2023 October 23

Deep Learning for Music Analysis and Generation

MIDI Generation (II)

Generation with high-level structure
(x → MIDI)

Yi-Hsuan Yang Ph.D.
yhyangtw@ntu.edu.tw

Objectives

- Audio-domain music structure analysis and its application to MIDI generation
- Decoder-only Transformer → encoder-decoder Transformer

Outline

- **Music structure analysis**
- Objective evaluation metrics for MIDI generation
- Structured MIDI generation
- HW3

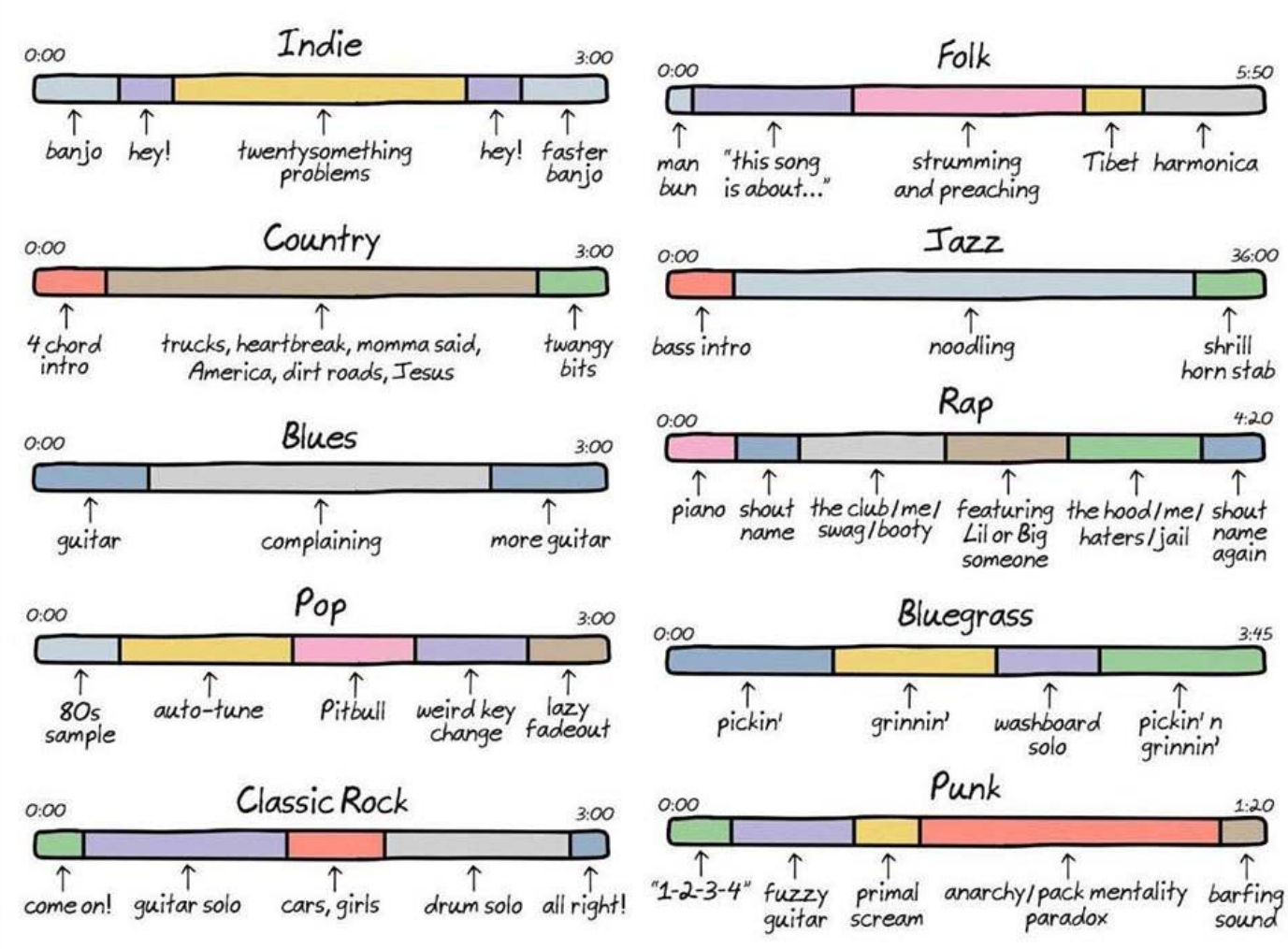
Reference: FMP Notebook

https://www.audiolabs-erlangen.de/resources/MIR/FMP/C4/C4.html

Part	Title	Notions, Techniques & Algorithms	HTML	IPYNB
B	Basics	Basic information on Python, Jupyter notebooks, Anaconda package management system, Python environments, visualizations, and other topics	[html]	[ipynb]
0	Overview	Overview of the notebooks (https://www.audiolabs-erlangen.de/FMP)	[html]	[ipynb]
1	Music Representations	Music notation, MIDI, audio signal, waveform, pitch, loudness, timbre	[html]	[ipynb]
2	Fourier Analysis of Signals	Discrete/analog signal, sinusoid, exponential, Fourier transform, Fourier representation, DFT, FFT, STFT	[html]	[ipynb]
3	Music Synchronization	Chroma feature, dynamic programming, dynamic time warping (DTW), alignment, user interface	[html]	[ipynb]

Part	Title	Notions, Techniques & Algorithms	HTML	IPYNB
4	Music Structure Analysis	Similarity matrix, repetition, thumbnail, homogeneity, novelty, evaluation, precision, recall, F-measure, visualization, scape plot	[html]	[ipynb]
5	Chord Recognition	Harmony, music theory, chords, scales, templates, hidden Markov model (HMM), evaluation	[html]	[ipynb]
6	Tempo and Beat Tracking	Onset, novelty, tempo, tempogram, beat, periodicity, Fourier analysis, autocorrelation	[html]	[ipynb]
7	Content-Based Audio Retrieval	Identification, fingerprint, indexing, inverted list, matching, version, cover song	[html]	[ipynb]
8	Musically Informed Audio Decomposition	Harmonic/percussive separation, signal reconstruction, instantaneous frequency, fundamental frequency (F0), trajectory, nonnegative matrix factorization (NMF)	[html]	[ipynb]

Music Structure / Form



(Image from the Internet)

Hungarian Dance No. 5 by Johannes Brahms (Ormandy)

MIR tasks

1. Alignment
2. Transcription
3. Instrument recognition
4. Tempo estimation
5. Key detection
6. Dynamics
7. Emotion

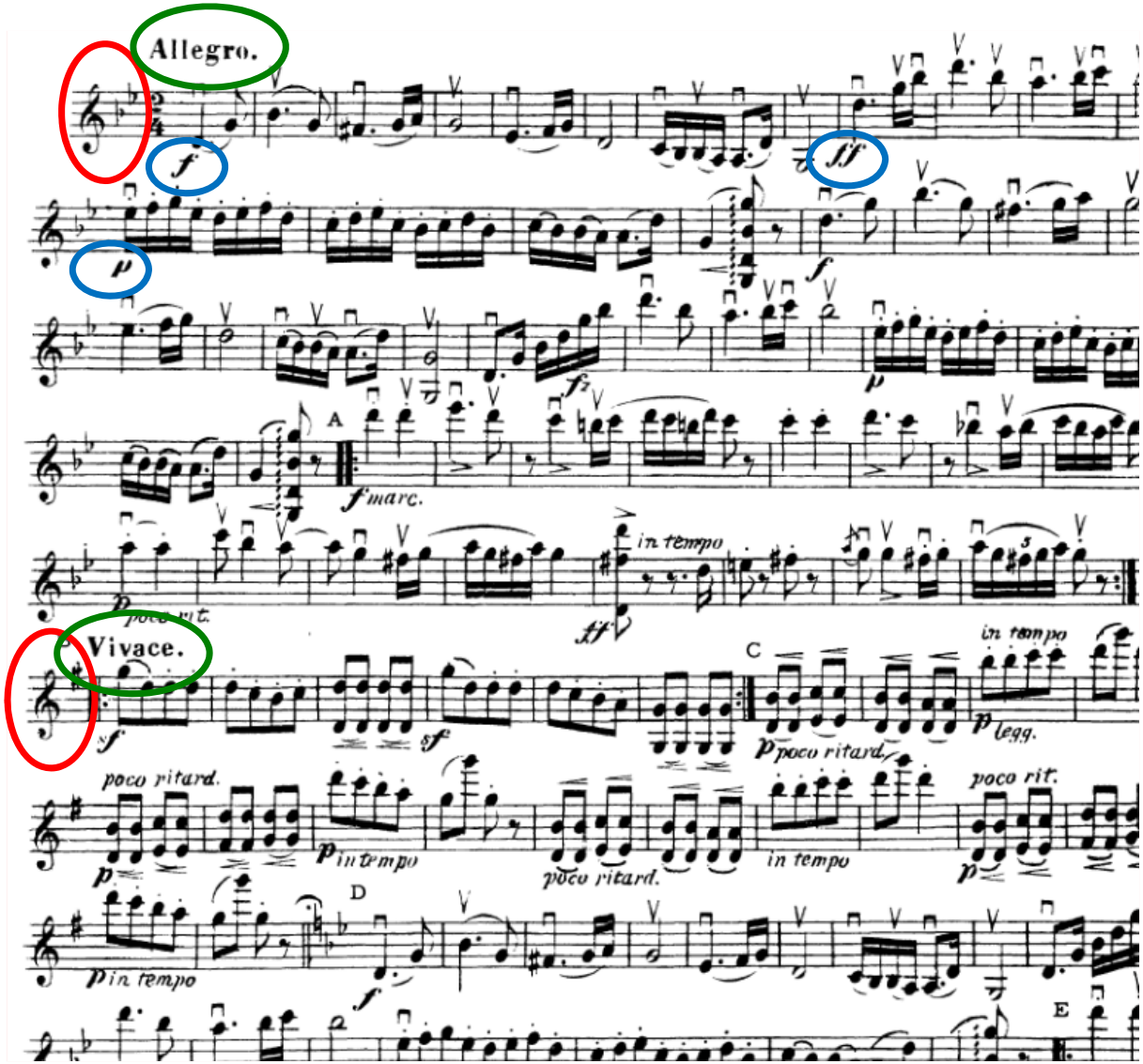
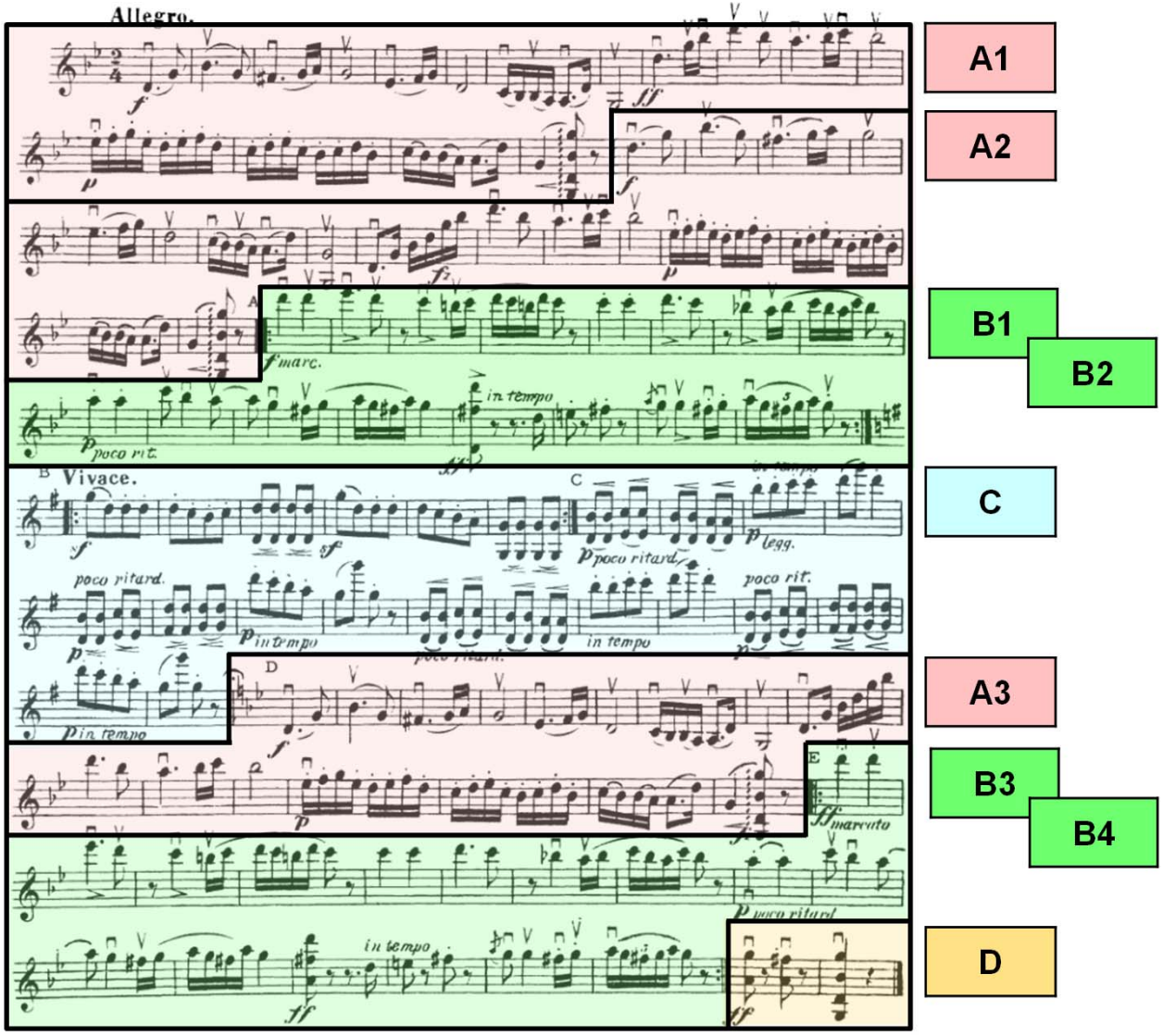


Figure from [Mueller, FPM, Chapter 4, Springer 2015]

Hungarian Dance No. 5 by Johannes Brahms (Ormandy)

8. Structure

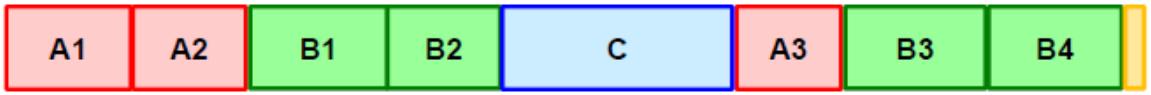


https://www.youtube.com/watch?v=QAMxkietiik

Figure from [Mueller, FPM, Chapter 4, Springer 2015]

Music Structure Analysis

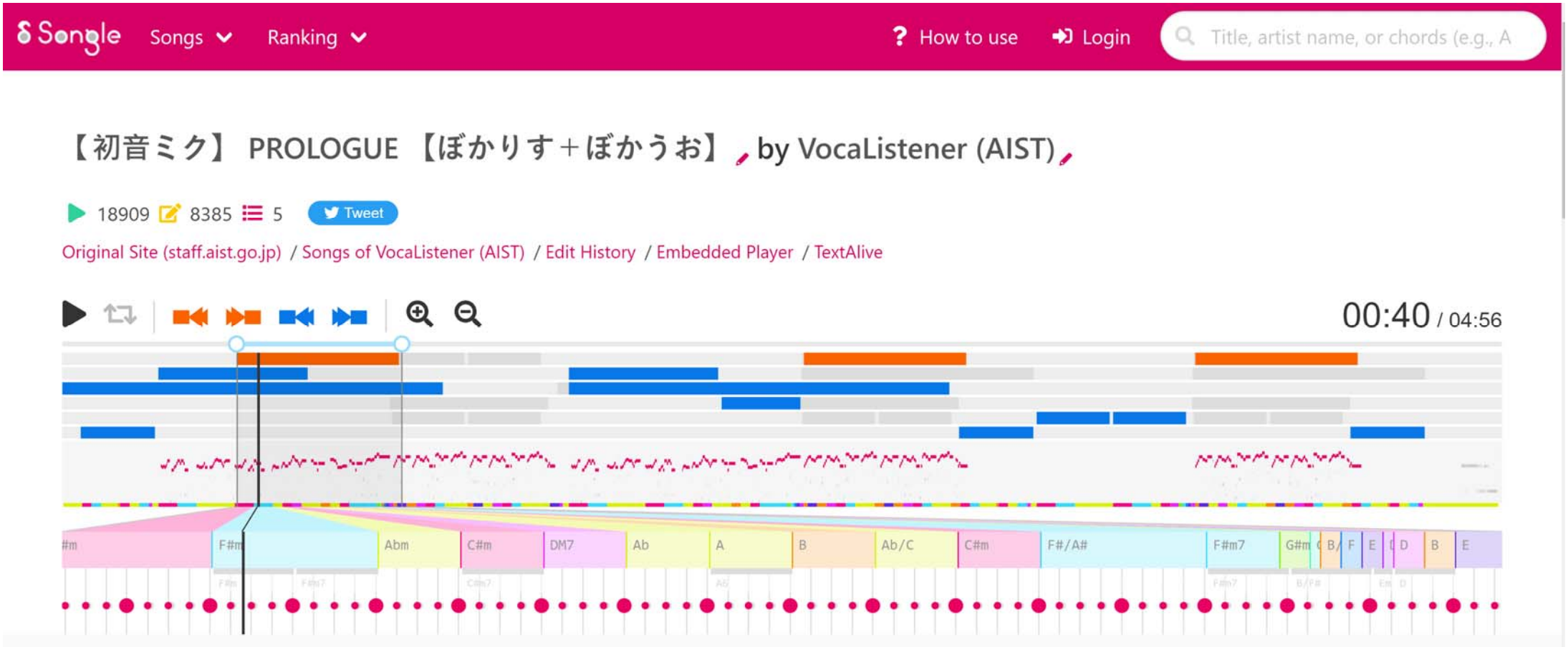
- General goal:
  1. **Segmentation:** **Divide** an audio recording into temporal segments corresponding to musical parts
  2. **Grouping**: **Cluster** or **label** these segments into musically meaningful categories



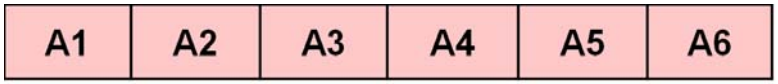
- Examples
  - *Intro, verse, chorus, bridge, outro* of a pop song
  - *Exposition, development, recapitulation, coda* of a sonata

Demonstration: Songle

https://songle.jp/



Music Structure / Form



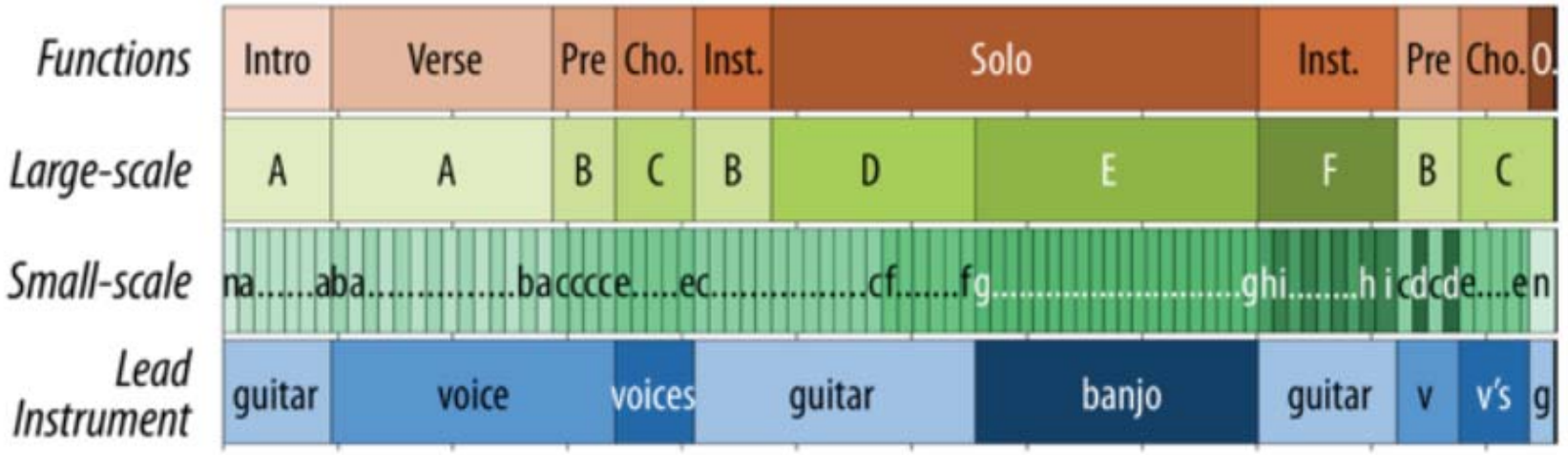
- Strophic form
- Chain form
- Rondo form

- Sonata form
- “Tell Me Why” by Beatles
- “Yesterday” by Beatles

Figure from [Mueller, FPM, Chapter 4, Springer 2015]

Ambiguity in Structure Analysis

- "Musical structure naturally exhibits a **hierarchical** organization where a variety of cues can trigger boundaries between segments of different length, depending on the time scale at which they are observed"



Ref 1: Buisson et al, "Learning multi-level representations for hierarchical music structure analysis," ISMIR 2022

Ref 2: Smith et al, "Design and creation of a large-scale database of structural annotations," ISMIR 2011

General Principles of Music Structure Analysis

1) Novelty, 2) Homogeneity, 3) Repetition

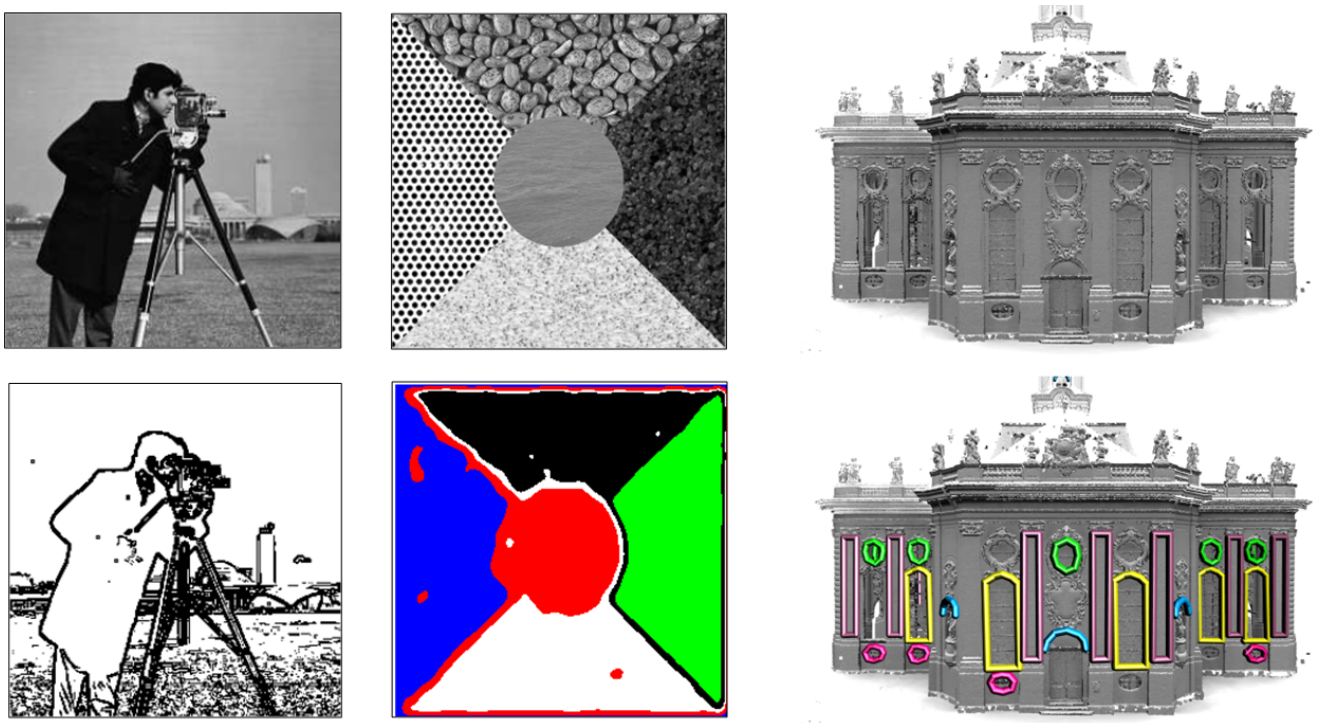
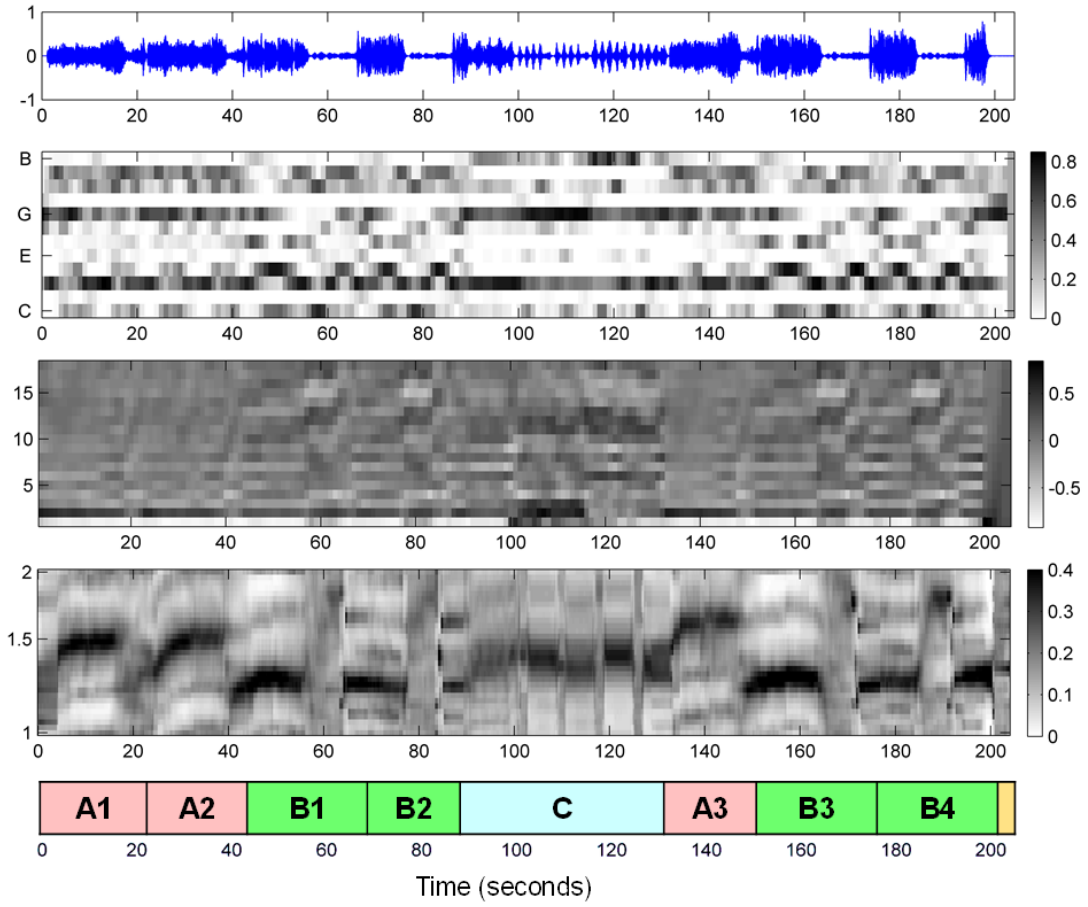


Figure from [Mueller, FPM, Chapter 4, Springer 2015]

Acoustic Features Cues

https://www.audiolabs-erlangen.de/resources/MIR/FMP/C4/C4S1_MusicStructureGeneral.html

- Wav
- Chromagram
- MFCC
- Tempogram
- Groundtruth



Feature Representation

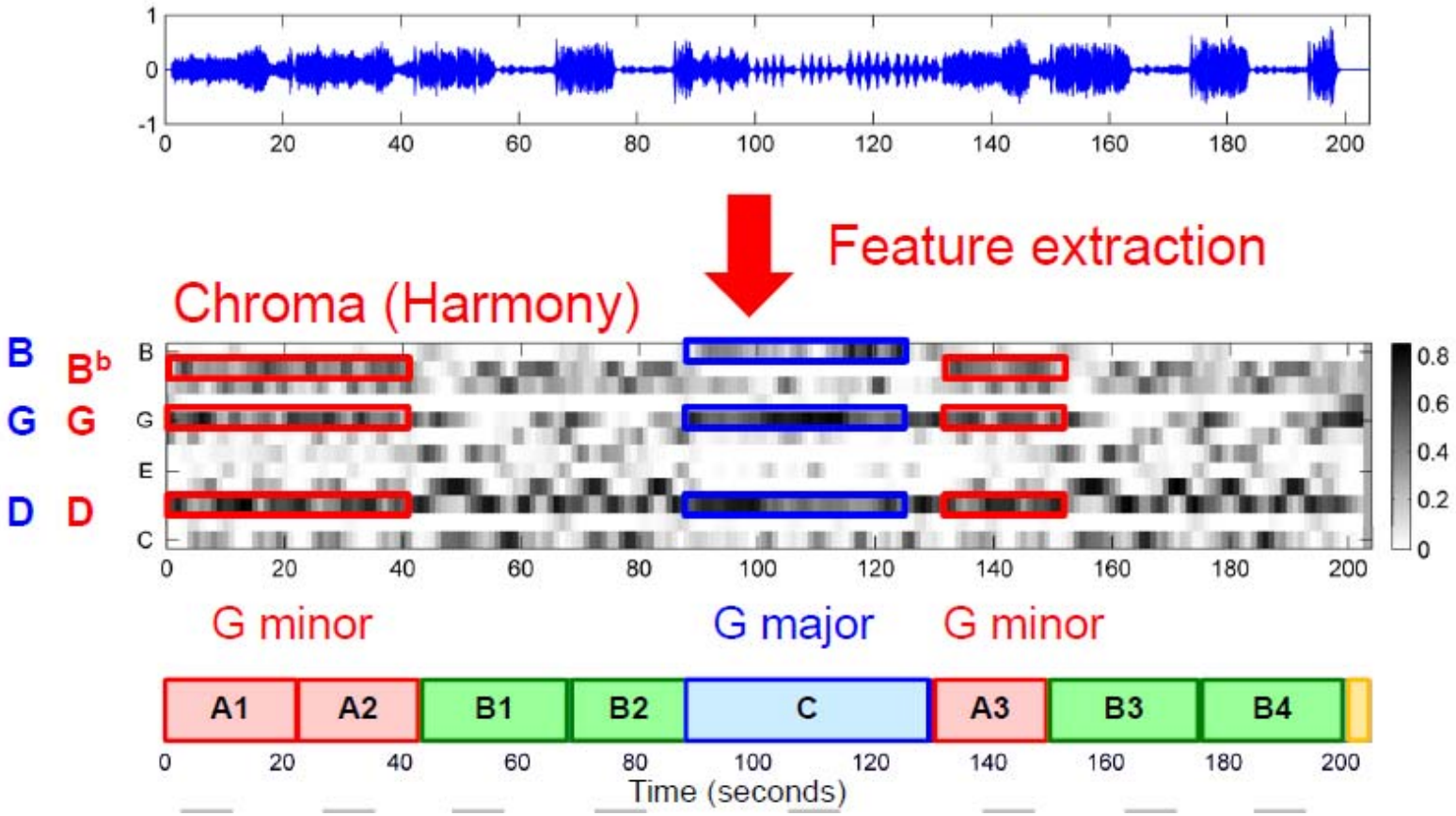


Figure from [Mueller, FPM, Chapter 4, Springer 2015]

Self-Similarity Matrices (SSM)

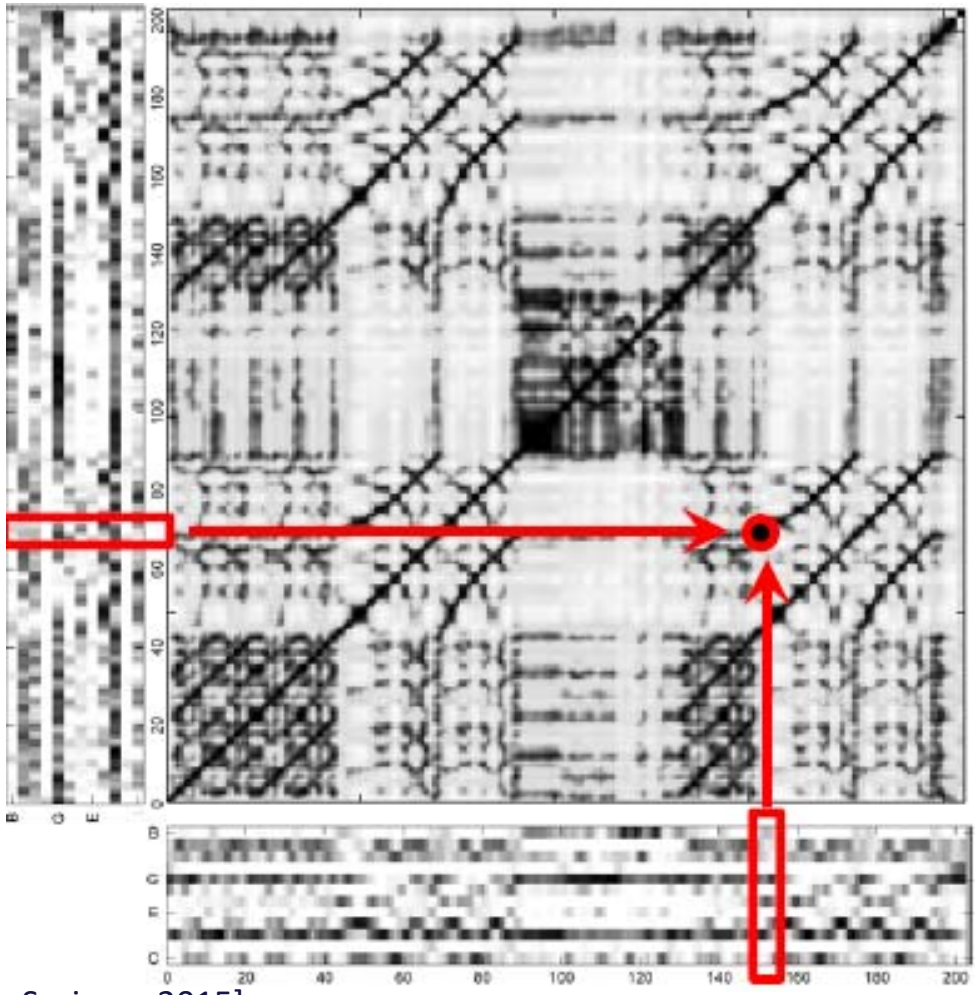


Figure from [Mueller, FPM, Chapter 4, Springer 2015]

Blocks → Homogeneity

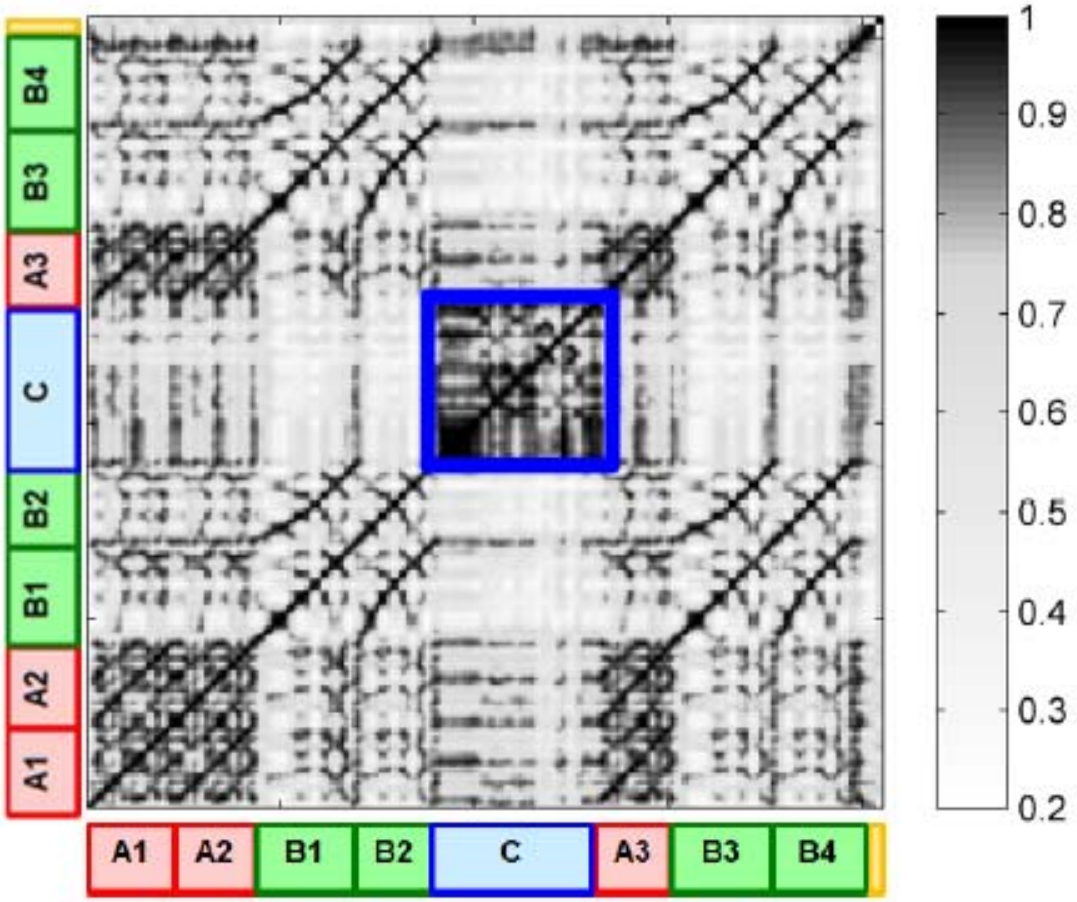


Figure from [Mueller, FPM, Chapter 4, Springer 2015]

Paths → Repetition

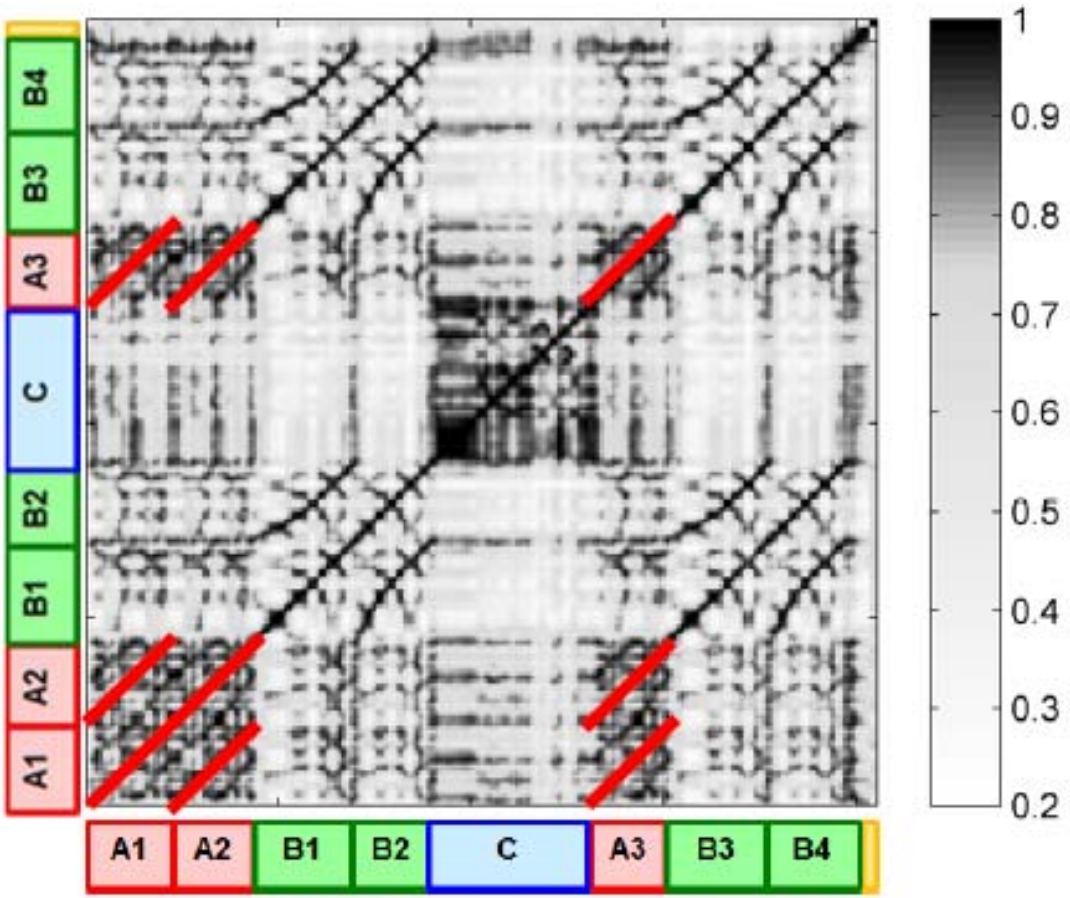


Figure from [Mueller, FPM, Chapter 4, Springer 2015]

Paths → Repetition

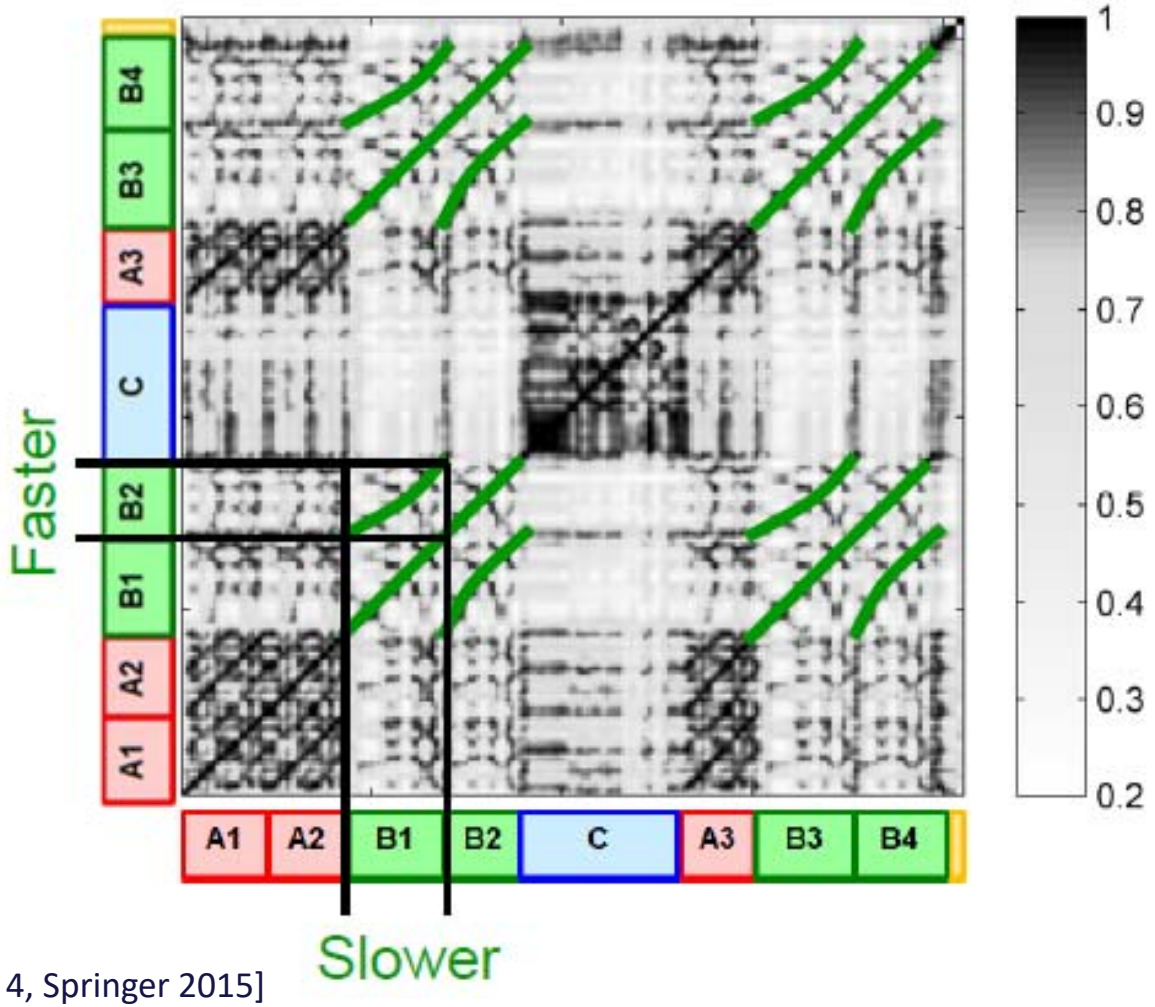


Figure from [Mueller, FPM, Chapter 4, Springer 2015]

Self-Similarity Matrices (SSM)

https://www.audiolabs-erlangen.de/resources/MIR/FMP/C4/C4S2_SSM.html

Blocks: Homogeneity

Paths: Repetition

Corners: Novelty

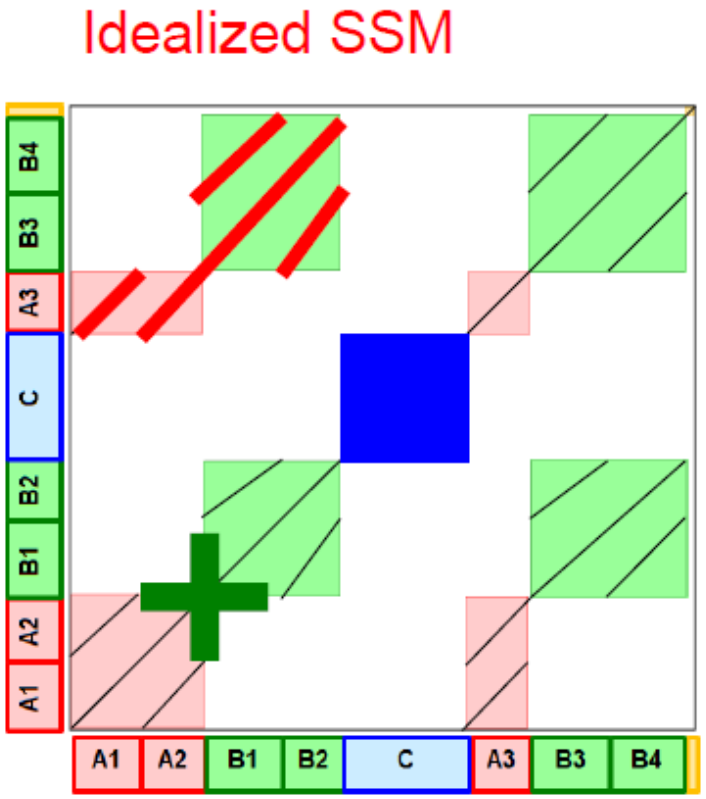
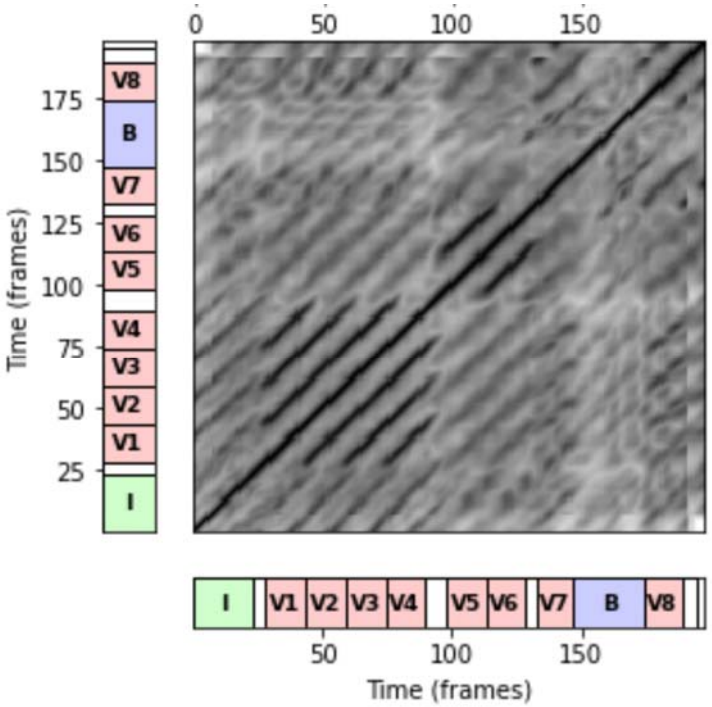


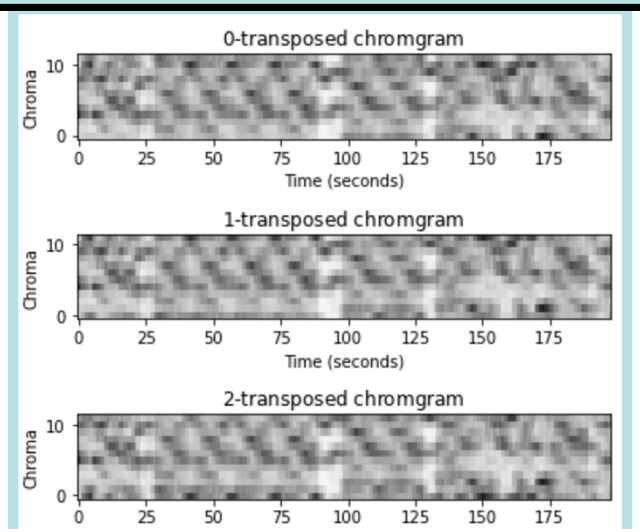
Figure from [Mueller, FPM, Chapter 4, Springer 2015]

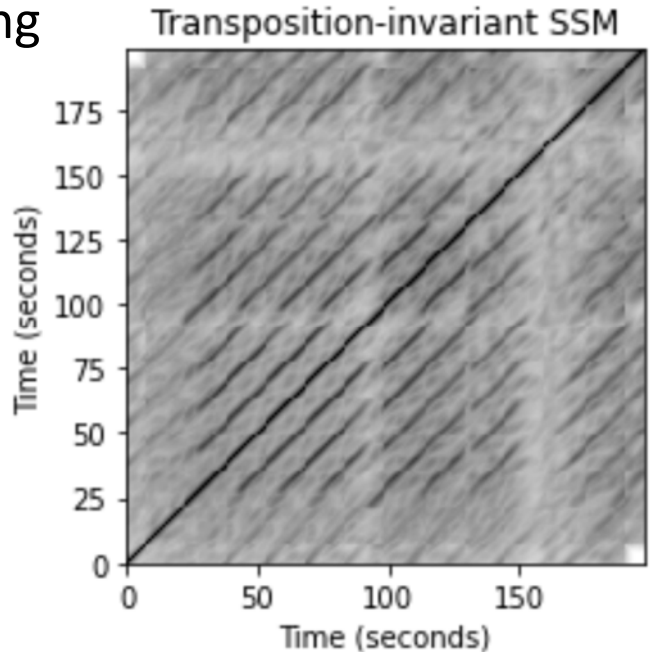
Transposition-Invariant SSM

https://www.audiolabs-erlangen.de/resources/MIR/FMP/C4/C4S2_SSM-TranspositionInvariance.html

1. **Cyclic shift** the chromagram
2. Compare the original recording with the transposed ones
3. Cell-wise **max** over 12 shifts

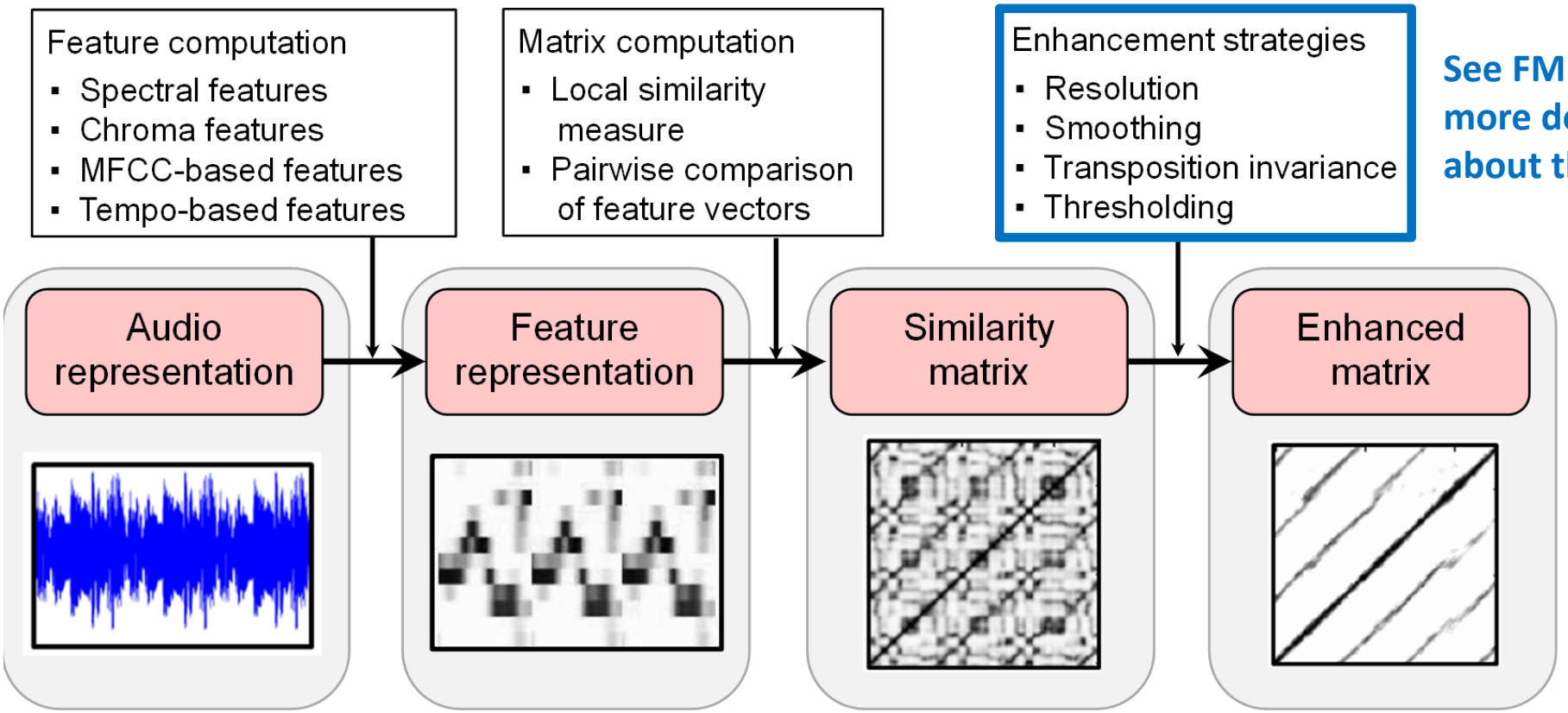






Procedure

https://www.audiolabs-erlangen.de/resources/MIR/FMP/C4/C4S1_MusicStructureGeneral.html



See FMP for more details about this part

Novelty-based Segmentation

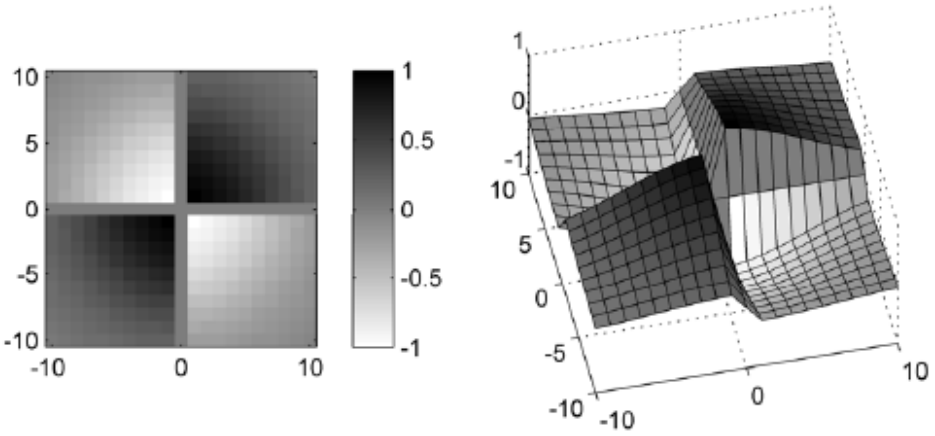
https://www.audiolabs-erlangen.de/resources/MIR/FMP/C4/C4S4_NoveltySegmentation.html

General goals:

- Find instances where musical changes occur.
- Find transition between subsequent musical parts.

Idea (Foote):

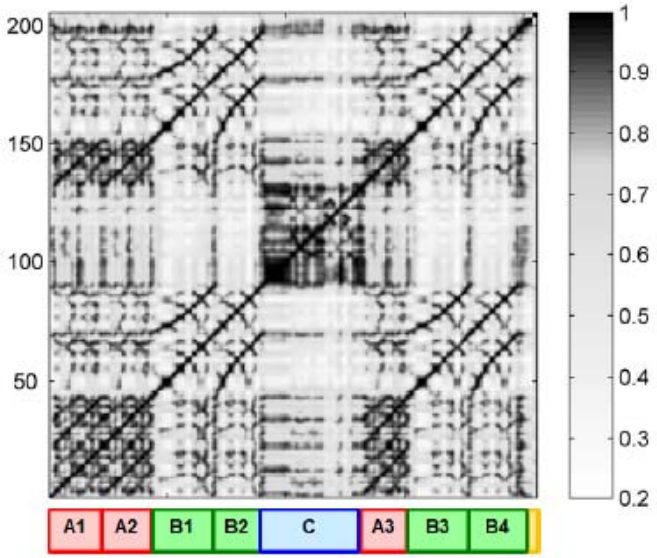
Use checkerboard-like kernel function to detect corner points on main diagonal of SSM.



Gaussian checkerboard kernel

Figure from [Mueller, FPM, Chapter 4, Springer 2015]

Novelty-based Segmentation

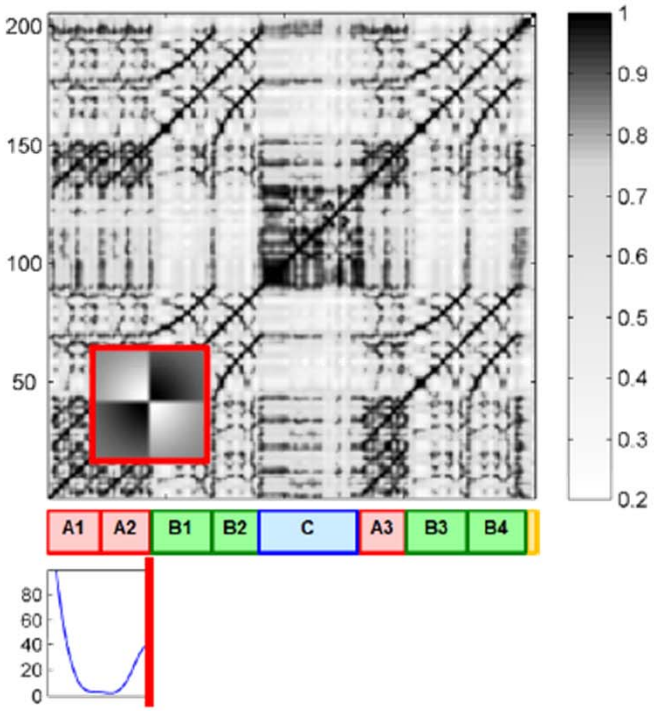


Idea (Foote):

Use checkerboard-like kernel function to detect corner points on main diagonal of SSM.

Figure from [Mueller, FPM, Chapter 4, Springer 2015]

Novelty-based Segmentation

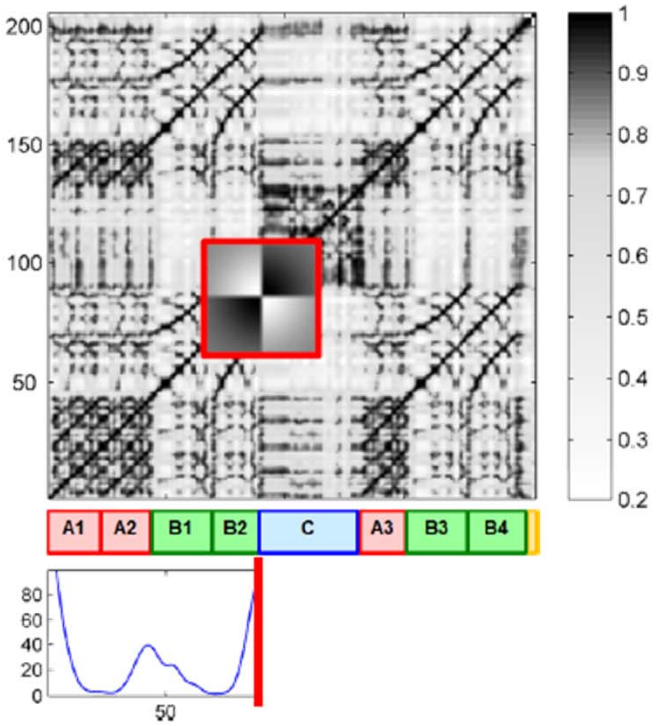


Idea (Foote):

Use checkerboard-like kernel function to detect corner points on main diagonal of SSM.

Figure from [Mueller, FPM, Chapter 4, Springer 2015]

Novelty-based Segmentation

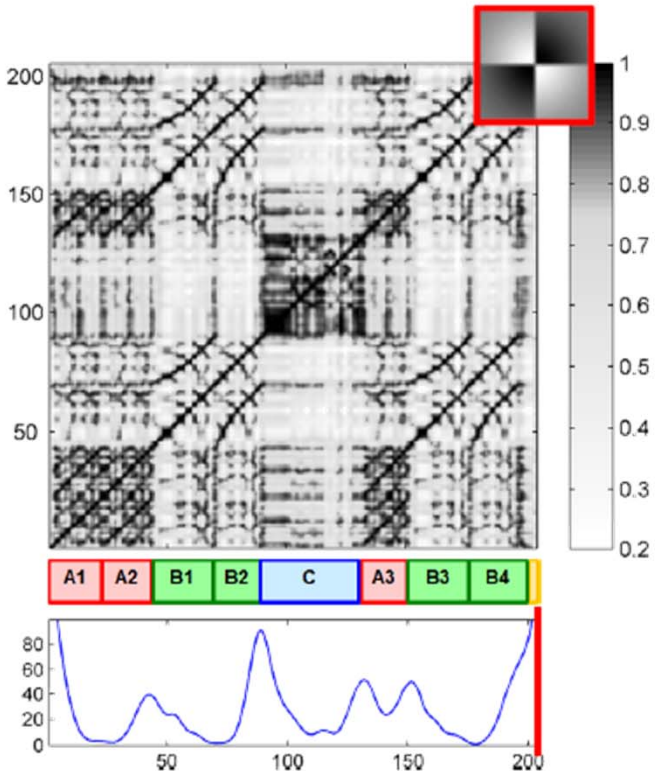


Idea (Foote):

Use checkerboard-like kernel function to detect corner points on main diagonal of SSM.

Figure from [Mueller, FPM, Chapter 4, Springer 2015]

Novelty-based Segmentation

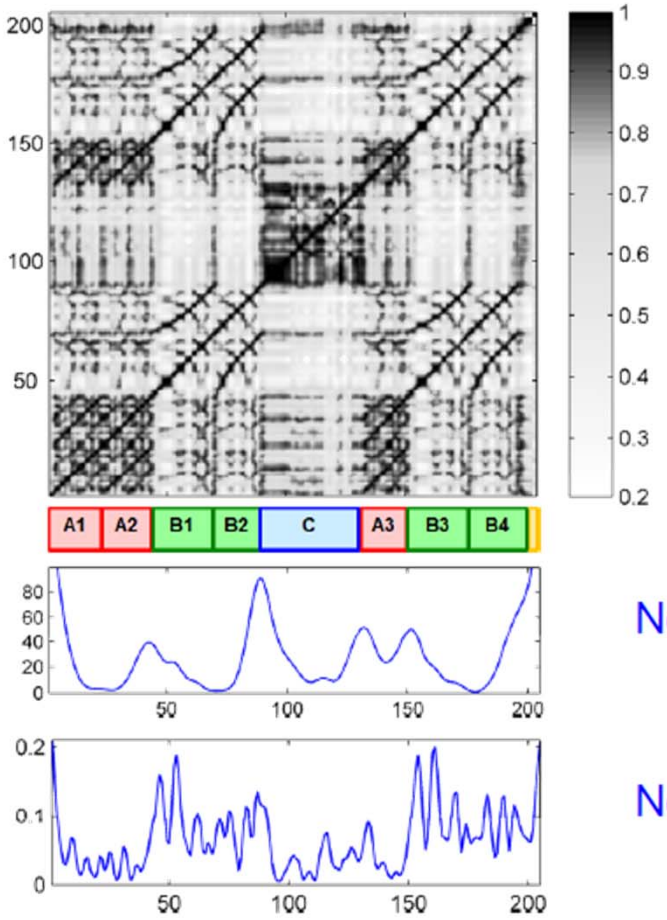


Idea (Foote):

Use checkerboard-like kernel function to detect corner points on main diagonal of SSM.

Figure from [Mueller, FPM, Chapter 4, Springer 2015]

Novelty-based Segmentation



Idea (Foote):

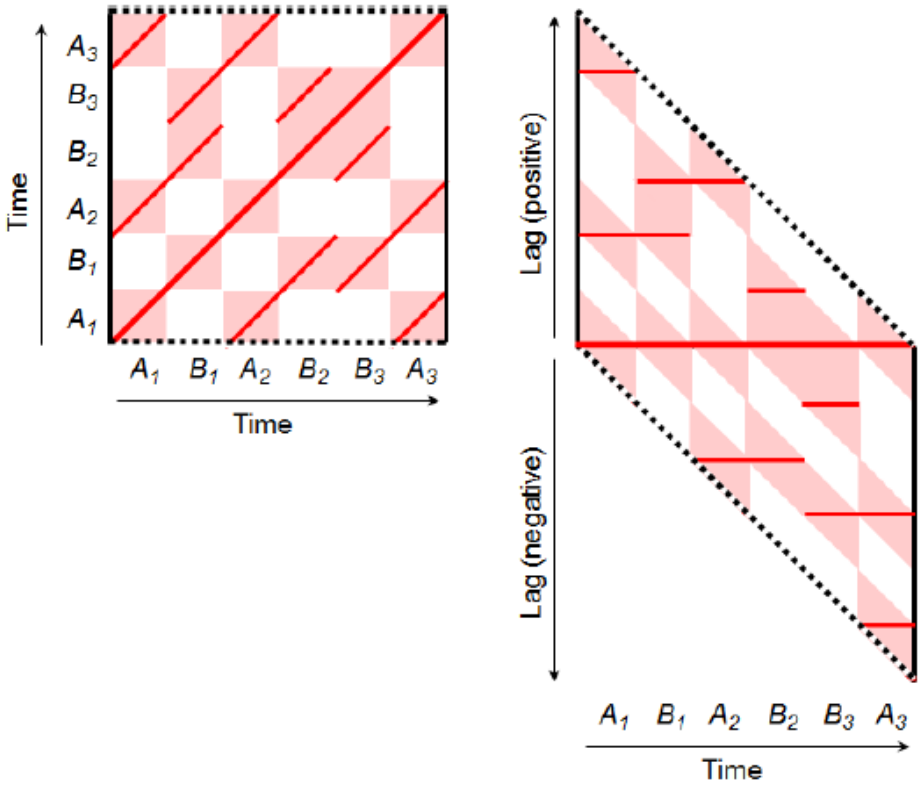
Use checkerboard-like kernel function to detect corner points on main diagonal of SSM.

Novelty function using 

Novelty function using 

Figure from [Mueller, FPM, Chapter 4, Springer 2015]

Structure Feature-based Segmentation

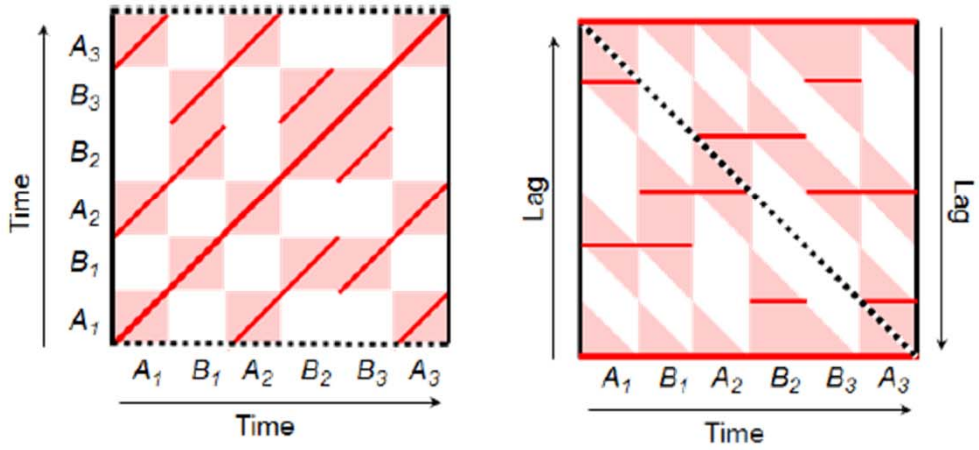


Structure features

- Enhanced SSM
- Time-lag SSM

Figure from [Mueller, FPM, Chapter 4, Springer 2015]

Structure Feature-based Segmentation



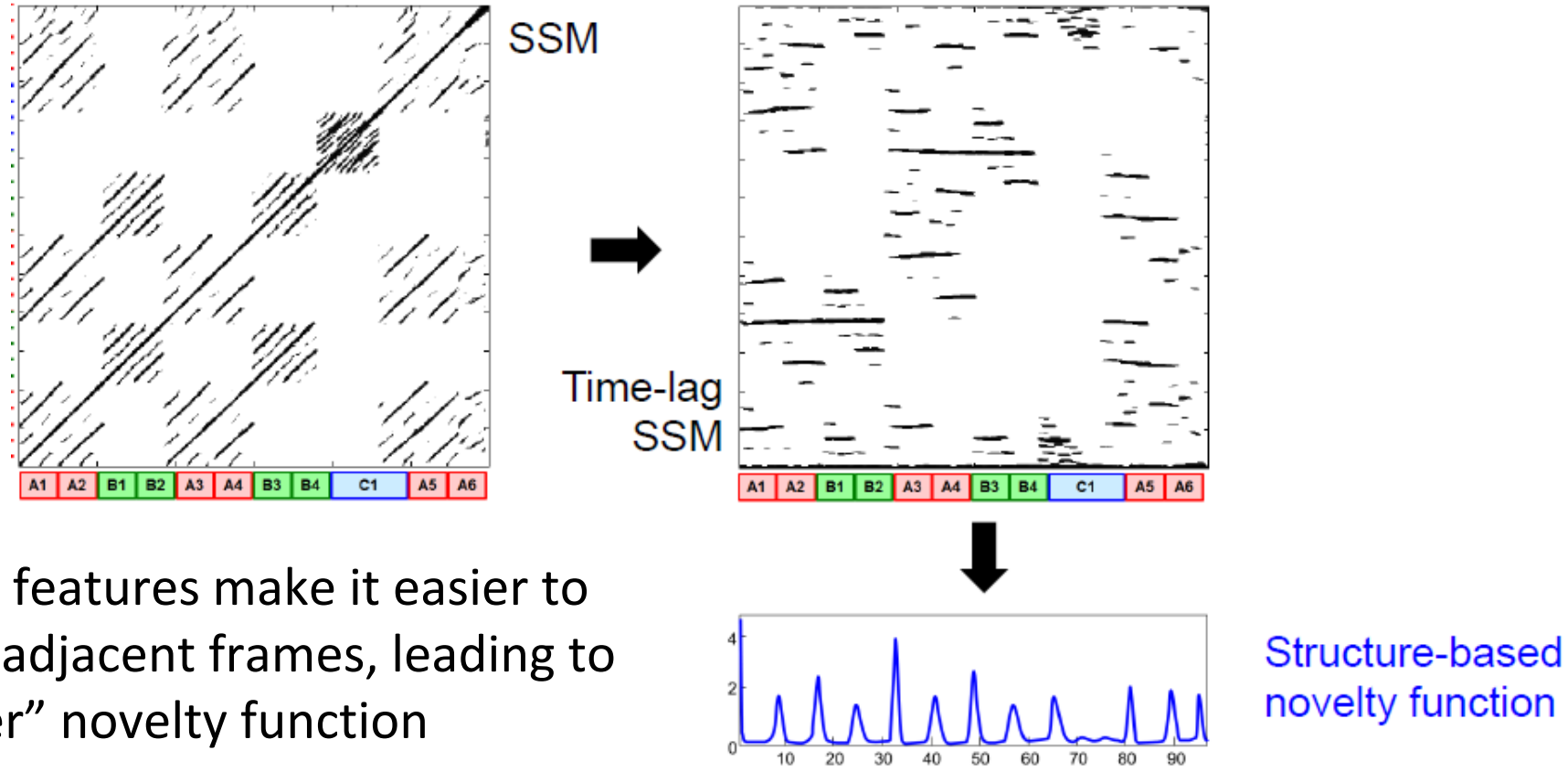
Structure features

- Enhanced SSM
- Time-lag SSM
- Cyclic time-lag SSM

Figure from [Mueller, FPM, Chapter 4, Springer 2015]

Structure Feature-based Segmentation

https://www.audiolabs-erlangen.de/resources/MIR/FMP/C4/C4S4_StructureFeature.html

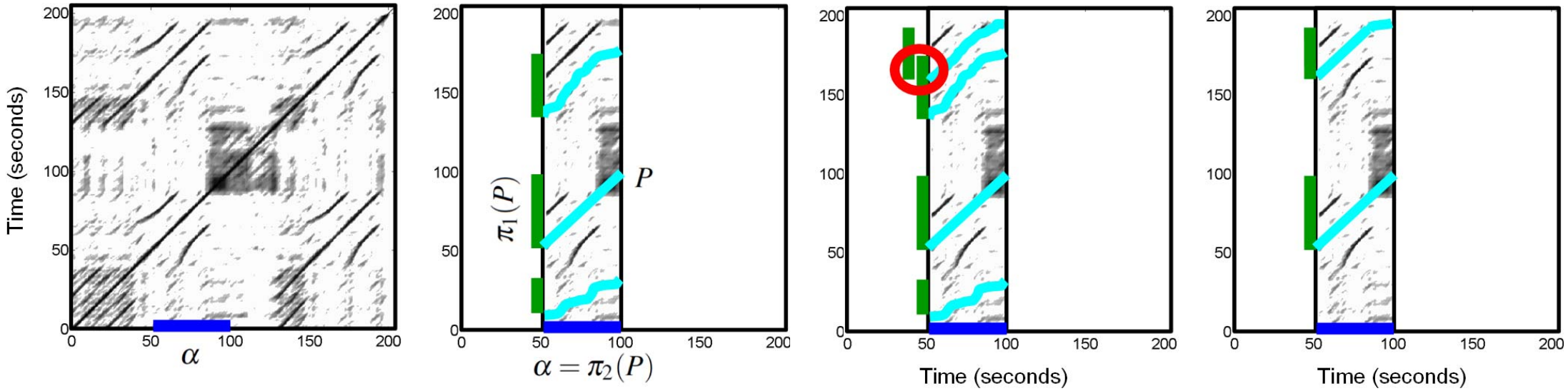


- Structure features make it easier to compare adjacent frames, leading to a “sharper” novelty function

Figure from [Mueller, FPM, Chapter 4, Springer 2015]

Application of SSM: Thumbnailing

https://www.audiolabs-erlangen.de/resources/MIR/FMP/C4/C4S3_AudioThumbnailing.html

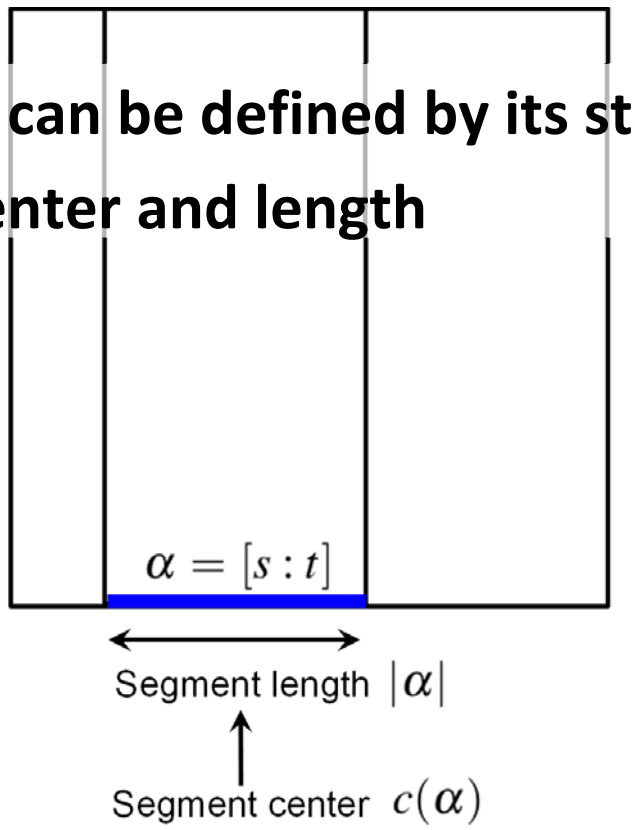


- **Fitness**: how well a given segment explains other related segments (by **DTW**) & how much of the overall music recording is covered by all these related segments (considering **non-overlapping** segments with **certain similarity**)
- **Audio thumbnail:** the segment of maximal fitness

Application: Scape Plot Representation

https://www.audiolabs-erlangen.de/resources/MIR/FMP/C4/C4S3_ScapePlot.html

- **A segment can be defined by its starting and end points**
- **Or by its center and length**



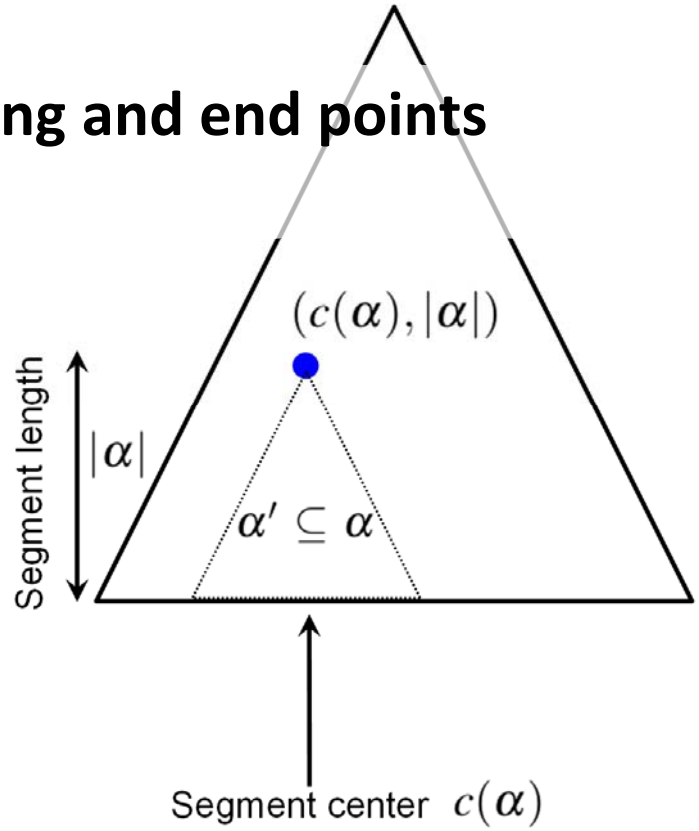


Figure from [Mueller, FPM, Chapter 4, Springer 2015]

Application: Scape Plot Representation

https://www.audiolabs-erlangen.de/resources/MIR/FMP/C4/C4S3_ScapePlot.html

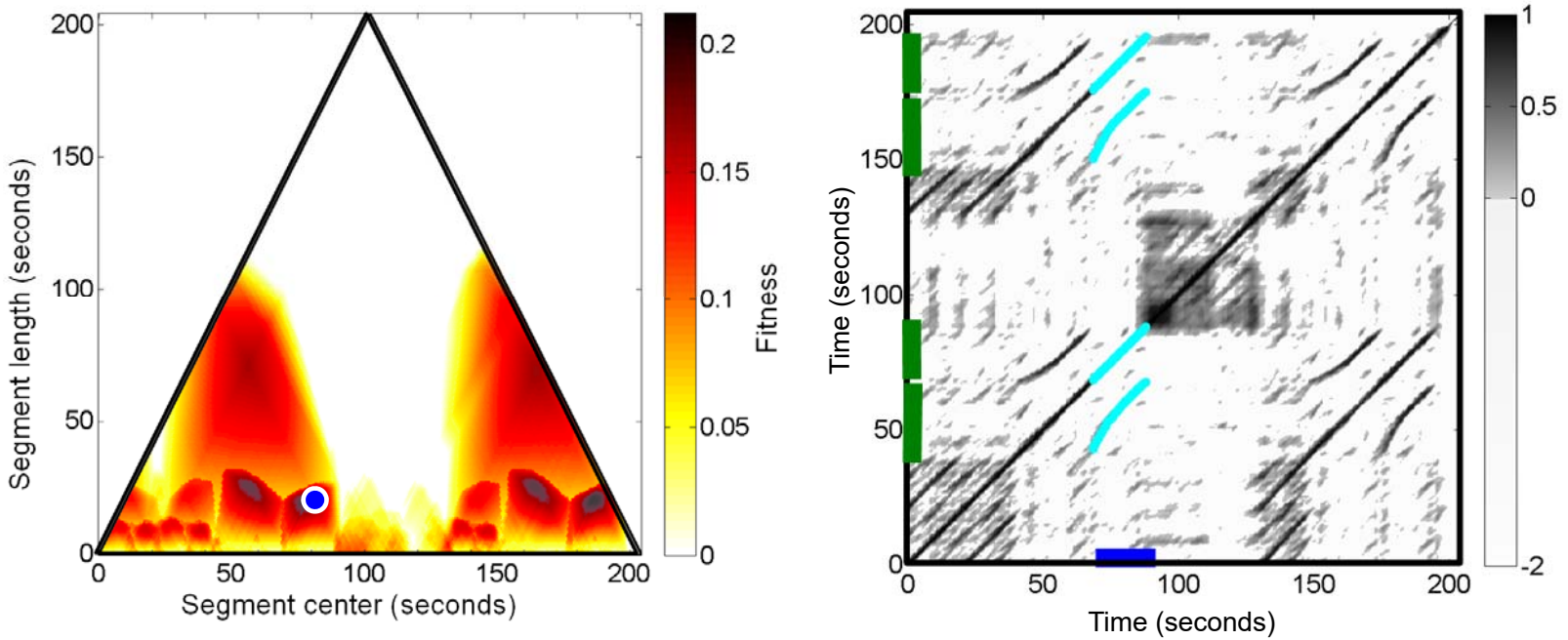


Figure from [Mueller, FPM, Chapter 4, Springer 2015]

Application: Scape Plot Representation

https://www.audiolabs-erlangen.de/resources/MIR/FMP/C4/C4S3_ScapePlot.html

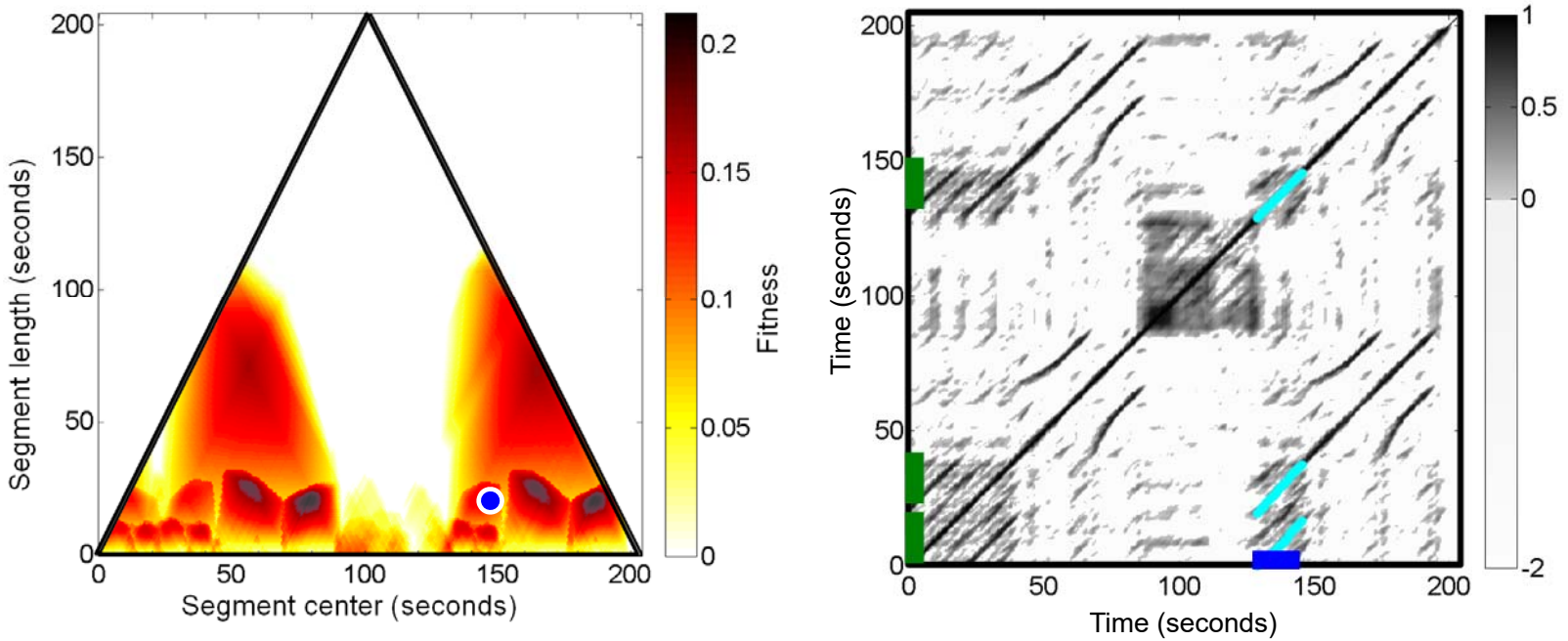


Figure from [Mueller, FPM, Chapter 4, Springer 2015]

Application: Scape Plot Representation

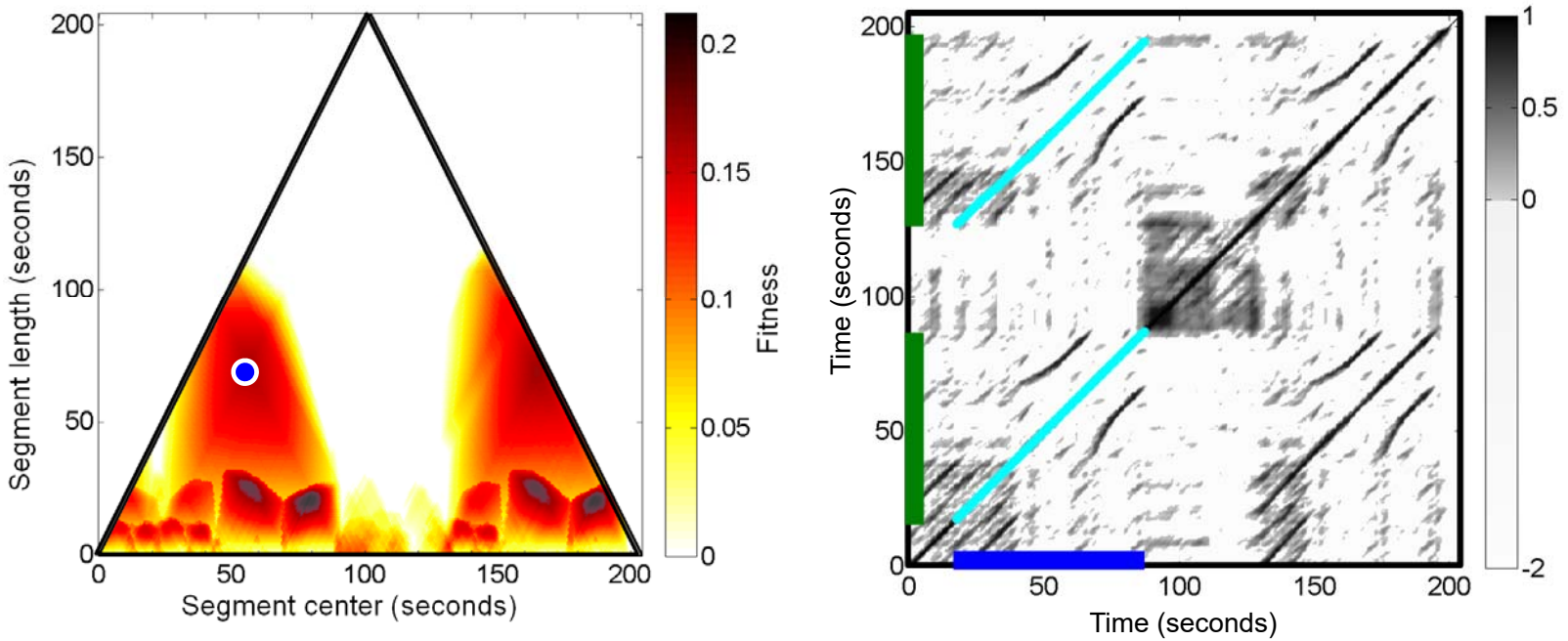


Figure from [Mueller, FPM, Chapter 4, Springer 2015]

Library: Librosa

https://librosa.org/doc/latest/segment.html

Temporal segmentation

Recurrence and self-similarity

`recurrence_matrix` (data[, k, width, metric, ...])	Compute a recurrence matrix from a
`recurrence_to_lag` (rec[, pad, axis])	Convert a recurrence matrix into a l
`lag_to_recurrence` (lag[, axis])	Convert a lag matrix into a recurren
`timelag_filter` (function[, pad, index])	Filtering in the time-lag domain.

Temporal clustering

`agglomerative` (data, k[, clusterer, axis])	Bottom-up temporal segmentation.
`subsegment` (data, frames[, n_segments, axis])	Sub-divide a segmentation by featu

Library: MSAF

https://github.com/urinieto/msaf

Boundary Algorithms

- Improved C-NMF (Nieto & Jehan 2013)
- Checkerboard-like Kernel (Foote 2000)
- OLDA (McFee & Ellis 2014) (original source code from here)
- Spectral Clustering (McFee & Ellis 2014) (original source code from here)
- Structural Features (Serrà et al. 2012)

Labeling Algorithms

- Improved C-NMF (Nieto & Jehan 2013)
- 2D Fourier Magnitude Coefficients (Nieto & Bello 2014)
- Spectral Clustering (McFee & Ellis 2014) (original source code from here)

Outline

- Music structure analysis
- **Objective evaluation metrics for MIDI generation**
- Structured MIDI generation
- HW3

Evaluation Metrics for MIDI Generation

- Important aspects of MIDI generation
  - **Correctness**: "Is the music free of inharmonious notes, unnatural rhythms, and awkward phrasing?"
  - **Coherence**: over time & across different tracks
  - **Structureness**: "Are recurring motifs / phrases / sections, and reasonable musical development present?"
  - **Richness**: "Is the music intriguing and full of variations within?"
  - **Controllability**: Can the generation process be easily steered? Does the model allow for easy human-AI co-generation?

Ref: Wu & Yang, "Compose & Embellish: Well-structured piano performance generation via a two-stage approach", ICASSP 2023

Objective Measures for Correctness and Coherence

https://salu133445.github.io/muspy/doc/metrics.html

- For general MIDI → use **MusPy**

- drum_in_pattern_rate
- drum_pattern_consistency
- empty_beat_rate
- empty_measure_rate
- groove_consistency
- n_pitch_classes_used
- n_pitches_used
- pitch_class_entropy
- pitch_entropy
- pitch_in_scale_rate
- pitch_range
- polyphony
- polyphony_rate
- scale_consistency

Ref 1: Dong et al,"MusPy: A toolkit for symbolic music generation", ISMIR 2020
Ref 2: Dong et al, "MuseGAN: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment", AAAI 2018

Objective Measures for Correctness and Coherence

https://github.com/slSeanWU/MusDr

- For single-track MIDI → use **MusDr**
  - Pitch-Class Histogram Entropy (**H**)
    -- measures erraticity of **pitch usage** in shorter timescales (e.g., 1 or 4 bars)
  - Grooving Pattern Similarity (**GS**)
    -- measures consistency of **rhythm** across the entire piece
  - Chord Progression Irregularity (**CPI**)
    -- measures consistency of **harmony** across the entire piece

Ref: Wu & Yang, "The Jazz Transformer on the front line: Exploring the shortcomings of AI-composed music through quantitative measures", ISMIR 2020

Pitch Class Histogram Entropy

- "If a piece's **tonality** is clear, several **pitch classes** should dominate the pitch histogram (e.g., the tonic and the dominant), resulting in low-entropy"

To gain insight into the usage of different pitches, we first collect the notes appeared in a certain period (e.g., a bar) and construct the 12-dimensional pitch class histogram $\overrightarrow{\mathbf{h}}$, according to the notes' pitch classes (i.e. `C`, `C#`, ..., `A#`, `B`), normalized by the total note count in the period such that $\sum_i h_i = 1$. Then, we calculate the entropy of $\overrightarrow{\mathbf{h}}$:

$$\mathcal{H}(\overrightarrow{\mathbf{h}}) = -\sum_{i=0}^{11} h_i \log_2(h_i)\,. \tag{2}$$

The entropy, in information theory, is a measure of "uncertainty" of a probability distribution [40], hence we adopt it here as a metric to help assessing the music's quality in tonality. If a piece's tonality is clear, several pitch classes should dominate the pitch histogram (e.g., the tonic and the dominant), resulting in a low-entropy $\overrightarrow{\mathbf{h}}$; on the contrary, if the tonality is unstable, the usage of pitch classes is likely scattered, giving rise to an $\overrightarrow{\mathbf{h}}$ with high entropy.

Ref: Wu & Yang, "The Jazz Transformer on the front line: Exploring the shortcomings of AI-composed music through quantitative measures", ISMIR 2020

Grooving Pattern Similarity

- If a piece possesses a clear sense of **rhythm**, the **grooving patterns** between pairs of bars should be similar, thereby producing high GS scores

The grooving pattern represents the positions in a bar at which there is at least a note onset, denoted by $\overrightarrow{\mathbf{g}}$, a 64-dimensional binary vector in our setting.[6] We define the similarity between a pair of grooving patterns $\overrightarrow{\mathbf{g}}^a$, $\overrightarrow{\mathbf{g}}^b$ as:

$$\mathcal{GS}(\overrightarrow{\mathbf{g}}^a, \overrightarrow{\mathbf{g}}^b) = 1 - \frac{1}{Q}\sum_{i=0}^{Q-1} \mathrm{XOR}(g_i^a, g_i^b)\,, \qquad (3)$$

where Q is the dimensionality of $\overrightarrow{\mathbf{g}}^a$, $\overrightarrow{\mathbf{g}}^b$, and $\mathrm{XOR}(\cdot,\cdot)$ is the exclusive OR operation. Note that the value of $\mathcal{GS}(\cdot,\cdot)$ would always lie in between 0 and 1.

The grooving pattern similarity helps in measuring the music's rhythmicity. If a piece possesses a clear sense of rhythm, the grooving patterns between pairs of bars should be similar, thereby producing high $\mathcal{GS}$ scores; on the other hand, if the rhythm feels unsteady, the grooving patterns across bars should be erratic, resulting in low $\mathcal{GS}$ scores.

Ref: Wu & Yang, "The Jazz Transformer on the front line: Exploring the shortcomings of AI-composed music through quantitative measures", ISMIR 2020

Chord Progression Irregularity

- A well-composed Jazz piece should have a chord progression irregularity that is not too high

To measure the irregularity of a chord progression, we begin by introducing the term *chord trigram*, which is a triple composed of 3 consecutive chords in a chord progression; for example, (`Dm7`, `G7`, `CM7`). Then, *the chord progression irregularity* (*CPI*) is defined as the percentage of *unique chord trigrams* in the chord progression of an entire piece. Please note that 2 chord trigrams are considered different if any of their elements does not match.

It is common for Jazz compositions to make use of 8- or 12-bar-long templates of chord progressions (known as the *8-*, or *12-bar blues*), which themselves can be broken down into similar substructures [25, 35], as the foundation of a section, and more or less "copy-paste" them to form the complete song with, say, `AABA` parts. Therefore, a well-composed Jazz piece should have a chord progression irregularity that is not too high.

Ref: Wu & Yang, "The Jazz Transformer on the front line: Exploring the shortcomings of AI-composed music through quantitative measures", ISMIR 2020

Melody Harmonization Metrics

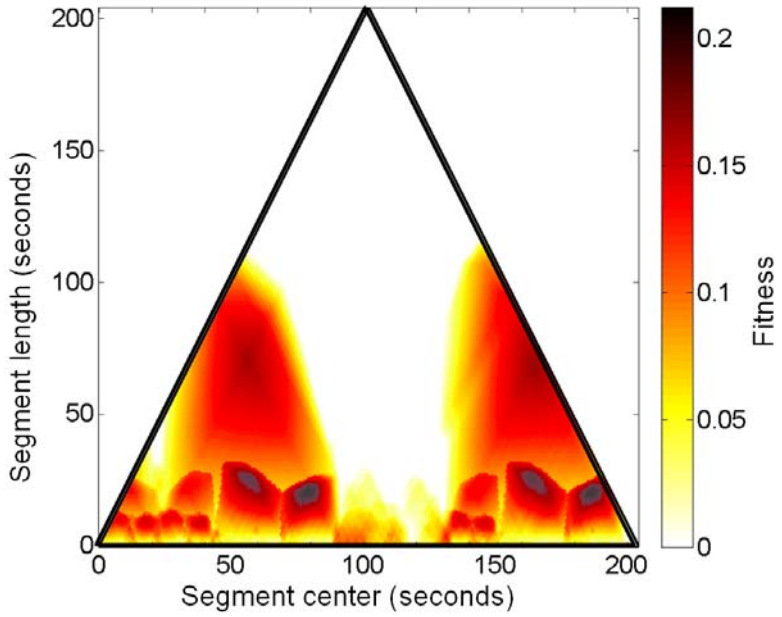
- **Chord progression metrics**: evaluate each chord sequence as a whole, independent from the melody
  - Chord histogram entropy
  - Chord coverage
  - Chord tonal distance
- **Chord/melody harmonicity metrics**: how well a chord sequence harmonizes a given melody sequence
  - Chord tone to non-chord tone ratio
  - Pitch consonance score
  - Melody-chord tonal distance

Ref: Yeh et al, “Automatic melody harmonization with triad chords: A comparative study”, JNMR 2021

Objective Measures for Structureness

https://github.com/slSeanWU/MusDr

- **MusDr**



Ref: Wu & Yang, "The Jazz Transformer on the front line: Exploring the shortcomings of AI-composed music through quantitative measures", ISMIR 2020

Our *structureness indicator* is based on the fitness scape plot and designed to capture the most salient repeat within a certain duration interval. For brevity of the mathematical representation, we assume the sampling frame rate of S is 1 Hz (hence N will be the piece's duration in seconds), and define the structureness indicator as follows:

$$\mathcal{SI}_l^u(S) = \max_{\substack{l \le i \le u \\ 1 \le j \le N}} S\,, \tag{4}$$

where l, u [8] are the lower and upper bounds of the duration interval (in seconds) one is interested in. In our experiments, we choose the structureness indicators of $\mathcal{SI}_3^8$, $\mathcal{SI}_8^{15}$, and $\mathcal{SI}_{15}$ to examine the short-, medium-, and long-term structureness respectively.

Objective Measures for Richness

Percentage of Distinct Pitch N-grams in Melody ($\mathcal{DN}$): following the popular *dist-n* [23] metric used in natural language generation to evaluate the diversity of generated content, we compute the percentage of distinct n-grams in the pitch sequence of the skyline extracted from each full performance. We regard **3∼5**, **6∼10**, and **11∼20** contiguous notes as short, medium, and long excerpts, and compute $\mathcal{DN}_{\text{short}}$, $\mathcal{DN}_{\text{mid}}$, and $\mathcal{DN}_{\text{long}}$ accordingly.

	Structureness (in %)			**Melody-line Diversity**		
	$\mathcal{SI}_{\text{short}}$	$\mathcal{SI}_{\text{mid}}$	$\mathcal{SI}_{\text{long}}$	$\mathcal{DN}_{\text{short}}$	$\mathcal{DN}_{\text{mid}}$	$\mathcal{DN}_{\text{long}}$
Naive repeats (1-bar)	83.6 ±8.6	88.3 ±5.4	75.7 ±11	1.2 ±0.6	1.2 ±0.6	1.3 ±0.6
Naive repeats (1-phrase)	75.5 ±15	86.2 ±6.8	73.9 ±11	8.2 ±4.8	10.0 ±5.8	10.8 ±6.5
CP Transformer [4]	32.5 ±3.3	29.9 ±4.4	17.9 ±6.3	82.0 ±8.9	99.6 ±0.7	100 ±0.0
Real data	43.8 ±7.1	43.1 ±8.4	34.8 ±12	50.0 ±14	74.1 ±14	88.3 ±11

Ref: Wu & Yang, "Compose & Embellish: Well-structured piano performance generation via a two-stage approach", ICASSP 2023

Objective Measures for Controllability

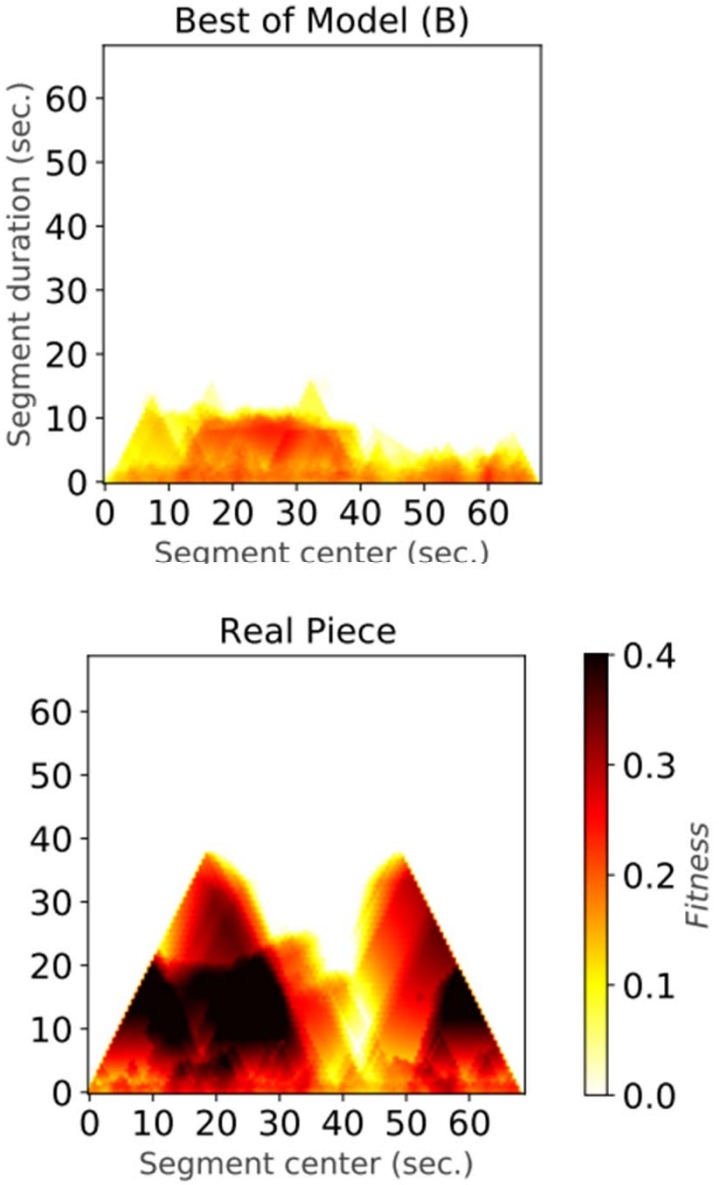
Model	Fidelity		Control			
	sim_{chr} ↑	sim_{grv} ↑	ρ_{rhym} ↑	ρ_{poly} ↑	$\lvert\rho_{\text{poly}\lvert\text{rhym}}\rvert$ ↓	$\lvert\rho_{\text{rhym}\lvert\text{poly}}\rvert$ ↓
MIDI-VAE [22]	75.6	85.4	.719	.261	.134	.056
Attr-Aware VAE [23]	85.0	76.8	**.997**	.781	.239	.040
Ours, AE objective	**98.5**	**95.7**	.181	.154	.058	.072
Ours, VAE objective	78.6	80.7	.931	.884	.038	**.003**
Ours, pre-attention	49.7	71.5	.962	**.921**	.035	.044
Ours, preferred settings	91.2	84.5	.950	.885	**.023**	.016

- The Spearman's rank correlation between an input attribute class and the attribute computed from the resulting generation
- The model should transfer an attribute without affecting other attributes

Ref: Wu & Yang, "MuseMorphose: Full-song and fine-grained piano music style transfer with one Transformer VAE", TASLP 2023

Transformers Do Not Easily Compose Music with Nice Structure

	Model (A)		Model (B)			Real
loss	*0.80*	*0.25*	*0.80*	*0.25*	*0.10*	- -
$\mathcal{H}_1$	2.29	2.45	2.26	2.20	**2.17**	1.94
$\mathcal{H}_4$	3.12	3.05	3.04	**2.91**	2.94	2.87
$\mathcal{GS}$	**0.76**	0.69	0.75	**0.76**	**0.76**	0.86
$\mathcal{CPI}$	81.2	77.6	79.2	**72.6**	75.9	40.4
$\mathcal{SI}_3^8$	0.18	0.22	0.25	**0.27**	0.26	0.36
$\mathcal{SI}_8^{15}$	0.15	0.17	**0.18**	**0.18**	0.17	0.36
$\mathcal{SI}_{15}$	0.11	**0.14**	0.10	0.12	0.11	0.35



Ref: Wu & Yang, "The Jazz Transformer on the front line: Exploring the shortcomings of AI-composed music through quantitative measures", ISMIR 2020

Outline

- Objective evaluation metrics for MIDI generation
- Music structure analysis
- **Structured MIDI generation**
  - Theme Transformer
  - MuseMorphose and VQVAE-Transformer
  - Compose-and-embellish
- HW3

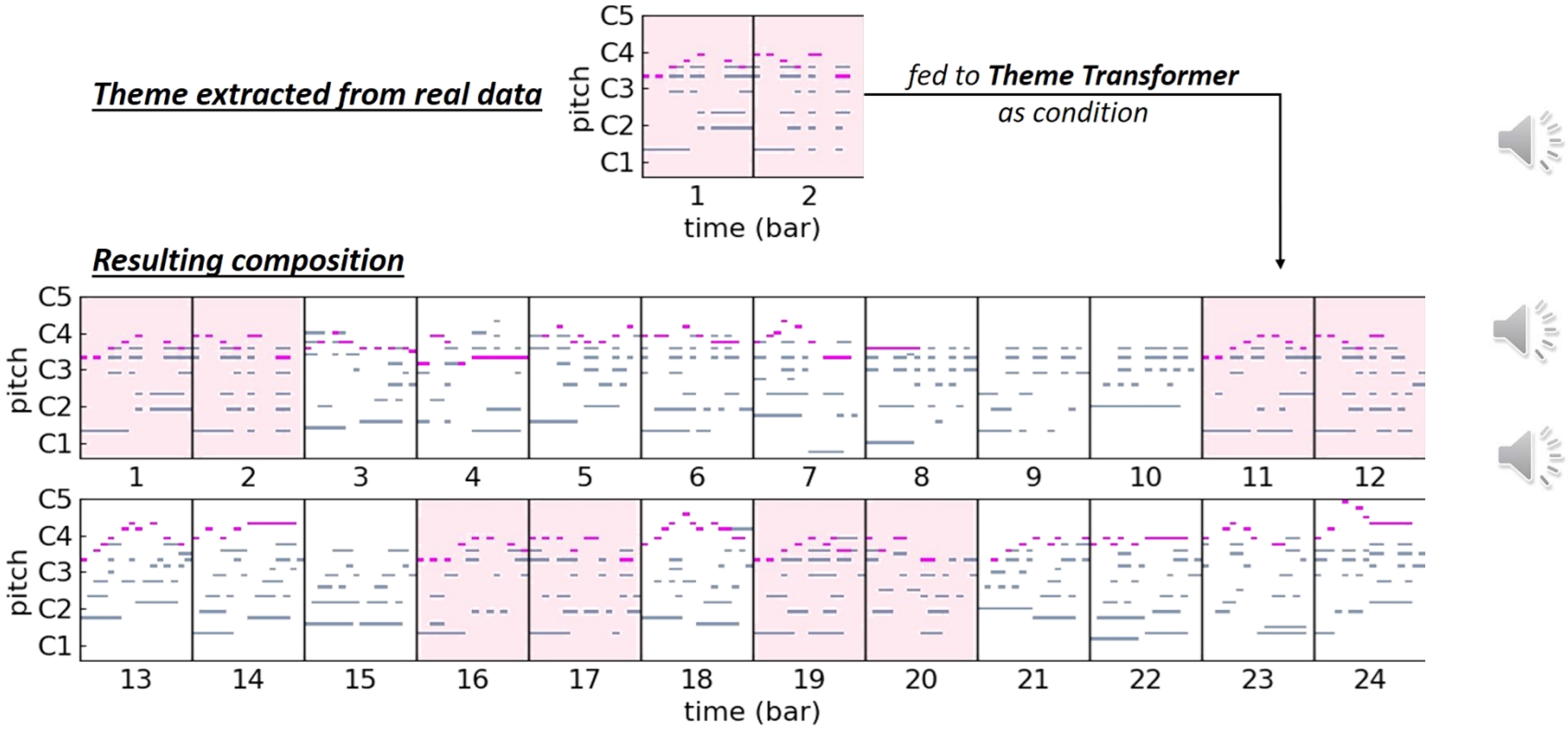
Theme Transformer: Theme-based Conditioning

- Conventional **prompt-based conditioning**
  - Cannot guarantee that the conditioning sequence would repeat or develop in the created continuation
  - Moreover, the influence of the prompt would diminish over time and the generated music *drifts away* from its beginning as the music gets longer
- Proposal: **theme-based conditioning**
  - **Automatic theme finding**: use fragments that have multiple occurrences in a music piece as the conditioning sequence at training time
  - Use **separate memory network** so that the model learns to **exploit the theme condition** when appropriate

Ref: Shih et al, “Theme Transformer: Symbolic music generation with theme-conditioned Transformer”, TMM 2023

Theme Transformer: Theme-based Generation

https://atosystem.github.io/ThemeTransformer/



Ref: Shih et al, “Theme Transformer: Symbolic music generation with theme-conditioned Transformer”, TMM 2023

Theme Transformer: Automatic Theme Finding

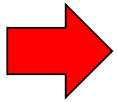
- Use **contrastive learning** to get **melody embedding**
  - **Negative pair**: fragments from different pieces
  - **Positive pair**: to be invariant to the following variations
    - Pitch shift on scale
    - Last note duration variation
    - Note splitting and combination

- Use DBSCAN to find clusters for each piece of music
- Pick the **largest cluster** as the theme for a piece of music

Theme Transformer: Token Design

- Use **THEME-START** and **THEME-END** tokens to indicate the beginning and ending of a ***theme region***

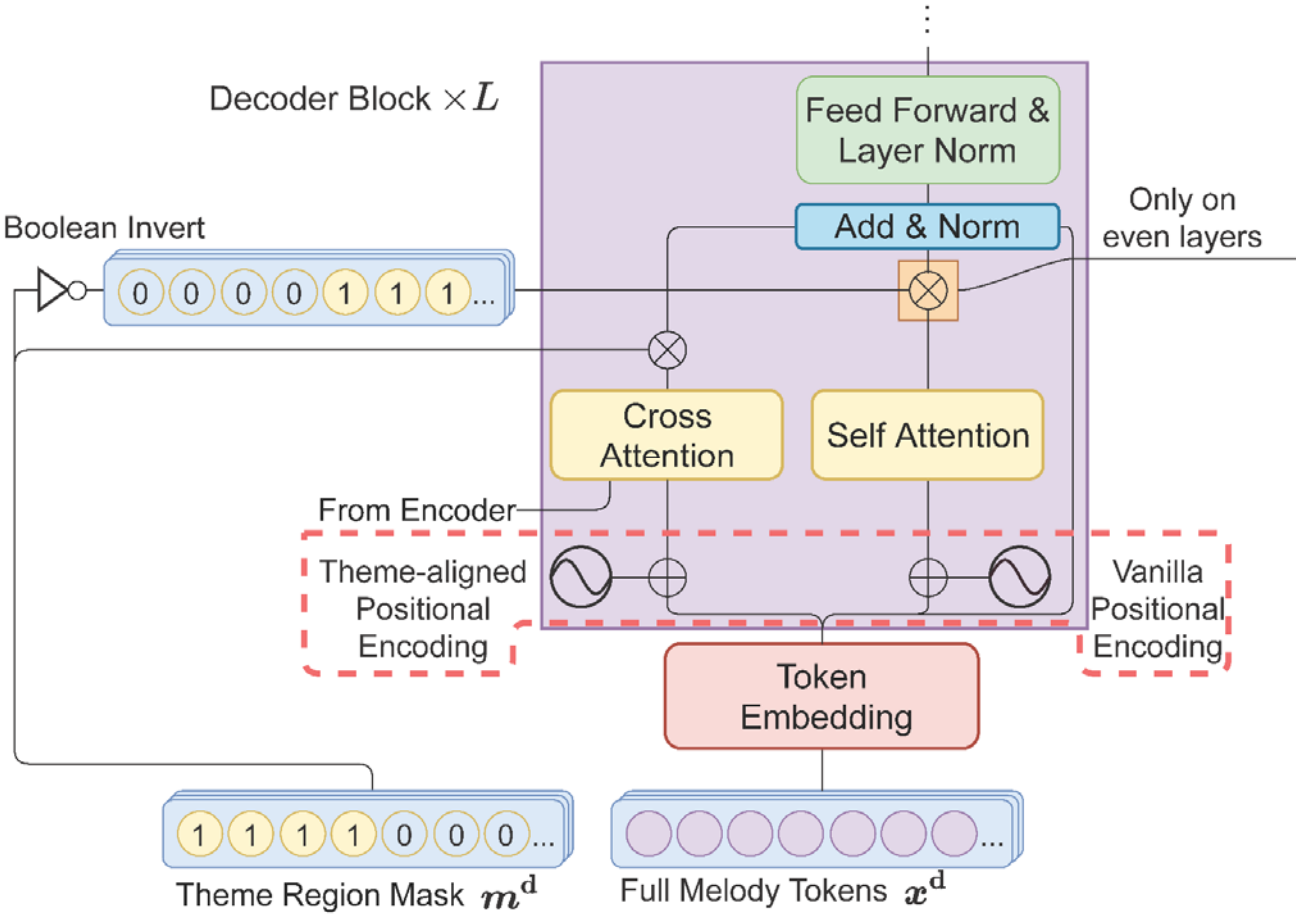
- At inference time, the model decides on its own when to enter a theme region



Token type	Piano Representation
NOTE-PITCH	127×2
NOTE-DURATION	64×2
NOTE-VELOCITY	126×2
TEMPO	76
BAR	1
SUBBEAT	16
THEME	2
PADDING	1
Total	730

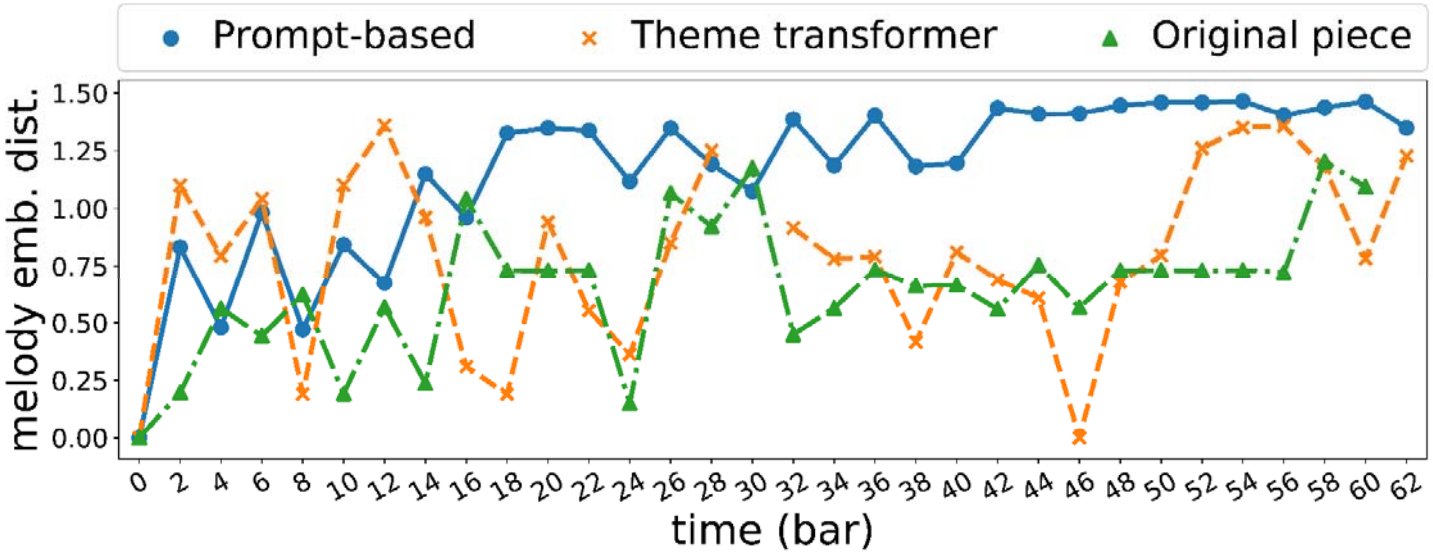
Theme Transformer: Attention Design

- **Separate memory network**
  - **Self-attention:** to attend to the past (local coherence)
  - **Cross-attention**: to attend to the theme
- **"Switch on"** cross-attention **only during theme regions**
- Dedicated **theme-aligned positional encoding**



Ref: Shih et al, "Theme Transformer: Symbolic music generation with theme-conditioned Transformer", TMM 2023

Theme Transformer: Evaluation



	Pitch class consistency↑	Melody inconsistency↓	Grooving consistency↑	Theme inconsistency↓	Theme uncontrollability↓
Baseline (Huang et al. 2019)	.59±.07	.33±.38	.84±.09	—	—
Seq2seq Transformer (proposed)	.61±.04	.46±.28	.90±.06	.22±.03	.45±.19
Theme Transformer (proposed)	.61±.06	.13±.24	.92±.07	.15±.06	.34±.13
Original pieces	.65±.05	.09±.18	.74±.10	.11±.06	.09±.06

Ref: Shih et al, “Theme Transformer: Symbolic music generation with theme-conditioned Transformer”, TMM 2023

Theme Transformer: Possible Extensions

- Extensions
  - Multiple themes
  - Application to other music genres
  - Use melody as the theme prompt (and make a UI)
  - CP token representation
  - Memory-based RNN
  - Variants of gating
  - To-copy pretrain
  - Better theme retrieval method
  - Better segmentation method

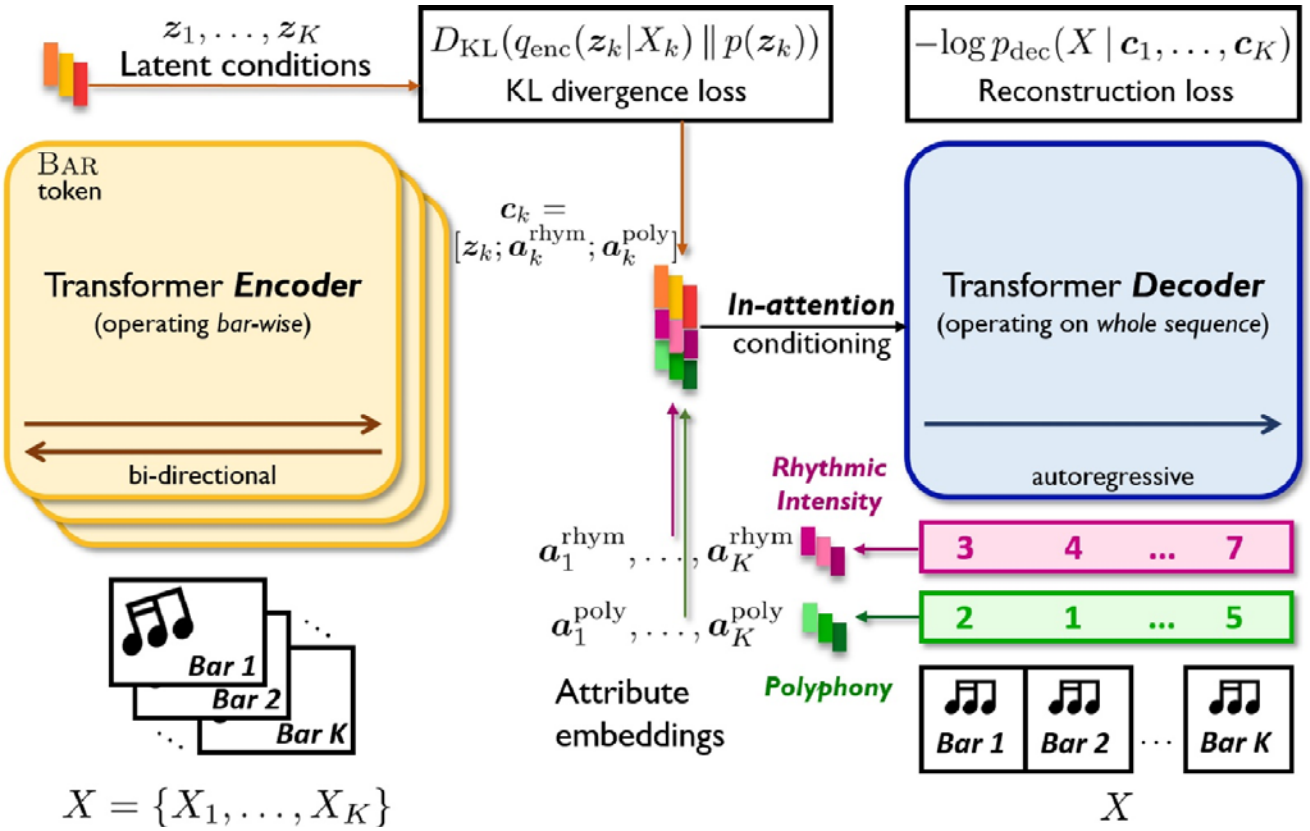
Theme Transformer: Strength and Weakness

- Allow us to repeat something over
- But
  - No development
  - Cannot control the structure

- Question: Can we generate a **skeleton/outline** of the music first, before we fill the details?

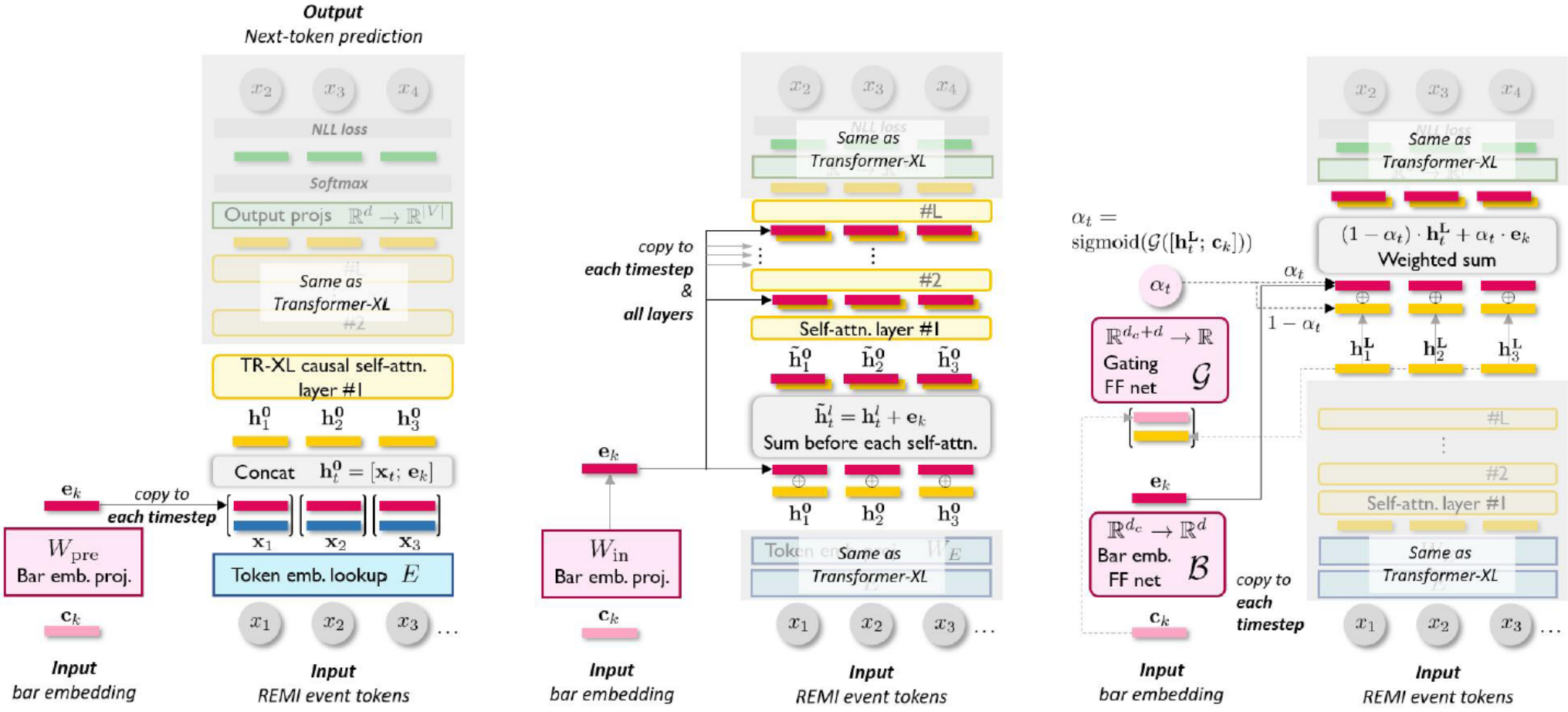
MuseMorphose: VAE-Transformer for Style Transfer

https://slseanwu.github.io/site-musemorphose/



Ref: Wu & Yang, "MuseMorphose: Full-song and fine-grained piano music style transfer with one Transformer VAE", TASLP 2023

MuseMorphose: Various Conditoining Mechansims



(a) Pre-attention conditioning (b) In-attention conditioning (c) Post-attention conditioning

Ref: Wu & Yang, "MuseMorphose: Full-song and fine-grained piano music style transfer with one Transformer VAE", TASLP 2023

MuseMorphose: VAE-Transformer for Style Transfer

Model	Best ckpt. at step	**Recons. NLL** *Train*	*Val.*	**KL divergence** *Train*	*Val.*
MIDI-VAE [19]	65K	0.676	0.894	0.697	0.682
Attr-Aware VAE [20]	190K	0.470	0.688	0.710	0.697
MuseMorphose (Ours), AE objective	90K	0.130	0.139	6.927	6.928
MuseMorphose (Ours), VAE objective	75K	0.697	0.765	0.485	0.484
MuseMorphose (Ours), preferred settings	140K	0.457	**0.584**	0.636	**0.636**

bold: leads the baselines under the same latent space constraints

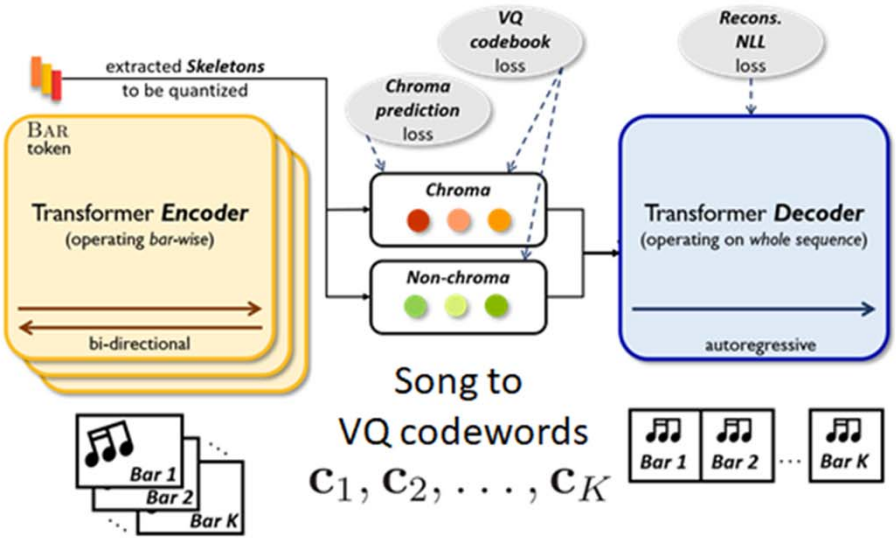
Model	**Fidelity** $sim_{\text{chr}}\uparrow$	$sim_{\text{grv}}\uparrow$	**Control** $\rho_{\text{rhym}}\uparrow$	$\rho_{\text{poly}}\uparrow$	**Fluency** PPL$\downarrow$	**Diversity** $sim_{\text{chr}}\downarrow$	$sim_{\text{grv}}\downarrow$
MIDI-VAE [19]	75.6	85.4	.719	.261	8.84	74.8	86.5
Attr-Aware VAE [20]	85.0	76.8	**.997**	.781	10.7	86.5	**84.7**
Ours, AE objective	**98.5**	**95.7**	.181	.154	**6.10**	97.9	95.4
Ours, VAE objective	78.6	80.7	.931	.884	7.89	**73.2**	84.9
Ours, preferred settings	91.2	84.5	.950	**.885**	7.39	87.1	87.6

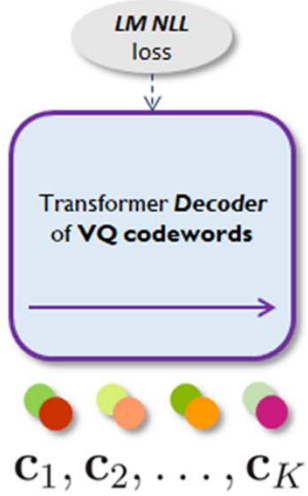
Ref: Wu & Yang, "MuseMorphose: Full-song and fine-grained piano music style transfer with one Transformer VAE", TASLP 2023

VQVAE Transformer for Structured Generation

Low-level Model:
Structure Extractor + Conditional Decoder

High-level Model:
Structure Generator





(Slide made by Shih-Lun Wu; unpublished)

VQVAE Transformer Result

- **Conditional generation:** given VQs from songs in test set
- **Better structure?** -- note n-gram repetition stats

		Short (3∼6 notes)	Mid (7∼15 notes)	Long (16∼40 notes)
Unique %	**Real data**	42.67	60.85	75.49
	VQ model	45.30	64.05	76.95
	Uncond. model	9.32	15.23	27.13
Reappearance interval (in bars)	**Real data**	16.61	23.68	29.22
	VQ model	16.33	23.74	29.08
	Uncond. model	2.74	3.56	4.70

- **Possible plagiarism?** -- len of longest common note string
  - **VQ gen** vs. **Ref real song:** 6.24 +/- 1.98 notes
  - Random real song pairs: 3.79 +/- 1.49 notes

(Slide made by Shih-Lun Wu; unpublished results)

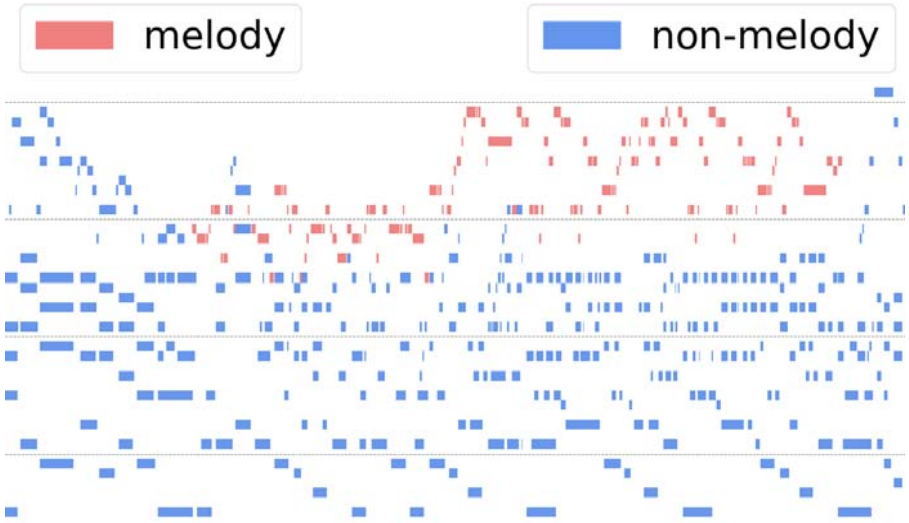
VQVAE Transformer

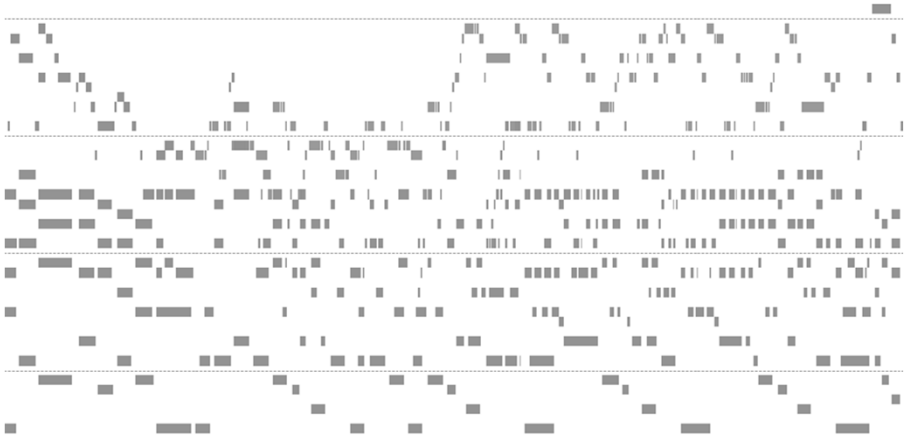
- May allow us to generate a **skeleton/outline** of the music first, before we fill the details
- But
  - Empirical result incomplete (not yet finished)

- Question: The VQVAE approach sounds quite brute-force, is there a **musically-inspired** approach?

Domain Knowledge Inspired Approach

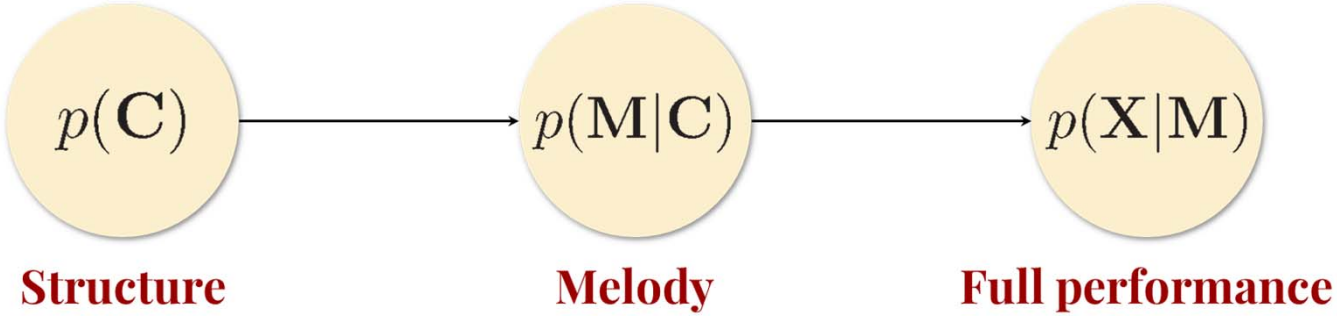
Modular approach > End-to-end approach





Generate melody first,
then the piano accompaniment

Generate Melody First, then the Piano

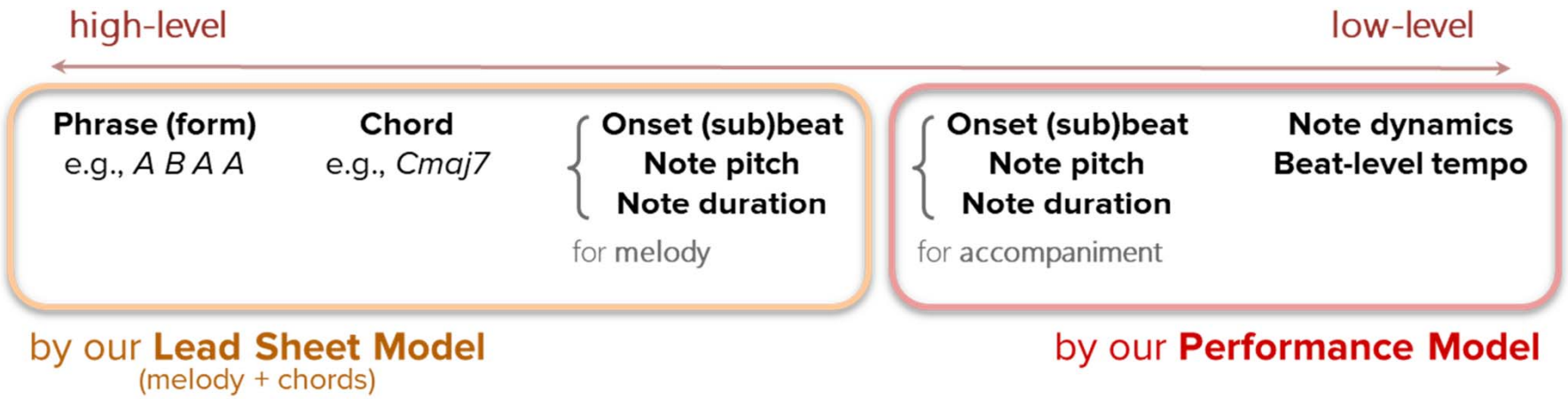


- End-to-end full-song generation is hard; will a 2-stage model make things easier? → Generate **lead sheet** (melody & chords) first, and then the **full piano performance**
- Can we pretrain the melody stage with non-piano data to do better?

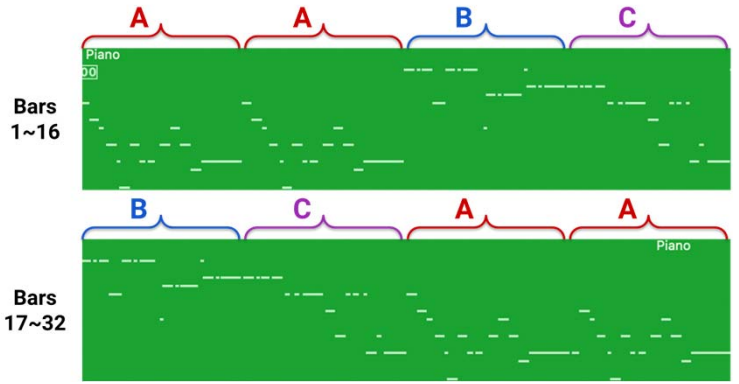
(Slide made by Shih-Lun Wu)

Compose & Embellish

(Figure made by Shih-Lun Wu)



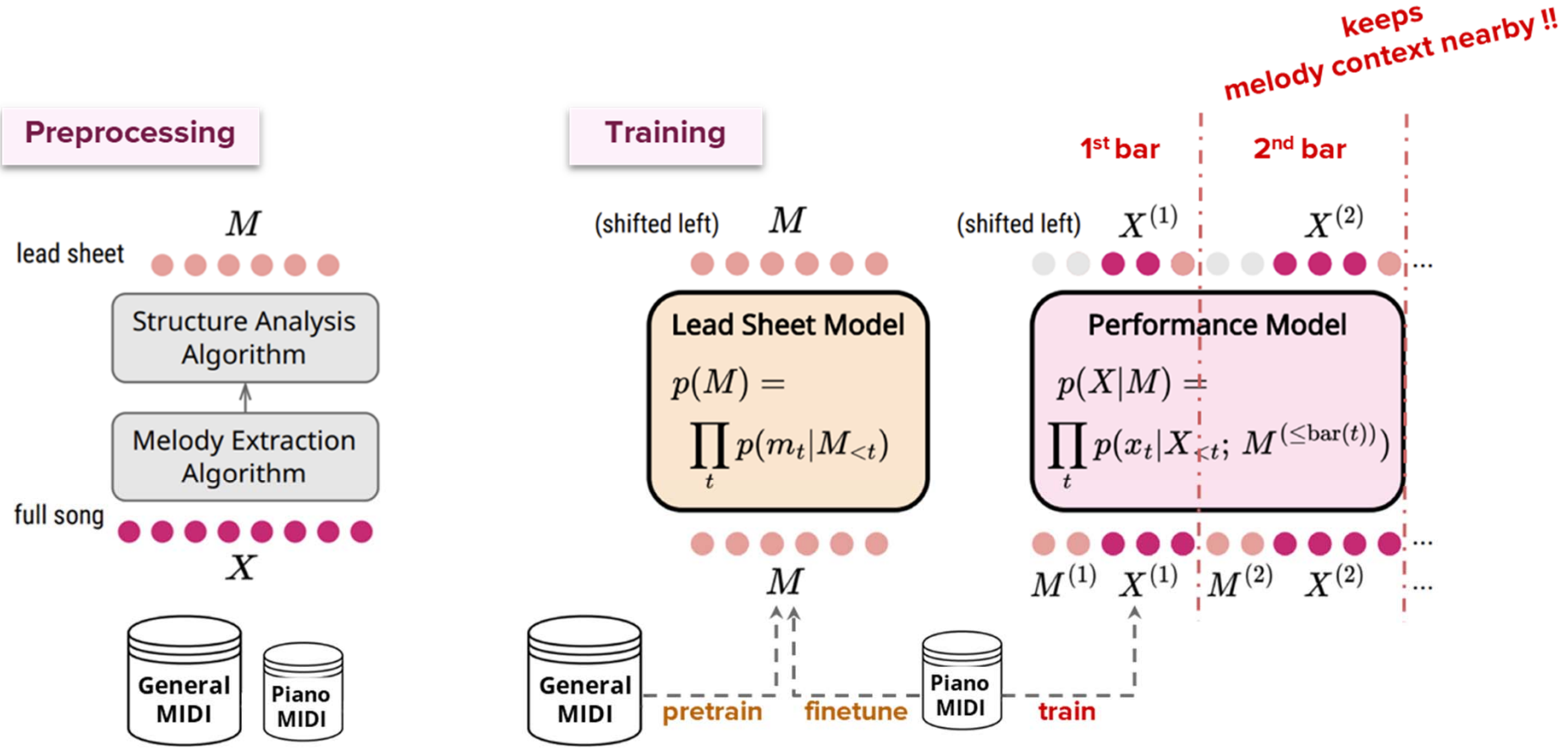
- A two-stage model
  - **Compose** lead sheet & form
  - **Embellish** it to create piano performance



Ref: Wu & Yang, "Compose & Embellish: Well-structured piano performance generation via a two-stage approach", ICASSP 2023

Compose & Embellish

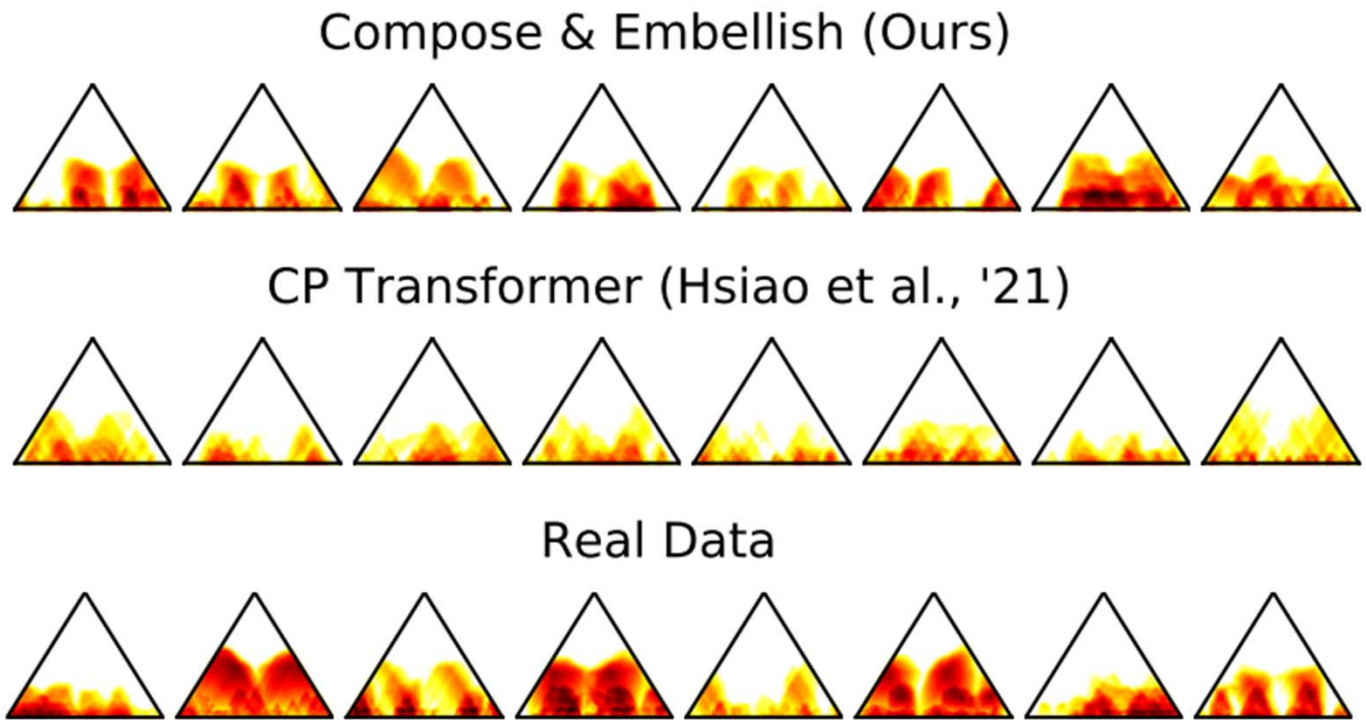
(Figure made by Shih-Lun Wu)



Ref: Wu & Yang, "Compose & Embellish: Well-structured piano performance generation via a two-stage approach", ICASSP 2023

Compose & Embellish: Result

https://github.com/slSeanWU/Compose_and_Embellish



Ref: Wu & Yang, “Compose & Embellish: Well-structured piano performance generation via a two-stage approach”, ICASSP 2023

Compose & Embellish: Result

- Greatly outperforms CP Transformer in structureness
- Ablations show that both the use of structure tokens ("struct") and the lead sheet data pre-training help

	Structureness (in %)			**Melody-line Diversity**		
	$\mathcal{SI}_{\text{short}}$	$\mathcal{SI}_{\text{mid}}$	$\mathcal{SI}_{\text{long}}$	$\mathcal{DN}_{\text{short}}$	$\mathcal{DN}_{\text{mid}}$	$\mathcal{DN}_{\text{long}}$
Naive repeats (1-bar)	83.6 ±8.6	88.3 ±5.4	75.7 ±11	1.2 ±0.6	1.2 ±0.6	1.3 ±0.6
Naive repeats (1-phrase)	75.5 ±15	86.2 ±6.8	73.9 ±11	8.2 ±4.8	10.0 ±5.8	10.8 ±6.5
CP Transformer [4]	32.5 ±3.3	29.9 ±4.4	17.9 ±6.3	82.0 ±8.9	99.6 ±0.7	100 ±0.0
COMPOSE & EMBELLISH	**36.8** ±6.7	**35.1** ±7.7	**25.8** ±12	**49.7** ±19	69.9 ±20	81.6 ±17
w/o struct	**36.8** ±6.8	34.2 ±9.2	23.8 ±11	48.0 ±17	68.7 ±17	82.7 ±14
w/o pretrain	36.6 ±7.5	33.1 ±8.8	19.6 ±10	53.2 ±19	**74.9** ±18	**87.9** ±14
w/o struct & pretrain	36.3 ±6.0	34.1 ±6.7	23.0 ±8.4	52.5 ±18	76.1 ±16	89.0 ±11
Real data	43.8 ±7.1	43.1 ±8.4	34.8 ±12	50.0 ±14	74.1 ±14	88.3 ±11

Ref: Wu & Yang, "Compose & Embellish: Well-structured piano performance generation via a two-stage approach", ICASSP 2023

Compose & Embellish: Result

- 5-point Likert scale on 5 aspects
  - coherence (Ch), correctness (Cr), structureness (S), richness (R), overall (O)

	Ch	Cr	S	R	O
CPT	2.38±0.9	2.49±0.9	2.33±0.9	2.64±0.9	2.33±0.9
C&E (ours)	3.53±0.9	3.11±1.0	3.36±1.2	3.29±1.0	3.18±0.9
Real data	4.42±0.7	4.13±0.8	4.44±0.8	4.24±0.8	4.40±0.7

(StDevs of *individual data points* shown after ±)

- Large (>1 point) gain over CP Transformer on coherence & structureness
- Still a long way to go to rival humans

(Figure made by Shih-Lun Wu)

Recap

- **Theme Transformer**
  - Allow for explicit **conditioning of "theme"** thru customized cross attention design
  - The idea is quite general and applicable to various types of music (melody, single-track piano, multi-track MIDI etc)

- **VAE Transformer (MuseMorphose)**
  - Allow for **time-varying condition**
  - For **style transfer**, and variation generation (as it needs a reference MIDI)
  - The idea can be extended to multi-track MIDI
  - The input can take audio files instead

Recap

- **VQVAE Transformer**
  - A brute-force approach for unconditional MIDI generation that generates a **blueprint** first, before filling the details
  - Can be extended to multi-track MIDI

- **Compose-and-embellish**
  - A musically-inspired approach for unconditional MIDI generation that generates a ***lead sheet* as the blueprint** first, before filling the details
  - More interpretable, interactive; can use unpaired lead sheet data for pretraining
  - Can be extended to multi-track MIDI
  - Can be combined with Theme Transformer
  - The input can take audio files instead

Outline

- Music structure analysis
- Objective evaluation metrics for MIDI generation
- Structured MIDI generation
- **HW3**