ECS7006 Music Informatics

Week 8 – Musical Structure and Source Separation

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

prepared by Simon Dixon
using material by Juan Bello, Emmanuel Vincent and Meinard Müller

`s.e.dixon@qmul.ac.uk`

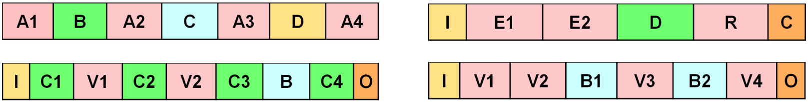
2023

Musical Structure and Form

- The goal of structure analysis is to identify structural elements and segment music into its constituent temporal parts
- Musical structure has multiple aspects: rhythmic, harmonic, melodic, and form
- Structure can be viewed at various levels of granularity: e.g. notes phrases, sections, movements and works
- Musical form arises from repetition or similarity of the segments of a piece
- Form can be observed in the high-level aspects of music (e.g. **repetition** in rhythm, melody, harmony or timbre), or in the **similarity** of lower-level features

Common Musical Forms

- **Binary form**: two contrasting segments (A and B) with optional repetition of each segment: AB or AAB or ABB or AABB
- **Ternary form**: two contrasting segments followed by a return to the first segment: e.g. ABA or AABA
- **Rondo form**: several contrasting segments alternating with a repeating segment: e.g. ABACADA
- **Chain form**: a sequence of unrelated segments: e.g. ABCDE
- **Strophic form**: a single section is repeated several times, usually with variation (e.g. verses of a song, theme & variations): A A′ A″ A‴
- For pop music, the structure can be expressed in terms of units such as *verse*, *chorus*, *introduction* and *bridge*



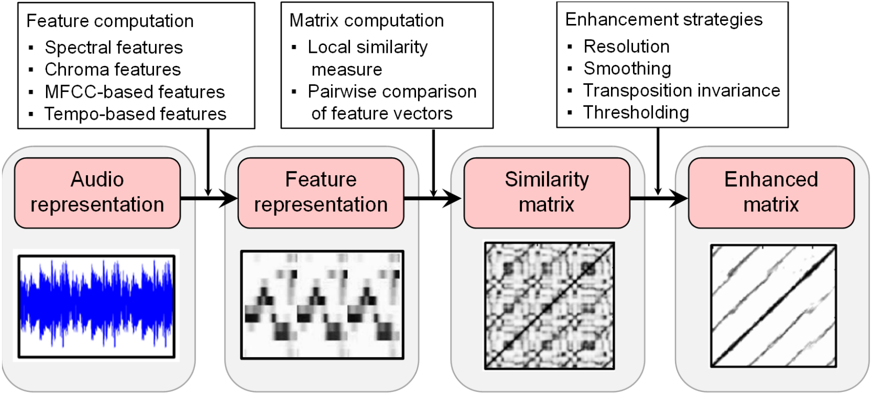
Hierarchical Form

- Form is (approximately) hierarchical: pieces contain sections which contain phrases (at multiple levels)
- e.g. **Sonata form** in Western classical music
  - Has three main sections: the *exposition*, where two contrasting themes are introduced, the *development*, based on thematic material from the exposition, and the *recapitulation*, where the two themes appear again
  - The exposition may be repeated
  - The parts may be embedded between an introduction and coda
  - A complex type of ternary form
  - Sonata form is often used for the first movement of a larger piece such as a symphony
- As in the case of metre, a precise definition of the structural level of interest is difficult

Segmentation and Structure

- **Segmentation** is the process of breaking down continuous media into discrete units
  - e.g. segmenting images into objects, video into scenes, or audio into speech, music, silence and applause
  - Involves finding boundaries between regions of high self-similarity (e.g. in texture, colour, or instrumentation)
- **Structure analysis** involves assigning meaning to segments (labelling)
  - What the segment represents (absolute, e.g. drum solo)
  - How the segment relates to other segments (relative, e.g. same as first segment)
- Can be based on repetition/similarity or homogeneity/novelty

Structure Analysis Pipeline



FMP Fig.4.9

Feature Extraction

- Audio waveforms and spectrograms are too specific for capturing repetition or similarity
- First step is to compute features
- What types of features can be used?
  - Pitch content (harmony, melody) is approximated by chroma features
  - Timbre can be appproximated by MFCCs
  - Tempo can be factored out by using *beat-synchronous* features
  - Temporal features (tempograms, Sec. 6.2.4 of text book) can also be used
- The parameters used for feature extraction (e.g. temporal resolution) greatly influence results
- Optimal parameter settings depend on the music being analysed

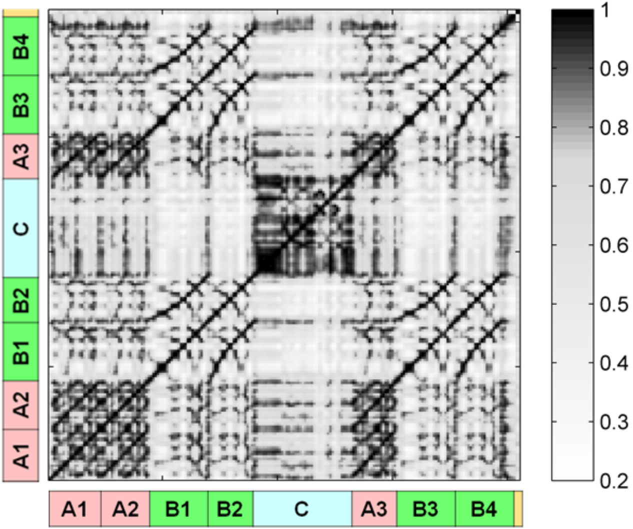
Self-Similarity Matrices

- The second step is to compute a *self-similarity matrix* (SSM), which stores the similarity of every pair of audio frames, based on the chosen features
- For the similarity function $s : \mathcal{F} \times \mathcal{F} \to \mathbb{R}$, where $\mathcal{F} = \mathbb{R}^D$ is the feature space, we use a normalised inner product:

$$s(x_n, x_m) = \frac{| < x_n, x_m > |}{(< x_n, x_n >< x_m, x_m >)^{0.5}}$$

- Results in matrix $S(n, m) = s(x_n, x_m) \in [0, 1]$
- Repetition of a segment results in a *path* of high values in S
- Homogeneous regions result in *blocks* of high values in S

Self-Similarity Matrix with Chroma Features: Paths



Processing Self-Similarity Matrices: Paths

- Given a feature sequence $X = (x_1, x_2, ..., x_N)$ and similarity matrix $S \in [0,1]^{N \times N}$, a *path* $P = ((n_1, m_1), ..., (n_L, m_L))$ with:

$$(n_l, m_l) \in [1 : N]^2$$

$$(n_{l+1}, m_{l+1}) - (n_l, m_l) \in A$$

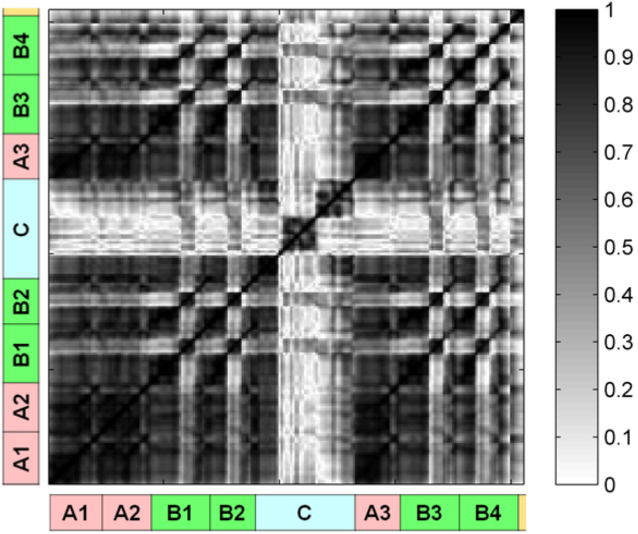
  defines a sequence of cells in S indexed by each (n_l, m_l)

- A defines the admissable step sizes, e.g. $A = \{(2,1), (1,2), (1,1)\}$
- The *score* $\sigma(P)$ of a path P is defined as:

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

- Repetition of segments can be discovered by finding high scoring paths

Self-Similarity Matrix with MFCC Features: Blocks



Processing Self-Similarity Matrices: Blocks

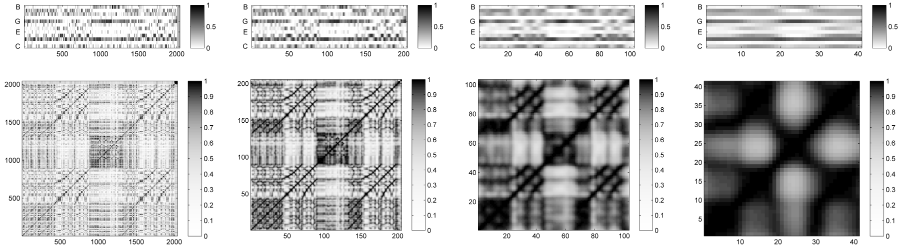
- Likewise, given two segments $\alpha_1 = [s_1 : t_1]$ and $\alpha_2 = [s_2 : t_2]$, a *block* $B = \alpha_1 \times \alpha_2$ defines a region of cells: $\{(n_l, m_l) : n_l \in \alpha_1 \text{ and } m_l \in \alpha_2\}$ in the similarity matrix S
- The score $\sigma(B)$ of a block $B = \alpha_1 \times \alpha_2$ is defined as:

$$\sigma(B) = \sum_{n \in \alpha_1, m \in \alpha_2} S(n, m)$$

- Homogeneous segments can be discovered by finding high scoring blocks

Processing Self-Similarity Matrices: Enhancement

- Path and block structures tend to be fragile due to the many types of variation that occur in music performance
- We will look at several approaches to address this problem
- Preprocessing: *feature enhancement*
  - Smoothing (e.g. median filtering) and downsampling
  - Enhance block structures at the cost of fine-grained path structures
  - Computational savings (time and space) for subsequent processing

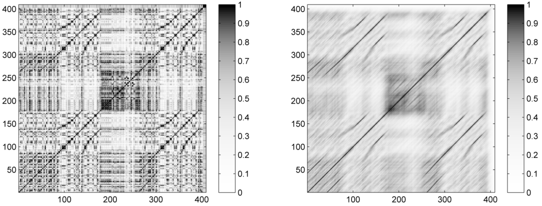


Processing Self-Similarity Matrices: Enhancement

- The second enhancement method is *path smoothing*
- The idea is to perform low-pass filtering in the direction of the diagonal

$$S_L(n,m) = \frac{1}{L}\sum_{l=1}^{L} S(n+l, m+l)$$

- Only works for fixed tempo



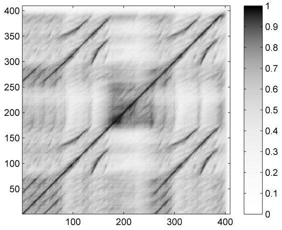
Original　　Smoothed

Processing Self-Similarity Matrices: Enhancement

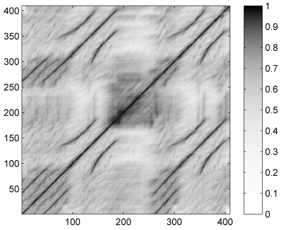
- Can add a parameter θ for tempo ratio and use:

$$S'(n,m) = \max_{\theta} \sum_{l=1}^{L} S(n+l, m+\theta l)$$

- Filter in both directions (forward and backward) for best results
- Diagonal median filtering should also work well



Tempo-invariant



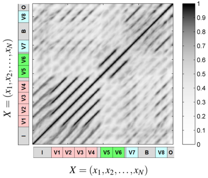
Bi-directional

- If it is desired that transposed versions should be matched, chroma features can be rotated by 0 to 11 positions, and the maximum similarity over all 12 rotations is then used:

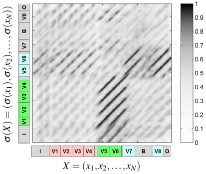
$$\rho^i(x(j)) = x((j+i) \mod 12)$$

$$\rho^i(S)(n,m) = s(\rho^i(x_n), x_m)$$

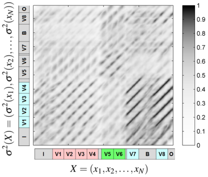
$$S^{\mathsf{TI}}(n,m) = \max_{i\in[0:11]} \rho^i(S)(n,m)$$



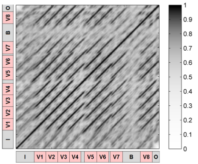
$\rho^0(S)$



$\rho^1(S)$

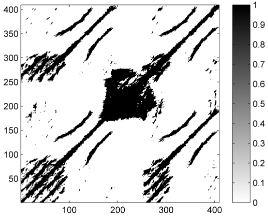


$\rho^2(S)$

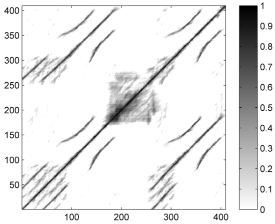


S^{TI}

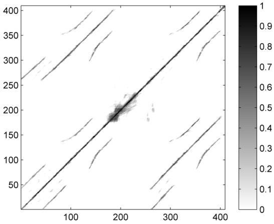
- Thesholding (with threshold τ):
  - With binarisation ($S' = (S > \tau)$) or scaling ($S' = C \times \max(S - \tau, 0)$)
  - Absolute or relative (e.g. set τ to 80th percentile)
  - Local (rowwise and columnwise) or global



Binarisation ($\tau = 0.75$)



Scaling (80th%)



Scaling (95th%)

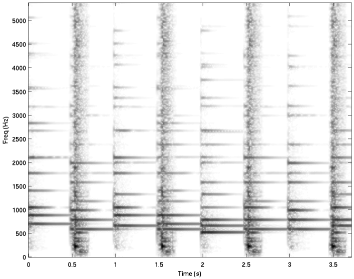
Source Separation

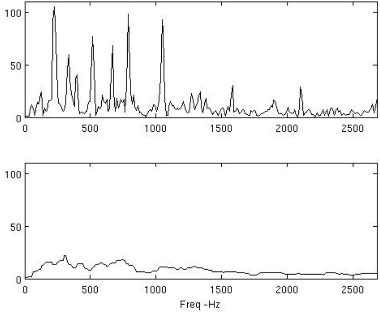
Source Separation

- Most audio signals are mixtures of several simultaneously active audio sources (voices, instruments, environmental sounds, recording noise)
- Source separation involves extracting one or several source signals in order to listen to or process them separately or remix them
- Techniques are based on properties of the sounds to be separated, or modelling and reverse engineering the mixing process
- Applications include:
  - real-time denoising for hearing aids
  - removal of vocals for karaoke
  - remixing (concerts, conferences, post-production, electronic music composition)
  - spatial enhancement (upmixing stereo to 5.1, adaptation to the listening environment)

Harmonic/Percussive Source Separation (HPSS)

- Separation of harmonic (periodic, steady-state) sounds from percussive (transient) sounds
  - Note that pitched percussive instrument sounds (e.g. piano, timpani) have both harmonic and percussive components
- Core idea: *harmonic* sounds form *horizontal structures* in a spectrogram representation (localised in frequency, spread out in time); *percussive* sounds form *vertical structures*
- Using filters to enhance one direction and suppress the other, each time-frequency point is classified as contributing either to the harmonic or percussive part of the signal
- A binary mask is used to remove time-frequency points not belonging to the desired component, and an inverse transform reconstructs the desired time-domain signal





Example: separation of piano and snare drum. Left: spectrogram; right: spectra for one time frame for the harmonic (upper) and percussive (lower) components. From: (Fitzgerald, DAFx-2010)

Filtering of Harmonic and Percussive Components

- Median filtering can be used (Fitzgerald, DAFx-2010)
- The median of a sorted list of N numbers is the middle value
  - If N is odd, it is the $\frac{N+1}{2}$-th highest value
  - If N is even, it is the average of the $\frac{N}{2}$-th and $(\frac{N}{2}+1)$-th values
- The harmonic filter F_h with width $2w+1$ is computed as:

$$F_h(X)(n,k) = \text{median}(X(n-w:n+w,k))$$

  and the percussive filter F_p of height $2h+1$ is computed as:

$$F_p(X)(n,k) = \text{median}(X(n,k-h:k+h))$$

  for a spectrogram $X(n,k)$
- Out-of-domain values are assumed to be 0

HPSS: Separation via Binary Masking

- A binary mask is a matrix containing values 0 and 1 only
- The harmonic mask M_h is given by:

$$M_h(n,k) = \begin{cases} 1, & F_h(X)(n,k) \geq F_p(X)(n,k) \\ 0, & \text{otherwise} \end{cases}$$

- The percussive mask M_p is the complement of the harmonic mask:

$$M_p(n,k) = \begin{cases} 0, & F_h(X)(n,k) \geq F_p(X)(n,k) \\ 1, & \text{otherwise} \end{cases}$$

- The masks are applied by pointwise multiplication with the original spectrogram:

$$X_h(n,k) = X(n,k)M_h(n,k)$$

$$X_p(n,k) = X(n,k)M_p(n,k)$$

- The time domain signals are computed by applying the inverse FFT, windowing and overlap-adding the frames together

Separating Instrumental Sources

- One approach to instrumental source separation is reverse engineering the mixing process
- The choice of algorithm depends firstly on the number of independent mixture channels c relative to the number of sources s:
  - If $c > s$, the system is *over-determined*
  - If $c = s$, the system is *determined*
  - If $c < s$, the system is *under-determined*
- The type of mixture also has an impact:
  - Instantaneous: trivial mixing filters (gains, no delays)
  - Anechoic: trivial mixing filters (gain and delay pairs)
  - Convolutive (or echoic): non-trivial mixing filters
  - Reverberant: mixing filters exhibit a realistic reverberation time
- The final factor is whether the mixing filters change over time

Identification and Filtering

- Source separation is often addressed as a two-part problem:
  - *identification* of the source properties (spatial directions, short-term power spectra)
  - *filtering* of the mixture channels based on the identified properties
- Identification algorithms include:
  - direction-based algorithms (ICA, DUET, ADRess)
  - transcription-based algorithms
  - hybrid algorithms
- Filtering techniques include:
  - beamforming
  - time-frequency masking

Degenerate Unmixing Estimation Technique (DUET)

- The Degenerate Unmixing Estimation Technique (DUET) is an identification algorithm for (under)-determined instantaneous stereo mixtures
- The model assumes that the sources have different spatial directions and that no more than one source is present at any given time-frequency point
- If source j only is active at (n, f), then the inter-channel intensity difference (IID):

$$\text{IID}(n, f) = 20 \log_{10} \left(\frac{|X_2(n, f)|}{|X_1(n, f)|} \right)$$

approximates the relative mixing gain, indicating the source direction

DUET

- DUET separates the sources as follows:
  - Compute IID for each time-frequency point
  - Find source positions corresponding to peaks in the IID histogram
  - Associate each time-frequency point with the closest source and perform binary masking
- The gains can be better estimated at time-frequency points with a high inter-channel coherence (correlation), which probably contain a single active source
- For convolutive mixtures, the inter-channel phase difference (IPD) also provides relevant information:

$$\text{IPD}(n, f) = \angle X_2(n, f) - \angle X_1(n, f)$$

- For a far-field microphone pair recording, where source j only is active at (n, f), then $\text{IPD}(n, f) = 2\pi f \tau_j \mod 2\pi$, where τ_j is the time delay of arrival between the two microphones

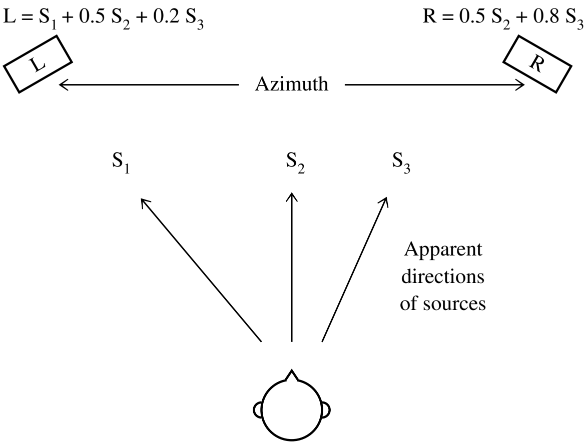
ADRess

- Azimuth Discrimination and Resynthesis (Barry et al., 2004) is a source separation method for under-determined stereo mixtures
- ADRess has a simple and efficient real-time implementation
- Assumes stereo images created with a pan-pot (phase coherence between channels)

$$x_L(t) = \sum_{j=1}^{J} P_{L,j} S_j(t) \qquad\qquad x_R(t) = \sum_{j=1}^{J} P_{R,j} S_j(t)$$

  where x_L, x_R are the mixture signals, $P_{i,j}$ the panning coefficient (gain) for channel i and source j, and S_j the unmixed source signals
- Assumes each source has a different fixed spatial location (pan setting)

ADRess Example



- ADRess uses the relative levels of a source in the two channels to locate spectral energy belonging to the source position

ADRess Algorithm

- The idea is to find the gain factor $g_j = \frac{P_{L,j}}{P_{R,j}}$ which equalises the intensity of source j in the left and right channels
- Then source j can be removed by subtraction: $x_L(t) - g_j x_R(t)$
- To avoid numerical problems, the left and right channels are interchanged if $P_{L,j} > P_{R,j}$, ensuring $g_j \leq 1$
- Apply this idea to each time-frequency bin
- To recover the sources, left and right frequency-azimuth planes A_L and A_R are computed, sampled at $2\beta + 1$ points in the azimuth dimension, representing the possible positions of instruments

$$A_L(n, k, i) = |X_L(n, k) - \frac{i}{\beta} X_R(n, k)|$$

$$A_R(n, k, i) = |X_R(n, k) - \frac{i}{\beta} X_L(n, k)|$$

where X_L and X_R are STFTs of the two channels

ADRess Algorithm (continued)

- (To simplify notation, we omit the time argument n here)
- The azimuth points corresponding to minima in A_L and A_R are singled out as sources at that position

$$A'_L(k,i) = \begin{cases} \max_i A_L(k,i) - \min_i A_L(k,i) & \text{if } i = \text{argmin}_i A_L(k,i) \\ 0 & \text{otherwise} \end{cases}$$

$$A'_R(k,i) = \begin{cases} \max_i A_R(k,i) - \min_i A_R(k,i) & \text{if } i = \text{argmin}_i A_R(k,i) \\ 0 & \text{otherwise} \end{cases}$$

- Harmonic overlap between sources causes “azimuth smearing”, i.e. the minima appear between the source positions
- To compensate, a window of width $h < \beta$ is constructed around the apparent source position d to capture harmonics that have drifted from their source

ADRess Algorithm (continued)

- Resynthesis of a single source is achieved by:
  - summing across the window:

$$Y_L(k) = \sum_{i=d-h/2}^{d+h/2} A'_L(k,i) \qquad\qquad Y_R(k) = \sum_{i=d-h/2}^{d+h/2} A'_R(k,i)$$

  - combining with the original phase values $\phi_L(k)$ and $\phi_R(k)$ and taking the IFFT
  - combining frames with a standard overlap-add scheme

Summary and Resources

- Direction-based source separation algorithms are simple and fast, but only work for multi-channel mixtures where the time-frequency overlap between sources and the reverberation are minimal
- *Informed* source separation algorithms take advantage of knowledge of the score or source instruments to achieve better separation (e.g. Open Unmix by Sony 2019; Spleeter by Deezer, 2019)
- For further reading:
  - S. Rickard, *The DUET Blind Source Separation Algorithm*, in S. Makino et al. (eds.), Blind Speech Separation, pp 217–241, 2007
  - D. Barry, B. Lawlor and E. Coyle, *Sound Source Separation: Azimuth Discrimination and Resynthesis*, 7th International Conference on Digital Audio Effects, 2004
  - F.-R. Stöter, S. Uhlich, A. Liutkus and Y. Mitsufuji, *Open-Unmix – A Reference Implementation for Music Source Separation*, Journal of Open Source Software, 2019