Center for Research in Entertainment and Learning

Discrete Musical Systems

Shlomo Dubnov

UCSD

CELFI Seminar, Lecture 4

Methods of Proposed Music Systems Theory

Combination of DES and Machine Learning

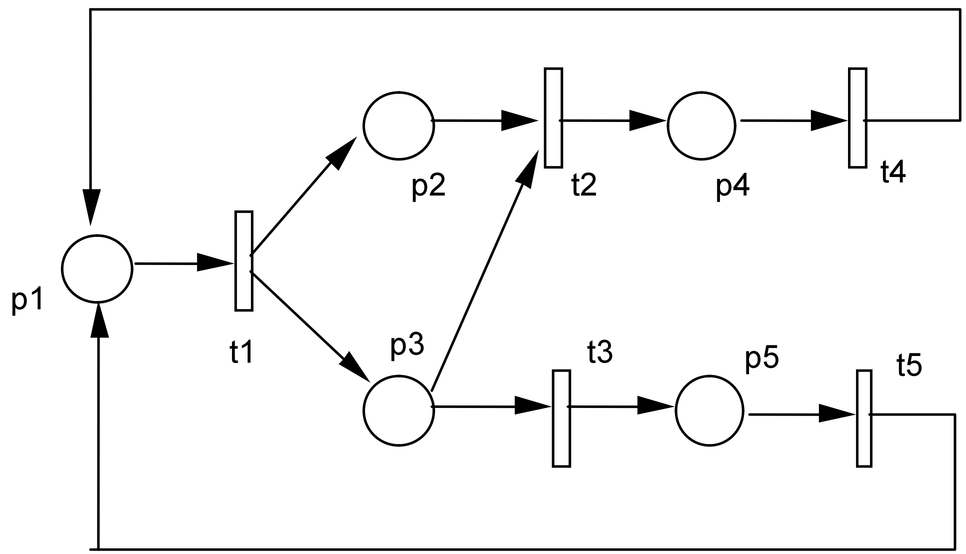
- DES:
  - Finite State Automata
  - Markov Processes
  - Petri Nets
- Machine Learning:
  - Learning FSA
  - Time Series Data mining
  - Reinforcement Learning *
  - Structure and Concurrence Modeling

Petri Nets and Music Structure

Show how to learn and control a large scale musical form using audio segmentation and Petri Nets

Places and transitions

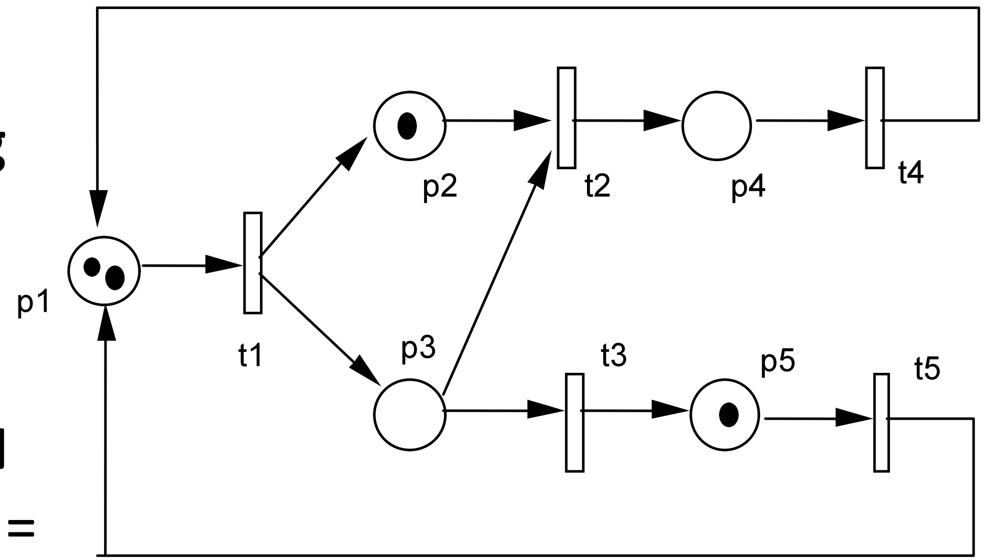
- **A PETRI NET** is a bipartite graph which consists of two types of nodes: **places** and **transitions** connected by directed arcs.
- Place = circle, transition = bar or box.
- An arc connects a place to a transition or a transition to a place.
- No arcs between nodes of the same type.
- **Input and output places** of a transition
- **Input and output transitions** of a place



Token and marking system state

Each place **pi** contains a number of **tokens**.

The distribution of tokens in the Petri net is called **marking** $M = (m_1, m_2, \ldots, m_n)$ where m_i = # of tokens in place **pi**

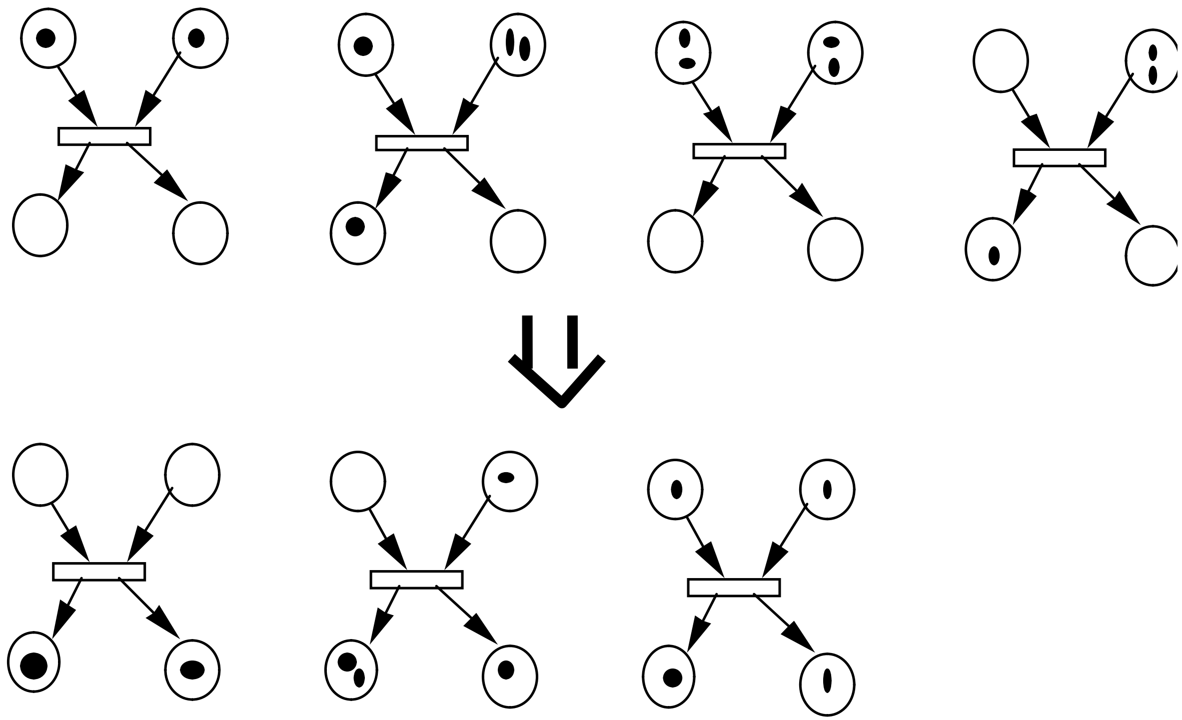


System State = marking of PN
The initial state of the system = initial marking M_0.

System dynamics by transition firing

- A **transition** is said **enabled** (firable) if each of its input places contains at least one token. An enabled transition can fire.
- **Firing a transition** removes a token from each input place and add one token to each ouput place.
- Firing a transition leads to a new marking that enables other transitions.
- The dynamic behavior of the corresponding system = evolution of the marking and transition firings
- Convention: **simultaneous transition firings are forbidden**.

Firing Example



Formal definitions

Petri Nets

A Petri net is a five-tuple PN = (P, T, A, W, M_0) where:

- P = { p_1, p_2, ..., p_n} is a finite set of places
- T = { t_1, t_2, ..., t_m } is a finite set of transitions
- A $\subseteq$ (P×T) $\cup$ (T×P) is a set of arcs
- W : A → { 1, 2, ... } is a weight function
- M_0 : P → { 0, 1, 2, ... } is the initial marking
- P $\cap$ T = Φ and P $\cap$ T = Φ

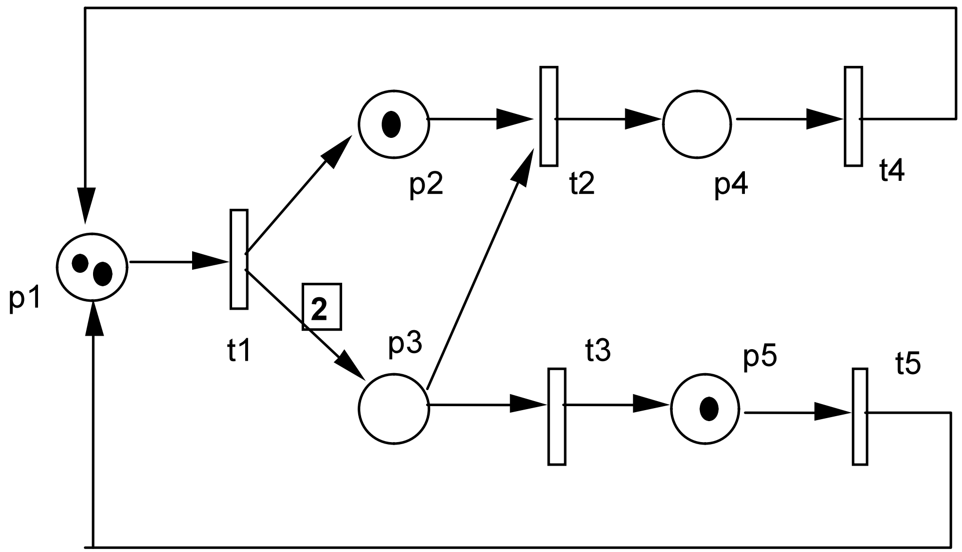
PN without the initial marking is denoted by N:

- N = (P, T, A, W)
- PN = (N, M0)

A Petri net is said **ordinary** if w(a) = 1, $\forall$a $\in$ A.

Graphic representation

Similar to that of ordinary PN but with default weight of 1 when not explicitly represented.



Transition firing

Rule 1: A **transition** t is **enabled** at a marking M if M (p) ≥ w(p, t) for any p ∈ ^{o}t where ^{o}t is the set of input places of t

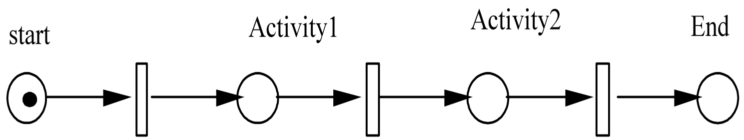
Rule 2: An enabled transition may or may not fire.

Rule 3: **Firing transition** t results in:

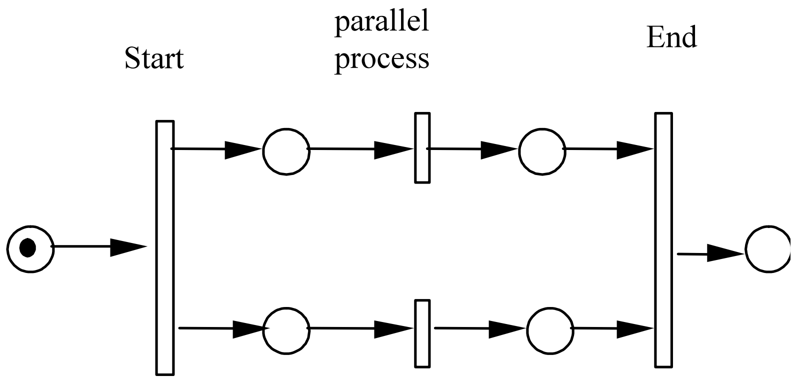
- removing w(p, t) tokens from each p ∈ ^{o}t
- adding w(t, p) tokens to each p ∈ t^{o} where t^{o} is the set of output places of t

PN models of key characteristics

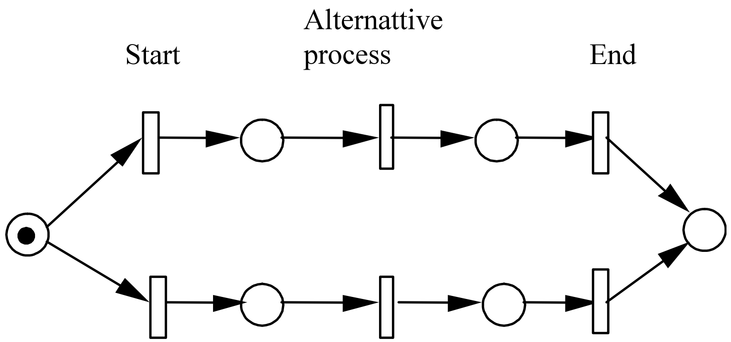
Precedence relation:



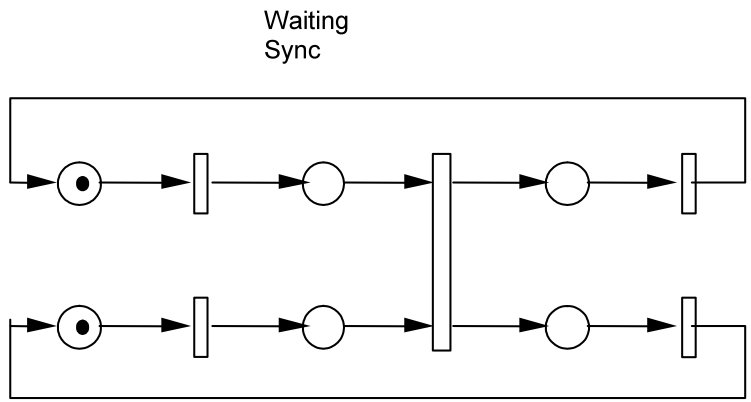
Parallel processes:



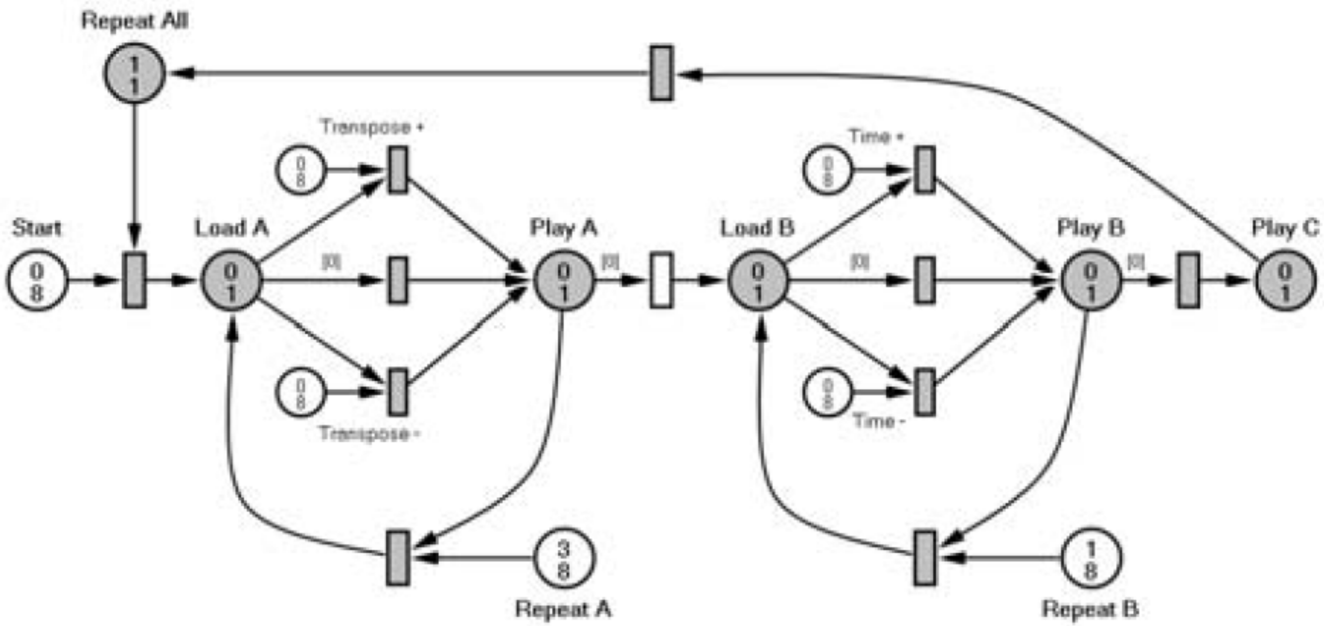
Alternative processes:



Synchronization:



Music Modeling with PN



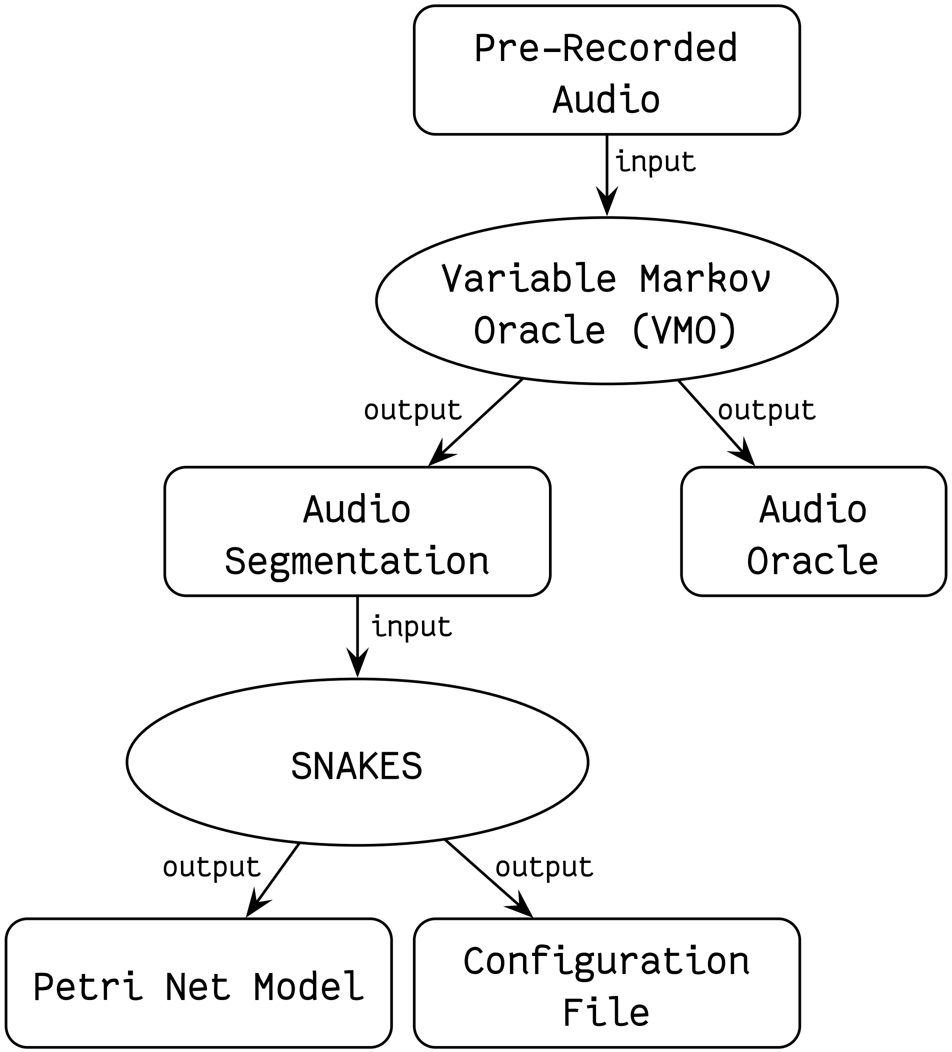
Frank Zappa “Peaches and Regalia” analysis by A. Baratè

Automatic Modeling of Musical Structure using PN

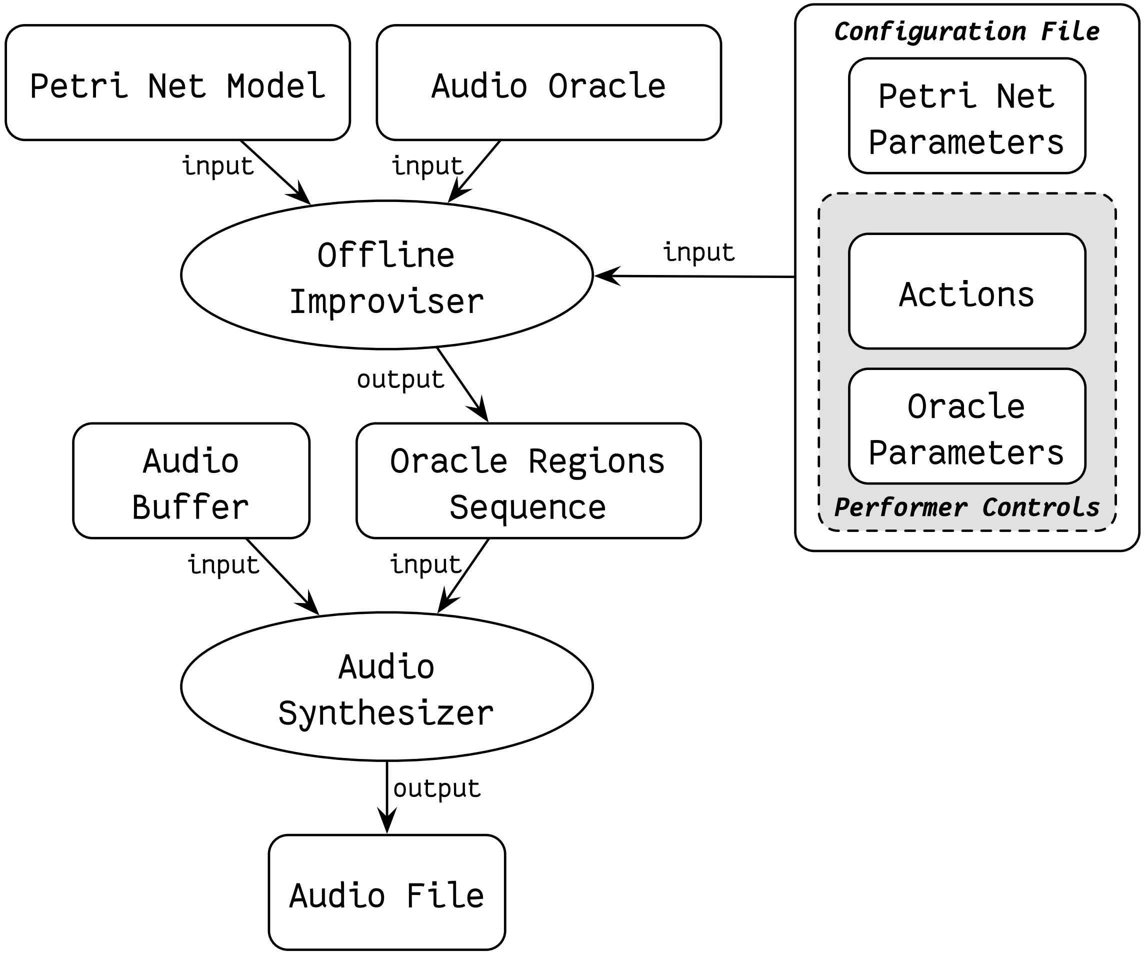
Overview of the system

- PyOracle: https://gitlab.com/himito/PyOracle_I-score
- i-score: https://github.com/himito/i-score
- VMO-Score: https://himito.github.io/vmo_i-score_generator

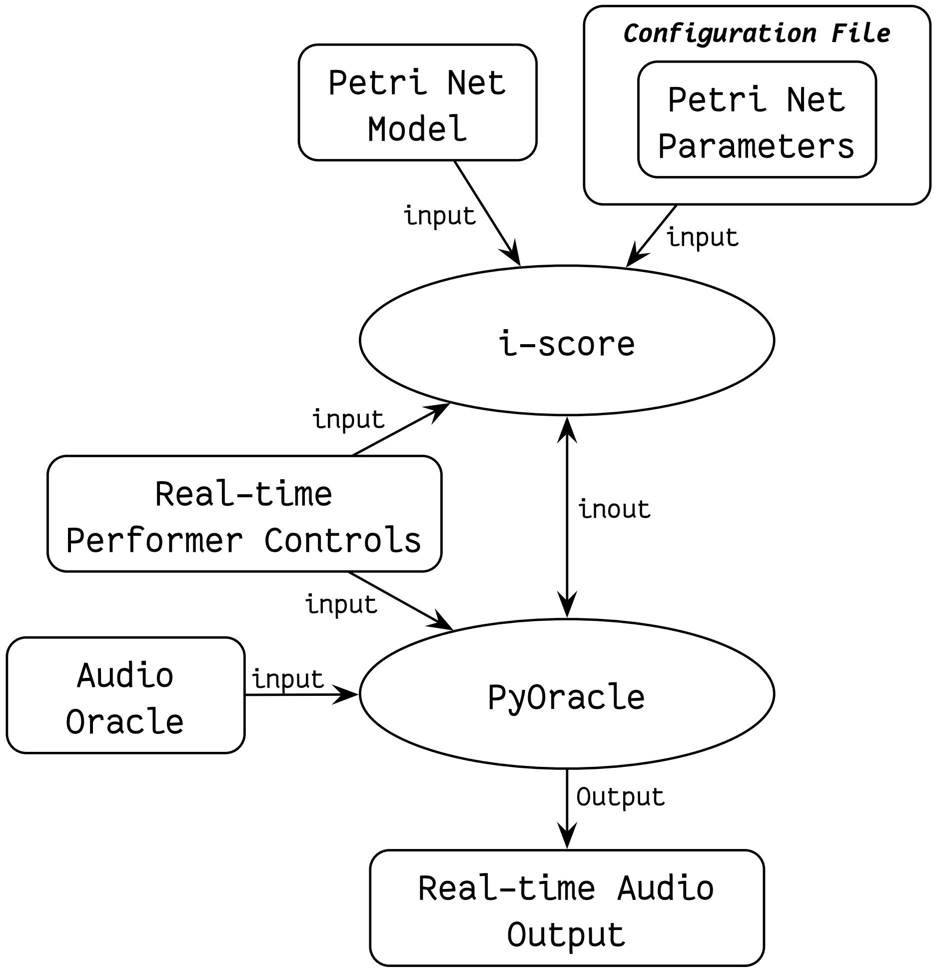
Overview of the system



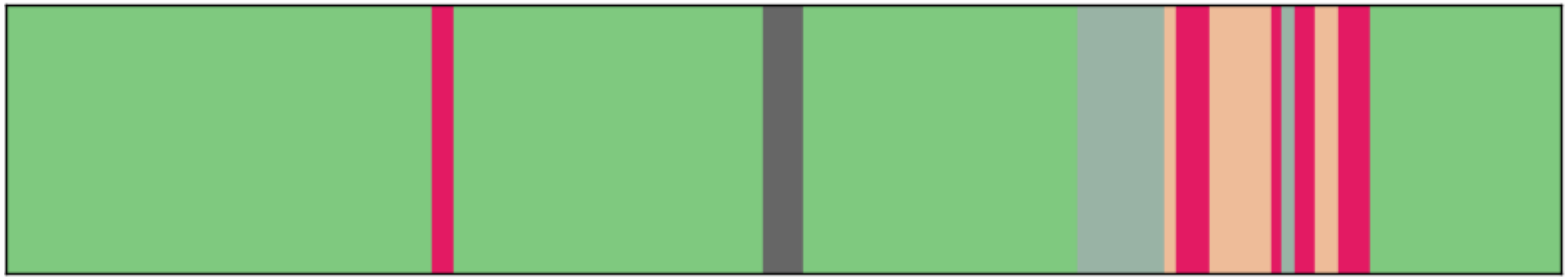
Offline Improvisation

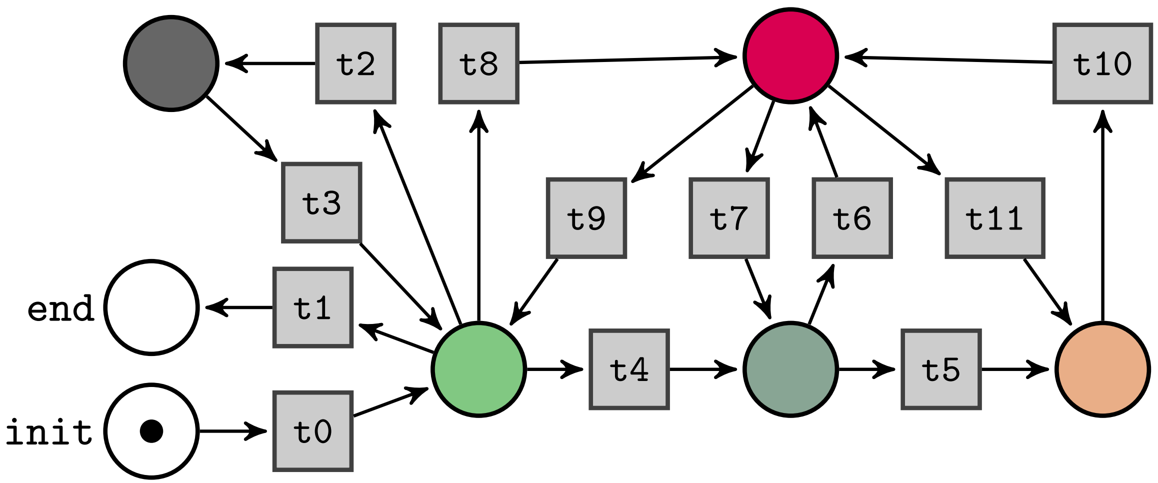


Real Time Improviser



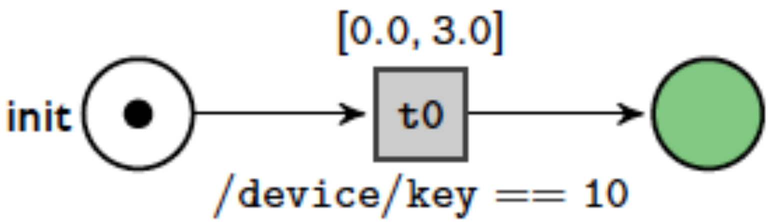
Segmentation





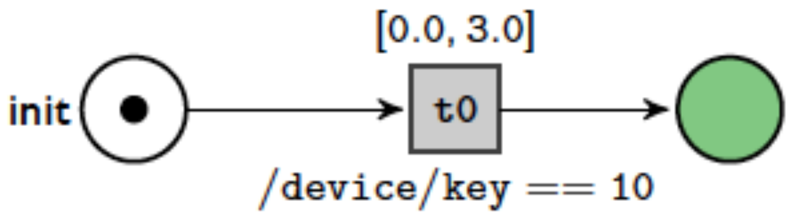
Composition with PN

```
# file: configuration.yml

conditions:
  - transition : 't0'
    time-min   : 0.0
    time-max   : 3.0
    condition  : '/device/key == 10'

  - transition : 't1'
    ...
```

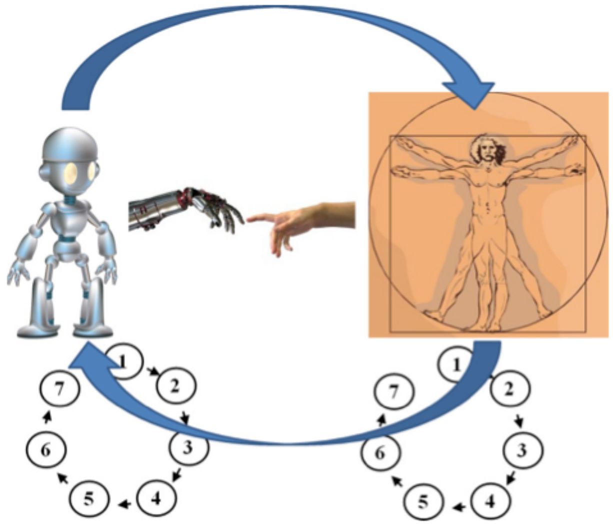


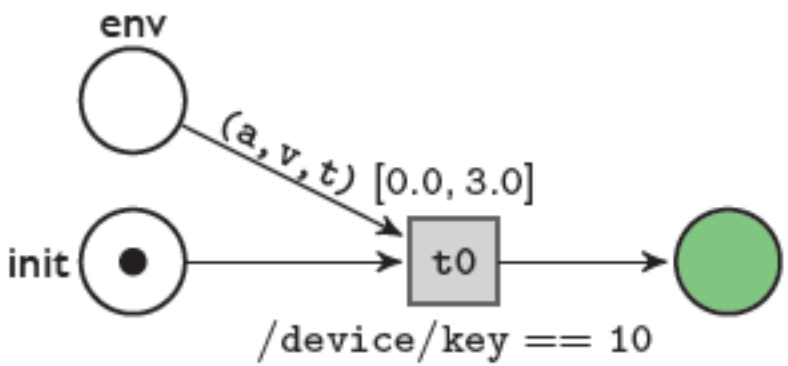
Improvisation with PN

```yaml
# file: configuration.yml

actions:
  - address : '/volume/sensor/pos_x'
    value   : 10
    time    : 250

  - address : ...
```

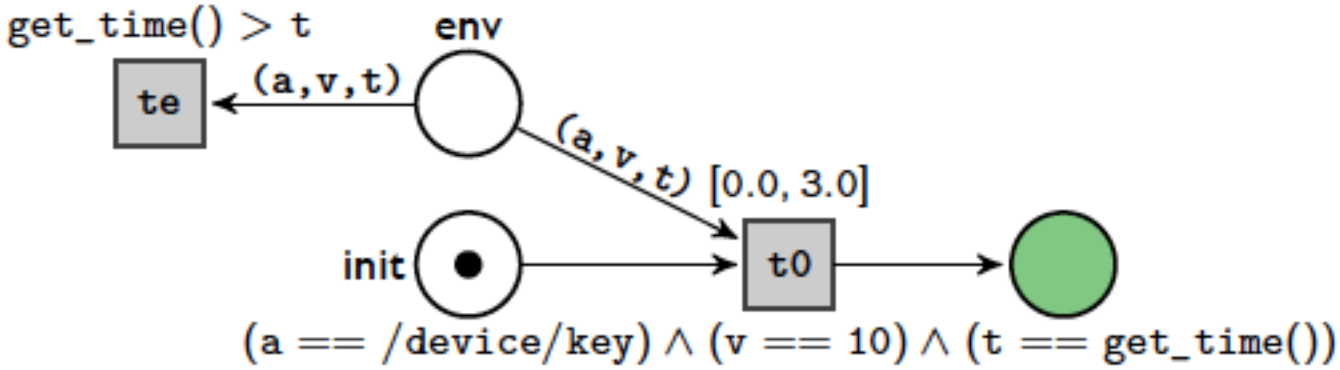


Modeling Interaction

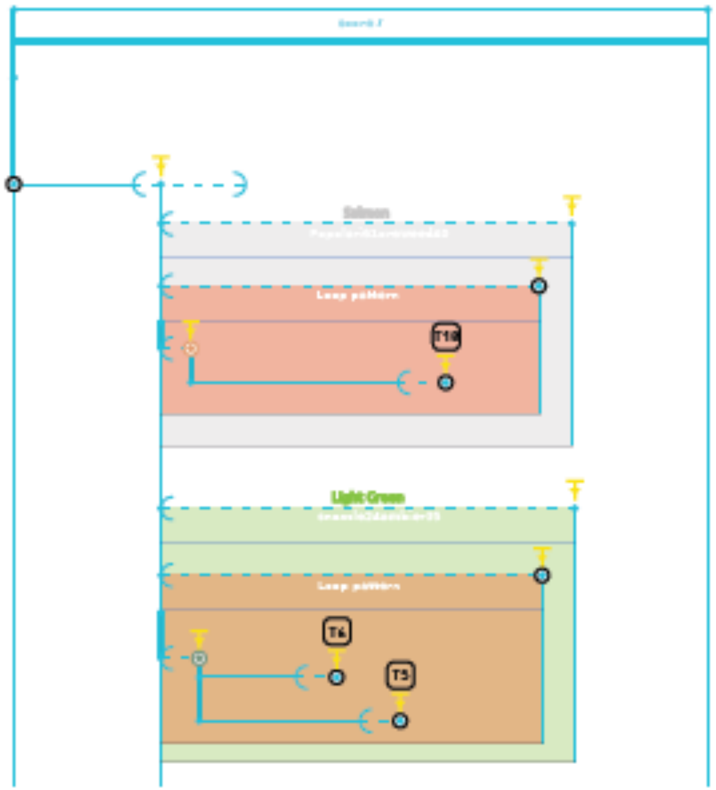




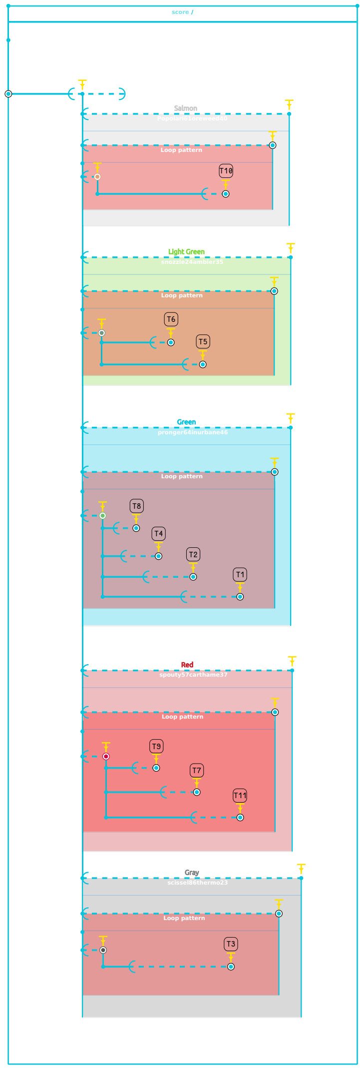
Adding Environment

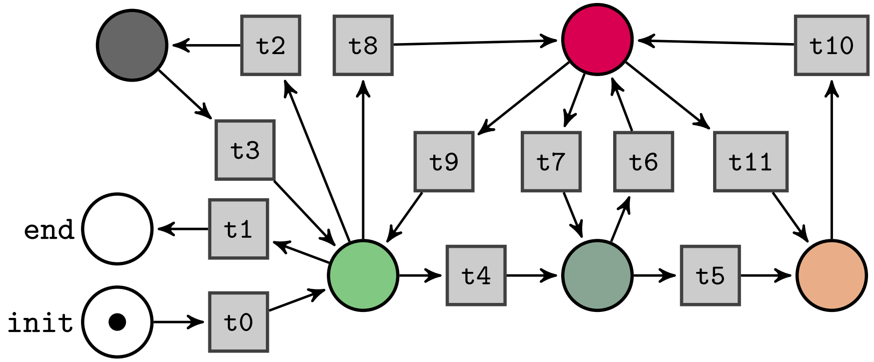


i-score representation



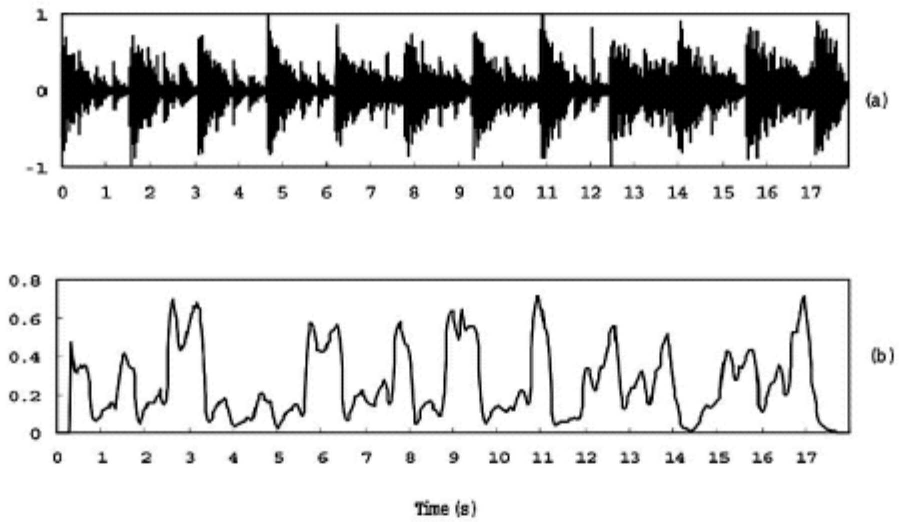
PN to i-score mapping

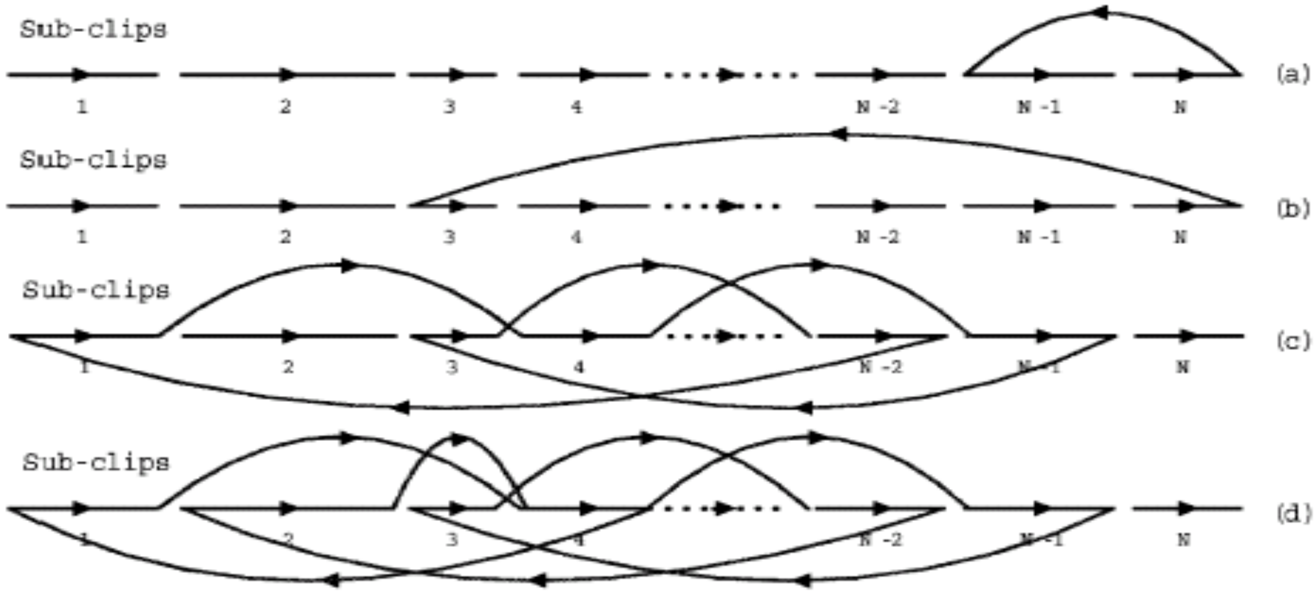




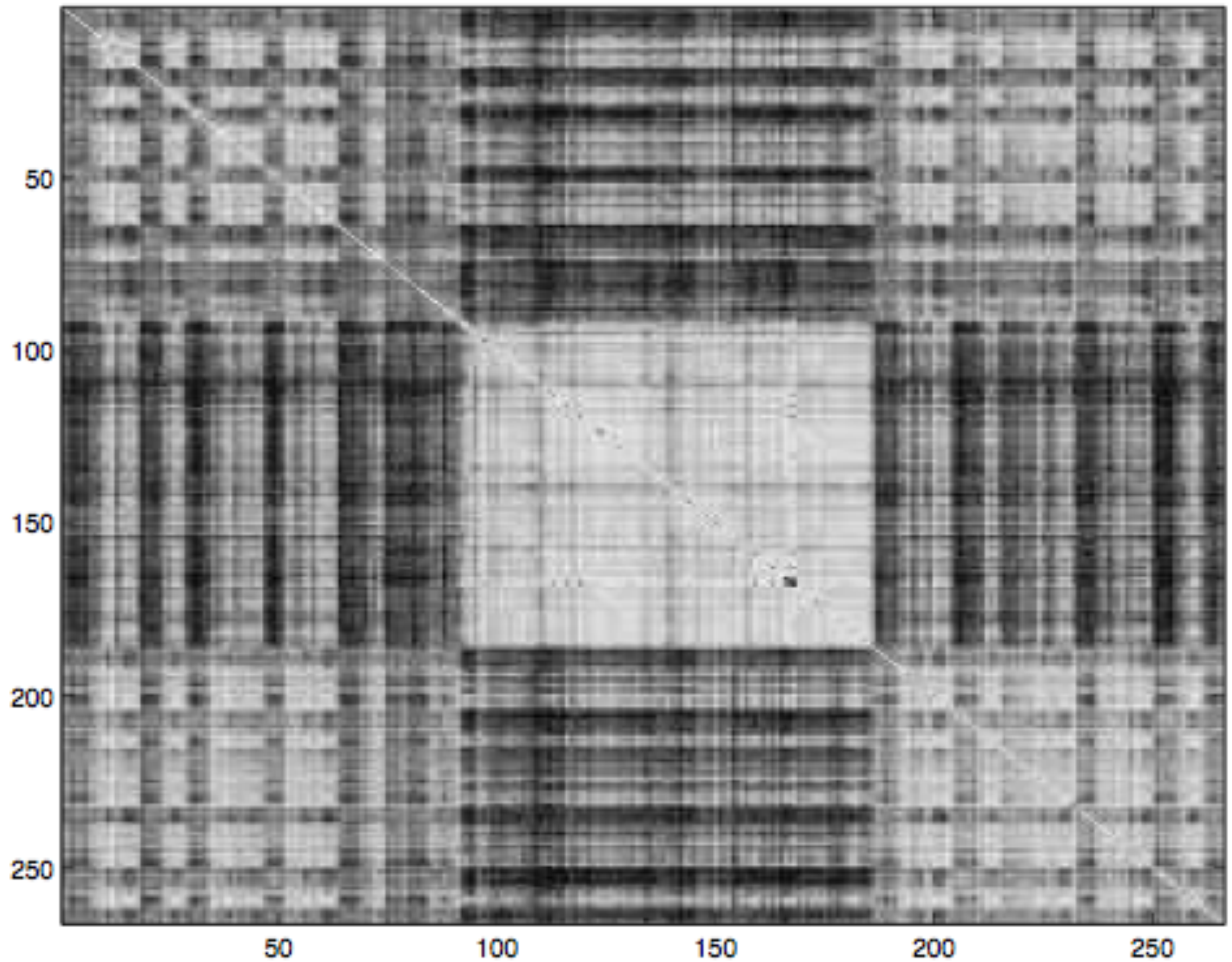
Segmentation

Using self-similarity (recurrence) to partition an audio file inter-connected regions by spectral clustering


1
0
-1
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
(a)
0.8
0.6
0.4
0.2
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
(b)
Time (s)


Sub-clips
1
2
3
4
N -2
N -1
N
(a)
(b)
(c)
(d)

Recurrence Matrix



$$d(i, j) = \frac{\langle X_i, X_j \rangle}{\|X_i\| \|X_j\|}.$$

SVD

$$\mathbf{X}_{n\times p} = \mathbf{U}_{n\times n}\mathbf{\Lambda}_{n\times p}\mathbf{V}^T_{p\times p}$$

where

$$\mathbf{U}^T\mathbf{U} = I_{n\times n}, \text{ and } \mathbf{V}^T\mathbf{V} = \mathbf{I}_{p\times p}.$$

SVD – Geometric Interpretation

'spectral decomposition' of the matrix

$$\begin{bmatrix} \\ X \\ \\ \end{bmatrix} = \begin{bmatrix} | & | \\ \boldsymbol{u_1} & \boldsymbol{u_2} \\ | & | \end{bmatrix} \cdot \begin{bmatrix} \lambda_1 & \varnothing \\ \varnothing & \lambda_2 \end{bmatrix} \cdot \begin{bmatrix} \text{---} & \boldsymbol{v_1} & \text{---} \\ \text{---} & \boldsymbol{v_2} & \text{---} \end{bmatrix}$$

SVD – Geometric Interpretation

'spectral decomposition' of the matrix:

m

n

$$\left[\; X \;\right] = \lambda_1 \;\; \boldsymbol{u}_1 \;\; \boldsymbol{v}^T{}_1 + \lambda_2 \;\; \boldsymbol{u}_2 \;\; \boldsymbol{v}^T{}_2 + \ldots$$

SVD – Geometric Interpretation

'spectral decomposition' of the matrix:

m

n

r terms

$$\left[\; X \;\right]_{n \times m} = \lambda_1 \; \boldsymbol{u}_1 \; \boldsymbol{v}^T_1 + \lambda_2 \; \boldsymbol{u}_2 \; \boldsymbol{v}^T_2 + \ldots$$

n x 1

1 x m

SVD – Geometric Interpretation

Approximation / dim. Reduction - by keeping the first few terms (how many?)

m

n

$$X = \lambda_1 \, \boldsymbol{u}_1 \, \boldsymbol{v}^T_1 + \lambda_2 \, \boldsymbol{u}_2 \, \boldsymbol{v}^T_2 + \ldots$$

assume: $\lambda_1 >= \lambda_2 >= \ldots$

$$\mathbf{X}_{n\times p} = \mathbf{U}_{n\times n}\mathbf{\Lambda}_{n\times p}\mathbf{V}^T_{p\times p}$$

where

$$\mathbf{U}^T\mathbf{U} = I_{n\times n}, \text{ and } \mathbf{V}^T\mathbf{V} = \mathbf{I}_{p\times p}.$$

$$\mathrm{X} = \mathrm{U}\mathrm{S}_x$$

U – orth. basis vectors

$$\mathrm{S}_x = \Lambda_x \mathrm{V}^T$$
$$\mathbf{D} = \mathrm{S}_x^T \mathrm{S}_x.$$

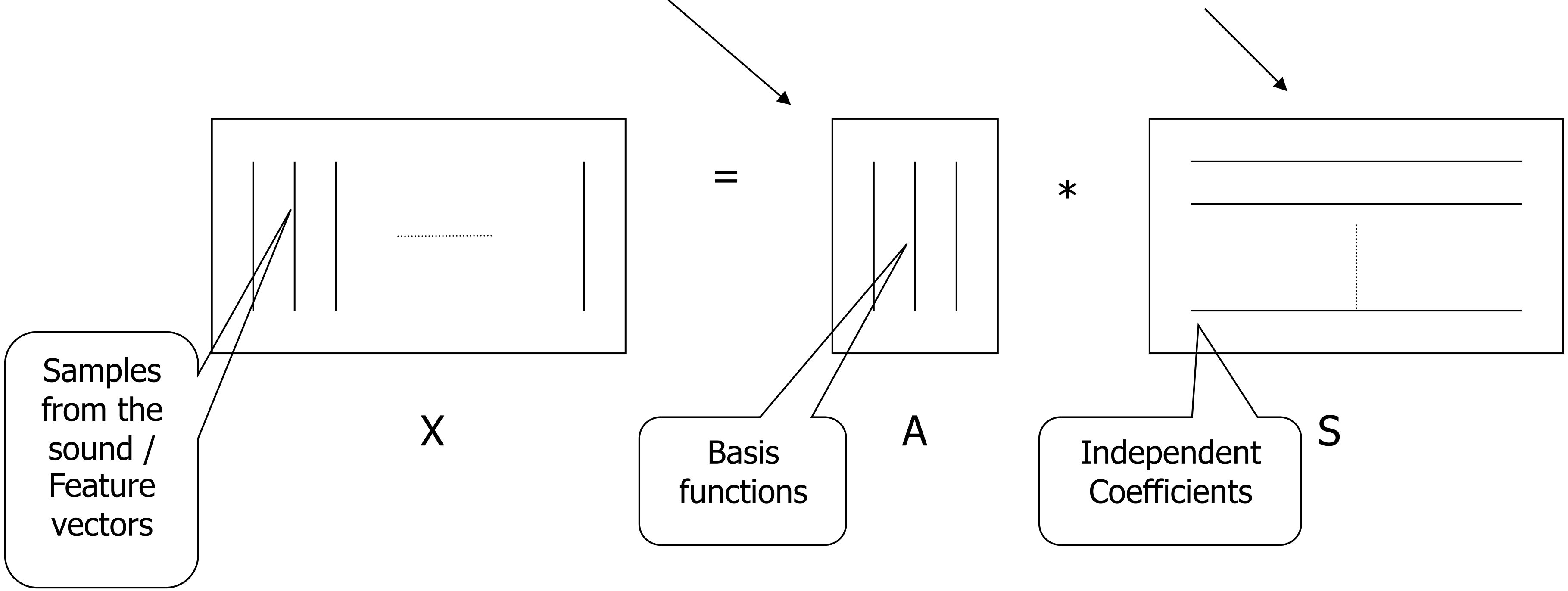
Property I: Eigenvectors S_d of $\mathbf{D}$ are transpose of the expansion coefficients of X, $S_d = S_x^T$.

Relation between SVD and Self-Similarity based clusted

- Matrix D is a correlation matrix
- When self-similarity is computed using dot-product, D becomes a similarity matrix
- Spectral Clustering uses eigenvectors of a normalized similarity matrix to find objects in the data (to be explained next)
  - Side Note: the name “Spectral Clustering” comes from spectral decomposition of a matrix (eigenvectors) and has nothing to do with spectrum of the audio signal

Dimension Reduction / Audio Basis

Eigenvector Clustering



Spectral Clustering

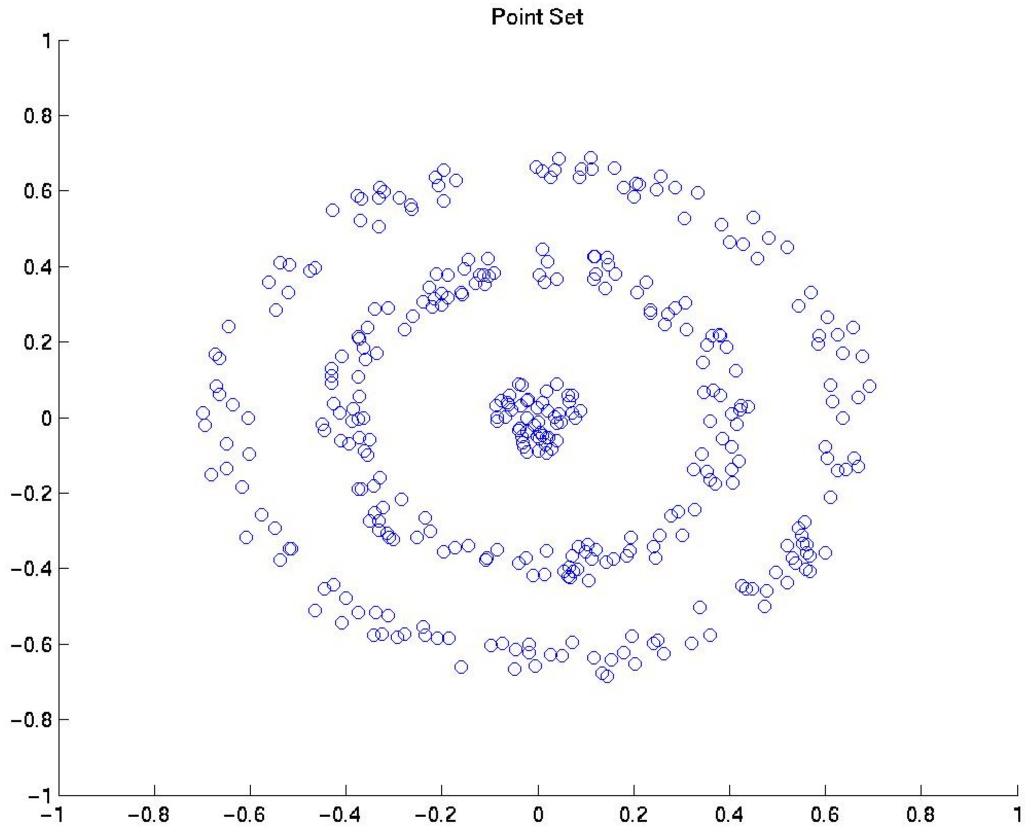
- Transform the similarity matrix D to a stochastic matrix:

$$P = Z^{-1}D$$

- Pij is the probability of moving from sound grain i to sound grain j in one step of a random walk

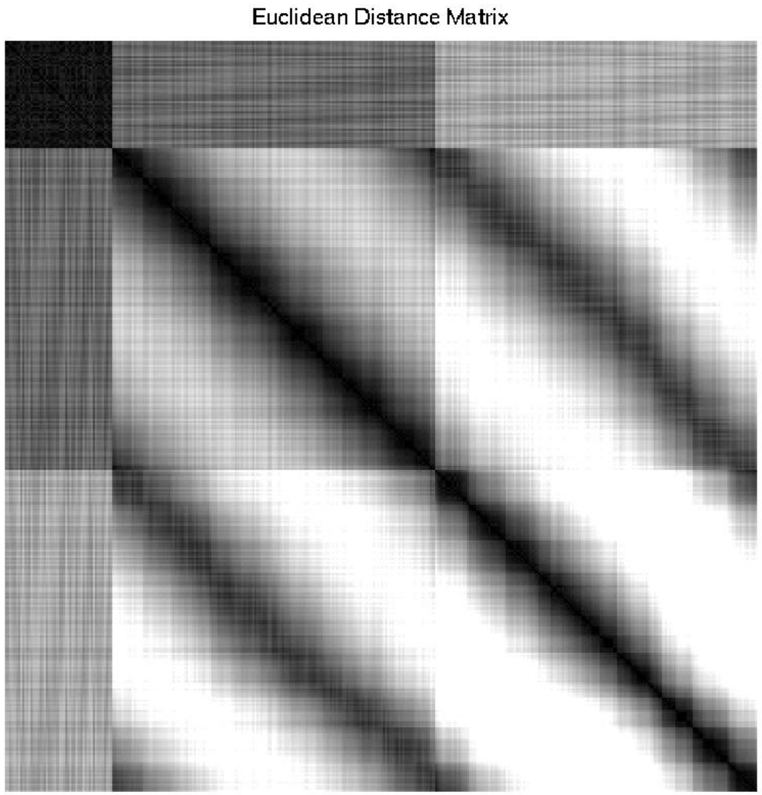
- Same eigenvectors: y =yP; eigenvalues: λ = 1-λP

Synthetic Example I

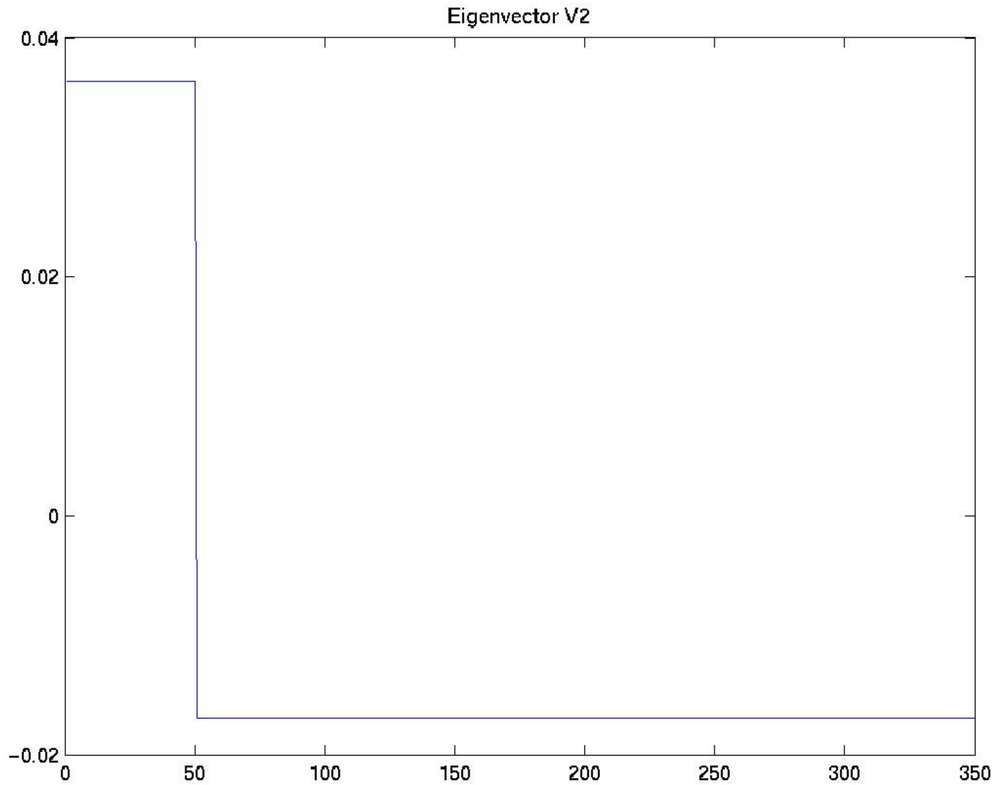


Note: Usual distance based clustering does not work for this type of data.

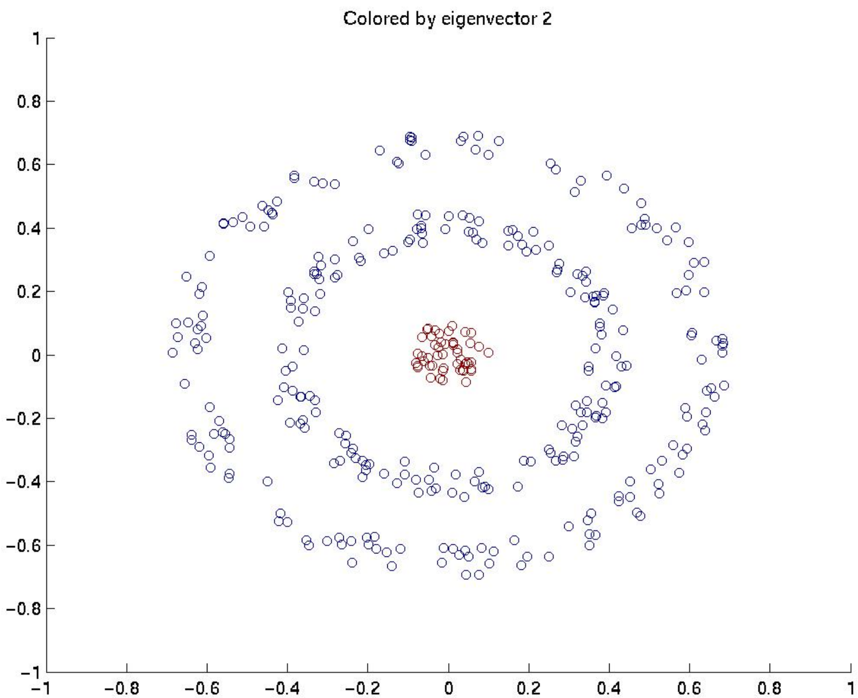
Distance Matrix



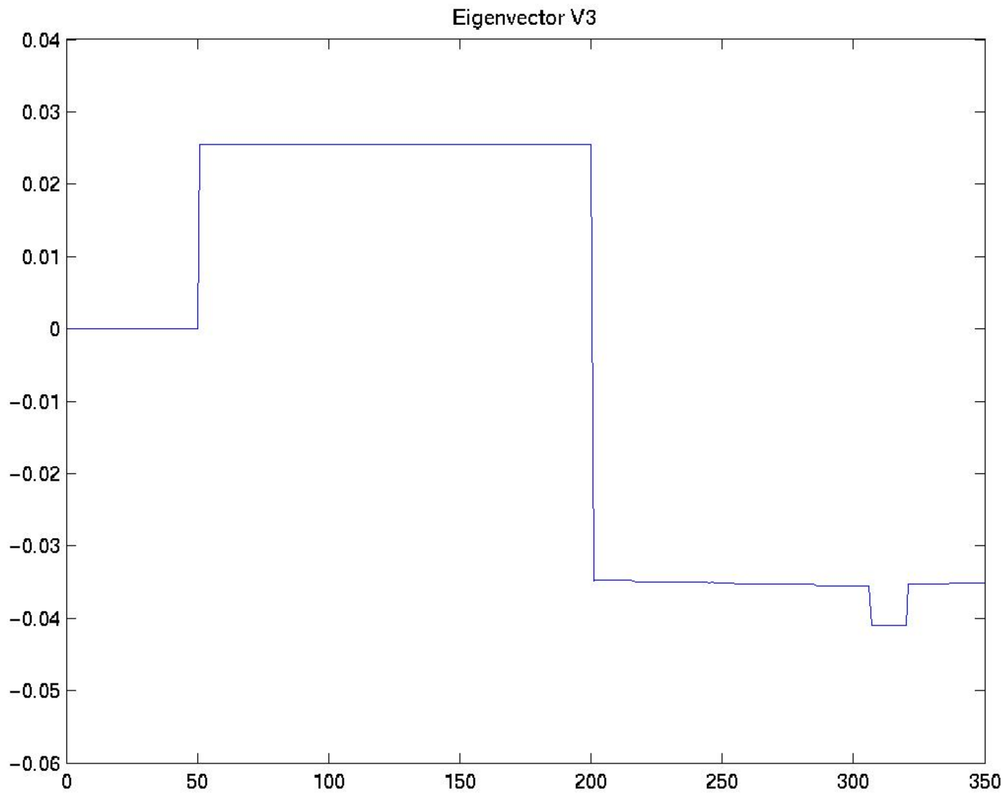
The second generalized eigenvector



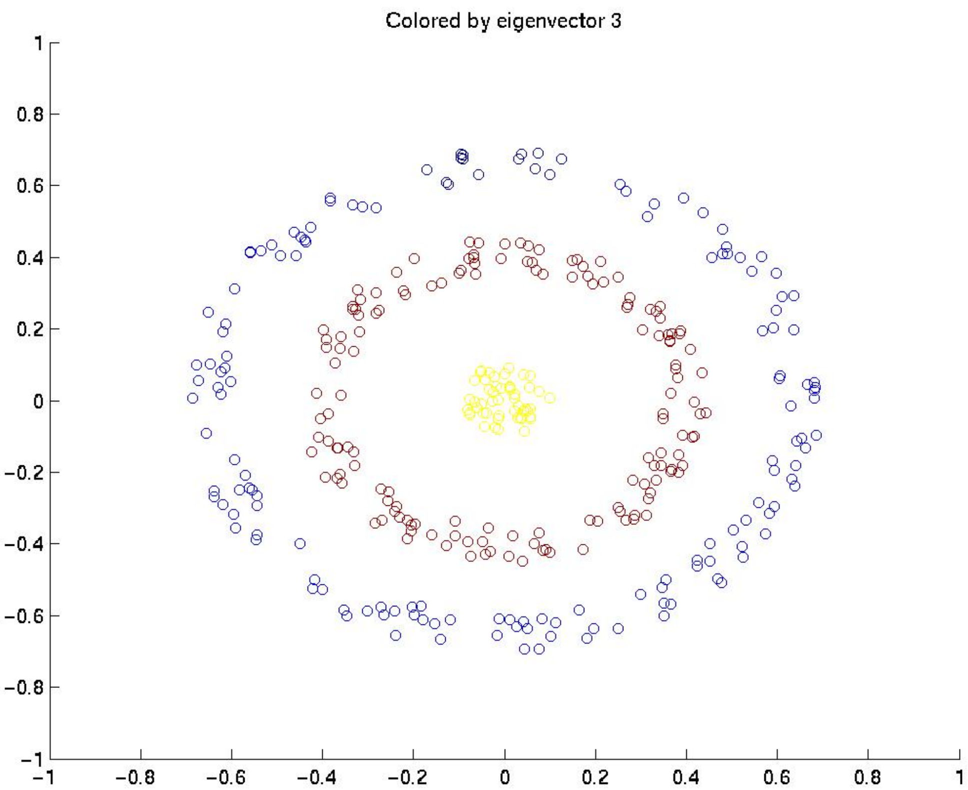
The first partition



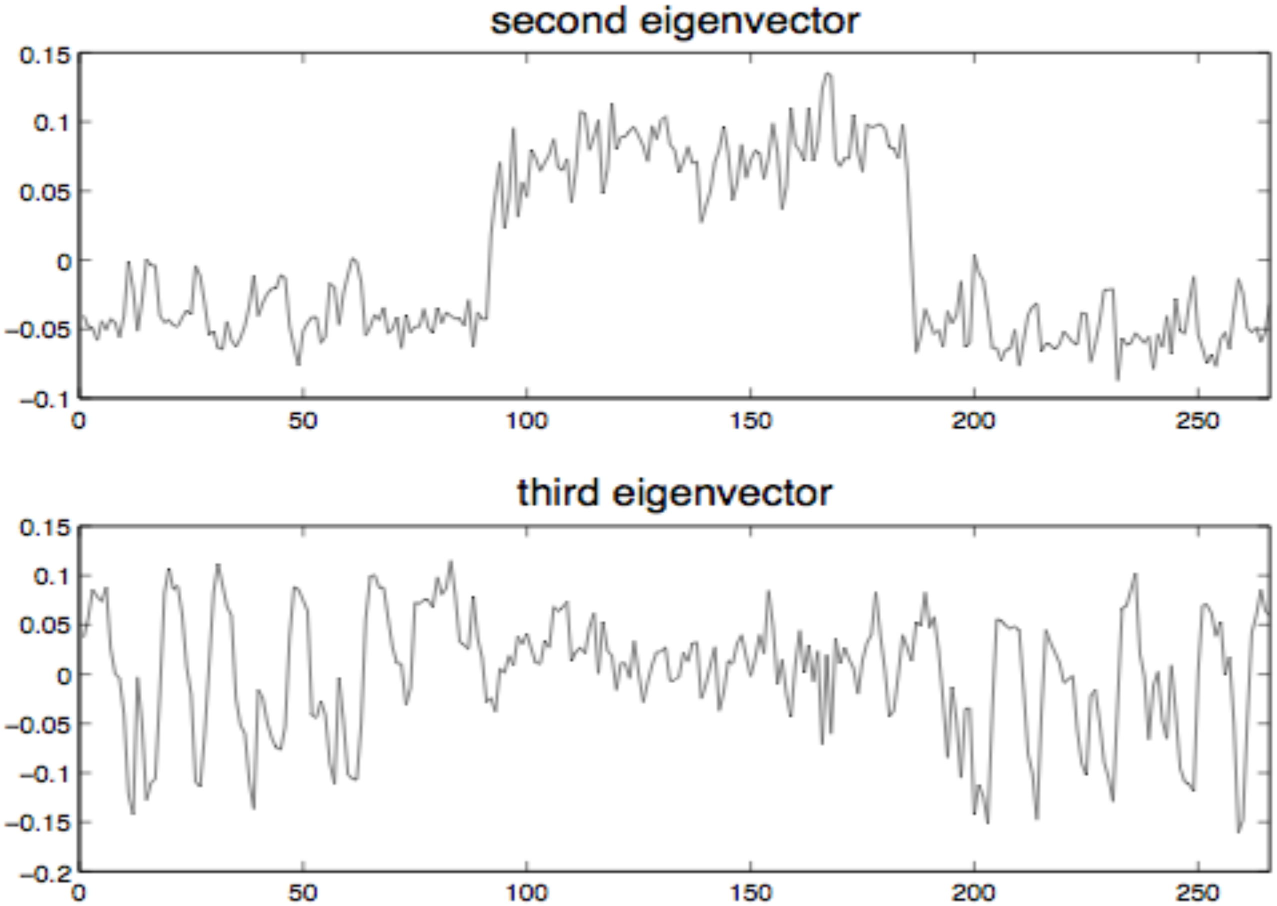
The second generalized eigenvector

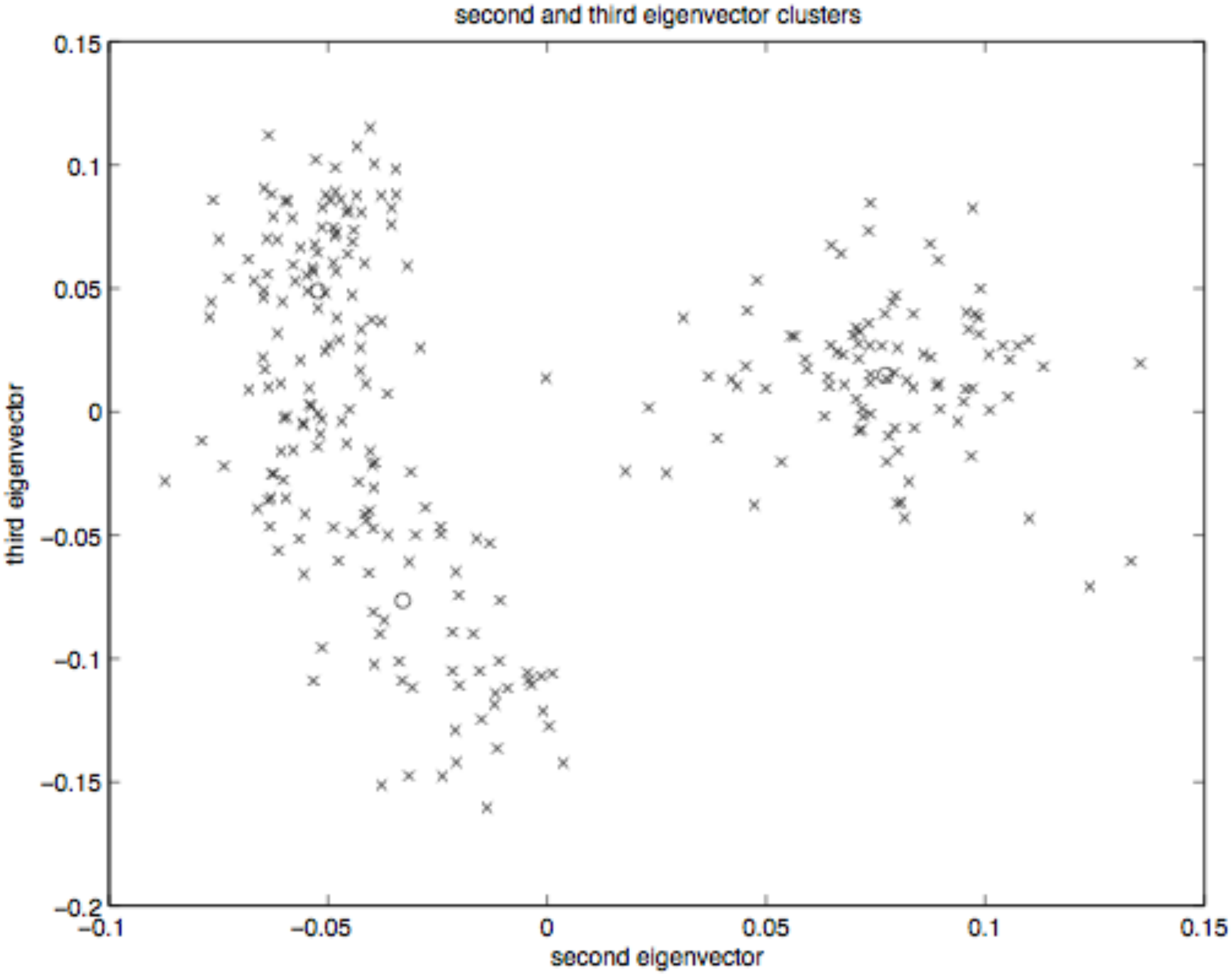


The second partition



Example II: Audio Segmentation using eigenvectors





second and third eigenvector clusters
0.15
0.1
0.05
0
−0.05
−0.1
−0.15
−0.2
third eigenvector
−0.1
−0.05
0
0.05
0.1
0.15
second eigenvector

Example

Original

Segmented

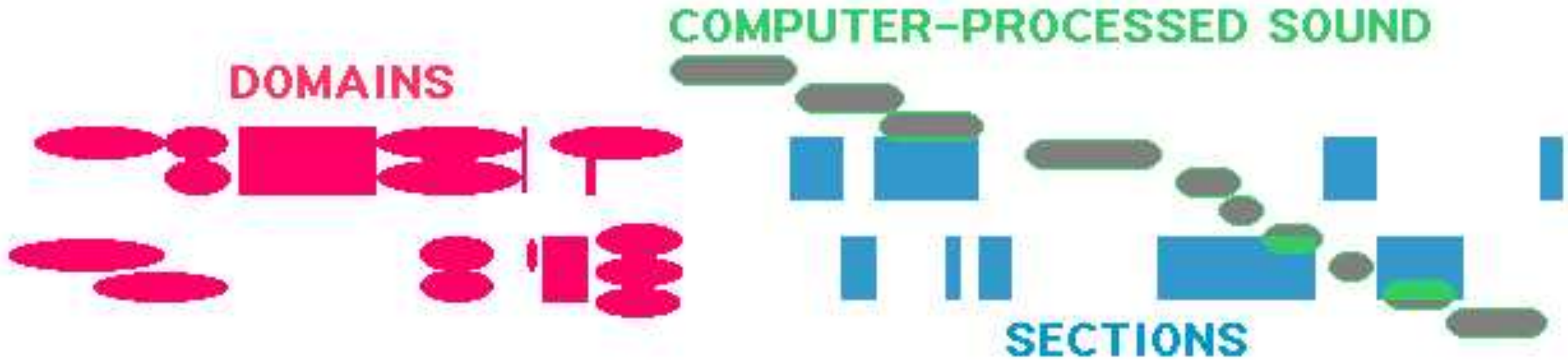
Example III:

The Angel of Death by Roger Reynolds

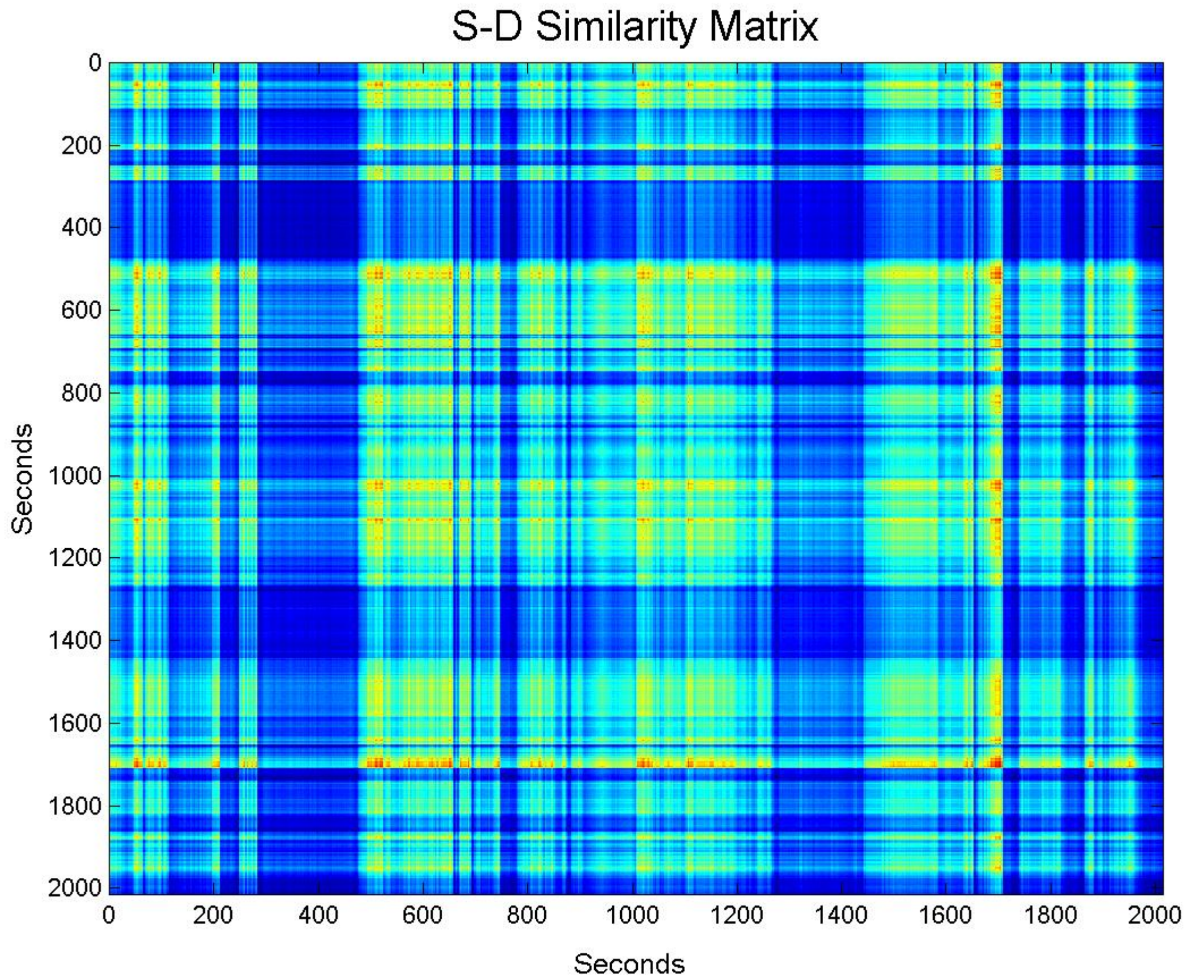
for piano, chamber orchestra and computer-processed sound

S-D VERSION

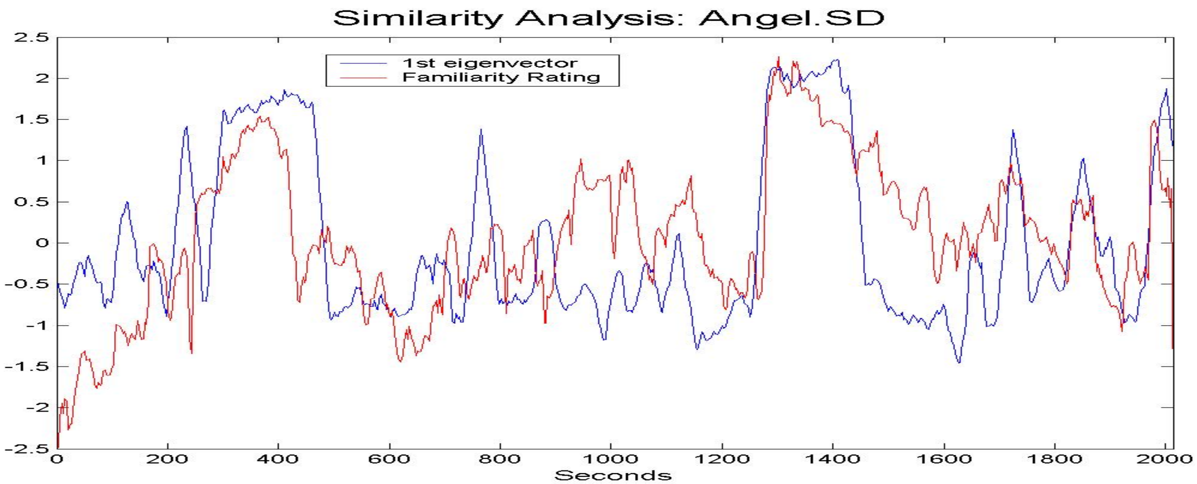
D-S VERSION

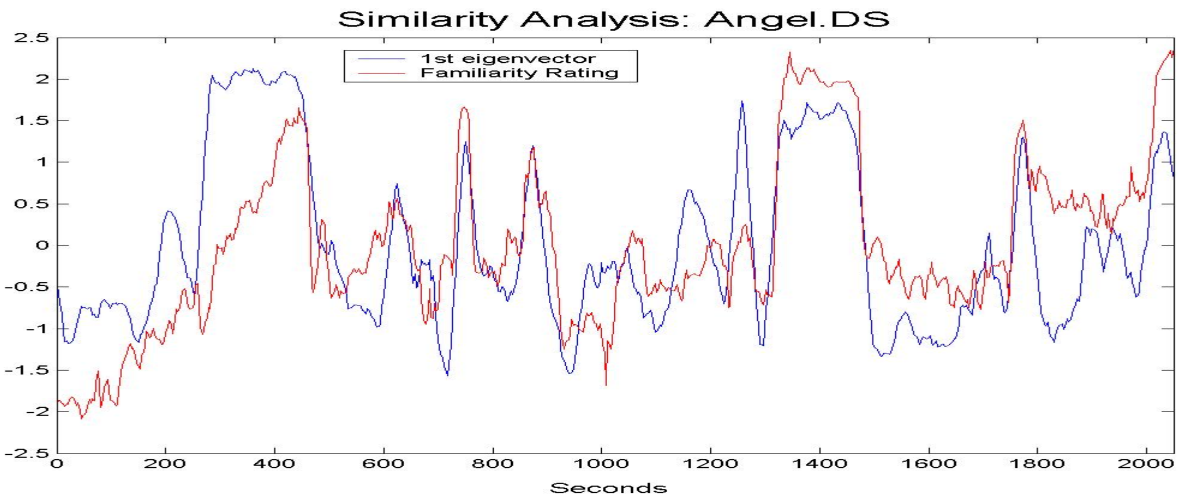


Familiarity vs. Recurrence



Matrix Eigenvector Profile





JASIST, 2006, Special Issue on Style

• ***Segmentation*** based on ***Recurrence*** (Similarity Matrix) can be done directly from the ***first eigenvectors***

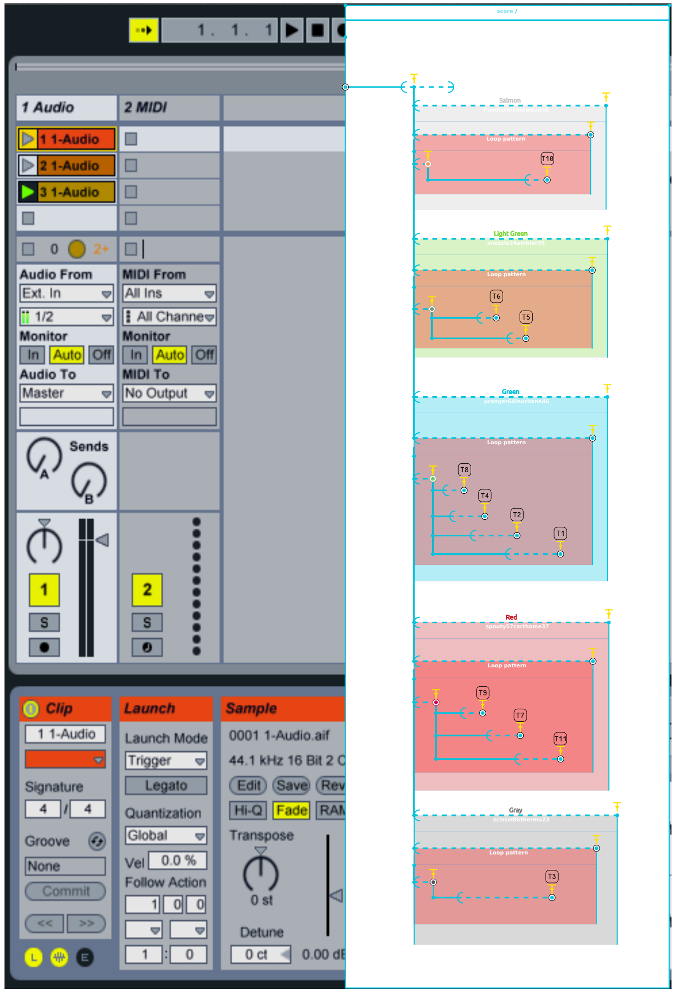
- after proper normalization
- automatic ways to find thresholds could be devised (such as k-means method in ICMC 2004 paper)

• If your ***distance*** function can be expressed as ***dot-product***, then you can ***use SVD*** directly on the ***data***

• segmentation is non-linear with respect to the original data space

• Spectral Clustering can be applied to graph derived from VMO analysis (not covered here – see references)

Summary: New type of DAW



- Automatic Clips Cut
- Improvise
- Follow conditions and actions
- Side-chaining query

References

- Foote, J. and M. Cooper (2001). Visualizing musical structure and rhythm via self-similarity. In Proceedings of the ICMC, pp. 419–422. ICMA.
- Lu, L., S. Li, L. Wenyin, and H. Zhang (2002). Audio textures. International Conference on Acoustics, Speech, and Signal Processing, 1761–1764.
- Shi, J. and J. Malik (2000). Normalized cuts and image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. 22(8), 888–905.
- Dubnov, S. and Apel, T. (2004). Audio Segmentation by Singular Value Clustering, ICMC
- Wang, C. and Mysore, G.J (2016), Structural segmentation with Variable Markov Oracle and boundary adjustment, ICASSP
- Arias, J., Desainte-Catherine M. and Dubnov, S. (2016) Automatic Construction of Interactive Machine Improvisation Scenarios from Audio Recordings, International Workshop on Musical Metacreation, AAAI Conference on Computational Creativity

The End