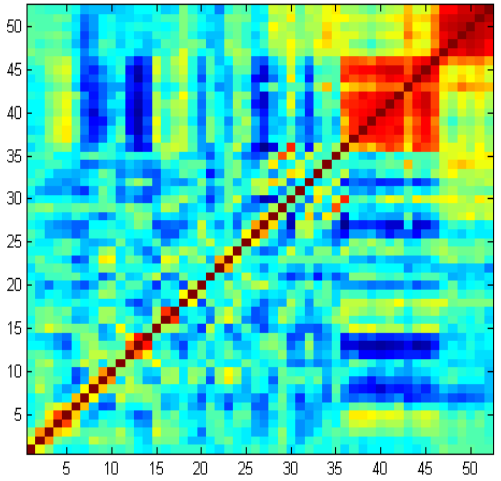
$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{(n-1)s_x s_y}$$

- The 0.5-percentile is used as aggregation function.



r_{xy}

CP

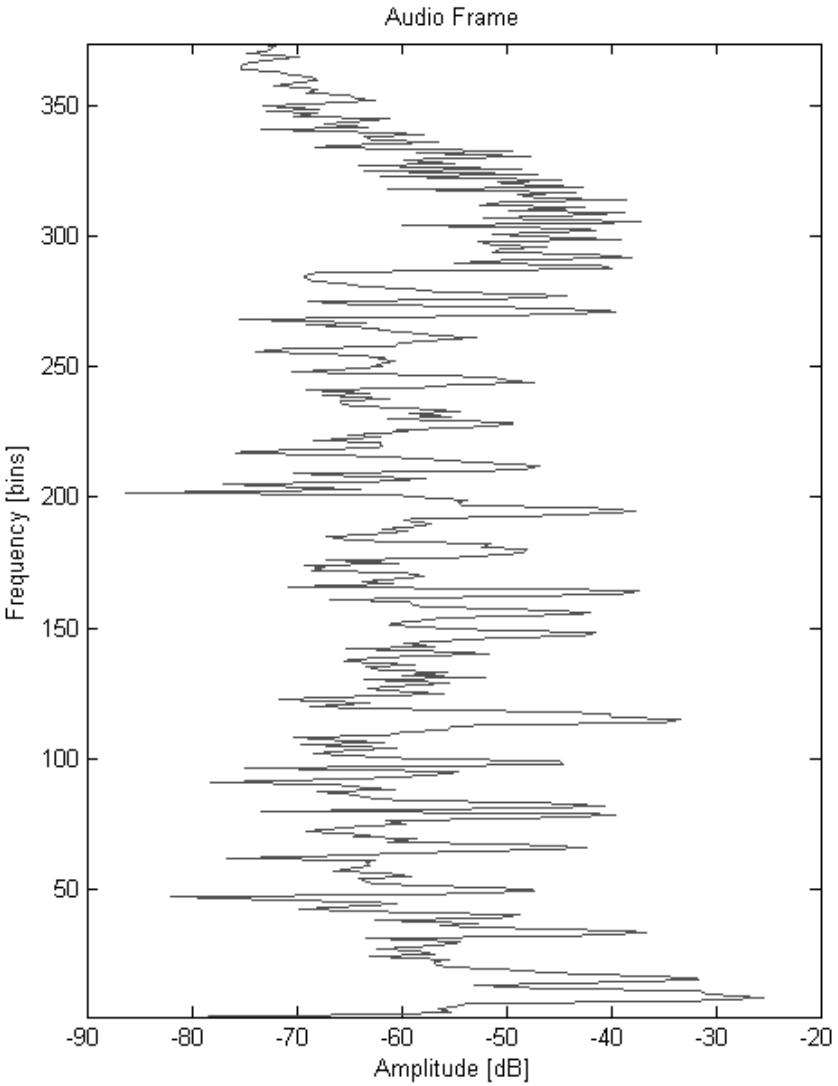


for all pairs

Department of Computational Perception

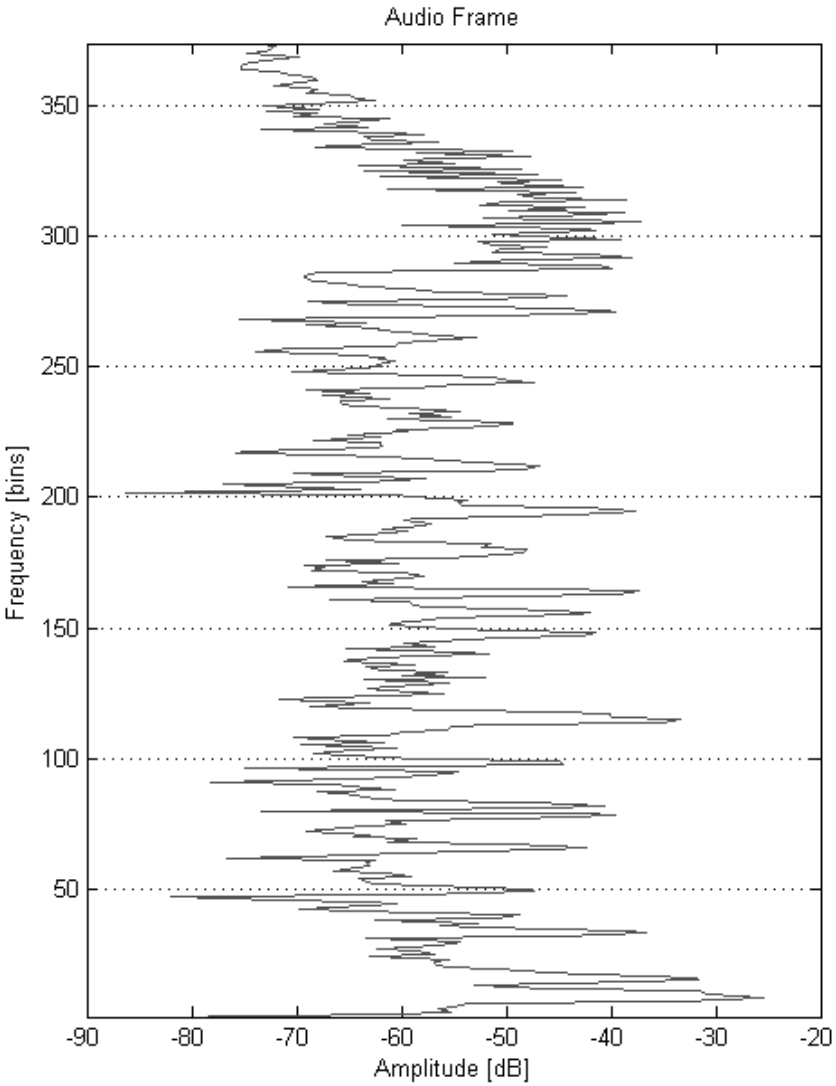
Spectral Contrast Pattern (SCP)

- Compute the **spectral contrast** per frame to estimate the *"tone-ness"*
- This is performed separately for 20 frequency bands of the Cent spectrum.
- Sort the spectral contrast values of each frequency band along the whole block.
- The aggregation function is the 0.1-percentile.



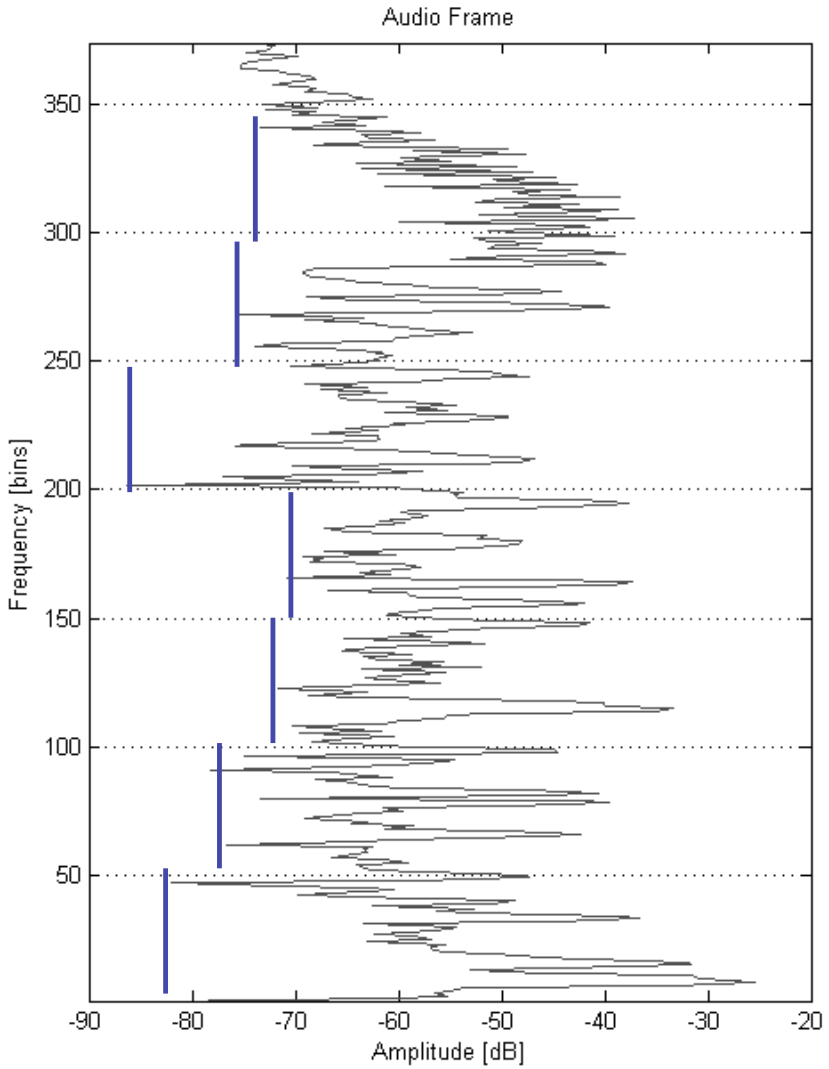
Spectral Contrast Pattern (SCP)

- Compute the **spectral contrast** per frame to estimate the *"tone-ness"*
- This is performed separately for 20 frequency bands of the Cent spectrum.
- Sort the spectral contrast values of each frequency band along the whole block.
- The aggregation function is the 0.1-percentile.



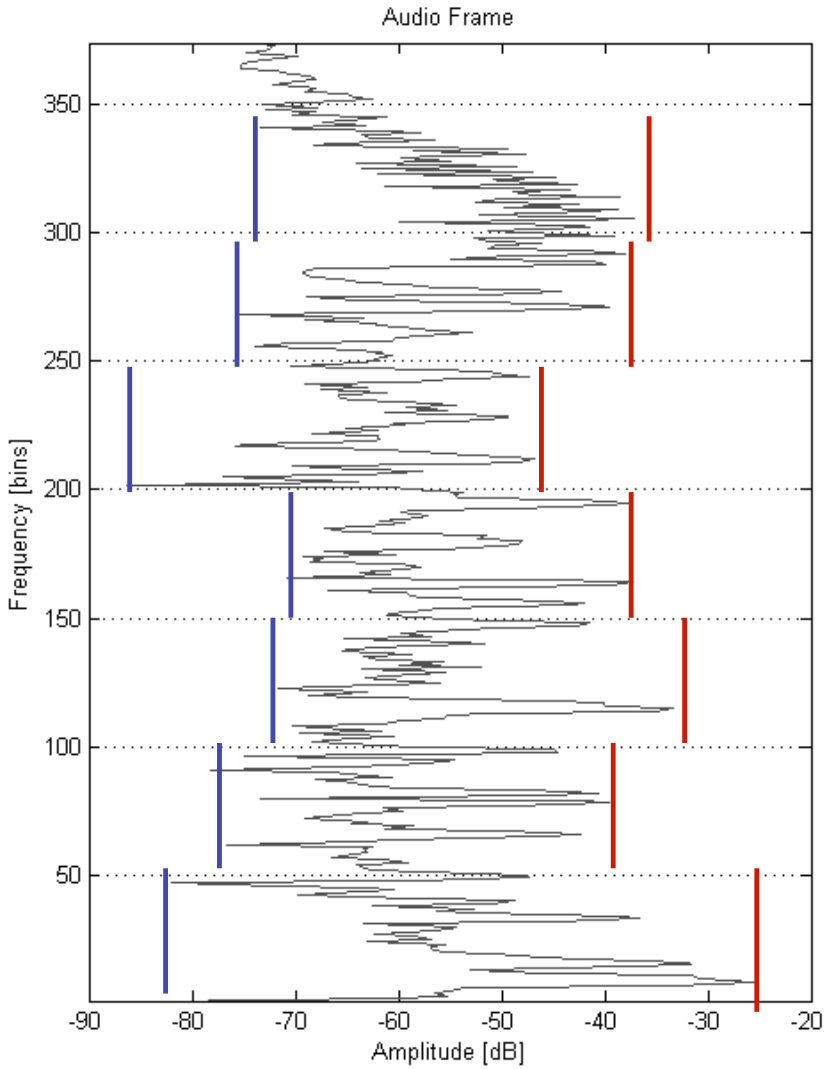
Spectral Contrast Pattern (SCP)

- Compute the **spectral contrast** per frame to estimate the *"tone-ness"*
- This is performed separately for 20 frequency bands of the Cent spectrum.
- Sort the spectral contrast values of each frequency band along the whole block.
- The aggregation function is the 0.1-percentile.



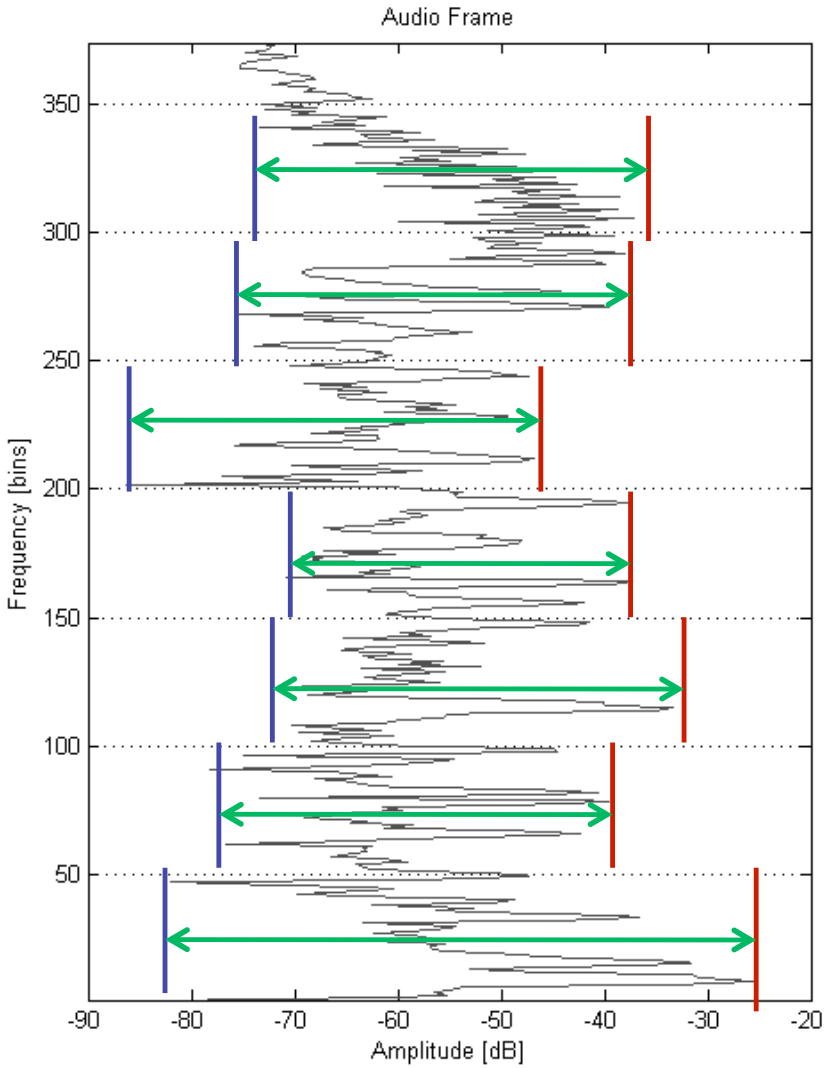
Spectral Contrast Pattern (SCP)

- Compute the **spectral contrast** per frame to estimate the *"tone-ness"*
- This is performed separately for 20 frequency bands of the Cent spectrum.
- Sort the spectral contrast values of each frequency band along the whole block.
- The aggregation function is the 0.1-percentile.



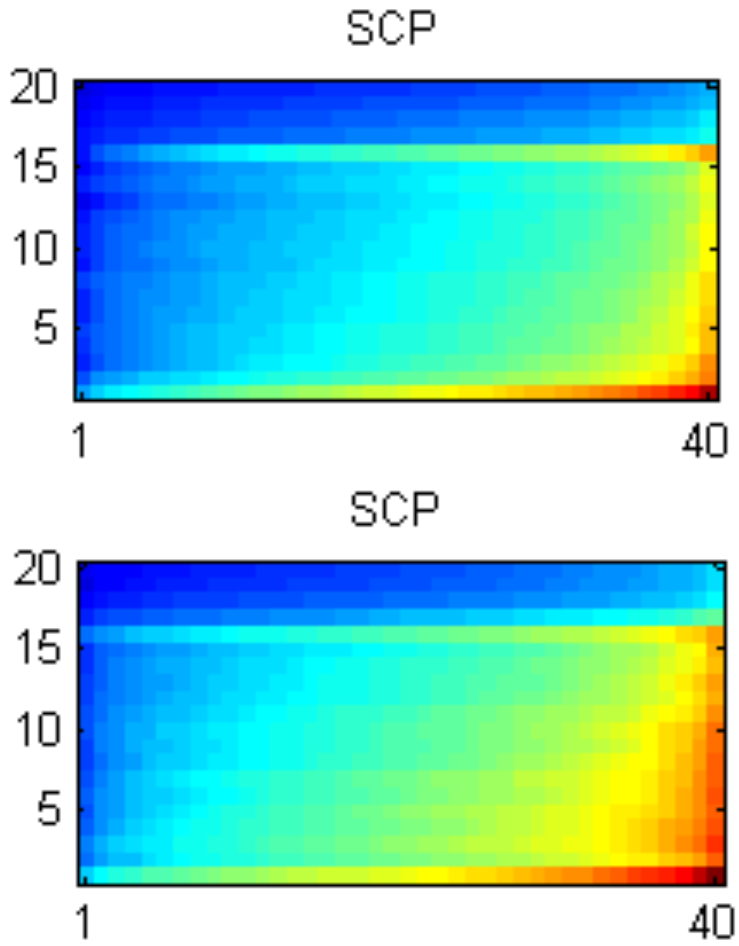
Spectral Contrast Pattern (SCP)

- Compute the **spectral contrast** per frame to estimate the *"tone-ness"*
- This is performed separately for 20 frequency bands of the Cent spectrum.
- Sort the spectral contrast values of each frequency band along the whole block.
- The aggregation function is the 0.1-percentile.



Spectral Contrast Pattern (SCP)

- Compute the **spectral contrast** per frame to estimate the *"tone-ness"*
- This is performed separately for 20 frequency bands of the Cent spectrum.
- Sort the spectral contrast values of each frequency band along the whole block.
- The aggregation function is the 0.1-percentile.



Defining Similarity in the BLF

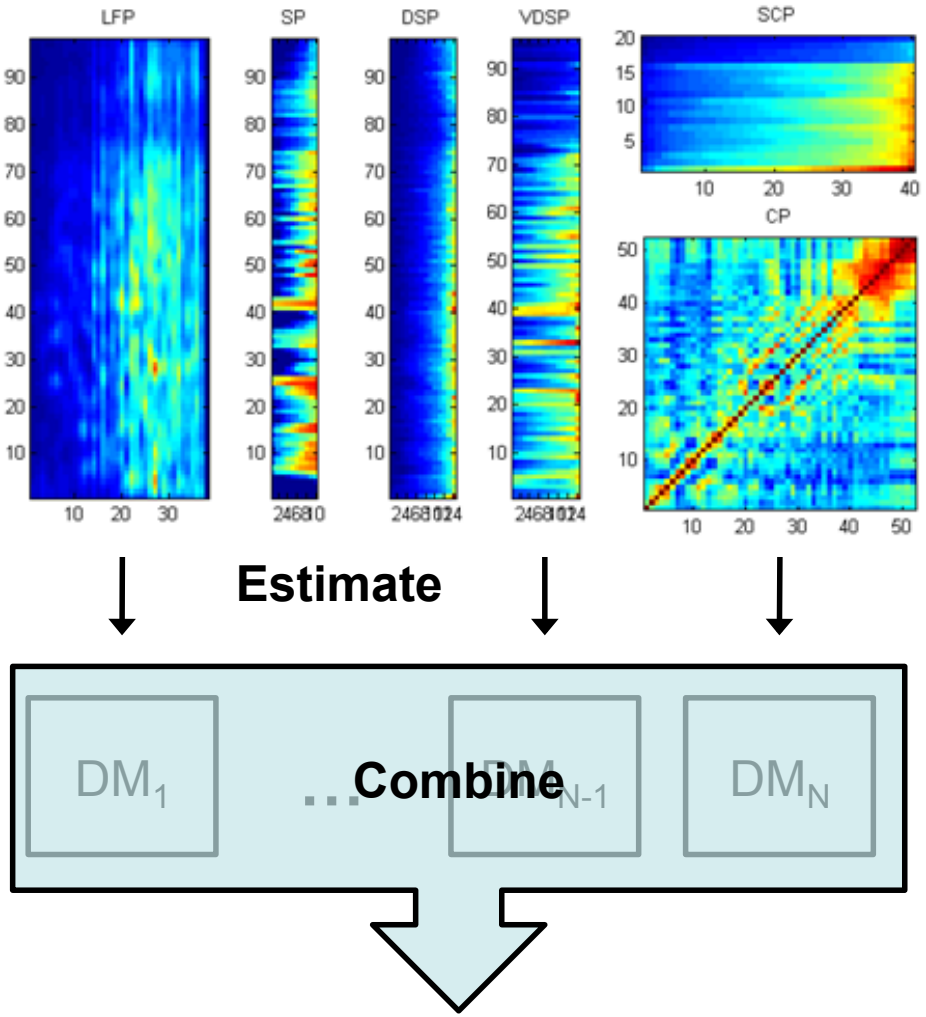
Estimate song similarities for multiple block-level features

- Calculate song similarities separately for each pattern (by computing **Manhattan distance**)
- **Fusion:** Combine the similarity estimates of the individual patterns into a single result

Naïve approach: linearly weighted combination of BLFs

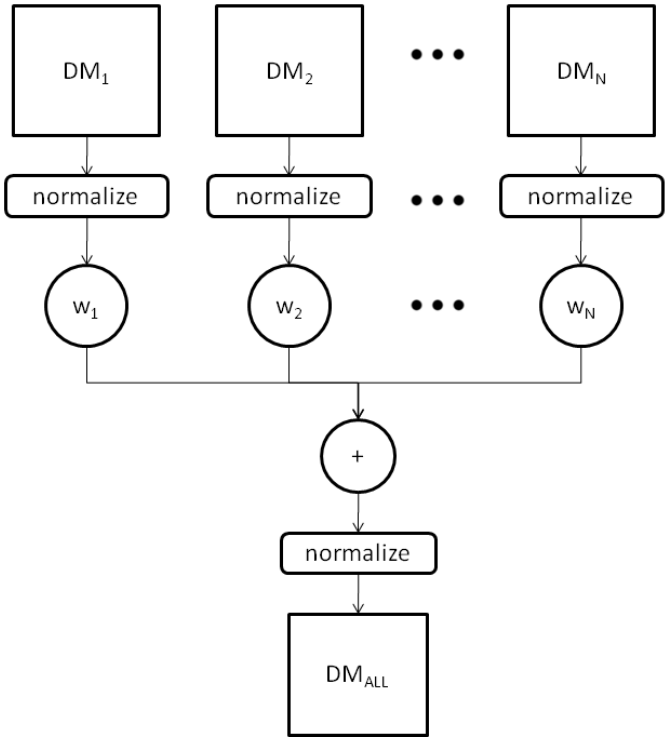
Problem: similarity estimates of the different patterns (block-level features) have different scales.

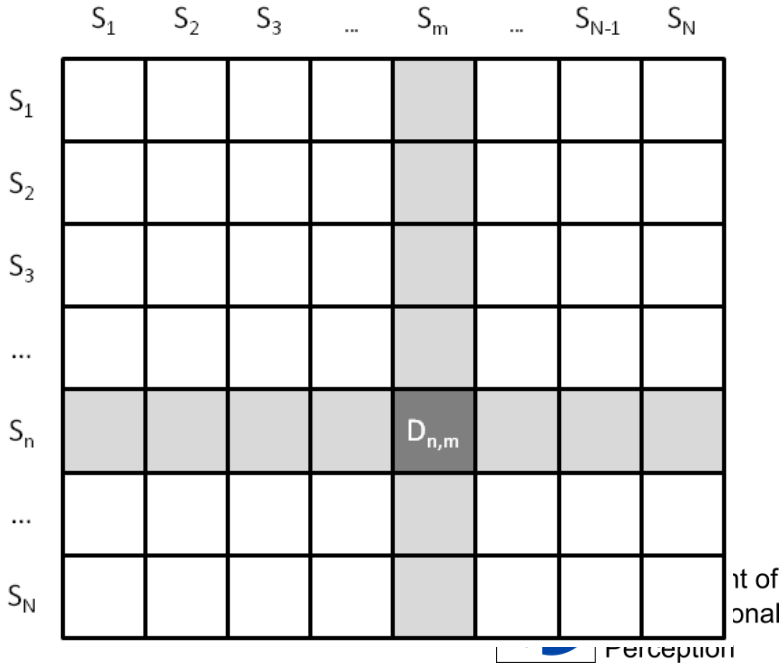
→ special normalization strategy is used: **Distance Space Normalization**



Distance Space Normalization (DSN)

- Operates on the distance matrix
- Each distance $D_{n,m}$ is normalized using Gaussian normalization.
- Mean and standard deviation are computed over both **column** and **row** of the distance matrix.
- Each distance has its own normalization parameters.
- **Observation:** The operation itself can improve the nearest neighbor classification accuracy.





Demo: Content-Based Music Browsing

Department of
Computational
Perception

nepTune – Structuring the Music Space

(Knees et al.; MM 2006)

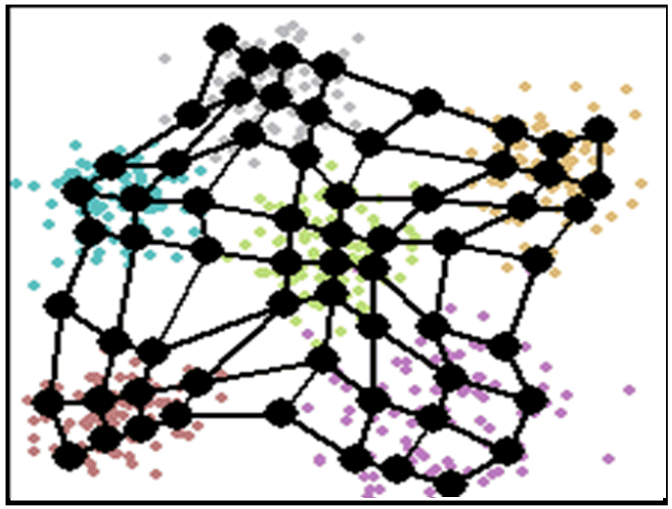
Clustering of music pieces

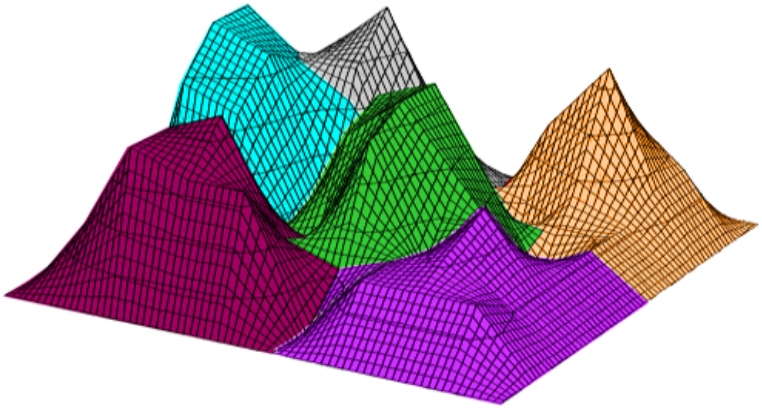
Each song corresponds to point in feature (similarity) space

Self-organizing Map

High-dimensional data (content-based features) is projected to 2-dim. plane

Number of pieces per cluster
→ landscape height profile





nepTune – Web-based Augmentation

(Knees et al.; MM 2006)

Automatic description of landscape via Web term extraction

artist names (ID3)

Music dictionary

$$G_{w,j} = \frac{F_{w,j}^2}{\sum_i F_{w,i}}$$

Term goodness

Google Images

Department of Computational Perception