

UNIVERSITY OF CALIFORNIA, SAN DIEGO

**Analysis and Generation of Music Signals with the Variable Markov Oracle**

A dissertation submitted in partial satisfaction of the  
requirements for the degree  
Doctor of Philosophy

in

Music

by

Cheng-i Wang

Committee in charge:

Professor Shlomo Dubnov, Chair  
Professor Alon Orlitsky  
Professor Miller Puckette  
Professor Lawrence Saul  
Professor Tamara Smyth

2018

Copyright  
Cheng-i Wang, 2018  
All rights reserved.

The dissertation of Cheng-i Wang is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

---

---

---

---

---

Chair

University of California, San Diego

2018

DEDICATION

To my mother, Ms. Li-Shan Kin.

## EPIGRAPH

*Our understanding of musical technique  
would have advanced much further  
if only someone had asked:*

*Where, when and how did music first develop its most striking and distinctive characteristic  
- repetition?*

— Heinrich Schenker

## TABLE OF CONTENTS

	Signature Page . . . . .	iii
	Dedication . . . . .	iv
	Epigraph . . . . .	v
	Table of Contents . . . . .	vi
	List of Figures . . . . .	ix
	List of Tables . . . . .	xi
	Acknowledgements . . . . .	xii
	Vita . . . . .	xiv
	Abstract of the Dissertation . . . . .	xvi
Chapter 1	Introduction . . . . .	1
	1.1 Modeling Temporality in Music . . . . .	1
	1.2 Modeling Musical Information Content . . . . .	3
	1.3 Dissertation Organizations . . . . .	6
Chapter 2	The Variable Markov Oracle . . . . .	7
	2.1 Factor Oracle and Audio Oracle . . . . .	7
	2.2 Variable Markov Oracle . . . . .	10
	2.2.1 VMO as an On-line Clustering Algorithm . . . . .	11
	2.2.2 Model Selection via Information Rate . . . . .	15
	2.2.3 Sequence Matching Algorithms . . . . .	19
	2.2.3.1 Query-matching algorithm . . . . .	20
	2.2.3.2 On-line alternative . . . . .	20
	2.2.3.3 A Probabilistic Interpretation . . . . .	23
	2.2.4 Context-Aware Hidden Markov Models . . . . .	25
	2.2.4.1 Variable Markov Oracle as Latent Model . . . . .	26
	2.2.4.2 VMO-HMM . . . . .	28
Chapter 3	Analysis with the VMO . . . . .	30
	3.1 Motif Discovery . . . . .	30
	3.1.1 Pattern Discovery by VMO . . . . .	31
	3.1.2 Experiments . . . . .	33
	3.1.2.1 Feature Extraction . . . . .	33
	3.1.2.1.1 Symbolic Representation . . . . .	33

	3.1.2.1.2	Audio Recording . . . . .	34
	3.1.2.2	Repeated Themes Discovery . . . . .	34
	3.1.2.3	Performance Recordings to Aid Pattern Discovery	37
	3.1.3	Evaluation . . . . .	37
	3.1.4	Discussion . . . . .	43
3.2		Structural Segmentation . . . . .	44
	3.2.1	SSM from Variable Markov Oracle . . . . .	46
	3.2.2	Segmentation Algorithm . . . . .	47
	3.2.2.1	Spectral Clustering . . . . .	47
	3.2.2.2	Structure Features and Segment Similarity . . . . .	50
	3.2.3	Boundary Adjustment . . . . .	51
	3.2.4	Music Structure Segmentation . . . . .	53
	3.2.4.1	Experiments . . . . .	53
	3.2.4.2	Evaluation . . . . .	54
	3.2.5	Discussions . . . . .	55
3.3		Music Analysis with VMO-HMM . . . . .	55
Chapter 4		Generate with VMO . . . . .	62
	4.1	Machine Improvisation . . . . .	62
	4.2	Query-guided Improvisation . . . . .	65
	4.2.1	Query-matching Algorithm for Generation . . . . .	67
	4.2.2	Examples - Guiding with Tonal Content . . . . .	68
	4.2.3	Examples - Guiding with Rhythmic Content . . . . .	76
	4.2.4	Related Works . . . . .	78
	4.3	Structural Improvisation . . . . .	79
	4.4	Improvisation with VMO-HMM . . . . .	81
	4.4.1	Jazz chord sequences . . . . .	82
	4.4.2	Random Walks on VMO-HMM . . . . .	83
	4.4.3	Query VMO-HMM by Chord Label Sequence . . . . .	84
	4.5	Discussions . . . . .	87
Appendix A		Human-Gesture Applications . . . . .	91
	A.1	3-D Gesture Recognition and Retrieval . . . . .	91
	A.2	Gesture Following . . . . .	95
Appendix B		<i>Memento</i> . . . . .	100
	B.1	Concert Proposal for MuMe 2017 . . . . .	100
	B.1.1	Description of the work . . . . .	100
	B.1.2	Description of the technology behind VMO-Score . . . . .	101
	B.1.3	Examples of previous performances using VMO-Score . . . . .	102
	B.1.4	Novel Technological Elements in the current work . . . . .	102
	B.1.5	Preparing for the MuMe 2017 Concert . . . . .	102
	B.2	Scores and performance plan . . . . .	103

Acronyms . . . . .	109
Glossary . . . . .	110
Bibliography . . . . .	111



## LIST OF FIGURES

Figure 2.1:	A Factor Oracle example . . . . .	9
Figure 2.2:	An Audio Oracle example . . . . .	10
Figure 2.3:	A Variable Markov Oracle example . . . . .	12
Figure 2.4:	Oracle structures with extreme $\theta$ values . . . . .	15
Figure 2.5:	Information Rate (IR) values versus different $\theta$ values . . . . .	19
Figure 2.6:	The decoding process for the VMO query-matching algorithm . . . . .	21
Figure 2.7:	Comparisons between different clustering algorithms . . . . .	27
Figure 3.1:	Chromagrams, found patterns and ground truths for motif discovery . . . . .	35
Figure 3.2:	Motif discovery with aid from performance audio . . . . .	38
Figure 3.3:	Motif discovery JKU-PDD evaluations . . . . .	41
Figure 3.4:	Motif discovery MIREX establishment scores . . . . .	42
Figure 3.5:	Motif discovery MIREX occurrence scores . . . . .	42
Figure 3.6:	Motif discovery MIREX $F_3$ scores . . . . .	42
Figure 3.7:	Motif discovery JKU-PDD evaluations (real audio) . . . . .	43
Figure 3.8:	Obtaining VMO-SSM from a <i>Variable Markov Oracle (VMO)</i> . . . . .	48
Figure 3.9:	Segmentation with spectral clustering and <i>VMO</i> . . . . .	50
Figure 3.10:	Segmentation with structural features and <i>VMO</i> . . . . .	51
Figure 3.11:	Midi-chormagram for VMO-HMM music analysis . . . . .	57
Figure 3.12:	Clusters of midi-chromagram from VMO-HMM music analysis . . . . .	58
Figure 3.13:	Lead sheet for VMO-HMM music analysis . . . . .	59
Figure 3.14:	Matrices from Markov transition tensor extracted from VMO-HMM . . . . .	61
Figure 4.1:	Guided audio synthesis example I - chromagrams . . . . .	69
Figure 4.2:	Guided audio synthesis example I - $L_2$ distances . . . . .	70
Figure 4.3:	Guided audio synthesis example II - chromagrams . . . . .	72
Figure 4.4:	Guided audio synthesis example II - $L_2$ distances . . . . .	73
Figure 4.5:	Guided midi synthesis example I . . . . .	76
Figure 4.6:	Guided midi synthesis example II . . . . .	76
Figure 4.7:	$lrs$ values for structural improvisation . . . . .	81
Figure 4.8:	Random walks on VMO-HMM . . . . .	84
Figure 4.9:	Comparison between actual and translated midi-chromagram . . . . .	86
Figure 4.10:	Example of decoded chord label sequence from VMO-HMM Viterbi . . . . .	86
Figure A.1:	Example frames of 3-D skeletal joints . . . . .	92
Figure A.2:	Query following example . . . . .	96
Figure A.3:	Example Kinect depth images for query following . . . . .	97
Figure A.4:	System diagram for interactive dance/graphics system . . . . .	97
Figure A.5:	Dance system snapshots . . . . .	99
Figure B.1:	Mumento–Machine Improvisation and Ensemble timeline . . . . .	104
Figure B.2:	Mumento–viola instruction . . . . .	105

Figure B.3: Mumento–cello instruction . . . . .	106
Figure B.4: Mumento–bass clarinet instruction . . . . .	107
Figure B.5: Mumento–percussion instruction . . . . .	108

## LIST OF TABLES

Table 3.1:	Motif discovery JKU-PDD evaluations . . . . .	39
Table 3.2:	Segmentation on the Beatles-ISO evaluations . . . . .	53
Table A.1:	Number of instances from the subset of the MSRC-12 dataset. These actions were meant for gaming scenarios. . . . .	92
Table A.2:	Precision for Kinect skeletal gesture retrieval - Consider type of gesture only - (upper rows) performances using <i>VMO</i> and (lower three rows) with other approaches. . . . .	93
Table A.3:	Precision for Kinect skeletal gesture retrieval - Both type and participant are considered . . . . .	93

## ACKNOWLEDGEMENTS

Deep thanks for the continuous, generous and constructive support from my advisor, Prof. Shlomo Dubnov. Thanks to Prof. Tamara Smyth and Prof. Miller Puckette for their sharing of invaluable knowledge and academic guidances. I would also like to thank Prof. Alon Orlitsky and Prof. Lawrence Saul for their inspection over my dissertation. Lastly I would like to thank the Center for Research in Entertainment and Learning (CREL) at UCSD for its support throughout my years in UCSD. Lastly, I would like to thank my family and friends for their companion and supports.

Chapter 2 is adapted from published materials in "*Variable Markov Oracle: A Novel Sequential Data Points Clustering Algorithm with Application to 3D Gesture Query-Matching*". Wang, Cheng-i. & Dubnov, Shlomo. International Symposium on Multimedia, 2014, 215-222, "*The Variable Markov Oracle: Algorithms for Human Gesture Applications*". Wang, Cheng-i. & Dubnov, Shlomo. IEEE MultiMedia, IEEE, 2015, 22, 52-67 and "*Context-Aware Hidden Markov Models of Jazz Music with Variable Markov Oracle*". Wang, Cheng-i. & Dubnov, Shlomo. 5th International Workshop on Musical Metacreation (MUME 2017) at the Eight International Conference on Computational Creativity, ICC3 2017, 2017.

Chapter 3 is adapted from published materials in "*Pattern Discovery from Audio Recordings by Variable Markov Oracle: A Music Information Dynamics Approach*". Wang, Cheng-i. & Dubnov, Shlomo. Acoustics, Speech, and Signal Processing (ICASSP), 2015 IEEE International Conference on, 2015, "*Music Pattern Discovery with Variable Markov Oracle: A Unified Approach to Symbolic and Audio Representations*". Wang, Cheng-i.; Hsu, Jennifer. & Dubnov, Shlomo. International Society for Music Information Retrieval Conference, 2015, 176-182, "*Structural Segmentation with the Variable Markov Oracle and Boundary Adjustment*". Wang, Cheng-i. & Mysore, Gautham. J. Proceedings of ICASSP 2016, Shanghai., IEEE, 2016 and "*Context-Aware Hidden Markov Models of Jazz Music with Variable Markov Oracle*". Wang, Cheng-i. & Dubnov, Shlomo. 5th International Workshop on Musical Metacreation (MUME

2017) at the Eight International Conference on Computational Creativity, ICCCC 2017, 2017.

Chapter 4 is adapted from published materials in "*Guided Music Synthesis with Variable Markov Oracle*". Wang, Cheng-i. & Dubnov, Shlomo. Tenth Artificial Intelligence and Interactive Digital Entertainment Conference, 2014, "*Machine Improvisation with Variable Markov Oracle: Toward Guided and Structured Improvisation*". Wang, Cheng-i.; Hsu, Jennifer. & Dubnov, Shlomo. Computers in Entertainment (CIE), ACM, 2016, 14, 4 and "*Context-Aware Hidden Markov Models of Jazz Music with Variable Markov Oracle*". Wang, Cheng-i. & Dubnov, Shlomo. 5th International Workshop on Musical Metacreation (MUME 2017) at the Eight International Conference on Computational Creativity, ICCCC 2017, 2017.

Appendix A is adapted from published materials in "*Variable Markov Oracle: A Novel Sequential Data Points Clustering Algorithm with Application to 3D Gesture Query-Matching*". Wang, Cheng-i. & Dubnov, Shlomo. International Symposium on Multimedia, 2014, 215-222 and "*The Variable Markov Oracle: Algorithms for Human Gesture Applications*". Wang, Cheng-i. & Dubnov, Shlomo. IEEE MultiMedia, IEEE, 2015, 22, 52-67.

## VITA

2018	Doctor of Philosophy in Music, University of California, San Diego
2013-2018	Research Assistant at CREL, University of California, San Diego.
2013	Master of Music in Music Technology, New York University, New York
2007	Bachelor of Business Administration in Finance, National Taiwan University, Taipei, Taiwan

## PUBLICATIONS

C. Wang, G. J. Mysore, and S. Dubnov, “Re-visiting the Music Segmentation Problem with Crowdsourcing,” in The 18th International Society for Music Information Retrieval Conference, 2017.

C. Wang and S. Dubnov, “Context-Aware Hidden Markov Models of Jazz Music with Variable Markov Oracle,” in 5th International Workshop on Musical Metacreation (MUME 2017) at the Eight International Conference on Computational Creativity, ICC3 2017, 2017.

C. Wang and G. J. Mysore, “Structural Segmentation with the Variable Markov Oracle and Boundary Adjustment,” Proceedings of ICASSP 2016, Shanghai., 2016.

C. Wang, J. Hsu, and S. Dubnov, “Machine Improvisation with Variable Markov Oracle: Toward Guided and Structured Improvisation,” Computers in Entertainment (CIE), vol. 14, no. 3, p. 4, 2016.

C. Wang, J. Hsu, and S. Dubnov, “Music Pattern Discovery with Variable Markov Oracle: A Unified Approach to Symbolic and Audio Representations,” in International Society for Music Information Retrieval Conference, 2015, pp. 176–182.

C. Wang and S. Dubnov, “The Variable Markov Oracle: Algorithms for Human Gesture Applications,” IEEE MultiMedia, vol. 22, no. 4, pp. 52–67, 2015.

C. Wang and S. Dubnov, “Pattern Discovery from Audio Recordings by Variable Markov Oracle: A Music Information Dynamics Approach,” in Acoustics, Speech, and Signal Processing (ICASSP), 2015 IEEE International Conference on, 2015.

C. Wang and S. Dubnov, “Variable Markov Oracle: A Novel Sequential Data Points Clustering Algorithm with Application to 3D Gesture Query-Matching,” in International Symposium on Multimedia, 2014, pp. 215–222.

C. Wang and S. Dubnov, “Guided Music Synthesis with Variable Markov Oracle,” in Tenth Artificial Intelligence and Interactive Digital Entertainment Conference, 2014.

C. Wang, T. Smyth, and Z. C. Lipton, “Estimation of Saxophone Control Parameters by Convex Optimization,” in 9th Conference on Interdisciplinary Musicology - CIM14, 2014.

T. Smyth and C. Wang, “Toward Real-Time Estimation of Tonehole Configuration,” in the 40th International Computer Music Conference (ICMC) joint with the 11th Sound & Music Computing conference (SMC), 2014.

ABSTRACT OF THE DISSERTATION

**Analysis and Generation of Music Signals with the Variable Markov Oracle**

by

Cheng-i Wang

Doctor of Philosophy in Music

University of California, San Diego, 2018

Professor Shlomo Dubnov, Chair

Music is temporal in nature, and each music piece has its own way of building expectations and surprises while the piece unfolds itself in time. Musical expectations are created by regularities of series of musical events while surprises are created by either the variation or breaking of such regularities. In that sense, identifying repeated sequences and distinguishing between variations is an essential task for either human listening to music, or modeling music algorithmically.

In this dissertation, a model is proposed to have the capabilities modeling both the temporal and expectation/surprise nature of music signals. The proposed model is named the Variable Markov Oracle since it is derived from a string matching method called Factor Oracle, that is capable of detecting arbitrarily long repetitions and could emulate variable-order Markov



chain behavior. The Variable Markov Oracle, in short, is a compressed suffix tree indexing each time instance in a music piece, while in the same time tracing the repeated sub-sequences in the piece. The model selection for the Variable Markov Oracle allows detection of inexact repetitions and utilizes information theoretic measurements which corresponds to the concept of expectation and surprises.

Motif identification and structural segmentation are two of the music research problems that are closely related to the concept of repeated sub-sequences in music, and in this dissertation the Variable Markov Oracle is used to solve these two problems and proved to be effective. The Variable Markov Oracle is also used in the context of machine improvisation to improve previously Factor Oracle based systems by providing query-guided and structural improvisation. Besides being applied to music signals, the uses of Variable Markov Oracle for retrieval and creative use on other time series data, such as human gesture, are also presented.

# Chapter 1

## Introduction

### 1.1 Modeling Temporality in Music

Music is an art form manifesting itself as temporal structures [1] meaning that it is the sequential relationships, repetitions and variations of musical elements that makes up a music piece [2, p. 327] [3, p. 154]. Since temporality is an inherent and fundamental concept of music, it is crucial for computational models to encompass such concept, so that the models could be used for analyzing or generating music materials.

Traditionally, “bag-of-features” was the dominant approach used in information retrieval circumstances. “Bag-of-features” refers to modeling songs by extracting features from either audio recordings or symbolic music representations (MIDI for example) with features aggregated into fixed-length vectors by statistical descriptors of the extracted features. Each fixed-length vector represents a single music piece, and the fixed-length vector representation facilitates querying, comparison, classification and clustering easily between music pieces having varied length [4, 5]. Although it is proven to be effective for inter-song applications, “bag-of-features” approach is not able to deal with intra-song applications such as motif discovery, structural segmentation, audio thumb-nailing, etc, since “bag-of-features” does not include temporal information within each

music piece.

In order to tackle intra-song application problems, models that take temporal information into their considerations are proposed. Especially Hidden Markov Models (HMMs) are widely adopted from speech recognition [6] to intra-song music applications, such as chord recognition [7], source separation [8], structural segmentation [9], etc. In [10], the dynamic texture model is proposed to model temporal dependencies of music signals for the music auto-tagging task. The difference between previous HMM based model and the dynamic texture model is that the dynamic texture model uses a linear dynamic system which has a continuous state space unlike the discrete state space assumed by HMMs.

In general the use of HMM or dynamic texture model falls under the unsupervised learning regime from a machine learning point of view. For modeling sequential relationships by supervised learning, Linear Chain Conditional Random Fields (CRFs) are used in intra-song music emotion modeling [11] and audio-to-score matching [12]. The main difference between CRFs and HMMs is that HMMs learn the joint probabilities between the observations and hidden states while CRFs learn the conditional probabilities of the output variables given input variables. Recurrent neural networks was used in [13] to generate Western tonal music. With the recent developments of using deep neural networks to learn feature transformation and non-linear mappings, recurrent neural networks are revived and used in jazz music transcription and generation [14], chord recognition [15], piano music transcription [16], etc. Convolutional neural networks are also used for music information retrieval [17, 18, 19] and music generation [20, 21] tasks, but it exploits local patterns rather than modeling temporal dependencies.

The aforementioned approaches, whether being unsupervised or supervised, could also be seen as model-based approaches, where parameters of compact mathematical models are trained or learned to describe temporal relationships between observations. Model-based approaches rely on carefully choosing of models and abundant amount of samples (in this dissertation, music pieces.) available for training. On the other hand, approaches differ from from model-based approaches,

are described as agnostic approaches in this dissertation. The agnostic approaches utilize distance (or similarity) calculations between observations and use these distances to build algorithms or data structures for further applications. For retrieval tasks such as query-by-content, Dynamic Time Warping (DTW) has been the dominant distance metric/algorithm used in time series query-by-content task by practitioners but is limited by its quadratic computational complexity [22] and monotonicity conditions. Another example is the *Guidage* system for query-by-audio, proposed in [23]. *Guidage* uses a structure called the *Audio Oracle (AO)* [24] to represent an audio signal in a suffix structure. This representation allowed efficient sub-clips retrieval by traversing the *AO* structure in forward direction, according to an input query. For symbolic sub-sequence matching, one function absent from *AO* was a solution for traversing the suffix links. For music generation applications, previous research using the *Factor Oracle (FO)* [25, 26] suffix tree data structure for symbolic music generation and the *AO* [24, 27] for audio content generation were developed for machine improvisation applications. Details of both *FO* and *AO* are provided in chapter 2. In [28], the bijection between Bregman divergence and exponential families is used to derive distance metrics for audio similarity computations with *AO* for on-line segmentation applications.

Since the *FO* can be used to find varied length of repeated sub strings in a symbolic sequence and *AO* extends *FO* to accept multi-variate time series as input, the *Variable Markov Oracle (VMO)* data structure, that combines *FO* and *AO* with new functionalities, is proposed in this dissertation. The use of the *VMO* on music analysis and generation applications are elaborated as well.

## 1.2 Modeling Musical Information Content

Models and methods mentioned in section 1.1 deal with describing temporal relationships in a music piece both systematically and mathematically. However, there is another way to characterize the temporal structure of music signals by measuring the “information content” of

the music signal by information theoretic measurements such as mutual information, entropy, etc [29, 30, 31]. The motivation of using information theoretic measurements to characterizing the temporal structure of music signals is that measurements such as mutual information and entropy could be related to the concept of “expectation” and “surprise” in music [32]. In [33], it is argued that the emotional content brought by music arises from composer’s creation and manipulation of expectation. Then in [32, 34], it is further elaborated that the concept of “expectation” is tightly related to regularities and the possibilities of explaining the “expectation” concept by statistical models. The following quote from [35] is a concise description of the connections between expectations in music and statistical models;

Once a musical style has become part of the habit responses of composers, performers, and practiced listeners it may be regarded as a complex system of probabilities . . . out of such internalized probability systems arise the expectations - the tendencies - upon which musical meaning is built . . . the probability relationships embodied in a particular musical style together with the various modes of mental behavior involved in the perception and understanding of the materials of the style constitute the norms of the style, . . .

By assuming musical events, structures and expectations could be approximated by probabilistic models, the information content of music signals could then be measured by measurements such as mutual information or entropy. In [30], multiple information theoretic based measurements are proposed. Among the proposed measurements, the Predictive Information Rate (PIR) measures the average amount of future information gained from observing the present observations, given the past observations, and is define as

$$PIR(X, Y|Z) = H(Y|Z) - H(Y|X, Z)$$

where  $X$  stands for current observation,  $Z$  for past observations and  $Y$  for future observations. It is shown in [30] that the PIR values could reflect changes of phrases, sections or parts in a music

piece. In [29, 31], the Information Rate (IR) is proposed and is defined as

$$IR(X, Z) = H(X) - H(X|Z)$$

where it measures how much of knowing the past reduce the uncertainty observing the current event. For both PIR and IR, one way to calculate their values given actual music signal observations is to assume a parametric probabilistic source emitting the observations, then calculating the measurements with the parametric probabilistic models. In [30], 2nd-order Markov chain is used to model the sequential relationships between the past, present and future observations. The music signals used in the experiment for PIR was limited to minimalist melodies in symbolic form to accommodate the use of 2nd-order Markov models. For the calculation of IR, in [29], it is shown that IR is inversely derivable from spectral flatness assuming spectral features are from a Gaussian process. Assuming probabilistic sources for music signals has the advantages of model interpretability and closed form solutions for PIR and IR calculations for certain probabilistic models, but such assumption also in the same time suffers from model biases and limitations choices of traceable models for information theoretic measurement calculations.

The other way to calculate information theoretic measurements is to approximate them by replacing the entropy terms with complexity measurement from compression algorithms if appropriate ones exist. In [31], the IR calculation on audio feature signals is approximated by compression algorithms on the *AO* suffix structures. The advantage of calculating information theoretic measurements by approximating them with compression algorithms instead of assuming probabilistic sources is that it is easier to account for variable-length context. Compression algorithms based on encoding repeated sub-strings such as the Lempel-Ziv-Welch algorithms [36] or the Compror algorithm based on the *FO* suffix tree data structure [37] both could handle variable length past context.

Since the *FO* and *AO* suffix tree data structure on time series is capable of modeling

sequential relationships and calculating information theoretic measurements for information content, further developments of a suffix tree based data structure and algorithms extending both *FO* and *AO* to have on-line clustering and query-retrieval capabilities are conducted. The newly developed suffix tree data structure is the *VMO*.

### **1.3 Dissertation Organizations**

The following chapters are organized as follows; in chapter 2, the construction, model selection, on-line clustering and query-matching algorithms of *VMO* are elaborated. Music information retrieval and analysis tasks using *VMO* including motif discovery, structural segmentation and music analysis are provided in chapter 3. Generative use of *VMO* is shown in chapter 4. Lastly, the use of *VMO* for human 3-D gesture applications is in appendix A and a computer music composition using *VMO* is presented in appendix B.

# Chapter 2

## The Variable Markov Oracle

### 2.1 Factor Oracle and Audio Oracle

*Audio Oracle (AO)* [24] is the signal extension of *Factor Oracle (FO)*, a suffix automaton for symbolic sequences. *FO* was proposed in [38] originally for bio sequence pattern matching and later extended to generating symbolic musical sequences [25]. In this chapter a new data structure is proposed, called the *Variable Markov Oracle (VMO)* that extends both *AO* and *FO* by combining strengths from both of them. First, it models time series observations and its symbolized version simultaneously rather than just the symbolized version (*FO*) or just the time series (*AO*). Second, it possesses a set of algorithms that allow the searching of signal fragments both in forward and backward directions, to match and recombine an input query in terms of fragments of stored data. The forward and backward search directions allowed by *VMO* enable nonlinear recombination paths which are not possible with DTW. The *VMO* is also equipped with an on-line construction algorithm inherited from *FO*. Technical details for *VMO* are provided in this chapter.

*FO* is a compressed suffix tree (suffix automaton) that aims at fast retrieval of repeated sub-strings (factors) and patterns (repeated suffixes) of a symbolic sequence,  $Q$ . *FO* could be



constructed in linear time and space, which is shown to be more appropriate than two other variable-order models based on incremental parsing and probabilistic suffix trees [25] for real-time applications. For a sequence of symbols  $Q = q_1, q_2, \dots, q_t, \dots, q_T$ , an  $FO$  is constructed with  $T$  states and each symbol  $q_t$  is associated with a state. Two kinds of link, forward link and suffix link, are created during the construction of  $FO$ . Two types of forward links are in an oracle structure. The first is an internal forward link which is a pointer from state  $t - 1$  to  $t$  labeled by the symbol  $q_t$ , denoted by  $\delta(t - 1, q_t) = t$ . The other forward link is an external forward link which is a pointer from state  $t$  to  $t + k$  labeled by  $q_{t+k}$  with  $k > 1$ . An external forward link  $\delta(t, q_{t+k}) = t + k$  is created in  $FO$  when

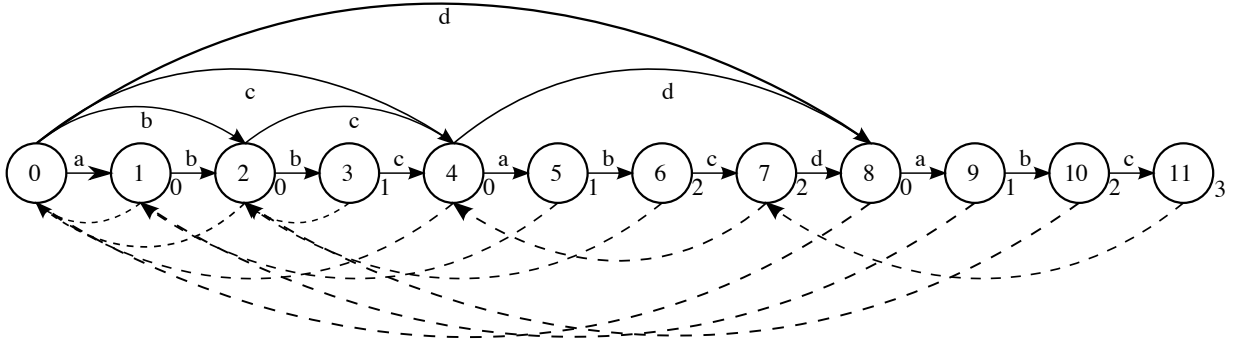
$$\begin{aligned} q_{t+1} &\neq q_{t+k}, \\ q_t &= q_{t+k-1}, \\ \delta(t, q_{t+k}) &= \emptyset. \end{aligned}$$

In other words, an external forward link is created when the most recent internal forward link,  $\delta(t + k - 1, q_{t+k}) = t + k$ , is never preceded by the previous occurrence of  $q_t$  (given that  $q_t = q_{t+k-1}$ ). The function of the forward links is to provide an efficient way to retrieve any of the factors of  $Q$ , starting from the beginning of  $Q$  and following a unique path formed by forward links.

A suffix link ( $\text{sfx}$ ) is a backward pointer that points state  $t$  to  $k, t > k$ , without a label and is denoted by  $\text{sfx}[t] = k$ . The condition for when to create a suffix link is

$$\begin{aligned} \text{sfx}[t] = k &\iff \text{the longest repeated suffix of} \\ &\{q_1, q_2, \dots, q_t\} \text{ is recognized in } k. \end{aligned}$$

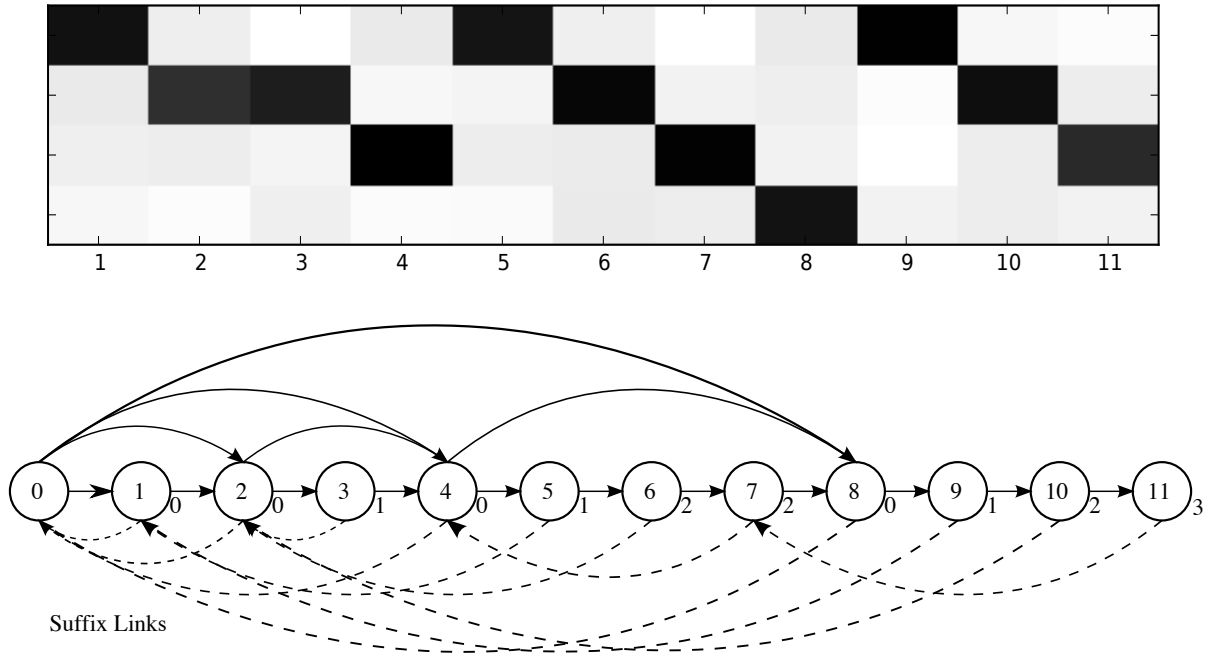
Suffix links are used to find repeated suffixes in  $Q$ . In order to track the longest repeated suffix at



**Figure 2.1:** *FO* of  $Q = \text{“abbcabcdabc”}$ . Dashed arrows are the suffix links and solid arrows are the forward links with labels of each symbol.

each time index  $t$ , the length of longest repeated suffixes ( $\text{lrs}$ ) at each state  $t$  (denoted as  $\text{lrs}[t]$ ) is computed by the algorithm described in [39]. The construction of *FO* could be done incrementally by appending newly appearing symbols to the end of  $Q$ . The algorithms for constructing *FO* are provided in [39]. An example of *FO* structure is depicted in figure 2.1. The example will be further exploited in section 2.2.3.1 to explain the decoding steps in the query-matching algorithm.

*AO* is the signal extension of *FO*. The input for an *AO*,  $O[t]$ , is a continuous, possibly multi-dimensional, time series sampled at discrete times. To extend the domain of *FO* from symbolic sequences to time series, such as an audio signal, a threshold  $\theta$  is introduced to determine if  $O[t]$  is similar to states found in  $O[1 \dots t - 1]$  by following suffix links.  $\theta$  is associated with the metric between observations of the given time series. Two samples  $O[i]$  and  $O[j]$  are considered similar if  $|O[i] - O[j]| \leq \theta$ . The metric should be chosen according to the application domain and features used. Different  $\theta$  values will lead to different oracle structures, the details of how to choose an appropriate  $\theta$  based on music information dynamics are given in section 2.2.2. Unlike *FO*, the *AO* does not symbols associated with the forward links. An example of *AO* is depicted in figure 2.2.



**Figure 2.2:**  $AO$  (bottom) of  $O = O[1] \dots O[T]$  (top). Dashed arrows are the suffix links and solid arrows are the forward links **without** labels of each frame.

## 2.2 Variable Markov Oracle

As mentioned above,  $AO$  resembles the suffix structure of  $FO$ , without symbols attached to states. This fact prevents  $AO$  from having efficient navigating algorithms for states linked by suffix links. From this observation,  $VMO$  is devised to improve  $AO$  by explicitly assigning labels to frames (or states) linked by suffix links during the  $AO$  construction. The symbols formed by gathering states connected by suffix links have the following properties; 1) pairwise distances between states connected by suffix links are less than  $\theta$ , 2) the symbolized signal formed by the oracle could be interpreted as a sample from a variable-order Markov model since the states connected by suffix links share common suffixes with variable length, 3) each state is labeled by only one symbol because each state has only one suffix link, 4) the alphabet size of the assigned symbols is unknown before the construction and is determined by  $\theta$ .

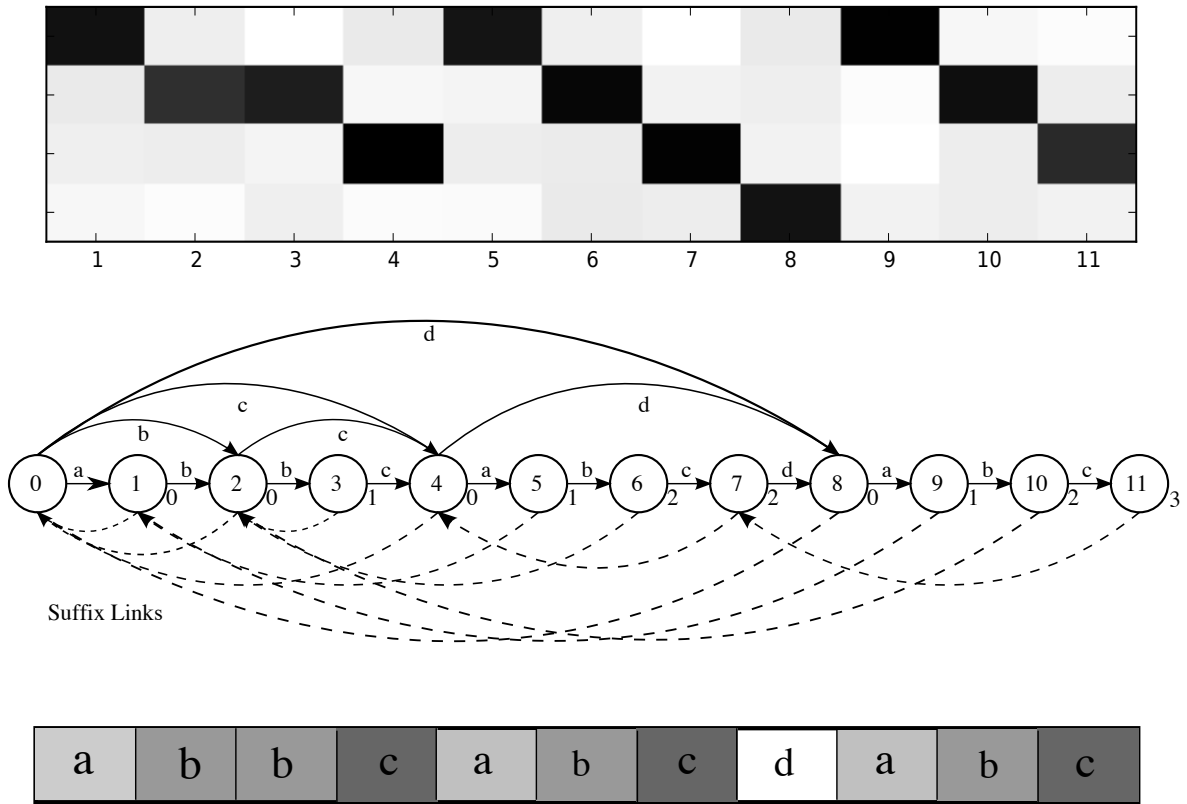
To explicitly keep track of the linked states and also to maintain the on-line nature of the

construction algorithm, the construction of *VMO* combines *FO* and *AO* by treating the sequence of assigned labels  $Q$  as the symbolic sequence for an *FO*. Pointers to  $O$  are tracked by introducing a list of pointers,  $\Sigma = [\sigma_1 \dots \sigma_m \dots \sigma_M]$ , with  $M$  the number of labels formed and  $\sigma_m$  a list containing the pointers (frame numbers) for the  $m$ th label. In summary, *VMO* accepts  $O$  as input and returns an oracle structure keeping track of the cluster label sequence  $Q$  and also the lists of pointers to  $O$ . The lists of pointers are stored in  $\Sigma$  and indexed by  $Q$ . An example *VMO* following the previous *FO* and *AO* examples is shown in figure 2.3.

Let  $O$  be the incoming new signal and  $t$  the time index.  $O[t]$  represents the newly observed value or vector at  $t$ . A forward link from state  $i$  to state  $j$  labelled by  $q$  is denoted by  $\delta(i, q) = j$ . A suffix link from state  $j$  to state  $i$  is denoted by  $\text{suffix}[j] = i$  without labeling.  $Q = q_1, \dots, q_T$  denotes the label sequence for labels of observations  $O = O[1] \dots O[T]$ . The initialization of a *VMO* is provided in algorithm 1. In algorithm 2, the incremental algorithm to add an incoming signal observation is provided. For each new incoming sample  $O[t]$ , a new state is constructed with the internal forward link  $\delta(t-1, q_t) = t$  created. The cluster (pointer) label  $q_t$  for  $O[t]$  is initialized as null. The while loop from line 5 to line 15 in algorithm 2 is the standard process to assign external forward links and suffix links introduced in [39]. Line 16 to line 25 in algorithm 2 is the newly introduced part of *VMO* that assigns the cluster label to  $q_t$  then appends the pointer of  $O[t]$  to  $\sigma_{q_t}$ . In this chapter, for the algorithms described in pseudocode,  $X[i]$  means retrieving the item from an array  $X$  in its  $i$ th location;  $[a; b]$  means appending  $b$  to the end of  $a$ ;  $X_{i,j}$  means accessing the item at  $i$ th row and  $j$ th column of a matrix  $X$ ; and  $X(i, :)$  means retrieving the whole  $i$ th row in a matrix  $X$ .

## 2.2.1 VMO as an On-line Clustering Algorithm

The on-line construction algorithms presented in section 2.2 build a clustering of frames and stores the clustering in  $\Sigma$ . How the construction algorithms could be viewed as an on-line clustering algorithm with Markov constraints is conjectured in this section. An on-line clustering



**Figure 2.3:** *VMO* (middle) of  $O = O[1] \dots O[T]$  (top) with symbolized/clustered frames  $Q =$  “abbcabcbabc” (bottom). Dashed arrows are the suffix links and solid arrows are the forward links with labels of each symbol.

---

**Algorithm 1** On-line construction of *VMO*

---

**Require:** Time series as  $O = O[1], O[2] \dots O[T]$

- 1: Create an oracle  $P$  with initial state  $p_0$
  - 2:  $\text{sfx}_P[0] \leftarrow -1, \Sigma \leftarrow \emptyset, N \leftarrow 1$
  - 3: **for**  $t = 1 : T$  **do**
  - 4:  $\text{Oracle}(P = p_1 \dots p_t) \leftarrow \text{Add-Frame}(\text{Oracle}(P = p_1 \dots p_{t-1}), O[t])$
  - 5: **end for**
  - 6: **return**  $\text{Oracle}(P = p_1 \dots p_T)$
-

---

**Algorithm 2** Add-Frame

---

**Require:** Oracle  $P = p_1 \dots p_t$ , time series instance  $O[t + 1]$

- 1: Create a new state  $t + 1$
  - 2:  $q_{t+1} \leftarrow 0$ ,  $\text{sfx}_P[t + 1] \leftarrow 0$ ,  $M \leftarrow 0$
  - 3: Create a new transition from  $t$  to  $t + 1$ ,  $\delta(t, q_{t+1}) = t + 1$
  - 4:  $k \leftarrow \text{sfx}_P[t]$
  - 5: **while**  $k > -1$  **do**
  - 6:      $D \leftarrow$  distances between  $O[t + 1]$  and  $O[\delta(k, :)]$
  - 7:     **if** all distances in  $D$  is greater than  $\theta$  **then**
  - 8:          $\delta(k, q_{t+1}) \leftarrow t + 1$
  - 9:          $k \leftarrow \text{sfx}_P[k]$
  - 10:     **else**
  - 11:         Find the forward link from  $k$  that minimizes  $D$
  - 12:          $k' \leftarrow \delta(k, :)[\text{argmin}(D)]$
  - 13:          $\text{sfx}_P[t + 1] \leftarrow k'$
  - 14:         **break**
  - 15:     **end if**
  - 16: **end while**
  - 17: **if**  $k = -1$  **then**
  - 18:      $\text{sfx}_P[t + 1] = 0$
  - 19:     Initialize a new cluster with current frame index
  - 20:      $\sigma_{M+1} \leftarrow t + 1$
  - 21:      $\Sigma \leftarrow [\Sigma; \sigma_{M+1}]$
  - 22:     Assign a label to the new cluster,  $q_{t+1} \leftarrow M + 1$
  - 23:     Update number of clusters,  $M \leftarrow M + 1$
  - 24: **else**
  - 25:     Assign cluster label based on assigned suffix link
  - 26:      $q_{t+1} \leftarrow q_{k'}$
  - 27:      $\sigma_{q_{k'}} \leftarrow [\sigma_{q_{k'}}; t + 1]$
  - 28: **end if**
  - 29: **return** Oracle  $P = p_1 \dots p_{t+1}$
-

algorithm refers to incrementally aggregating a data stream into clusters without observing all the data points in the first place. In [40], an on-line clustering algorithm, the Leader-Follower Clustering, is proposed. The leader-follower clustering algorithm is provided in algorithm 3.

---

**Algorithm 3** Leader-Follower Clustering

---

**Require:** A data stream with  $x$  the incoming sample

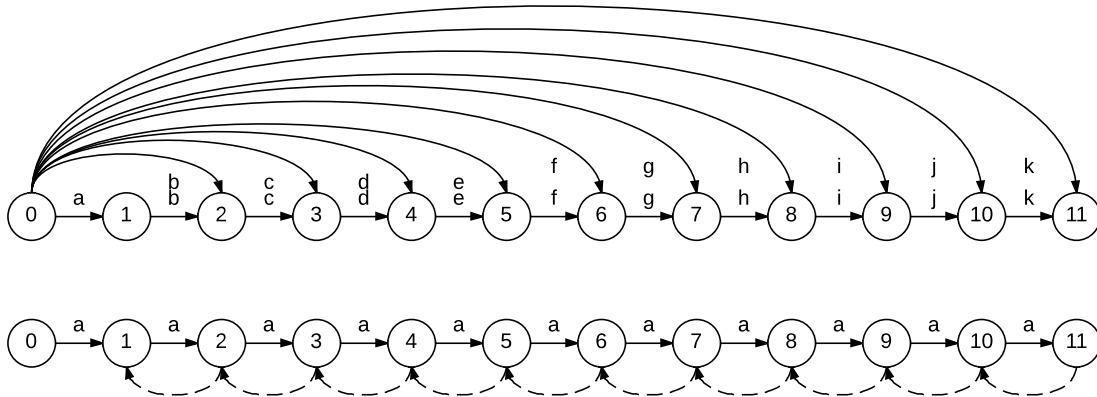
```

1: Initialize  $\eta, \theta$ 
2:  $w_1 \leftarrow x$ 
3: while  $x$  is presented do
4:    $j \leftarrow \operatorname{argmin}_j \|x - w_j\|$  (find nearest cluster)
5:   if  $\|x - w_j\| < \theta$  then
6:      $w_j \leftarrow w_j + \eta x$ 
7:   else
8:     add new  $w \leftarrow x$ 
9:      $\|w\| \leftarrow w / \|w\|$  (normalizing the weights)
10:  end if
11: end while
12: return  $w_1, w_2, \dots$ 

```

---

Comparing algorithm 2 to algorithm 3, one can observe that the suffix link finding routine in algorithm 2 is similar to line 4 in the Leader-Follower Clustering where a possible cluster label is identified. Then the label-assigning routine in algorithm 2 is where either an existing label gets assigned or a new label is formed, and is similar to the If-else part in algorithm 3 as well. The  $\theta$  used in both algorithms plays the same role in the sense that it controls the number of clusters created. Two major differences separates these two algorithms. First, the suffix link finding routine in algorithm 2 places a Markov constraint on the states to be considered as candidate states for the suffix link of the incoming sample. Markov constraint refers to the constraints placed in constraint satisfaction problem (CSP) to enforce Markov properties in the generated sequence from solving the CSP [41]. The concept is borrowed here and used to explain the effect of line 6 in algorithm 2 where, instead of having Markov constraint cost functions in an optimization scenario of CSP, *VMO* directly limits only the states sharing the same suffix with the incoming observation to be considered as candidates to where suffix links point. Then variable-order Markov properties are injected because of the shared suffixes. The second difference is that



**Figure 2.4:** Two oracle structures with extreme values of  $\theta$ . The characters near each forward link represent the assigned labels. (Top) The oracle structure with  $\theta = 0$  or extremely low  $\theta$  value. (Bottom) The oracle structure with a very high  $\theta$  value. In both cases the oracles are incapable of capturing any type of structure from the time series.

*VMO* does not parametrize the clustered frames in terms of centroids or other statistics. From another point of view, since the clusters in *VMO* are formed by whether assigning linkages between incoming frames and existed frames in established clusters or creating a new cluster by criteria related to distance or similarity measures, *VMO* is more like a variant of single linkage hierarchical clustering.

### 2.2.2 Model Selection via Information Rate

As mentioned earlier, different  $\theta$  values lead to different oracle structures. For an extreme  $\theta$  value, *VMO* may assign different symbols to every frame in  $O$  ( $\theta$  being excessively low), or *VMO* may assign the same symbol to every frame in  $O$  ( $\theta$  being excessively high). In these two cases, *VMO* is incapable of capturing variable-order dependencies in the time series. Selecting the optimal  $\theta$  in the context of music information dynamics with Information Rate (IR) is described in this section. An example of the oracle structure with extreme  $\theta$  values is shown in figure 2.4.

With different  $\theta$  values, *VMO* constructs different suffix structures and different symbol-



ized sequences from the signal. To select the one symbolized sequence with the most informative variable-order structure, IR is used as the criterion in model selection between different structures generated by different  $\theta$  values. IR is an information theoretic measure capable of measuring the information content of a time series [24] in terms of the predictability of its source process on the present observation given past ones. *VMO* uses the same approach as *AO* [31] to calculate IRIR. Let  $x_1^N = \{x_1, x_2, \dots, x_N\}$  denote time series  $x$  with  $N$  observations,  $H(x) = -\sum P(x) \log_2 P(x)$  the entropy of  $x$ , the definition of IR is

$$IR(x_1^{n-1}, x_n) = H(x_n) - H(x_n | x_1^{n-1}). \quad (2.1)$$

IR is the mutual information between the present and past observations, which is maximized when there is a balance between variation and repetition in the symbolized signal. The value of IR could be approximated by replacing the entropy terms in (2.1) with a complexity measure associated with a compression algorithm. This complexity measure is the number of bits used to compress  $x_n$  independently using the past observations  $x_1^{n-1}$ .

$$IR(x_1^{n-1}, x_n) \approx C(x_n) - C(x_n | x_1^{n-1}). \quad (2.2)$$

*Compror*, a lossless compression algorithm based on *FO* and *lrs*, is provided in [37]. The detailed formulation of combining *Compror*, *AO* and IR is provided in [31] and also the following paragraphs.

Before getting into the compression algorithms using *VMO*, a motivating example of why IR is an appropriate measurement is provided here. Assuming a symbolic sequence  $X = [a, b, a, b, a, b, \dots]$ , that repeats forever and  $a$  is always followed by  $b$  and  $b$  always followed by  $a$ . If  $X$  is modeled by a 1st-order Markov chain, then the IR of  $X$  is easily 1 since  $H(X) = 1$  and  $H(X^n | X_1^{n-1}) = H(X^n | X^{n-1}) = 0$ . Also  $X$  has the maximum IR value among all possible 2-event sequences. In [30], it is argued that the aforementioned statements about  $X$  is why IR is

not suitable for modeling information dynamics of music signals, since the repeating structure of  $[a, b, a, b, \dots]$  is not interesting by the sequence itself. Here the argument opposed to [30] is provided to support why IR is a suitable measurement for capturing information dynamics. Essentially, if there exists some algorithms or models that could *reduce* or *decompose* surface musical elements into the form of  $X$ , then that implies that such algorithms or models are capable of finding the basic building blocks  $a$  and  $b$  of the music piece as similar concepts were proposed by [42] and [43]. In other words, unlike in [30] that only “surface” musical events (note values having the same length) are modeled by Predictive Information Rate (PIR), IR is capable of measuring how well a latent space or state space model is capable of modeling a reduced structure of the surface musical events.

For a *VMO*, the on-line generation of code words  $C$  for equation (2.2) is as follows [37]: When a new state is added to an oracle structure at time  $(t + 1)$ , the suffix link and length of the longest repeated suffix for this state along the *VMO* are computed. If the resulting suffix link points to state 0 that means no suffix was found since the distance between the new state and all previous states exceeded the threshold  $\theta$ . In such a case the new state has to be individually encoded. Otherwise, if suffix link to a previous location in the state sequence is found and the length of the longest repeating suffix is smaller than the number of steps passed since the last encoding event, then a complete preceding block of states is encoded in terms of a pair  $(length, position)$ . In the method here individual new state will be denoted as a pair  $(0, position)$ . Let  $\kappa[i]$  be the array that contains the states where encoding occurs during the compression pass. An algorithm for computing  $\kappa$  is described in algorithm 4 with `lrs` an array containing the lengths of the longest repeated suffixes of the oracle at each step. `lrs` could be obtained during the construction of an oracle. The collection of code pairs  $\Phi_\kappa$  resulting from algorithm 4 is passed to the incremental IR algorithm, as described in algorithm 5.

A visualization of the sum of IR values versus different  $\theta$ s is depicted in figure 2.5. When algorithm 4 and algorithm 5 are applied to a *VMO*, higher  $\theta$  value creates higher  $C(x_n)$  and

---

**Algorithm 4** Compression Pass over VMO

---

**Require:** Array containing the length of repeated suffixes for every state  $\text{lrs}[t], t = 1 \dots T$

- 1: Create an array  $\kappa$  with initialization  $\kappa = \{1\}$
  - 2: **for**  $t = 1$  to  $T$  **do**
  - 3:     **if**  $\text{lrs}[t+1] < t - \text{lrs}[\text{end}] + 1$  **then**
  - 4:          $\kappa \leftarrow \kappa \cup \{t\}$
  - 5:     **end if**
  - 6: **end for**
  - 7: **return** Vector  $\kappa$
- 

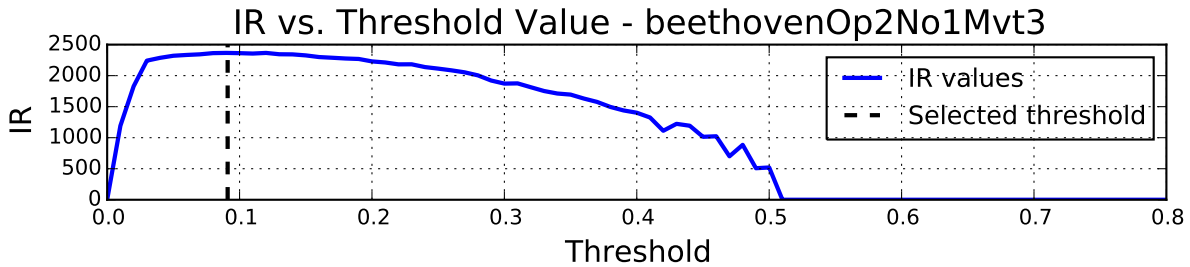
---

**Algorithm 5** Incremental IR from code  $\Phi$ 

---

**Require:** A sequence of codeword pairs  $\Phi = (\text{length}, \text{position})$ , with position information,  $\kappa$ , from compression algorithm, and length information from  $\text{lrs}$ .

- 1: Compute length  $T$  by summing all  $\Phi \rightarrow \text{length}$  values
  - 2: Create vectors  $C(x_n)$  and  $C(x_n|x_1^{n-1})$
  - 3: **for**  $t = 1$  to  $T$  **do**
  - 4:      $M \leftarrow$  number of symbols up to  $t$  by  $\sum_{i=1}^t (\text{lrs}[i] == 0)$
  - 5:      $C(x_t) \leftarrow \log_2(M)$
  - 6:      $C(x_t|x_1^{t-1}) \leftarrow \frac{\log_2(\# \text{ of codewords } \Phi \text{ up to } t)}{\Phi \rightarrow \text{length to which state } t \text{ belongs}}$
  - 7: **end for**
  - 8: **return** Vector  $IR = C(x_n) - C(x_n|x_1^{n-1})$
-



**Figure 2.5:** IR values are shown on the vertical axis while  $\theta$  are on the horizontal axis. The solid blue curve shows the relationship between IR and  $\theta$ , and the dashed black line indicates the chosen  $\theta$  by locating the maximum IR value. Empirically, IR curves exhibit quasi-concave function shapes, thus a global maximum can be located.

$C(x_n|x_1^{n-1})$  while lower  $\theta$  creates lower  $C(x_n)$  and  $C(x_n|x_1^{n-1})$ . Thus IR is maximized with an intermediate  $\theta$  value when  $C(x_n)$  and the negative  $C(x_n|x_1^{n-1})$  are balanced. A *VMO* with higher IR value captures more of the repeating sub-clips (ex. patterns, motifs, themes, gestures, etc) than the ones with lower IR values.

### 2.2.3 Sequence Matching Algorithms

One of the advancement that the *VMO* brings is the capability of performing query-retrieval or query-matching tasks on either symbolic or multi-variate time series. The sequence matching functionality is enabled by the clustering functionality introduced in *VMO* and it achieves sequence matching by concatenation of sub-sequences in the target sequence found by the repeated suffix structure. The algorithms for query-matching are described here in the following section and referred to in following chapters.

Let  $R$  be the query observation indexed by  $n$ , denoted as  $R = R[1], \dots, R[n], \dots, R[N]$ . The matching algorithm provided in algorithm 6 takes  $R$  as input and matches it to the target *VMO*,  $Oracle(Q = q_1, q_2, \dots, q_T, O = O[1], O[2], \dots, O[T])$ , constructed by a target time series,  $O$ . The algorithm returns a cost and a corresponding recombination path using  $O$  resembling  $R$ . The cost is the reconstruction error between the query and the best match from  $O$  given a metric on a

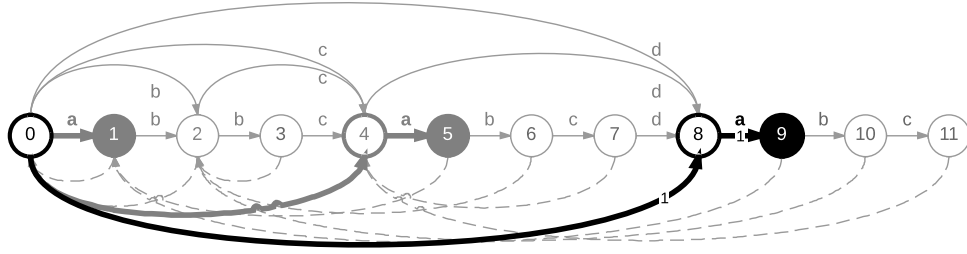
frame-by-frame basis. The recombination path corresponds to the sequence of indices that will reconstruct a new sequence from  $O$  that best resembles the query,  $R$ .

### 2.2.3.1 Query-matching algorithm

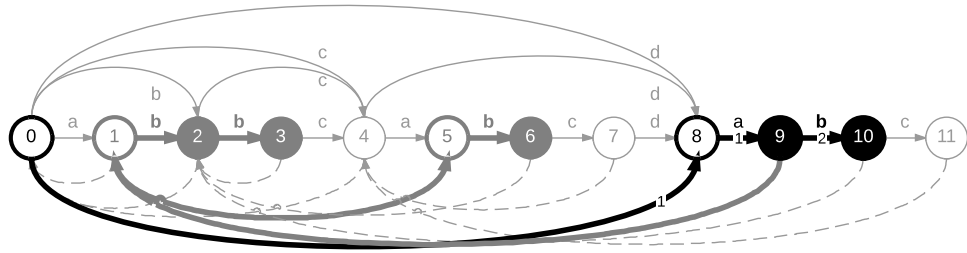
The query-matching algorithm described in this section is a dynamic programming algorithm. The algorithm is separated into two steps, initialization and decoding. In algorithm 6, the initialization is in line 1 to line 6. During initialization, the size of the alphabet,  $M$ , is obtained from the cardinality of  $\Sigma$ . Then for the  $m$ th list, the frame within the  $m$ th list that is closest to the first query frame,  $R[1]$ , is found and stored. After the initialization step, the decoding step (line 7~13 in algorithm 6) iterates over the rest of the query frames from 2 to  $N$  to find  $M$  paths, with each path beginning with the state found corresponding to the respective label in the initialization step. It could be observed that the proposed query-matching algorithm is similar to the Viterbi decoding algorithm for HMM and max-sum inference algorithm for graphical models [44] in the sense that each update in the decoding step depends only on its neighboring findings, thus making it efficient to compute and of no need to search over the whole state space. A visualization of algorithm 6 from initialization to decoding for one path among the  $M$  paths is shown in figure 2.6. The query-matching algorithms are used in section 4.2.1 for guided synthesis applications.

### 2.2.3.2 On-line alternative

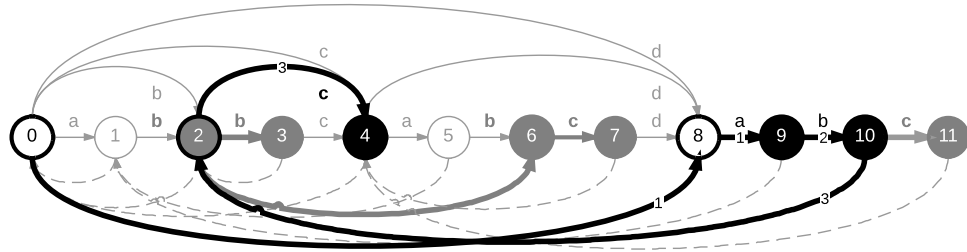
To satisfy certain applications that require real-time capabilities, the query-matching algorithm described in algorithm 6 is adjusted. The on-line query-matching algorithm will be referred to as the query-following algorithm. The query-following algorithms are provided in algorithm 7 and algorithm 8. The query-following algorithm updates the current “time progression” and “likelihood” on a frame-by-frame basis instead of one final backward pass at the end of the query time series for the offline version. In algorithm 7,  $M$  possible paths are initialized according



(a) At  $t = 1$  (Initialization for label **a**);  $\{\mathbf{a}, 9\}$ , the pair of label initialized and frame matched. At initialization, for label **a** the choices for the first frame are stored in the list,  $\{1, 5, 9\}$  from  $\Sigma$ . Assuming the closest frame in  $O$  to  $R[1]$  with label **a** is  $O[9]$ , then the first frame for path beginning with label **a** will be 9. With the help of keeping track of  $\Sigma$ , the calculation between  $R[1]$  and  $\{O[1], O[5], O[9]\}$  is straight forward.



(b) At  $t = 2$  (Decoding);  $\{\mathbf{b}, 10\}$ , the pair of label identified and frame matched. At  $t = 2$ , the only possible label following label **a** from  $t = 1$  is **b**, thus making frames in  $\{2, 3, 6, 10\}$  the possible candidates. Let  $O[10]$  be the closest frame from the candidates to  $R[2]$



(c) At  $t = 3$  (Decoding):  $\{\mathbf{c}, 4\}$ , the pair of label identified and frame matched. At  $t = 3$ , the possible labels following label **b** from  $t = 2$  is **b** and **c** by examining the forward links from state 10. The possible frames are now the union of labels **b** and **c**,  $\{2, 3, 4, 6, 7, 10, 11\}$ . Let the closest frame from the candidates to  $R[3]$  be  $O[4]$ , the result path beginning at label **a** is  $\{9, 10, 4\}$ . The steps from (a) to (c) are done for all other 3 possible paths as well

**Figure 2.6:** Decoding steps: Consider the target time series represented as the *VMO* shown above, the same from figure 2.1. The light gray parts of each subplot are the same from figure 2.1. In each subplot, parts marked by black with thick arrows indicate the path for the chosen state, dark gray ones with thick arrows represent possible paths and filled circle represents the candidate states. Numbers on the thick black arrows are step numbers. In this example, the query  $R$ , is assumed to have 3 frames and the subplots demonstrate hypothetical steps for the path started with frames in  $O$  in cluster labeled by **a** (among 4 possible paths started via **a**, **b**, **c** or **d**). Here the visualization of the query time series is omitted and the path is chose generically to demonstrate algorithm 6.

---

**Algorithm 6** Query-Matching

---

**Require:** Target signal in *VMO*,  $Oracle(Q = q_1, q_2, \dots, q_T, O = O[1], O[2], \dots, O[T])$  and query time series  $R = R[1], R[2], \dots, R[N]$

- 1: Get the number of clusters,  $M \leftarrow |\Sigma|$
- 2: Initialize cost vector  $C \in \mathbb{R}^M$  and path matrix  $P \in \mathbb{R}^{M \times N}$ .
- 3: **for**  $m = 1 : M$  **do**
- 4:      $P_{m,1} \leftarrow$  Find the state,  $t$ , in the  $m$ th list from  $\Sigma$   
          with the least distance,  $d_{m,1}$ , to  $R[1]$
- 5:      $C_m \leftarrow d_{m,1}$
- 6: **end for**
- 7: **for**  $n = 2 : N$  **do**
- 8:     **for**  $m = 1 : M$  **do**
- 9:          $P_{m,n} \leftarrow$  Find the state,  $t$ , in lists with labels  
              corresponding to forward links from state  
               $P_{m,n-1}$  with the least distance,  $d_{m,n}$  to  $R[n]$
- 10:          $C_m += d_{m,n}$
- 11:     **end for**
- 12: **end for**
- 13: **return**  $P[\text{argmin}(C)], \min(C)$

---

to the number of labels assigned by a *VMO* and the first entry for each path is located according to the distance between the first observation from the input stream to every frame in each label. Algorithm 7 returns a path vector  $P$  and a cost vector  $C$  and passes them to the query-following algorithm. In algorithm 8, the algorithm is run once to update  $P$  and  $C$  for each path for each incoming sample, then  $P_{\text{argmin}(C)}$  is returned as the found index in target time series  $O$  indicating the best match between the incoming time series  $R$  and stored time series  $O$ .

---

**Algorithm 7** Query Following - Initialization

---

**Require:** Target signal in *VMO*,  $Oracle(Q = q_1, q_2, \dots, q_T, O = O[1], O[2], \dots, O[T])$  and first frame of incoming time series  $R[1]$

- 1: Get the number of clusters,  $M \leftarrow |\Sigma|$
- 2: Initialize cost vector  $C \in \mathbb{R}^M$  and path vector  $P \in \mathbb{R}^M$ .
- 3: **for**  $m = 1 : M$  **do**
- 4:      $P_m \leftarrow$  Find the state,  $t$ , in the  $m$ th list from  $\Sigma$   
          with the least distance,  $d_m$ , to  $R[1]$
- 5:      $C_m \leftarrow d_m$
- 6: **end for**
- 7: **return**  $P_{\text{argmin}(C)}, P, C$

---

---

**Algorithm 8** Query Following - On-line Update

---

**Require:** Target signal in *VMO*,  $Oracle(Q = q_1, q_2, \dots, q_T, O = O[1], O[2], \dots, O[T])$ , path vector  $P$ , cost vector  $C$  from algorithm 7 and incoming time series  $R[n]$  at time  $n$ .

- 1: **for**  $m = 1 : M$  **do**
  - 2:      $P_{temp} \leftarrow P_m$
  - 3:      $P_m \leftarrow$  Find the state,  $t$ , in lists with labels  
                    corresponding to forward links from state  
                     $P_{temp}$  with the least distance,  $d_m$  to  $R[n]$
  - 4:      $C_m += d_m$
  - 5: **end for**
  - 6: **return**  $P_{\text{argmin}(C)}, P, C$
- 

### 2.2.3.3 A Probabilistic Interpretation

In this section, the analogy between the inference problem in HMM and the query-matching algorithm used by *VMO* will be elaborated. Given an HMM learned from  $O = O[1], O[2], \dots, O[T]$  specified as, initial probabilities  $\pi_i$  of being at state  $i$ , transition probabilities  $a_{ij}$  for transitioning from state  $i$  to state  $j$ , the hidden state at time step  $n$  denoted by  $x_n$  and emission probabilities  $P(R[n]|m)$  for observed variable  $R[n]$  at time step  $n$  generated by state  $m$  and. Given an observed time series  $R = R[1], R[2], \dots, R[N]$ . The most likely hidden states sequence  $x_1, \dots, x_N$  that generates  $R$  could be found by the recurrence relations:

$$V_{1,m} = P(R[1]|m) \cdot \pi_m,$$
$$V_{n,m} = \max_{m'} (P(R[n]|m) \cdot a_{m'm} \cdot V_{n-1,m'}).$$

Then given a *VMO* indexing  $O = O[1], O[2], \dots, O[T]$ , we can replace the terms in the recurrence relations in HMM with attributes in *VMO*. First of all, we treat the clusters of frames  $\sigma_1, \dots, \sigma_M$  stored in  $\Sigma$  as hidden states in an HMM. With  $\Sigma$  available from *VMO*, initial probability is replaced by an empirical frequency estimation from *VMO* as

$$\pi_i = \frac{|\sigma_i|}{T},$$



with  $|\sigma_i|$  denoting the number of frames stored in  $\sigma_i$ . The transition probability is replaced as

$$a_{ij} = \frac{1(\exists \delta(t, j), t \in \sigma_i) |\sigma_j|}{\sum_{j'=1}^M 1(\exists \delta(t, j'), t \in \sigma_i) |\sigma_{j'}|},$$

with  $1(\cdot)$  an indicator function returns 1 if the condition enclosed is true or 0 otherwise. The replacement for the transition probability is an approximation with conditional empirical frequency estimation from transition candidates by forward links. For the emission probability, we replace it by

$$P(R[n]|m) \propto \exp\left(\frac{-d(R[n], m)}{\alpha}\right),$$

with  $d(R[n], m) \geq 0$  and  $\alpha \geq 0$ .  $\alpha$  is a scalar controlling the variance which is assumed to be 1.  $d(R[n], m)$  is a cost function, and we define the cost function as

$$d(R[n], m) \triangleq \min_{t' \in \sigma_m} \|R[n] - O[t']\|.$$

The definition of the cost function refers to finding the closest frames in  $\sigma_m$  to the incoming observation  $R[n]$  and returning their distance. If we take log on both sides of the recurrence relations along with combining the above replacements, the recurrence relations could be re-

written as:

$$\begin{aligned}
\log(V_{1,m}) &= \log(\exp(-\min_{t' \in \sigma_m} \|R[1] - O[t']\|) \cdot \frac{|\sigma_m|}{T}) \\
&= -\min_{t' \in \sigma_m} \|R[1] - O[t']\| + \log\left(\frac{|\sigma_m|}{T}\right), \\
\log(V_{n,m}) &= \max_{m'} \log((\exp(-\min_{t' \in \sigma_m} \|R[n] - O[t']\|)) \\
&\quad \cdot \frac{1(\exists \delta(t, m), t \in \sigma_{m'})|\sigma_m|}{\sum_{j=1}^M 1(\exists \delta(t, j), t \in \sigma_{m'})|\sigma_j|} \cdot V_{n-1,m'}) \\
&= \min_{m'} (\min_{t' \in \sigma_m} \|R[n] - O[t']\| \\
&\quad + \log\left(\frac{1(\exists \delta(t, m), t \in \sigma_{m'})|\sigma_m|}{\sum_{j=1}^M 1(\exists \delta(t, j), t \in \sigma_{m'})|\sigma_j|} \cdot V_{n-1,m'}\right)).
\end{aligned} \tag{2.3}$$

then the comparison between the above re-written recursive relations to the query-matching algorithm proposed in algorithm 6 can be made. One can first notice that algorithm 6 not only identifies the hidden states ( $\sigma_{m'}$ ) but also the extracted frame  $O[t']$  from that hidden state, which is consistent with the cost function  $\min_{t' \in \sigma_m} \|R[n] - O[t']\|$  proposed above. Then the next observation is that the probabilities for transitioning to different  $\sigma_m$  is the number of frames in  $\sigma_m$  proportional to the sum of number of frames possible to transition to from  $\sigma_{m'}$ . Combine the above two observations, it could be established that the probability distribution for candidate frames before considering the emission probability is actually uniform. Thus one only need to consider the cost function in (2.3) which means  $\log(V_{1,m})$  and  $\log(V_{n,m})$  is actually exactly  $C_m$  in algorithm 6.

## 2.2.4 Context-Aware Hidden Markov Models

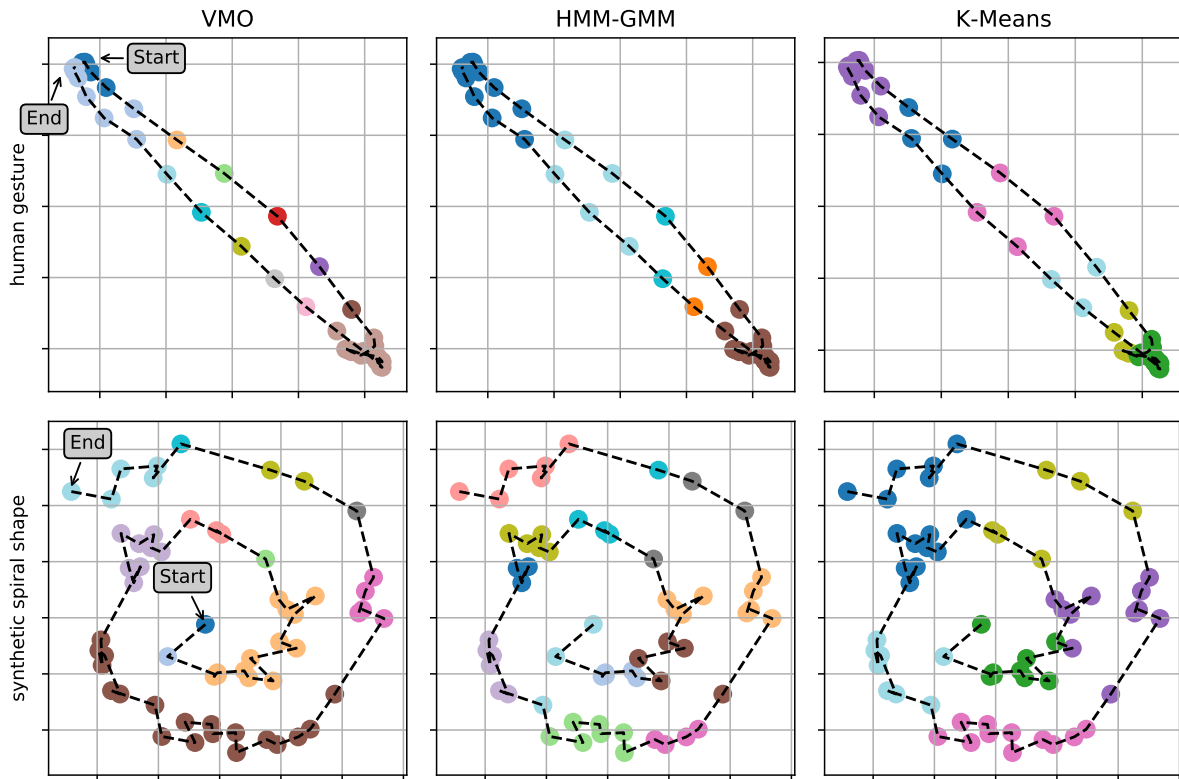
It is shown in [45] and previous paragraphs that a *VMO* is capable of clustering data points of a multivariate time series based on their temporal relations, and tracking the sequential transitions between these clusters. This effectively makes *VMO* into a latent temporal model with

special behavior that is different from the common HMM due to its variable length modeling property. In this section, a further study of how a *VMO* differs from a HMM is presented. A visual demonstration of the context aware aspect of a HMM constructed by *VMO* (*VMO*-HMM hereafter) is depicted in figure 2.7. In the example these observations are clustered into groups according to different clustering methods. Figure 2.7 shows these results by marking the cluster assignments with different colors. Examples of time series modeled by a *VMO*, an HMM Gaussian Mixture Models (HMM-GMM) and a K-Means clustering algorithm are shown. It should be noted that the K-Means clustering algorithm clusters observations based on spatial positions only (in the feature space), and that both the *VMO* and the HMM-GMM take the time trajectories into account. In these examples it is evident that only *VMO* is capable of distinguishing between observations that are spatially but not sequentially (temporally) close to each other and assigning them to different clusters. After examining the results it is clear that the establishment of clusters by an HMM-GMM and a K-means clustering algorithm is mainly determined by spatial relationships between observations but not temporal relationships. Although the possibility of forming a latent model was discussed in previous studies of *VMO* and Viterbi-like query-matching algorithms were described in previous sections, the *VMO* data structure still has to be kept in order for those algorithms to work. In this section, the proposed latent model is a compact version of the *VMO* data structure with a novel statistical model and probabilistic interpretations.

#### 2.2.4.1 Variable Markov Oracle as Latent Model

A *VMO* could be viewed as an on-line clustering algorithm without the need to specify the number of clusters as described earlier. A *VMO* could also be considered as a data structure that traces repeated sub-sequences in the latent space. These two properties make *VMO* capable of both modeling and generating multivariate time series.

The clustering property of a *VMO* is explained in detail in [45] and section 2.2.1. The *VMO* was introduced as a data structure that allowed symbolization and clustering, but without an



**Figure 2.7:** Two time series examples modeled by three different approaches. The gesture in the top row is a real world 3-D human skeletal joints gesture projected onto its first two principle components. The bottom one is a synthetic spiral sequence. The bottom one is a shape commonly tested in manifold discovering and clustering. Three approaches are used to model these time series. (From right to left) K-Means, HMM-GMM and *VMO*. Each observation along the time series is represented as a colored circle with its color represents the label (hidden/latent state) that the observation is assigned to. observations with the same color in the same plot belong to the same label. The starting and ending positions of the two time series are annotated in the left most column. Dashed lines connecting the data points represent the time progression trajectories of each time series.

underlying statistical model. In [46] and section 2.2.3.3, an HMM analogy between the Viterbi algorithm and the *VMO* query-matching algorithm was made as the first attempt to establish a statistical interpretation for the *VMO*. It took into account the inference of emission probabilities from observation and but did not model transition probabilities. The work described in this section is the most complete analogy or statistical interpretation of the *FO* structure after IR optimization to an HMM.

In short, a *VMO* records where and how long the longest repeated suffixes happened for every time step in the time series, and stores them in two arrays, `sfx` and `lrs` respectively. Since for each observation only one longest repeated suffix is recorded, each observation in a time series is indexed by the *VMO* by assigning a label that is based on the unique paths that are defined by suffix records. The labels are stored in  $\Sigma$ . Observations assigned the same label possess two properties that are utilized in this section: The first one is that the distances between the observations connected by suffix links are below a found threshold  $\theta$  during the model selection process. The second one is that they all share common suffixes in the latent space. The first property sets the basis of modeling observations with a latent variable, the second property provides a Markovian relationships between the latent variables.

#### 2.2.4.2 VMO-HMM

The HMM-like model extracted from a *VMO* is called the VMO-HMM. To extract a VMO-HMM from a *VMO*, each latent variable is represented by the centroid extracted from the clustered observations. The choice of centroid could be flexible, such as the mean or the median, depending on the applications. To extract the Markov transition probabilities from a *VMO*, the `lrs` array is used. The `lrs` arrays contains the lengths of the longest repeated suffixes, thus it also provides variable-length Markov transition information. To obtain these information, a 3-D Markov transition tensor is created instead of a 2-D Markov transition matrix. In algorithm 9, a simple algorithm is provided to show how the tensor is extracted.

---

**Algorithm 9** HMM tensor extraction

---

**Require:** An indexed VMO  $V$ , max variable Markov order length  $M$

```
1:  $N \leftarrow$  the number of latent variables in  $V$ 
2: Create a 3-D tensor  $S$  with dimensions  $\{M, N, N\}$ 
3:  $T \leftarrow$  the number of data points in  $V$ 
4: for  $t = 2 : T$  do
5:    $i \leftarrow \text{latent}_V[t-1]$ 
6:    $j \leftarrow \text{latent}_V[t]$ 
7:   if  $\text{lrs}_V[t] < 2$  then
8:      $S[1, i, j] += 1$ 
9:   else
10:     $S[1 : \text{lrs}_V[t] - 1, i, j] += 1$ 
11:   end if
12: end for
13: for  $m = 1 : M$  do
14:   Normalize each row in  $S[m, :, :]$ 
15: end for
16: return  $S$ 
```

---

In algorithm 9, the counts of occurrences between consecutive latent variables are accumulated across the first dimension of  $S$ , with the index of the first dimension representing the order of each Markov transition matrix. In section 3.3, jazz music analysis with the help from VMO-HMM is presented, and in section 4.4, the use of VMO-HMM in generating new musical materials is shown.

Chapter 2 is adapted from published materials in "*Variable Markov Oracle: A Novel Sequential Data Points Clustering Algorithm with Application to 3D Gesture Query-Matching*". Wang, Cheng-i. & Dubnov, Shlomo. International Symposium on Multimedia, 2014, 215-222, "*The Variable Markov Oracle: Algorithms for Human Gesture Applications*". Wang, Cheng-i. & Dubnov, Shlomo. IEEE MultiMedia, IEEE, 2015, 22, 52-67 and "*Context-Aware Hidden Markov Models of Jazz Music with Variable Markov Oracle*". Wang, Cheng-i. & Dubnov, Shlomo. 5th International Workshop on Musical Metacreation (MUME 2017) at the Eight International Conference on Computational Creativity, ICC3 2017, 2017. All of the algorithms described here could be found on <https://github.com/wangsix/vmo>.

# Chapter 3

## Analysis with the VMO

### 3.1 Motif Discovery

Automatic discovery of musical patterns (motifs, themes, sections, etc.) is a task defined as identifying salient musical ideas that repeat at least once within a piece [47, 48] with computational algorithms. In contrast to “segments” found in the music segmentation task [49], the patterns found here may overlap with each other and may not cover the entire piece. In addition, the occurrences of these patterns could be inexact in terms of harmonization, rhythmic pattern, melodic contours, etc. Lastly, hierarchical relations between motifs, themes and sections are also desired outputs of the pattern discovery task.

Two major approaches for symbolic representations are the string-based and the geometric methods. A string-based method treats a symbolic music sequence as a string of tokens and applies string pattern discovery algorithms on the sequence [50, 51]. A geometric method views musical patterns as shapes appearing on a score and enables inexact pattern matching as similar shapes imply different occurrences of one pattern [52, 53]. For a comprehensive review of pattern discovery with symbolic representations, readers are directed to [48]. For audio representations, geometric methods for symbolic representations have been extended to handle audio signals

by multi  $F_0$ -estimation with beat tracking techniques [54]. Approaches adopted from music segmentation tasks using self-similarity matrices and greedy search algorithms are proposed in [55, 56]. Most of the research involving audio representations has been focused on “deadpan audio” rendered from MIDI. In [54], the pattern discovery task is extended to live performance audio recordings with a single recording for each music piece. In the current study, instead of directly applying the proposed framework on performance recordings, multiple recordings are gathered for each musical piece to aid the pattern discovery on deadpan audio. In following sections, a string-based approach utilizing the clustering and repeated suffixes properties of the *Variable Markov Oracle (VMO)* is described and evaluated.

### 3.1.1 Pattern Discovery by VMO

Algorithm 10 shows the string-based algorithm for the automatic pattern discovery task using the *VMO*. The idea behind algorithm 10 is to track patterns by following suffix link (*sfx*), *rsfx* and length of longest repeated suffixes (*lrs*). *rsfx* stands for the reverse suffix links and stores where the suffix links are from for each frame. *sfx* and *rsfx* provides the locations of patterns, and *lrs* indicates the length of these patterns. In line 5 of algorithm 10, checks are made so that redundant patterns are avoided, and the lengths of patterns are larger than a user-defined minimum  $L$ . From line 6 to 10, the algorithm recognizes occurrences of established patterns, and from line 11 to 15 it detects new patterns and stores them into *Pttr* and *PttrLen*.

Algorithm 10 returns *Pttr*, *PttrLen* and  $K$ . *Pttr* is a list of lists with each  $Pttr[k], k \in \{1, 2, \dots, K\}$ , a list containing the ending indices of different occurrences of the  $k$ th pattern found.  $K$  is the total number of patterns found. *PttrLen* has  $K$  values representing the length of the  $k$ th pattern in *Pttr*.



---

**Algorithm 10** Pattern Discovery using *VMO*

---

**Require:** *VMO*,  $V$ , of length  $T$  and minimum pattern length  $L$ .

**Ensure:**  $sfx, rsfx, lrs \in V$

```
1: Initialize Pttr and PttrLen as empty lists.
2: Initialize  $prevSfx = -1, K = 0$ 
3: for  $i = T : L$  do
4:    $pttrFound = False$ 
5:   if  $i - lrs[i] + 1 > sfx[i] \wedge sfx[i] \neq 0 \wedge lrs[i] \geq L$  then
6:     if  $\exists k \in \{1, \dots, K\}, sfx[i] \in Pttr[k]$  then
7:       Append  $i$  to  $Pttr[k]$ 
8:        $PttrLen[k] \leftarrow \min(lrs[i], PttrLen[k])$ 
9:        $pttrFound = True$ 
10:    end if
11:    if  $prevSfx - sfx[i] \neq 1 \wedge pttrFound == False$  then
12:      Append  $\{sfx[i], i, rsfx[i]\}$  to  $Pttr$ 
13:      Append  $\min\{lrs[\{sfx[i], i, rsfx[i]\}]\}$  to  $PttrLen$ 
14:       $K \leftarrow K + 1$ 
15:    end if
16:     $prevSfx \leftarrow sfx[i]$ 
17:  else
18:     $prevSfx \leftarrow -1$ 
19:  end if
20: end for
21: return  $Pttr, PttrLen, K$ 
```

---

## 3.1.2 Experiments

The dataset chosen for the music pattern discovery is the JKU Pattern Development Dataset (JKU-PDD) [47]. This dataset consists of five polyphonic classical music pieces or movements in both symbolic and audio representations. The ground truth of repeated patterns (motifs, themes, sections) for each piece is annotated by musicologists. The details of the experimental setup are provided in the following sections.

### 3.1.2.1 Feature Extraction

For the automatic musical pattern discovery task, the chromagram is the input feature to algorithm 10 for both the symbolic and audio representations. The chromagram is a feature that characterizes harmonic content and is commonly used in musical structure discovery [57].

**3.1.2.1.1 Symbolic Representation** For the experiments described in this section, the symbolic representation chosen is MIDI, but other symbolic representations may be used instead. The chromagram derived from the symbolic representation is referred to as the “midi-chromagram”.

The midi-chromagram is similar to the midi-histogram described in [58] and represents the presence of pitch classes during each time frame. To create a midi-chromagram with quantization  $b$  in terms of MIDI whole note beats, frame size  $M$ , and hop size  $h$ , the MIDI file is first parsed into a matrix where each column is a MIDI beat quantized by  $b$  and each row is a MIDI note number (0 – 127). For each analysis frame, the velocities are summed over  $M$  MIDI beats, and then folded and summed along the MIDI notes to create a single octave of velocities. In other words, all velocities that correspond to MIDI notes that share the same modulo 12 are summed. The analysis frame then hops  $h$  MIDI beats forward in time, repeats the folding and summing, and continues on until the end of the MIDI matrix is reached. The bottom plot in figure 3.1 is an example of the midi-chromagram extracted from the Beethoven minuet MIDI file in the JKU-PDD dataset.

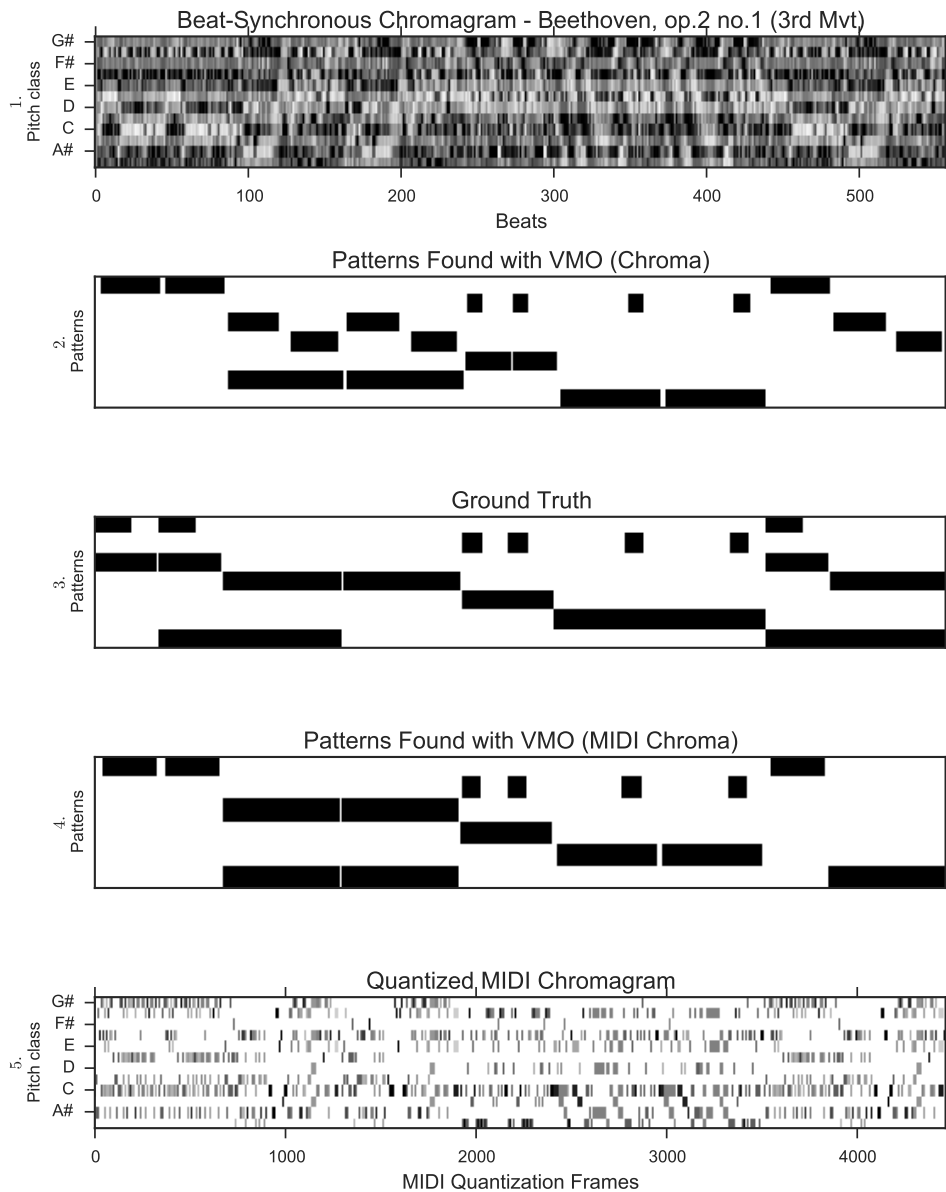
**3.1.2.1.2 Audio Recording** The routines for extracting the chromagram from an audio recording used in this section is as follows. For a mono audio recording sampled at 44.1 kHz, the recording is first down-sampled to 11025 Hz. Next, a spectrogram is calculated using a Hann window of length 8192 with 128 samples overlap. Then the constant-Q transform of the spectrogram is calculated with frequency analysis ranging between  $f_{min} = 27.5$  Hz to  $f_{max} = 5512.5$  Hz and 12 bins per octave. Finally, the chromagram is obtained by folding the constant-Q transformed spectrogram into a single octave to represent how energy is distributed among the 12 pitch classes.

To achieve the pattern discovery on a music metrical level, the chroma frames are aggregated with a median filter according to the beat locations found by a beat tracker [59] conforming to the music metrical grid. For finer rhythmic resolution, each beat identified is spliced into two sub-beats before chroma frame aggregation. Lastly, the sub-beat-synchronous chromagram is whitened with a *log* function. Whitening boosts the harmonic tones implied by the motifs so that the difference between the same motif with and without harmonization is reduced. See the top plot in figure 3.1 for an example of the the beat-synchronous chromagram extracted from the Beethoven minuet deadpan audio in the JKU-PDD dataset.

### 3.1.2.2 Repeated Themes Discovery

For both symbolic and audio representations, after the chroma feature sequence  $O$  is extracted from the music piece as described in section 3.1.2.1.1 and 3.1.2.1.2,  $\theta \in (0.0, 2.0]$  is used to construct multiple *VMOs* with  $O$ . The  $L_2$ -norm is used to calculate the distance between incoming observations and the ones stored in a *VMO*. The single *VMO* with the highest Information Rate (IR) is fed into algorithm 10 with  $L$  to find patterns and their occurrences. Instead of setting  $L = 5$  for all pieces as in [60],  $L$  is set according to `lrs` as  $L = \frac{\gamma}{T} \sum_{t=1}^T \text{lrs}[t]$ , where  $L$  is adaptive to the average length of repeated suffixes found in the piece.  $\gamma$  is a scaling parameter which is set to 0.5 empirically.

To consider transposition (moving chroma patterns up or down by a constant pitch



**Figure 3.1:** Features, found patterns, and ground truth for the Beethoven minuet in the JKU-PDD. 1. Beat-synchronous chromagram from the deadpan audio recording. 2. Patterns found by algorithm 10 using the chromagram shown above. 3. Ground truth from JKU-PDD. 4. Patterns found by algorithm 10 using the midi-chromagram. 5. Quantized midi-chromagram. For 2., 3. and 4., each row is a pattern place holder with dark regions representing the occurrences on the timeline. The order of found patterns is manually sorted to best align with the ground truth for visualization purposes. Notice the hierarchical relations of patterns embedded in the ground truth and how they are found from the algorithms.

interval), the distance function used for *VMO* structures is a cost function with transposition invariance. For a transposition invariant cost function, a cyclic permutation with offset  $k$  on an  $n$ -dimensional vector  $\mathbf{x} = (x_0, x_1, \dots, x_{n-1})$  is defined as

$$cp_k(\mathbf{x}) := \{x_i \rightarrow x_{(i+k \bmod n)}, \forall i \in (0, 1, \dots, n-1)\},$$

and the transposition invariant dissimilarity  $d$  between two vectors  $x$  and  $y$  is defined as,  $d = \min_k \{\|x - cp_k(y)\|_2\}$ .  $n = 12$  for the chroma vector, and the cost function is used during the *VMO* construction and model selection.

In addition to the regular chromagram, a stacked chromagram using time-delay embedding with  $M$  steps of history as in [61] is also used. Experiments reveal that choices for  $b$ ,  $M$ , and  $h$  for both the midi-chromagram and the stacked midi-chromagram can greatly alter the accuracy of patterns discovered. The values used in the experiments were quantization sizes  $b = [\frac{1}{8}, \frac{1}{16}, \frac{1}{32}]$ , frame size  $M = [1, 8, 16, 32]$ , and hop lengths  $h = [1, 2, 4]$  where  $M$  and  $h$  are described in terms of MIDI beats of size  $b$ . It was found that the stacked midi-chromagram with  $b = \frac{1}{32}$ ,  $M = 16$ , and  $h = 2$  resulted in the best pattern discovery. For the audio representation, there is no significant difference in terms of the patterns found or the evaluation metrics between regular and stacked chromagrams.

Figure 3.1 shows the chromagram found from audio and MIDI for the Beethoven minuet in the JKU-PDD along with the patterns found by the *VMO* structure and the ground truth patterns. The patterns found by the audio and symbolic representations share similarities and visually resemble the ground truth patterns. In section 3.1.3, quantitative measures for evaluating the patterns found by the *VMO* are explained and reported.

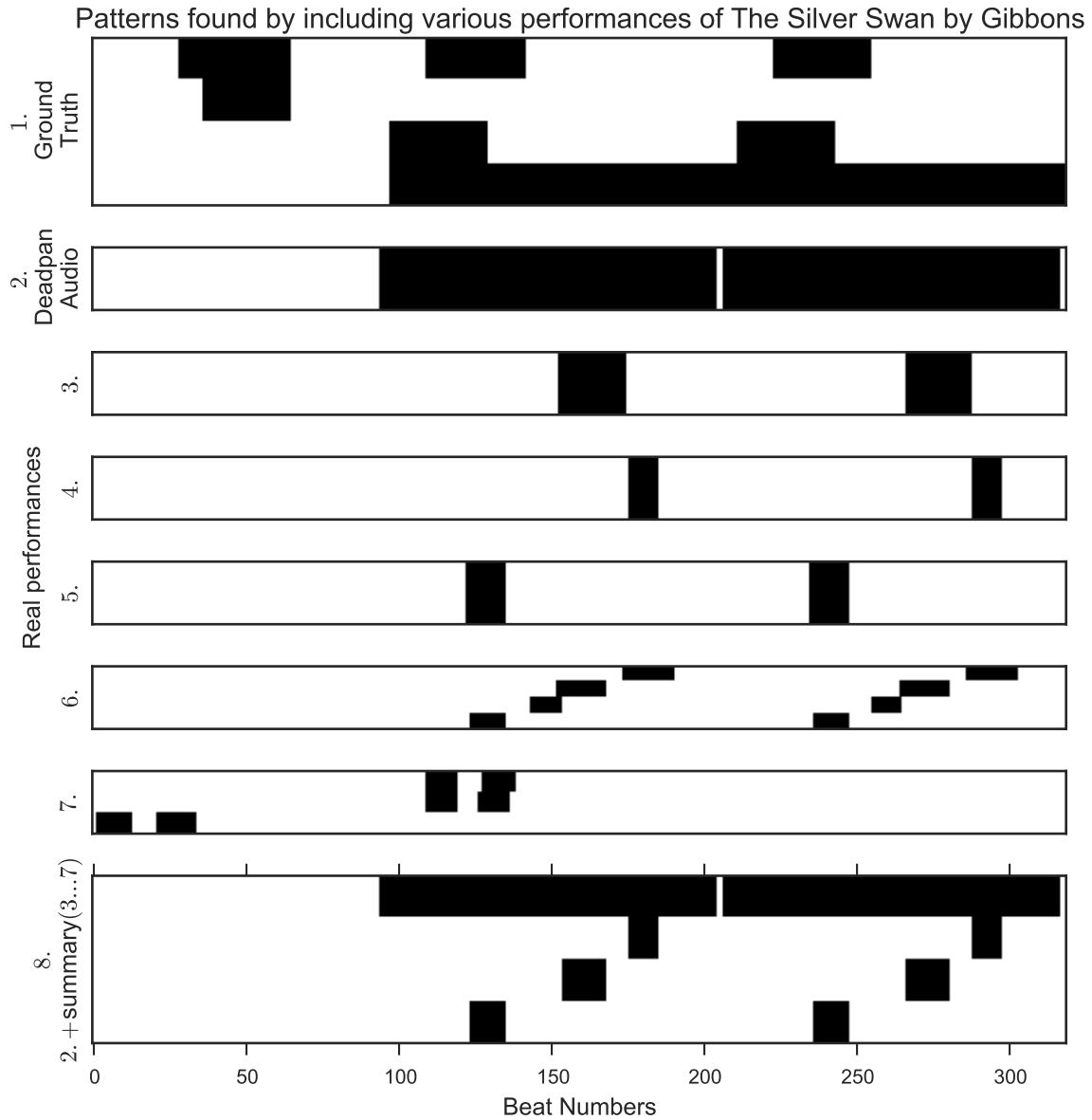
### 3.1.2.3 Performance Recordings to Aid Pattern Discovery

Five performance recordings for each of the pieces included in the JKU-PDD are collected in order to further explore the discovery of repeated themes. The motivation behind this experiment is to explore the notion that music performances contain information about how performers interpret the musical structure embedded in the score [62] and to examine whether or not the patterns found on deadpan audio could be improved with the addition of such information.

For each of the performance recordings, the chromagram is extracted and aggregated along the beats as described in section 3.1.2.1.2. DTW [63] is used to align the beat-synchronous chromagram from the performance audio with the beat-synchronous chromagram of the deadpan audio. Since motif annotations on these performance recordings do not exist yet, the alignment between the deadpan audio and performance recordings are necessary so that the patterns found from the performance recordings can be compared to the ground truth or added to the found patterns from the deadpan audio. The drawback of the alignment is that timing variations containing the performer’s structural interpretations are lost. Although timing variations are lost in this experiment, velocity variations applied across time and different voices are retained. The aligned performance audio chromagram is then whitened, normalized and fed into the *VMO* pattern finding algorithm. For patterns found across multiple performances of one piece, the intersection of patterns for any two performances of one piece that are longer than  $L$  are kept and added to the found patterns from the deadpan audio. Figure 3.2 is an example of how incorporating performance recordings can change the discovered patterns from deadpan audio.

### 3.1.3 Evaluation

The evaluation follows the metrics proposed in the Music Information Retrieval Evaluation eXchange (MIREX) [53]. Three metrics are considered for inexact pattern discovery. For each metric, standard  $F_1$  score, defined as  $F_1 = \frac{2PR}{(P+R)}$ , precision  $P$  and recall  $R$  are calculated. The first



**Figure 3.2:** 1. Ground truth from JKU-PDD. 2. Patterns found from deadpan audio with the *VMO*. 3 – 7. Patterns found from the five performances. 8. Patterns from deadpan and performance audio.

**Table 3.1:** Results from various algorithms on the JKU-PDD for both symbolic (upper three) and audio (bottom four) representations. Scores are averaged across pieces. Missing values were not reported in their original publications.

Algorithm	$F_{est}$	$P_{est}$	$R_{est}$	$F_{o(.5)}$	$P_{o(.5)}$	$R_{o(.5)}$	$F_{o(.75)}$	$P_{o(.75)}$	$R_{o(.75)}$	$F_3$	$P_3$	$R_3$	Time (s)
VMO symbolic	<b>60.79</b>	<b>74.57</b>	56.94	71.92	<b>79.54</b>	68.78	<b>75.98</b>	<b>75.98</b>	75.99	<b>56.68</b>	<b>68.98</b>	53.56	<b>4333</b>
[54]	33.7	21.5	<b>78.0</b>	<b>76.5</b>	78.3	<b>74.7</b>	—	—	—	—	—	—	—
[64]	50.20	43.60	63.80	63.20	57.00	71.60	68.40	65.40	76.40	44.20	40.40	<b>54.40</b>	7297
VMO deadpan	<b>56.15</b>	<b>66.8</b>	57.83	<b>67.78</b>	72.93	<b>64.3</b>	<b>70.58</b>	<b>72.81</b>	<b>68.66</b>	<b>50.6</b>	<b>61.36</b>	52.25	<b>96</b>
deadpan + real	52.76	53.2	58.25	67.35	<b>74.42</b>	63.31	70.51	72.73	68.58	48.25	50.2	<b>52.84</b>	—
[56]	49.8	54.96	51.73	38.73	34.98	45.17	31.79	37.58	27.61	32.01	35.12	35.28	454
[54]	23.94	14.9	<b>60.9</b>	56.87	62.9	51.9	—	—	—	—	—	—	—
[65]	41.43	40.83	46.43	23.18	26.6	20.94	24.87	32.08	21.24	28.23	30.43	31.92	196

metric is the establishment score ( $est$ ) which measures how each ground truth pattern is identified and covered by the algorithm. The establishment score takes inexactness into account and does not consider occurrences. The second metric is the occurrence score ( $o(c)$ ) with a threshold  $c$ . The occurrence score measures how well the algorithm performs in finding occurrences of each pattern. The threshold  $c$  determines whether or not an occurrence should be counted. The higher the value for  $c$ , the lower the tolerance.  $c = \{0.5, 0.75\}$  are used in standard MIREX evaluation. The last metric is the three-layer score that considers both the establishment and occurrence score. The results of the proposed framework are listed in table 3.1 along with a comparison to previous work.

From the evaluations for both symbolic and audio representations, the establishment scores are generally lower than the occurrence scores, meaning that the proposed algorithm is better at finding occurrences of established patterns than finding all possible patterns. With the symbolic representation, the standard  $F_{est}$ ,  $F_{o(.75)}$ , and  $F_3$  scores are better than previously published results. The establishment, occurrence, and three-layer precision scores are also as good as or better than previous algorithms [54, 64]. The recall scores reveal that this is a part of the algorithm that could be improved as previous algorithms all scored higher on recall than the proposed algorithm. Similar to the symbolic results, the proposed audio algorithm achieves high  $F1$  and precision scores for the establishment, occurrence, and three-layer scores. The recall of the audio algorithm is higher than previously reported results [54, 55, 56]. The recall rates of the proposed framework are inferior when compared to the precision scores and previous work in

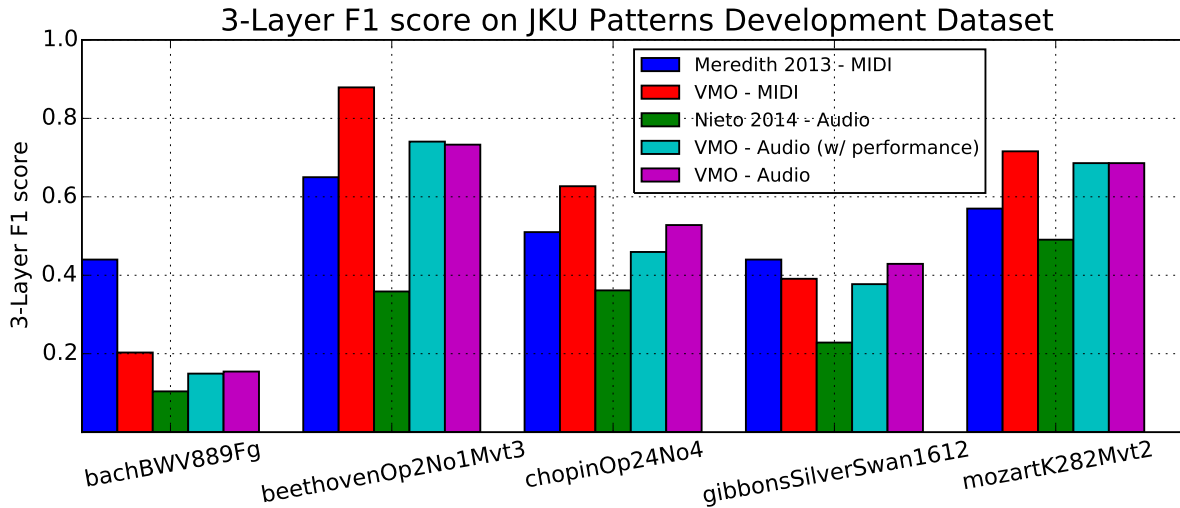


symbolic representation. This may occur because chroma features were used and the folding of the constant-Q spectrogram discards information contained in different voices.

The inclusion of performance recordings is the effort made in this work to improve both the coverage and accuracy of the pattern discovery framework for audio representations. Due to space limitations, the detailed metrics for each piece in the JKU-PDD is not shown here. The effects of including performance recordings are described here. The establishment recall rate and occurrence precision rate with threshold 0.5 are improved when performance recordings are included, but in general the pattern discovery task is not improved because the decrease in establishment precision rate is larger than the improvement on recall rates. This result indicates that more patterns and their occurrences could be discovered if different versions of the same piece are used in the pattern discovery task, but more false positive patterns will be found.

The proposed pattern finding algorithm completed in less time than previously reported algorithms on both symbolic and audio representations. Although the *VMO* data structure is used for both the proposed symbolic and audio algorithms, there is a discrepancy in the time that it takes to find the patterns for all five songs. The audio algorithm takes much less time because the analysis frames are larger than the frames used in the symbolic representation (32th note versus 8th note relatively). Thus, there are less frames to analyze with the audio representation and building a *VMO* takes less time.

Figure 3.3 is a summary of the three-layer  $F_1$  scores for each of the 5 pieces in the JKU-PDD for the proposed audio and symbolic frameworks along with the current state of the art results. The small quantization value for the MIDI representation leads to a higher score in the case of the Beethoven and Chopin pieces. The proposed audio and symbolic framework have the highest  $F_1$  value on the Beethoven minuet and the lowest  $F_1$  value with the Bach Fugue. When looking at the proposed method along with the current state of the art results, it is evident that the Bach Fugue and the Gibbons piece are songs where patterns are embedded in different voices, and that the Beethoven piece has more consistent repeated phrases. The algorithm for symbolic



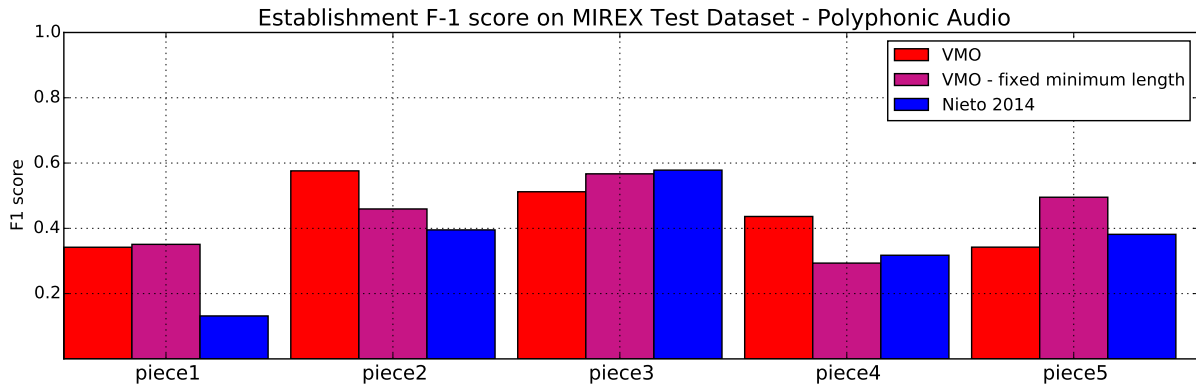
**Figure 3.3:** Three-layer  $F_1$  score ( $F_3$  in table 3.1) for the proposed deadpan audio and symbolic method on the 5 pieces in the JKU-PDD plotted along with state of the art results.

data described in [64] performs better with Bach and Gibbons in comparison to *VMO* and [56], most likely because of its capability to discover patterns embedded in different yet simultaneous voices.

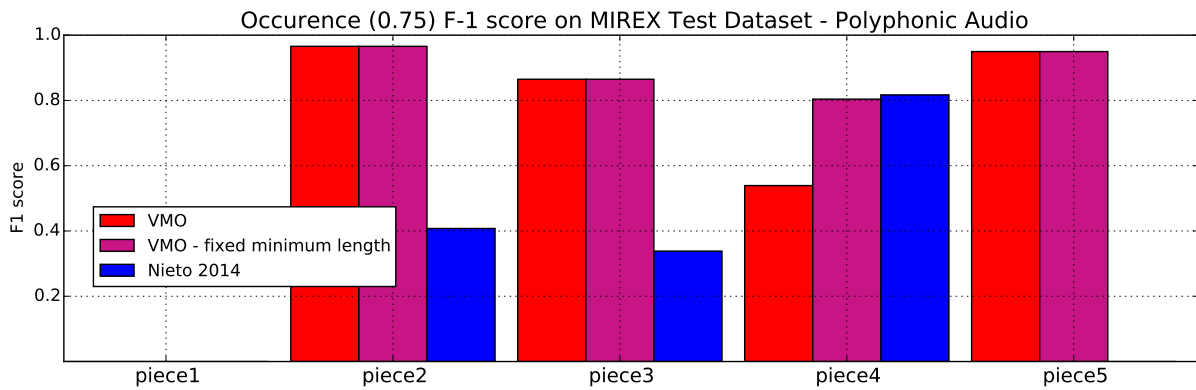
In figure 3.4 and 3.5 and 3.6, the competition results from MIREX on JKU-PDD deadpan audio test set are shown. It is obvious that the *VMO* approach outperform [56] in general and only is inferior for one piece out of five. A result on real performed audio from JKU-PDD development set is shown in figure 3.7, it is also obvious that the *VMO* approach is better than previous state of the art approach.

In summary, the method proposed here has improved upon the  $F_1$  and  $P$  scores as well as time to find patterns. The patterns found using audio and symbolic representations are similar and the evaluation scores reflect this similarity. Improving recall and allowing for inexact occurrences should be a focus for future studies. Source codes and details about the experiments are accessible via Github<sup>1</sup>.

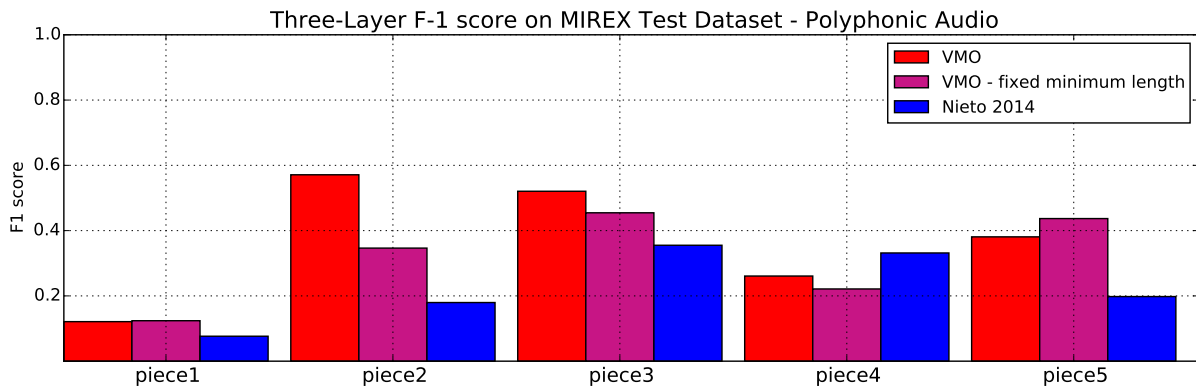
<sup>1</sup>[https://github.com/wangsix/VMO\\_repeated\\_themes\\_discovery](https://github.com/wangsix/VMO_repeated_themes_discovery)



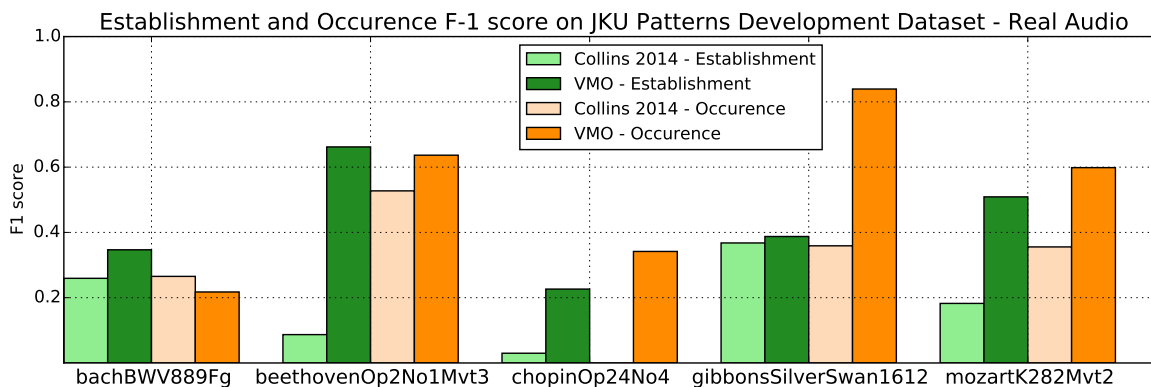
**Figure 3.4:** Establishment scores of the *VMO* and comparison to other participating approaches in MIREX competition.



**Figure 3.5:** Occurrence scores of the *VMO* and comparison to other participating approaches in MIREX competition.



**Figure 3.6:**  $F_3$  of the *VMO* and comparison to other participating approaches in MIREX competition.



**Figure 3.7:** Occurrence and establishment  $F_1$  score for the real audio of the 5 pieces in the JKU-PDD plotted along with state of the art results.

### 3.1.4 Discussion

In this section, a framework for automatic pattern discovery from a polyphonic music piece based on a *VMO* is proposed and shown to achieve state of the art performance on the JKU-PDD dataset and MIREX test set. With both the regular and stacked midi-chromagram, a smaller quantization value  $b$  results in better pattern discovery because finer details are captured with smaller quantization. From the results, it seems that a larger frame size  $M$  for smaller quantization  $b$  resulted in better pattern finding. For hop size  $h$ , it is observed that  $h = 2$  results in a hop of a 16th note which is the shortest note in the JKU-PDD ground truth annotations. Results from both the audio and MIDI representations show that the recall of discovered themes could be improved. Although it is possible for a *VMO* to identify inexact patterns from the input feature sequence with symbolization from  $\theta$ , different occurrences of the same pattern are sometimes not recognized because chroma features discard information from various voices in the music piece. Our framework could be improved if the feature used allows for separation of voices from polyphonic MIDI and audio. Incorporating techniques for identifying multiple voices in polyphonic audio would improve the proposed framework.

In addition to the proposed framework for both symbolic and audio representations, using multiple performance recordings in the repeated themes discovery task for deadpan audio is

another novelty presented in this dissertation. The work done in this dissertation differs from [54] in that the performance audio recordings are used as supplements to deadpan audio and not analyzed as separate musical entities. The original intention behind using deadpan audio for repeated themes discovery is to allow for the use of audio signal processing techniques, but deadpan audio contains the same amount of information as its symbolic counterpart with less accessibility because of its representation. This is evident by the similarity between the MIREX metrics for the MIDI and deadpan audio since similar techniques are applied. Performance recordings, on the other hand, contain expressive performance variations on phrasing and segmentation. In this dissertation, it is shown that adding performance recordings to the proposed framework achieved improvements on some of the standard metrics. The next step for advancing the repeated themes discovery task is to annotate the performance recordings so that these recordings can be used as a dataset directly without referencing back to the deadpan audio version. By observing the results from the pattern finding with performance recordings, the patterns found for each performance show informative cues as to how each rendition of the same piece differs from the others visually (figure 3.2). These visualizations are interesting discoveries on their own, even without a comparison to ground truth annotations, and could be further investigated for use in expressive performance analysis and music structural segmentation.

## **3.2 Structural Segmentation**

Automatically recognizing the segmentation of a music piece is not only a fundamental task in music information retrieval research for music structure analysis, but also leads to the development of efficient music content navigation and exploration applications. Reviews of existing work could be found in [66, 67]. Among various approaches, the Self-similarity matrix (SSM) has been the fundamental building block for several existing algorithms. An SSM captures global repetitive structures containing essential information for music segmentation. Matrix

decomposition of an SSM is widely adopted in existing work. In [68], non-negative matrix factorization (NMF) is used to decompose an SSM into basis functions representing different structural sections. The NMF idea is extended in [69] with a convexity constraint on the weights during decomposition, which leads to more stable results. In [70], ordinal linear discriminant analysis is used to learn feature representations from the singular value decomposition of the time-lag SSM. Spectral clustering is used in [49] to obtain low-dimensional repetition representations from an SSM. Approaches focus on deriving segmentation boundaries from an SSM are also popular. In [71], a checkerboard-like kernel is applied along the diagonal of the SSM to obtain a novelty curve for segmentation boundaries. Structure features are devised in [61] based on time-lag SSM and segmentation boundaries are inferred from structure features.

Approaches based on matrix decomposition or boundary detection represents two aspects of music segmentation problem; finding global structures and local change points. The two problems also corresponds to the categorization of repetition/homogeneous- and novelty-based approach proposed in [67]. In this section, algorithms to address the two aforementioned problems are proposed. The two algorithms can independently or jointly be plugged into various existing segmentation algorithms. For finding global repetitive structures, a novel method for obtaining SSMs is presented. The method is based on the *VMO* [60], a suffix automaton capable of symbolizing multi-variate time-series and keeping track of its repeated motifs. Since repeating sub-sequences are essential in music structure analysis, it is natural to experiment with the SSM obtained from the *VMO* (*VMO*-SSM) in the music structure segmentation task instead of the SSMs obtained with traditional approaches. Conventionally, an SSM is obtained by exhaustively calculating frame-by-frame pairwise distances. For music segmentation tasks, a binary SSM (recurrence plot) is often desired [70, 49, 61]. Nearest-neighbor criterion is often used to obtain a binary SSM from an SSM, but the number of nearest neighbors chosen for each frame is often determined heuristically. The *VMO*-SSM is directly in binary form and the reduction from continuous distance values to binary values is done implicitly according to information dynamics

[29, 30]. For improving the boundary detection accuracy, an iterative boundary adjustment algorithm to post-process the results from segmentation algorithms is proposed. The proposed algorithms are evaluated in the music structure segmentation task with the Beatles ISO dataset [72] against existing approaches based on SSMs.

### 3.2.1 SSM from Variable Markov Oracle

To obtain a binary SSM, the common approach is to apply  $k$ -nearest neighbor thresholding for each frame. Using  $k$ -nearest neighbor thresholding gains scale-invariance on the result binary SSM. With the VMO-SSM, instead of obtaining scale-invariance, the emphasis is on tracing similar trajectories in the metric space drawn by the time series.

The *VMO* is a suffix automaton capable of reducing a multi-variate time series down to a symbolic sequence but still retains repeating sub-sequences as shown in [60] and previous chapter. The *VMO* stores the information regarding repeating sub-sequences within a time series via *suffix links*. For each observation at time  $t$  of the time series with length  $T$  indexed by a VMO, a *suffix link*,  $\text{sf}_x[t] = k$ , is created pointing back in time  $k$  to where the longest repeated suffix happened. The suffix links not only contain the information regarding repeating sub-sequences, but also imply a frame-to-frame equivalency between  $t$  and  $k$  given  $\text{sf}_x[t] = k$  that leads to symbolization of the time-series. Given the symbolized sequence  $S$  by a VMO, a binary SSM (VMO-SSM),  $R \in \mathbb{R}^{T \times T}$ , could be trivially obtained via, with  $t > k$ ,

$$R_{tk} = \begin{cases} 1 & \text{if } \text{sf}_x[t] = k, \\ 0 & \text{otherwise,} \end{cases}$$

and filling the main diagonal line with 1.

The construction and model selection algorithms for *VMO* are documented in chapter 2. A visualization of how a VMO-SSM is obtained is shown in figure 3.8. The advantage of using

a *VMO* to create an SSM over the traditional frame-by-frame pair-wise distance approach is that a *VMO* selectively chooses frames to calculate distances with for each frame based on if common suffices are shared between two frames. The selective behavior leads to a more efficient calculation than the traditional exhaustive manner ( $O(T \log T)$  versus  $O(T^2)$  [38]) and also keeps track of recurrent motifs within the time series. The other difference of using a *VMO* for SSM calculation is that the reduction from a multivariate time series to a symbolic sequence utilizes the concept of information dynamics [29, 30] that aims at modeling the evolving information dynamics as the time series unfolds itself. In the case for the *VMO*, the information theoretic measurement, IR [29] as described in section 2.2.2, is maximized to determine the threshold for frame selection during suffix assignment.

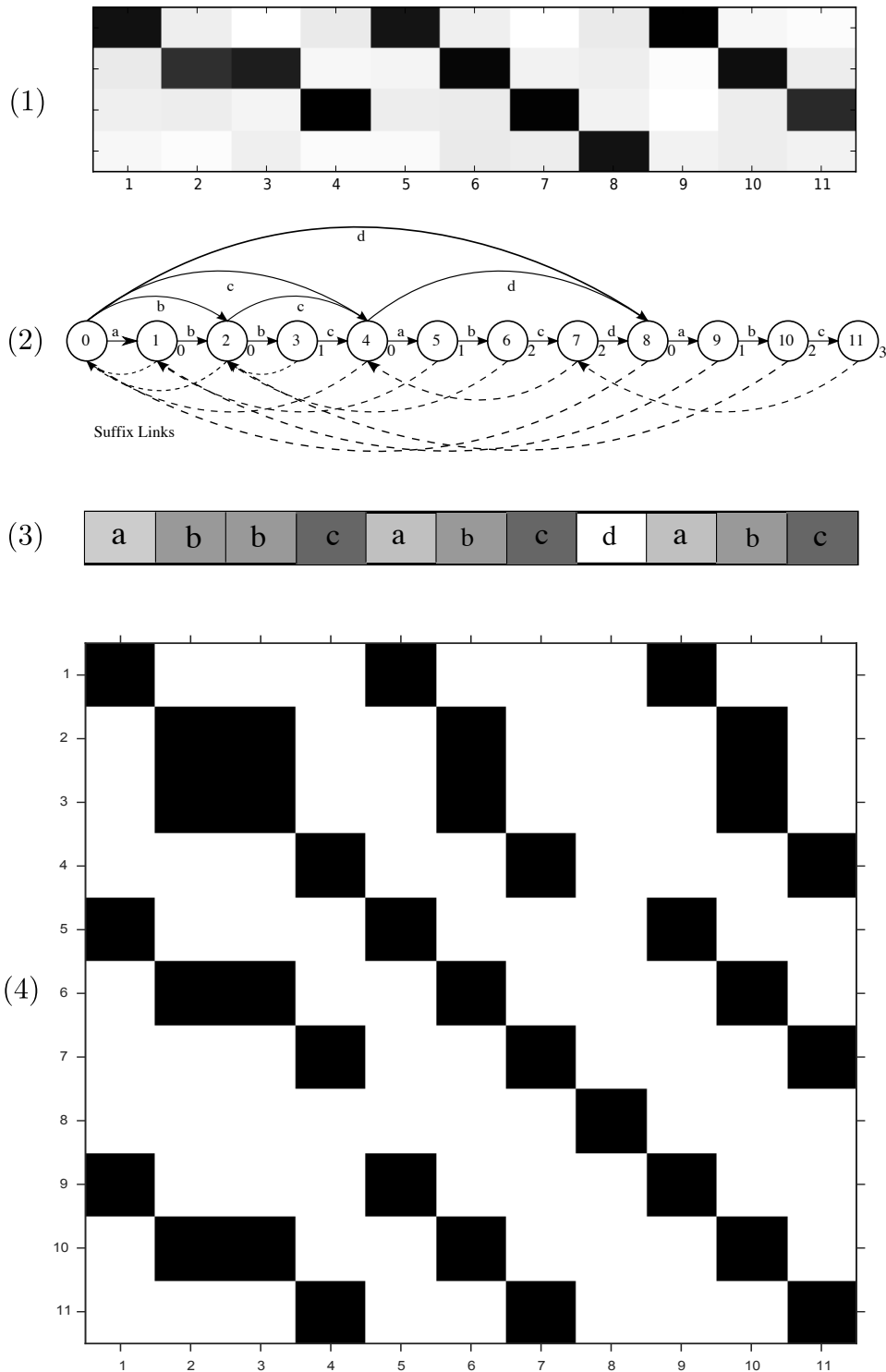
### 3.2.2 Segmentation Algorithm

To show how the VMO-SSM could help improving the music structure segmentation task, and to highlight the difference between the using a VMO-SSM and a traditional SSM, two existing work utilizing binary SSMs are adopted in this work. For both work, their original SSMs are replaced by VMO-SSMs. The first segmentation algorithm is the spectral clustering (SC) approach proposed in [49] and the second is the combination of structure features and segment similarity (SF) proposed in [61].

#### 3.2.2.1 Spectral Clustering

In [49], the observation is that the partition of the graph defined by a connectivity matrix into  $m$  connected components by spectral clustering is effectively the same as segmenting the time series with  $m$  distinguished sections (the total number of segments could be more than  $m$  with repetition of any of the sections). A series of operations are applied on the SSM to obtain a connectivity matrix, then spectral clustering is applied on the connectivity matrix to obtain a low-dimensional representation of repetitive structures. The operations include nearest neighbor





**Figure 3.8:** (1) A synthetic 4-dimensional time series. (2) A VMO structure with symbolized signal  $\{a, b, b, c, a, b, c, d, a, b, c\}$ , lower (dashed) are suffix links. Values outside of each circle are the length of longest repeated suffix. (3) Symbolized signal. (4) The VMO-SSM obtained from the symbolized signal in (3).

thresholding, smoothing with a median filter, adding local linkages, balancing local and global linkages, linkage weighting and feature fusion.

By replacing traditional SSM originally used in [49] with the binary VMO-SSM described in section 3.2.1, only median filtering and adding local linkages are needed to obtain the connectivity matrix  $R^+$  in this work. The median filter is applied in the diagonal direction to suppress erroneous entries, fill missing blanks and keep sharpening edges of the diagonal stripes in the binary SSM

$$R' = \text{median}(R_{i+t,j+t} | t \in -\omega, -\omega + 1, \dots, \omega).$$

The operation of adding local linkage is defined as

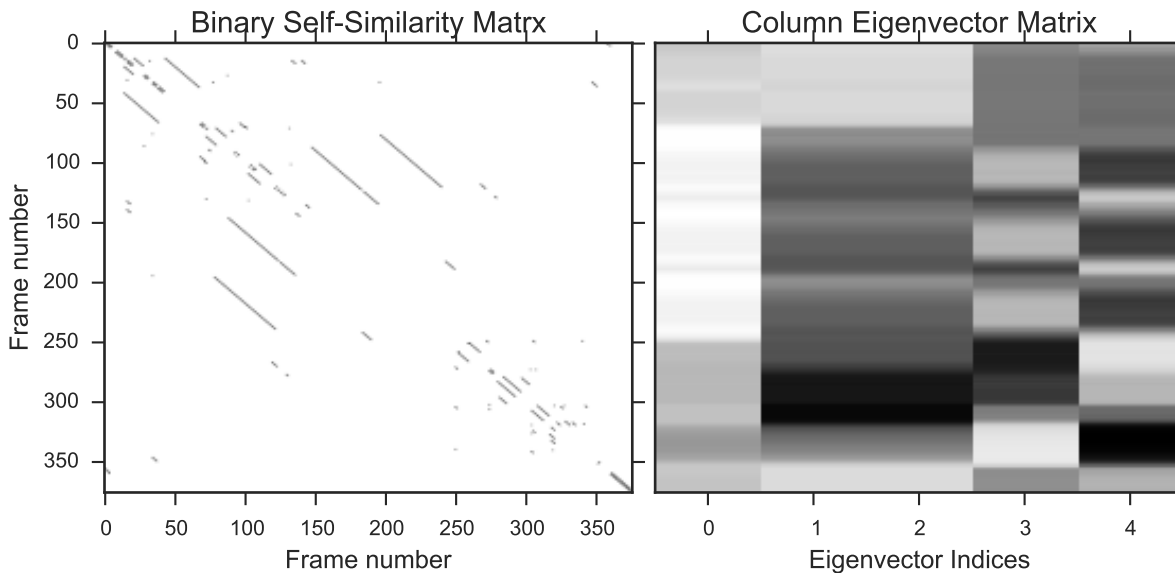
$$\delta_{ij} = \begin{cases} 1 & \text{if } |i - j| = 1, \\ 0 & \text{otherwise} \end{cases}$$

$$R_{ij}^+ = \max(\delta_{ij}, R'_{ij}).$$

Let  $I$  denote a identify matrix with dimension  $N$ , and  $D$  the diagonal degree matrix of  $R^+$ . The symmetric normalized Laplacian matrix of  $R^+$  is then calculated as

$$L = I - D^{-\frac{1}{2}} R^+ D^{-\frac{1}{2}}.$$

The eigenvectors of  $L$  could be interpreted as component membership functions of connected components on a graph defined by  $L$  [73]. The segmentation then follows standard spectral clustering algorithm as documented in [73]. In short, the first  $m$  eigenvectors with  $m$  smallest eigenvalues are concatenated to form a matrix  $Y \in \mathbb{R}^{T \times m}$  with rows normalized, then each row of  $Y$  is treated as one observation in  $k$ -means clustering with  $k = m$ . The assigned label from  $k$ -means clustering is the resulting segmentation labels. Boundaries are inferred from finding label changes between adjacent frames. Visualizations of the  $R^+$  matrix and  $Y$  matrix are depicted

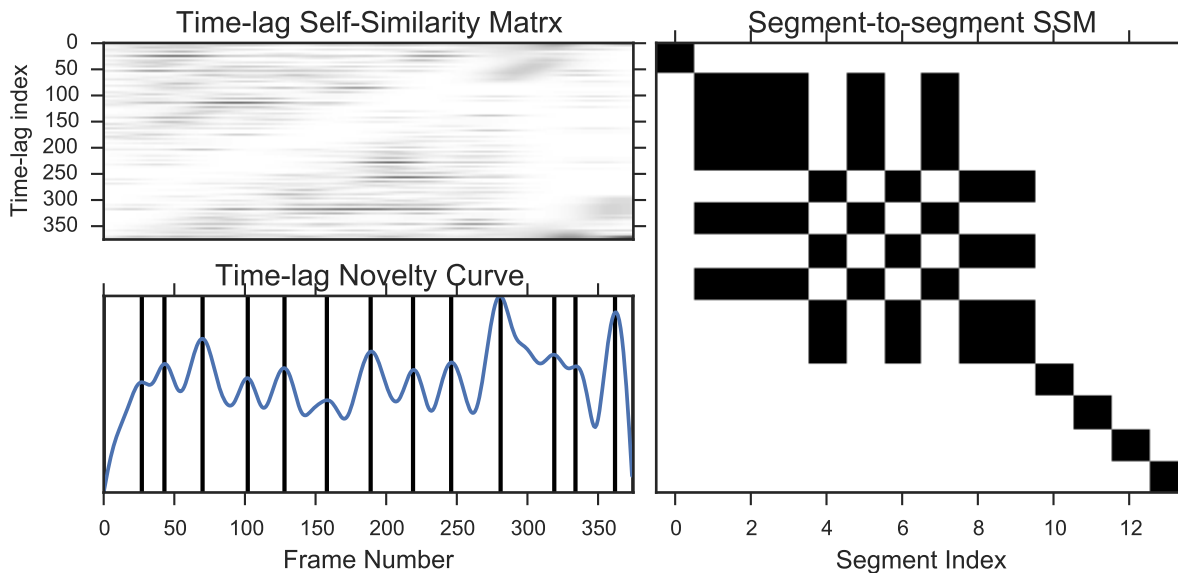


**Figure 3.9:** (Left) The binary VMO-SSM. (Right) The eigenvector matrix,  $Y$ , of “All You Need is Love” from Beatles.

in figure 3.9.

### 3.2.2.2 Structure Features and Segment Similarity

The details of the SF algorithm could be found in [61]. The goal of that work is to base the algorithm on a local presentation (frame-wise) of global structures (from tim-lag SSM). Similar to the replacement process described in section 3.2.2.1, the original binary SSM is replaced by the VMO-SSM. After obtaining  $R$  from the VMO, the following steps are applied to find the boundaries first; 1) a time-lag matrix  $L$  is obtained from  $R$ . 2)  $L$  is convolved with a 2-D Gaussian kernel. 3) Boundaries are identified via peak-picking on a novelty curve derived from  $L$ . To further obtain segment labels, segment-to-segment similarities are calculated based on a DTW-like score given  $R$ . The resulting similarities are stored in a square matrix  $\hat{S}$  with dimensions equal to the number of segments identified from boundary detection. A dynamic threshold based on the statistics of  $\hat{S}$  is used to discard non-similar segments. Transitivity between similar segments is induced by iteratively applying matrix multiplication of  $\hat{S}$  with itself and threshold.



**Figure 3.10:** (Left top) The smoothed time-lag matrix  $L$  from a VMO-SSM. (Left bottom) The novelty curve derived from the time-lag matrix. (Right) The segment-to-segment similarity matrix  $\hat{S}$  of “All You Need is Love” from Beatles.

Segment labels are then obtained from the rows of  $\hat{S}$ . Parameters for this algorithm include the standard deviations of the Gaussian kernel,  $\{s_L, s_T\}$ , for time-lag and time axis respectively, and peak-picking window length  $\lambda$ . An illustration of  $L$ , the novelty curve and  $\hat{S}$  derived from  $R$  is shown in figure 3.10.

### 3.2.3 Boundary Adjustment

Observations after examining the segmentation results from last sections reveal that often times the segmentation algorithm is capable of locating the boundaries between segments within a window of a few seconds but is not capable of locating the major change point within a window less than 1 second. The reason might be due to the smoothing on the SSM to obtain  $R'$  or  $L$ . To remedy the aforementioned situation, an iterative boundary adjustment algorithm is proposed to fine-tune the segmentation boundaries to nearby local maxima in terms of the dissimilarity between adjacent segments.

The criteria to refine boundaries is that the distance between two adjacent segments should be the farthest at the refined boundary points. Based on the criteria, the proposed algorithm adopts the method proposed in [74] where the distance between two segments are defined as the distance between the empirical distributions of the two segments. The distance criteria boils down to calculating the Kullback-Leibler divergence between the two segments, where the two segments are each modeled by a Multinomial distribution. Since the effect of changing one boundary point propagates to other adjacent segments of neighboring boundaries, an iterative algorithm is devised as shown in algorithm 11.

---

**Algorithm 11** Iterative Boundary Adjustment

---

**Require:** Boundary points  $B$  (without beginning and ending frame), features  $X$ , window size  $W$ , iteration limit  $N$  and adjustment cost  $C$ .

```

1:  $n \leftarrow 0$ 
2: while True do
3:    $c \leftarrow 0$ 
4:    $B' \leftarrow B$ 
5:   Randomly permute  $B'$ 
6:   for  $b \in B'$  do
7:      $\kappa \leftarrow$  K-L divergence of the two segments in  $X$  adjacent to  $b$ 
8:      $b' \leftarrow b$ 
9:     for  $t \in \{b - W : b + W\}$  do
10:       $\kappa' \leftarrow$  K-L divergence of the two segments in  $X$  adjacent to  $t$ 
11:      if  $\kappa' > \kappa$  then
12:         $\kappa \leftarrow \kappa'$ 
13:         $b' \leftarrow t$ 
14:      end if
15:    end for
16:     $b \leftarrow b'$ 
17:     $c += \text{abs}(b - b')$ 
18:  end for
19:   $B \leftarrow B'$ 
20:   $n += 1$ 
21:  if  $c \leq C || n \geq N$  then
22:    break
23:  end if
24: end while
25: return  $B$ 

```

---

**Table 3.2:**  $F$ ,  $P$  and  $R$  represent  $F_1$ -score, precision and recall respectively. Underscores of 0.5 and 3 represent 0.5 and 3 seconds window hit rate scores. *pair* stands for frame clustering.  $S_o$ ,  $S_u$  and  $S_f$  are the normalized conditional entropies of over-, under-segmentation and their  $F_1$ -scores. Results of SF and OLDA are from [70, 61]. Results of other algorithms are from [69]. Algorithms followed by (\*) are the ones with boundary adjustment algorithm. Parenthesis refers to the feature used for that algorithm. Numbers in bold are the highest  $F_1$ -score for each metric.

Algorithm	Boundaries						Segmentations					
	$F_{0.5}$	$P_{0.5}$	$R_{0.5}$	$F_3$	$P_3$	$R_3$	$F_{pair}$	$P_{pair}$	$R_{pair}$	$S_f$	$S_o$	$S_u$
SF (Chroma)[61]	—	—	—	<b>77.4</b>	75.3	81.6	<b>71.1</b>	78.7	68.1	—	—	—
VMO+SF (Chroma)	36.29	33.84	40.81	69.02	64.27	77.7	61.22	69.99	58.59	<b>67.38</b>	64.59	73.25
VMO+SF* (Chroma)	37.37	35.08	41.94	61.5	57.74	68.94	56.16	63.24	54.4	62.81	60.99	67.5
VMO+SC (CQT+MFCC)	34.34	29.38	43.52	64.46	55.09	81.64	55.9	68.63	49.87	62.50	57.59	70.54
VMO+SC* (CQT+MFCC)	<b>38.41</b>	34.28	45.47	60.98	54.29	72.26	52.84	61.08	49.05	60.02	55.87	64.84
VMO+SC (Chroma)	31.87	26.39	42.18	61.98	51.2	82.2	52.81	64.57	47.25	59.56	54.93	67.23
VMO+SC* (Chroma)	33.80	28.88	42.07	60.83	52.06	75.45	49.98	57.54	46.40	56.9	53.04	61.37
SC[49] (CQT+MFCC)	31.9	26.03	45.39	57.46	46.95	81.05	54	65.16	48.93	59.56	55.05	67.41
C-NMF[69] (Chroma)	24.89	24.52	26.41	60.41	59.84	63.45	53.53	58.29	52.65	57.2	55.85	60.63
OLDA[70] (Multi-feature)	29.6	29.7	32	53.5	55.3	55	—	—	—	—	—	—
SI-PLCA[75] (Chroma)	28.27	39.57	22.74	50.12	70.59	39.97	49.36	42.67	65.17	48.08	62.28	42.67
CC[76] (Chroma)	25.06	27.3	23.86	55.06	60.17	52.16	49.18	62.91	41.06	56.5	50.36	66.5

Algorithm 11 resembles an expectation-maximization algorithm in the sense that each iteration (outer for-loop in algorithm 11) stochastically cycles through all boundaries and adjusts them to maximize the K-L divergence of adjacent segments, then fixes the adjusted boundaries as new boundaries and proceeds to the next iteration until convergence criteria are met. The stopping criteria are the total number of iteration  $N$  and the total length of boundaries moved  $C$ . Empirical observation of running algorithm 11 shows that the total length of boundary moved at each iteration,  $c$ , monotonically decreases with number of iterations  $i$ .

### 3.2.4 Music Structure Segmentation

The Beatles-ISO dataset has 179 annotated songs and is widely used in evaluating segmentation algorithms [76, 68, 75, 70, 61]. The segmentation experiment aims at identifying a segmentation of a given audio recording and compare the segmentation with human annotations.

#### 3.2.4.1 Experiments

To evaluate the effect of the VMO-SSM and the boundary adjustment algorithm, the proposed framework is evaluated against the Beatles-ISO dataset and compared to existing work on the same dataset. Three standard features and their combinations are considered in this

experiment; constant-Q transformed spectra (CQT), chroma and MFCCs. All audio recordings are down-sampled to 22050Hz, analyzed with a 93ms window and 23ms hop. CQT are calculated between frequency range  $[0, 2093]$ Hz with 84 bins. Chroma are derived from CQT by folding the 8 octaves into 12 bins. MFCCs are calculated from 128 Mel bands and 12 MFCCs are taken. All features are beat-synchronized using a beat-tracker [59] with median-aggregation. Similar to [49, 61], features are then stacked using time-delay embedding with one step of history and one step of future. Each dimension of each feature is normalized along the time axis. To combine different features, they are simply stacked and different dimensions are assumed to have equal importance.

A parameter sweep is done to find the best set of parameters in this experiment. Cosine distance is used in the *VMO* distance calculation. For SC, the median filtering window  $\omega$  is 17. The number of basis in SC (or the number of distinguished sections),  $m$ , is 5. For SF, the standard deviations for time-lag and time axis,  $\{s_L, s_T\}$ , are 0.5 and 12. The peak-picking window length  $\lambda$  is 9. The parameters for the boundary adjustment algorithm,  $W$ ,  $N$  and  $C$ , are  $\{4, 10, 2\}$  respectively.

### 3.2.4.2 Evaluation

The evaluation results of the proposed framework along with the ones from other existing work are shown in Table 3.2. The metrics used follow the ones proposed in Music Information Retrieval Evaluation eXchange (MIREX). The evaluation could be understood in two aspects. The first aspect is the performance on retrieving boundaries and the second one is the performance on assigning labels to regions defined by retrieved boundaries. For boundary hit rate, the combination of the *VMO*, SC and boundary adjustment outperforms all other existing work by a margin of at least 7% (in [61] it is not reported) for 0.5s window tolerance. For 3s window tolerance, despite being inferior to SF, the approaches using the *VMO*-SSM are still superior to other existing work. The boundary adjustment algorithm turns out introducing a trade-off between short- and long-time

tolerance boundary hit rate. For SC the trade-off of  $F_{0.5}$  and  $F_3$  is acceptable with  $F_{0.5}$  always improved slightly more than the degradation of  $F_3$ . It could be observed that it is not worthwhile applying the boundary adjustment algorithm on SF since the degradation of  $F_3$  is far more than the improvement on  $F_{0.5}$ . The discrepancy between applying the boundary adjustment algorithm on SC and SF could be understood by the nature of the segmentation algorithms, since SF focuses on finding boundaries from SSM more directly than the matrix decomposition approaches, there might be less room left for improving boundary accuracies in the post-processing stage for SF. For segmentations, original SF ranks the highest in pair-wise clustering F-score and the combination of the VMO and SF is the runner-up. For the F-score of normalized conditional entropy, the VMO-SF combination returns the highest score (it is not reported in [61]). For matrix decomposition approaches, replacing traditional SSMs with VMO-SSMs achieves comparable or superior performances than existing work in segment labeling evaluation.

### 3.2.5 Discussions

In this section, an alternative SSM extracted from the *VMO* is shown to be reliable replacing the traditional SSM in music segmentation tasks. In general, using the VMO-SSM improves boundary detection accuracy and achieves comparable or superior performances in segment labeling to state-of-the-art. The reason that the VMO-SSM is better in boundary detection for matrix decomposition approaches is that the selective mechanism during the construction of the *VMO* suffix structure discards unnecessary calculations, and in turn leads to a cleaner binary SSM than the one from traditional approach.

## 3.3 Music Analysis with VMO-HMM

To exemplify the use of VMO-HMM on musical applications, a case study on analyzing Jazz music harmonic progression is conducted. The piece being analyzed is “Now’s the time”

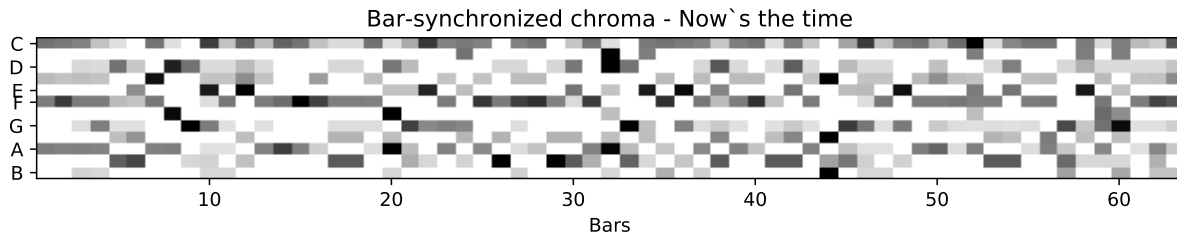


from Charlie Parker. The lead sheet in MusicXML and an accompaniment recording in midi are both available. To analyze the harmonic progression automatically, the MIDI accompaniment is used as the input to a *VMO*. The lead sheet comes with the melody and human annotated chord labels, and serves as the reference to the *VMO* analysis.

To obtain harmonic information from a MIDI file, the MIDI note events are first quantized to a piano roll matrix with dimension  $\{128, B\}$ , where  $B$  stands for the number of bars and the first dimension for MIDI pitch values. The values in the piano roll matrix are velocities ranging between  $[0, 127]$  with 0 representing a none event. There are several choices for the note velocity aggregation (pooling) within a bar, such as max, mean or median. For this case study, max-pooling is used to aggregate the note velocities within each bar along the time axis. Since the time signature and tempo are given in the MIDI file, the bar locations could easily be determined. After extracting the piano roll matrix, it is further folded over octaves to form a chroma-like matrix (midi-chromagram) with dimensions  $\{12, B\}$ , with the first dimension represents the pitch class  $\{C, Db, D, Eb, E, F, Gb, G, Ab, A, Bb, B\}$ . A normalization along the time axis is done to normalize the values of each pitch class to be between  $[0, 1]$ . A visualization of the final midi-chromagram matrix is depicted in figure 3.11.

To further strengthen the harmonic modeling, the 12-dimension data points from the midi-chromagram are projected onto the tonnetz space [77] during the construction of a *VMO* with the midi-chromagram. The  $L_2$  norm distance is used during the construction. After indexing the midi-chromagram with a *VMO*, the clusters could be retrieved by grouping the frames in  $\Sigma$  having the same label. The clusters of chroma frames for each clusters formed by the *VMO* could be visualized as in figure 3.12.

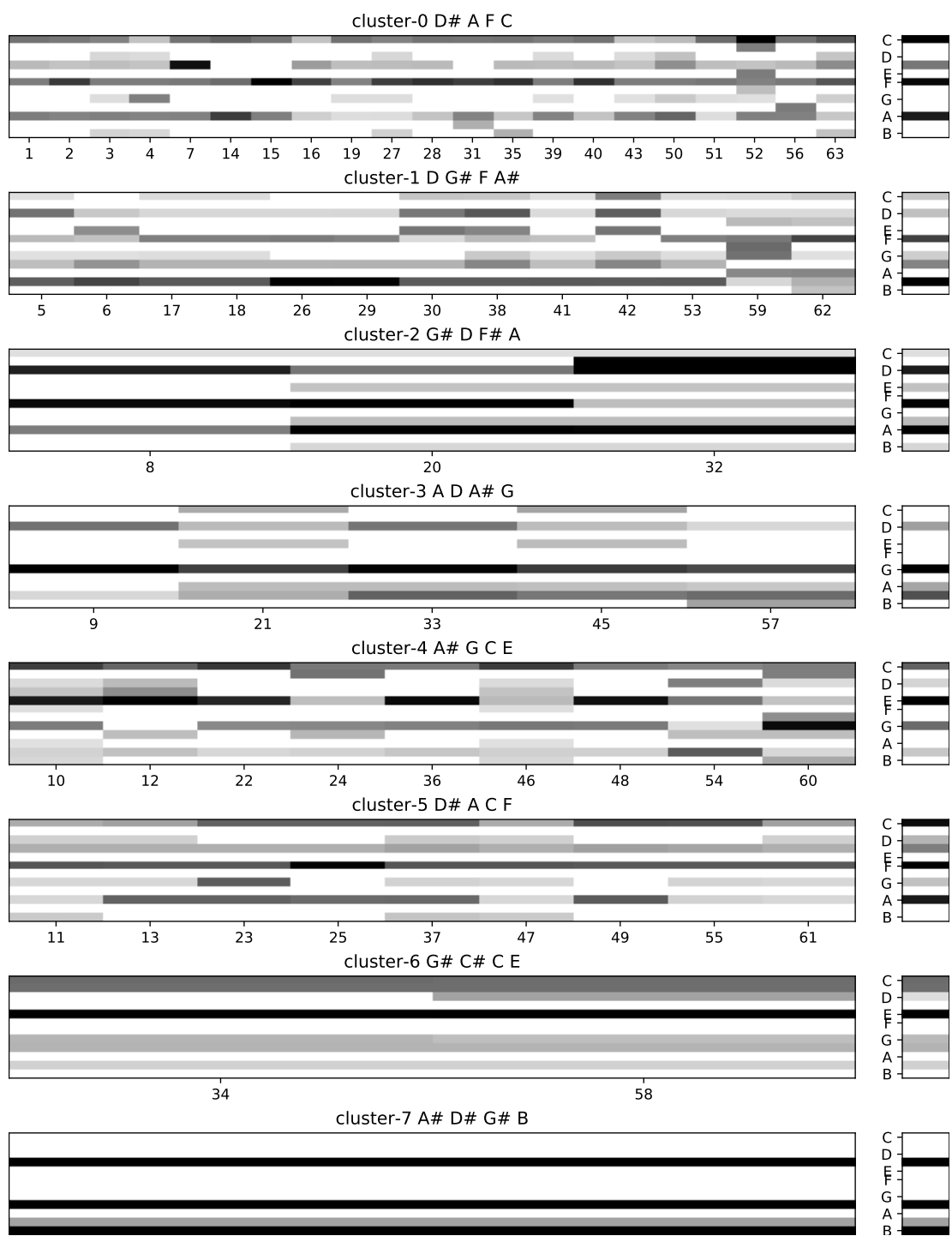
Parts of the reference lead sheet with chord labels are shown in figure 3.13. To examine how well a *VMO*-HMM capture the relationships between observations and latent variables. A qualitative comparison between the reference chord labels and the clusters from the *VMO*-HMM shows that the discovered clusters from the *VMO*-HMM do capture harmonic meanings and



**Figure 3.11:** The midi-chromagram extracted from the MIDI accompaniment recording of the Jazz piece “Now’s the time”.

provide more information than the reference chord labels from the lead sheet. Comparing the centroid chroma (right column) in figure 3.12 with the score in figure 3.13, cluster-0 and cluster-5 match with chord labels F and F7, cluster-1 matches with Bb7, cluster-2 with Am/D7, cluster-3 with Gm, cluster-4 with C7 and cluster-6 with C7 (b9). Since cluster-7 only has only 1 frame in it and paragraph considerations, its discussion will be skipped. The unsupervised discovery with the VMO-HMM is nearly perfect to the reference chord labels, which shows that the VMO-HMM is capable of capturing the groupings of midi-chroma frames into harmonically meaningful clusters. Furthermore, an interesting split of the chord label [F, F7] into cluster-0 and cluster-5 shows that the VMO-HMM does capture a level deeper than what the surface chord labels suggest. By examining where cluster-0 and cluster-5 are located in the score, the F7s associated with cluster-0 are undeniably the tonic with a flat 7 for which Major or Mixolydian scales could be suitable. On the other hand, the F7 associated with cluster-5 are passing chords between C7. Like the one in measure 11. Though it is marked as F7 in the lead sheet, in the accompaniment MIDI file it is actually played as F13 (#11), in which a Lydian Dominant scale could be used.

Finally, the Markov transition matrices obtained from algorithm 9 of different orders are shown in figure 3.14. The  $j$ th entry in the  $i$ th row in each matrix represents the probability from cluster- $i$  transition to cluster- $j$ . Matching the clusters to the chord labels verifies that this piece follows a tight [ii, V, I] progression of the chord sequence [D7, Gm, C7] and [Gm, C7, F7]. By examining these Markov transition matrices, it could be observed that the higher the order, the



**Figure 3.12:** The clusters formed by the *VMO* on “Now’s the time”. The left matrix of each row is the collection of frames from the mid-chromagram, the right single vector is the median centroid obtained by median-pooling along the time axis within that cluster. The four pitches for each cluster represent the four most dominant pitches within that cluster by velocity.

# Now's The Time

M1~M25

$\text{♩} = 132$   
F7

6 Bb7 F7 D7 Gm

10 C7 F7 C7 F7

14 F F7 Bb7 Bb7

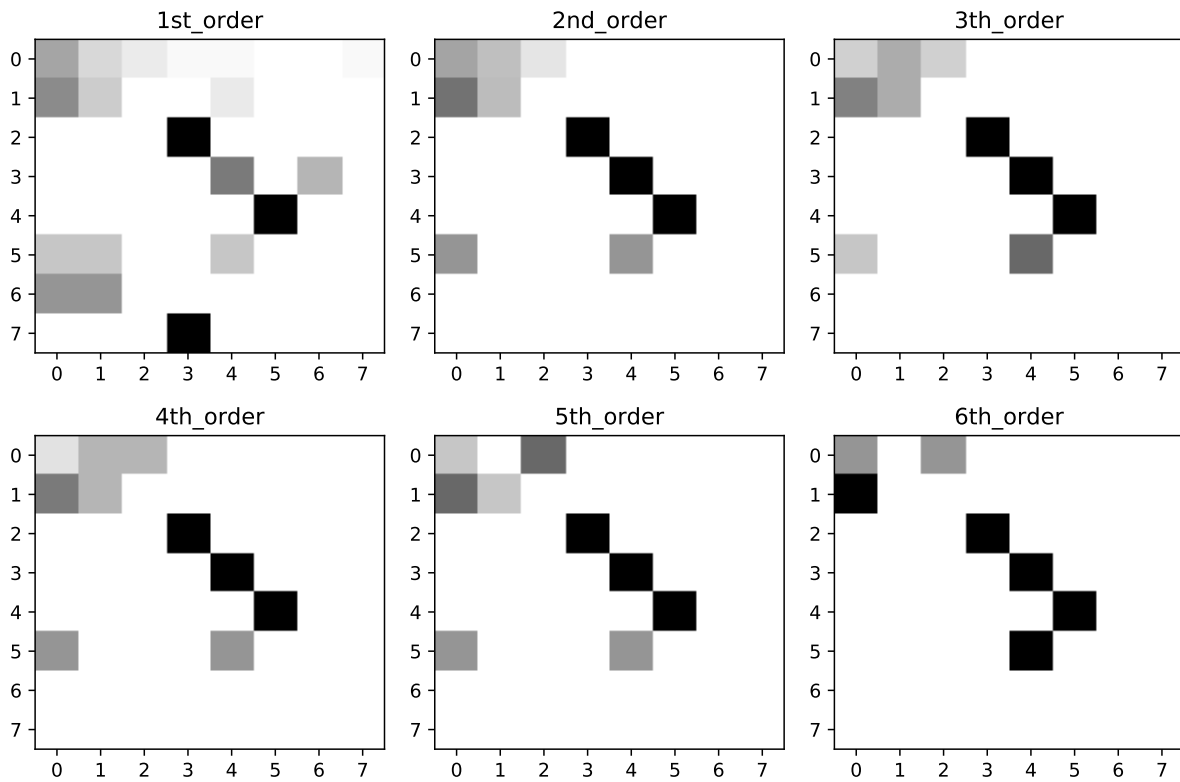
19 F7 Am D7 Gm

22 C7 F7 C7 F7

**Figure 3.13:** Partial score from the lead sheet of “Now’s the time” from bar 1 to 25. The annotated chord labels are on top of each bar.

sparser the Markov transition matrix is. As the transition matrix gets sparser, the more definite the transition probabilities and fewer choices for possible next states. The musical interpretation combining the observations of the transition matrices and the clusters is that lower order Markov transition matrices capture freer and more jazzy relationships between chord clusters while higher order ones capture stricter harmonic tonal function relationships between chord clusters. For example, in the 1st-order transition matrix, cluster-0, which matches to  $[F, F7]$ , could transition to any of the other clusters except for cluster-5, whose chroma content is similar to cluster-0. But as the order gets higher, the possible clusters that cluster-0 could transition to becomes fewer, and converges to cluster-1 and cluster-2, which matches to  $Bb7$  and  $Am/D7$  respectively. The converged transitions between  $F7$  to  $Bb7$  and  $Am/D7$  are standard  $[I, IV]$  or  $[I, vi]$  movements in tonal theory.

Chapter 3 is adapted from published materials in *"Pattern Discovery from Audio Recordings by Variable Markov Oracle: A Music Information Dynamics Approach"*. Wang, Cheng-i. & Dubnov, Shlomo. Acoustics, Speech, and Signal Processing (ICASSP), 2015 IEEE International Conference on, 2015, *"Music Pattern Discovery with Variable Markov Oracle: A Unified Approach to Symbolic and Audio Representations"*. Wang, Cheng-i.; Hsu, Jennifer. & Dubnov, Shlomo. International Society for Music Information Retrieval Conference, 2015, 176-182, *"Structural Segmentation with the Variable Markov Oracle and Boundary Adjustment"*. Wang, Cheng-i. & Mysore, Gautham. J. Proceedings of ICASSP 2016, Shanghai., IEEE, 2016 and *"Context-Aware Hidden Markov Models of Jazz Music with Variable Markov Oracle"*. Wang, Cheng-i. & Dubnov, Shlomo. 5th International Workshop on Musical Metacreation (MUME 2017) at the Eight International Conference on Computational Creativity, ICC3 2017, 2017.



**Figure 3.14:** Different Markov transition matrices extracted from the VMO-HMM, from lower to higher order. The entry  $[i, j]$  in each matrix represents the probability from latent variable  $i$  to  $j$ .

# Chapter 4

## Generate with VMO

### 4.1 Machine Improvisation

Using computers to generate musical content has a long history and has existed since the introduction of computational technologies [78, 79]. Recently, interactive and on-line music systems [80] have been the focus of this field. Previous research using the *Factor Oracle (FO)* [25, 26] for symbolic music generation and the *Audio Oracle (AO)* [24, 27] for audio content generation will be extended for machine music synthesis and improvisation in this chapter. Machine Improvisation refers to artificially augmented musical performances where machines contribute creatively to the musical outcome. With the use of machine learning and algorithmic techniques, programming compositional algorithms has been largely substituted by programming machines that extract patterns or rules from musical examples. Furthermore, it is now possible to rehearse or perform with machines that improvise in the style of live musicians. These machines capture musical style by using either live musical content or pre-recorded examples.

Currently, objective criteria to evaluate the quality of music produced by one independent artificial agent does not exist, as elaborated in [81, 82]. Thus, for such an agent, the qualitative goal proposed in [83, 25] to model the style from the input examples is followed. The goal

behind imitating the style of musical examples is to recreate the sense of creativity and virtuosity embedded within the style of musical examples. This creativity and virtuosity can hopefully be approximated by the musical outcome that shares a similar style with the musical examples. To capture the style of musical input, the sequential dependencies between musical surface events, such as pitch class profiles, melody, beat tatum, etc, are modeled. This allows the system to generate musical surface events that conform to the sequential dependencies from the input examples. The aforementioned intuition is supported by the experiment reported in [84] where it is shown that complex sources can be approximated by Markov models with limited orders. The argument of approximating a generative source with limited-order Markov models conforms with the argument made in [85]. In [85], the use of a dictionary-based approach to parse, store and model a musical sequence is proposed. The dictionary-based approach is proposed to solve the conflicting requirements of having arbitrarily long context (large order Markov model) for prediction and having enough samples for reliable conditional probability estimates. As a result, the Incremental Parsing algorithm used in Lempel and Ziv compression algorithm [36] is used to construct the dictionary-based model and leads to a variable-Markov model (since the context stored in the dictionary has different lengths) that echoes to the argument made in [84]. From the perspective of the dictionary-based approach, the *FO* also parses a sequence by identifying the longest repeated suffixes at each time instance and resembles the Incremental Parsing.

Previous work in machine improvisation has focussed on symbolic representations [25, 86, 87] where the granularity and expressivity of the musical outcomes are limited. Thus it is desirable to have a system that is capable of processing both symbolic and audio representations. The *FO* was originally introduced for fast pattern matching in bio-informatics [38], but it was shown in [25, 86, 87] that an *FO* could be used for symbolic machine music improvisation by imitating a musician's style. An *AO* is the signal extension of an *FO* that enables machine improvisation with an audio signal. Although the *AO* structure laid the foundations for audio machine improvisation, it lacks certain functionalities that the *FO* structure provides on symbolic



sequences.

Systems that implement stylistic machine improvisation through the *FO* and *AO* structures capture the style of the musical input but improvise with little relation to the musician. A machine listening component to the machine improvisation framework is added in this chapter. As a first step towards guided machine improvisation, a framework for guided musical synthesis in [88] based on the *Variable Markov Oracle (VMO)* is proposed. The *FO* and *AO* structures were combined to create the *VMO* data structure for off-line guided music synthesis using a query-matching algorithm, algorithm 6. The query-matching algorithm extends the single frame query function in the *AO* structure [27] to a sequence query-matching function. The sequence query-matching algorithm based on a *VMO* could also be viewed as an extension of the *Guidage* system proposed in [23] where both forward and suffix links are used instead of only using forward links. A *VMO* achieves this query-matching functionality by explicitly exploring how feature frames in a multivariate time series are clustered together by the oracle structure [45] as described in chapter 2. Using a *VMO* with the query-matching algorithm allows us to address rhythmic and harmonic issues between the improvised output and the musical input in a machine improvisation scenario. On the rhythmic side, a rhythmic beat phase feature that allows a live musician to query the machine for an improvised output that is synchronized in beat phase is defined. For harmonic relations between the improvised output and input music, a *VMO* allows for longer queries (as opposed to the single-frame query approach of an *AO*) that can improve the improvisation output.

Given a constructed *VMO*, a machine improvisation scheme without interactions with other musical agents is elaborated in [24, 27]. Given a starting state, a pointer navigates the oracle structure by suffix links to “reshuffle” the audio stream, creating novel audio sequences from the original one. Suffix links are used as jumping points during the navigation. By using suffix links as jumping points, the smoothness of the reshuffled audio signal is assured since suffix links point to the occurrences of the longest repeated suffixes of the current point. The basic algorithm

for machine improvisation using a *VMO* is provided in algorithm 12. Other heuristics of how to navigate the oracle to improve the quality of the improvised outcome could be found in [26].

---

**Algorithm 12** *VMO* Incremental Improvisation

---

**Require:** A *VMO*  $Q = q_1, q_2, \dots, q_T$ , current navigating index  $i$

- 1: Retrieve all the states following the states having the label  $q_i$ , store the retrieved states in a list  $L$
  - 2: **if**  $L$  is empty **then**
  - 3:      $i_n \leftarrow i + 1$
  - 4: **else**
  - 5:      $i_n \leftarrow$  Choose randomly from  $L$  either uniformly or by some specified weights
  - 6: **end if**
  - 7: **return** The next index to navigate to,  $i_n$
- 

## 4.2 Query-guided Improvisation

In previous experience with machine improvisation, scenarios are encountered where interactions between the machine and the human input are desired; i.e. the machine improvises along with another human musician. Such machine improvisation is referred to as guided improvisation. Guided or control improvisation refers to a framework that the improvisation from the artificial musical agent is regularized by, rather than interactive with, the input from the human musician, predefined rules or references. Under such frameworks, the problem could then be divided into two well-defined blocks. The first is the improvisation part which is usually cast as a sampling process or random walk given some stochastic process describing the music generating process. The second is the regularization part that could be implemented as cost functions, constraints or heuristics solving tractable optimization problems.

The simplest case of improvisation control and interaction appear in what are termed as the “cadence” problem, which is how to make the machine improvisation reach an ending point together with the human musician automatically. A simple, single chroma query to lead an *AO* to a certain tonal area where it matches the soloist so the *AO* could be stopped together with

the musician in a situation where both are chromatically consistent with respect to each other is proposed in [27]. In these applications, the musician would use a single long note as a query to a system that would emphasize regions in the oracle that have a related chroma.

Another generalized version of the “cadence” situation is the “queueing” problem, i.e. how to trigger changes in the selection of material that the oracle structure uses in a way that corresponds to the musical input. In previous experiments, it was not possible to distinguish among different types of musical material simply based on a single chroma or note. Moreover, changing the note or replacing the query in time creates a “moving target” situation that could fall into a desired trajectory, but without any guarantees or algorithmically efficient solutions. Therefore, one would want to allow the system to listen to more than a single note and be able to switch between regions or alternative oracles based on longer queries. In this case, the sequential nature within the query is taken into account. Intuitively, to make regions distinct, a longer context than a single note query is needed, especially if one would like to consider adding transformations, such as transposition or pitch shift in the future. In this way, identification of melodic phrases or even timbral “gestures” can be used to switch regions more generally.

Operation of the guided improvisation can also be understood in terms of ensembles and larger compositional relations between humans and artificial partners. In an “accompaniment” scenario, the oracle is trained on a selection of musical material that supports a live solo or lead musical instrument. One can think of it as a “duet” situation, where a solo input is driving an intelligent and creative playback of an accompaniment piano or even a complete Jazz ensemble. The role of the system then, would be instantaneously harmonizing, providing a bass line and even rhythm to a melodic or polyphonic lead query.

A different configuration and musical use of the proposed system is for constraining a machine generated solo for a given song or standard. In such a case, a recording of the backing ensemble is provided as a query into an oracle trained on an unrelated solo (from a different source). The purpose of the query is to navigate an unrelated solo in a way where the solo licks

would be extracted to match the song composition. In this case the query is used to create the solo, while the harmonic and rhythmic grids are fixed in the query track.

In short, as a next step to address the aforementioned “cadence” and “queueing” problem from the previous results in *AO* structures, the revised oracle structure, the *VMO*, is introduced to allow for querying with a sequence instead of a single frame. Although the examples presented in this chapter are created in an off-line fashion, the algorithms are capable of on-line deployment.

### 4.2.1 Query-matching Algorithm for Generation

Algorithm 6 or 7 and 8 are used to create new music signals by using a query music signal guiding a path traversing the target music signal. The signals could be either audio or MIDI representations. The returned path matches the target and query music signals in the feature space extracted from the signals. The target music signal in the guided synthesis/improvisation applications is treated as the material for concatenative synthesis, and the query music signal provides the map that illustrates how the materials should be recombined. For the examples described hereafter, the steps are as follows:

1. Both the target and query music signals are converted from their original representations to appropriate feature representations.
2. The target feature time series is indexed by an *VMO* as  $O$ , and the query feature time series is used as query input,  $R$ , for algorithm 6 or 7 and 8 to retrieve the recombination path  $P[\text{argmin}(C)]$ .
3. The recombination path  $P[\text{argmin}(C)]$  is used to index the target music signal to generate new music content.

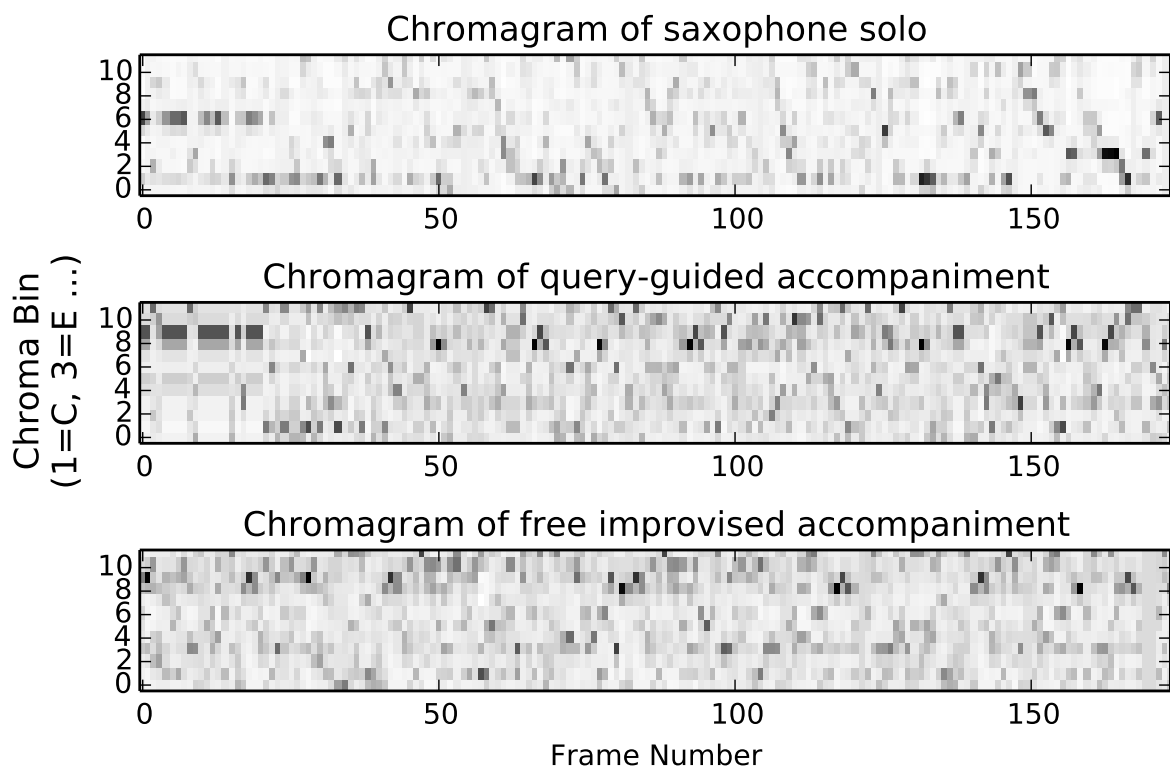
## 4.2.2 Examples - Guiding with Tonal Content

In this section, how to guide target music with tonal content from query music is described. Approaches for audio and MIDI are both described here. For audio representations, chroma is chosen to be the feature used to represent the tonal content of a music signal. The chroma is calculated using a window length of  $N$  with  $\frac{N}{2}$  overlap, frequency analysis ranging between  $f_{min} = 63.54Hz$  to  $f_{max} = 22050Hz$  and 12 bins per octave. The recombination path  $P[\text{argmin}(C)]$  is used to index the target music signal. These indices are used to synthesize a new music signal with the overlap-add method using a hamming window of length  $N$ , and an overlap of length  $\frac{N}{2}$ . In our settings,  $N$  is set empirically to  $2^{15} = 32768$  from listening evaluations made by the author. The music signals for the examples are obtained from free shared music recordings in the style of Jazz. Lead tenor saxophone recordings are cut into segments of  $50 \sim 100$  seconds long and are used as query music signals,  $R$ . Full band recordings (drums, bass, electric guitar and piano) are  $20 \sim 40$  seconds long and used as  $O$  to construct  $VMOs$ . Although in theory, the query-matching and improvisation algorithms could be applied to all genres and features, the combination of Jazz and chroma is chosen because so far, the query-matching algorithm still works based on frame level matching between the query and the target music signal. The relatively wider tolerance for rapidly changing harmonic or rhythmic structures of Jazz music allows for a natural progression even if the musical structure is broken on a larger scale (such as chord progression or sectional changes) due to the recombination of audio frames from the original audio. The audio examples can be found online <sup>1</sup>.

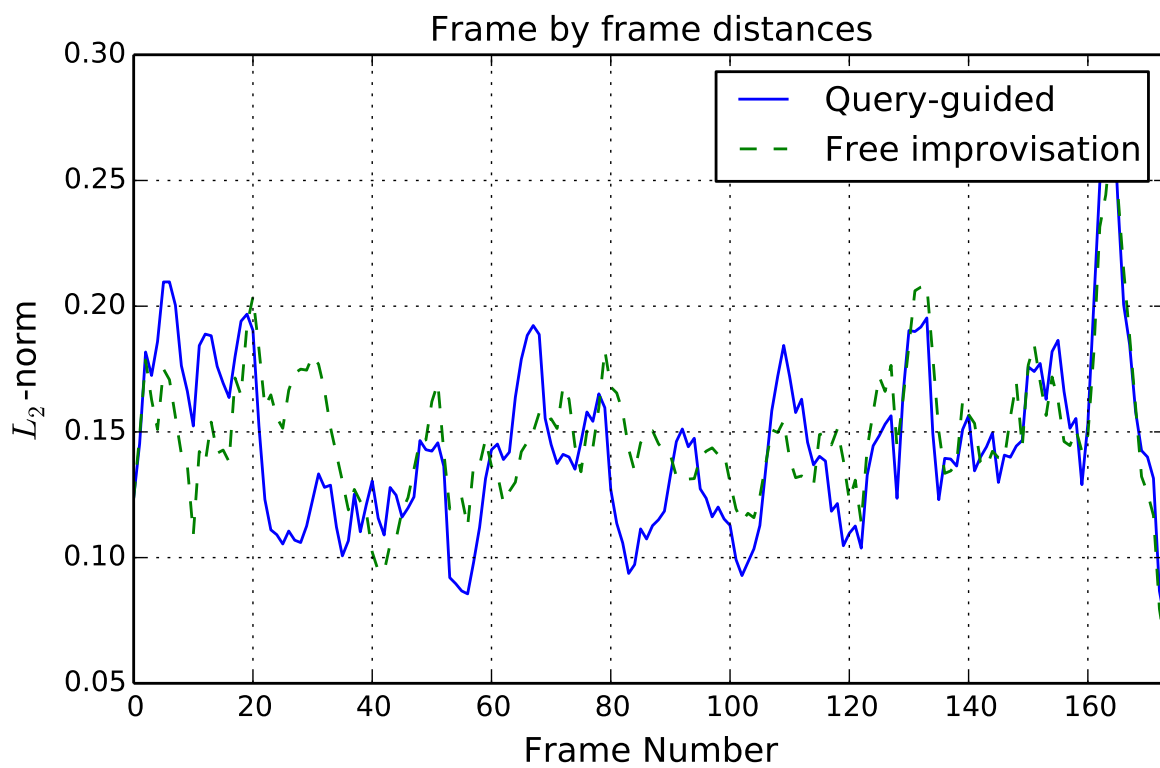
To qualitatively evaluate the query-guided synthesis from audio representations, a free (unguided) improvised accompaniment using the  $AO$  mentioned in [27] is synthesized as well to compare with the query-guided version. Example I is shown here and depicted in figure 4.1. To aid the evaluation with quantitative measures, frame-by-frame distances between the chromagrams

---

<sup>1</sup><https://soundcloud.com/pyoracle/sets/guided-free-improvisation-with-markov-oracle>



**Figure 4.1:** Example I. (Upper) Chromagram of the saxophone lead. (Middle) Chromagram of the query-guided accompaniment. (Bottom) Chromagram of the free improvised accompaniment.

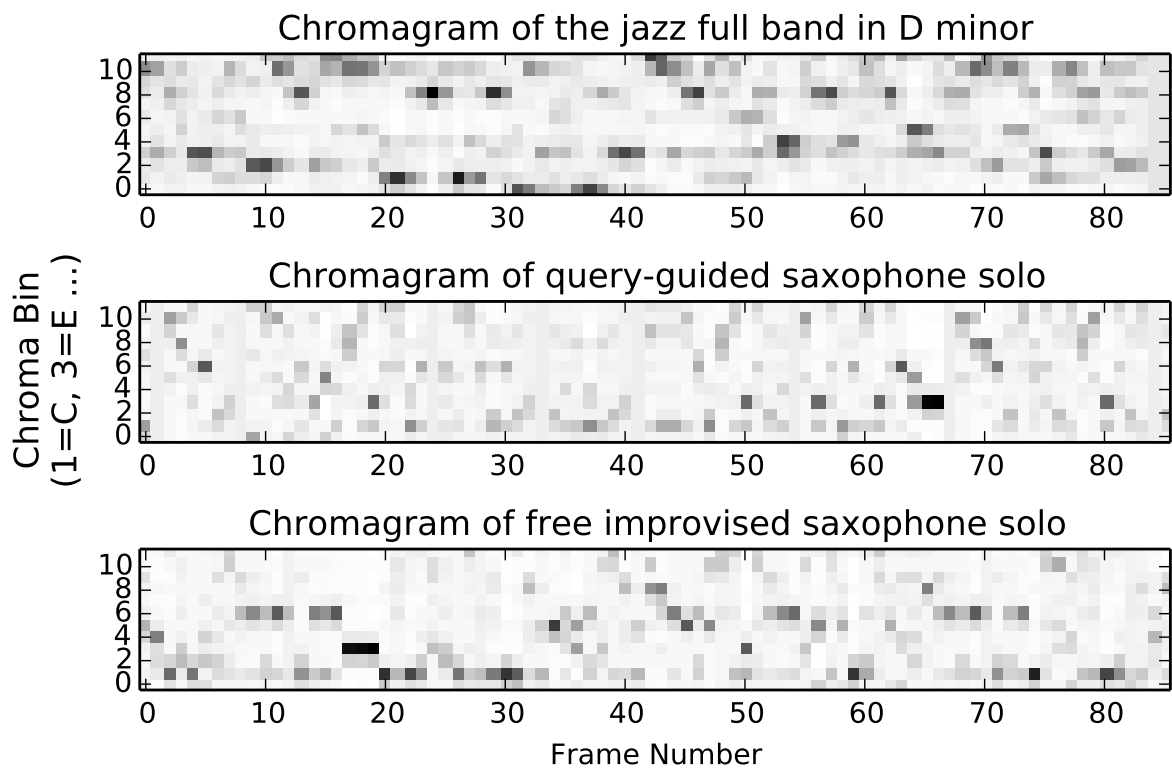


**Figure 4.2:** Frame-by-frame  $L_2$ -norm between chromagrams of lead and accompaniment for example I. The  $\{\mu, \sigma\}$  of  $L_2$ -norm error for guided-synthesis and free improvisation are  $\{0.13, 0.07\}$  and  $\{0.15, 0.06\}$  respectively.

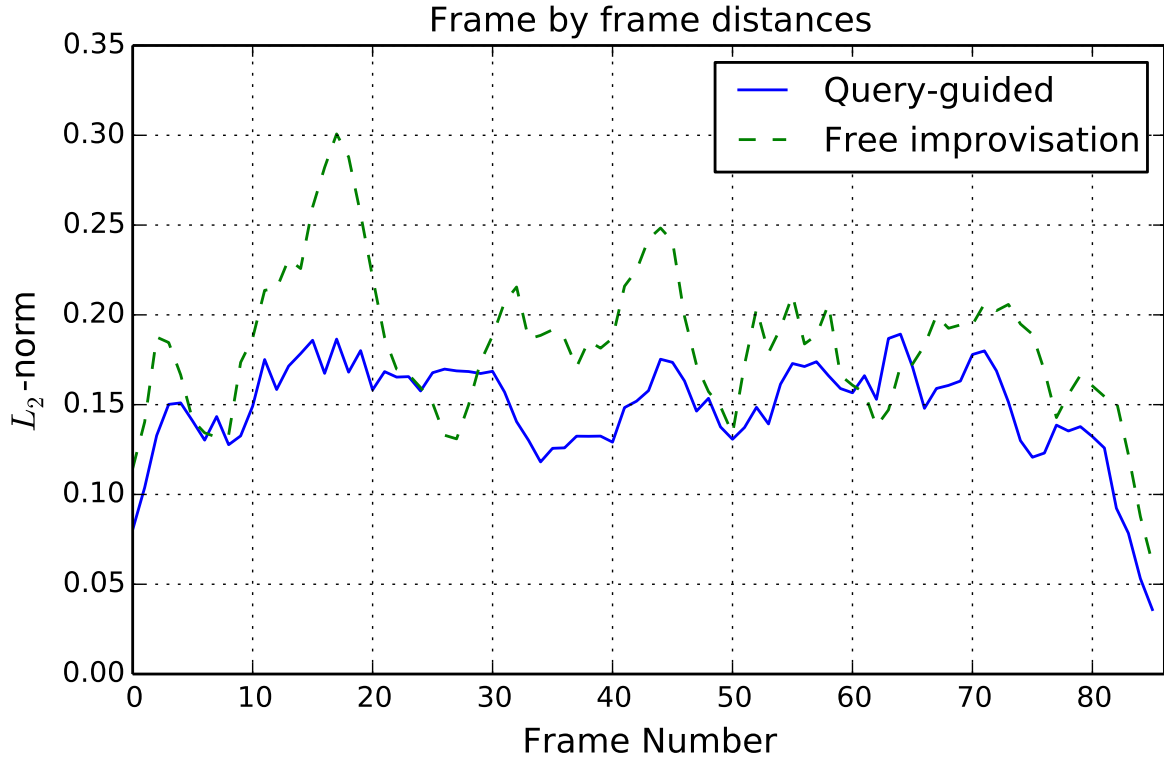
of query music signal and the two synthesized accompaniments are plotted in figure 4.2. From figure 4.2, it is observed that the errors could potentially be correlated. After examination of the parameters used, we conclude that the possible correlation is due to the fact that the query signal used in this case might be too far from the target signal in the feature space, thus leading to rather indistinguishable synthesis between the query-guided and free improvised version from the signal point of view. But in figure 4.1, it could also be observed that the query-guided accompaniment did have a more similar structure to the lead than the free improvised one. The phenomenon is clearer at the beginning of the signal. Aurally, when listening to the mix of the lead and the synthesized accompaniments, one can observe that the free improvised accompaniment has a better sense of continuation in terms of how sub-clips of the original signals are recombined. In contrast, the query-guided accompaniment sounds more “broken” due to the use of shorter sub-clips to match the query. Nevertheless, in the case of the free improvised accompaniment, the original lead and the accompaniment sound separately from each other even if the tonality of the lead and the original accompaniment recording were chosen to be compatible with each other. In listening to the mix with query-guided accompaniments, the synchronization between the lead and the accompaniment in terms of harmonicity is more obvious. The sound examples could be found at the project page (<https://soundcloud.com/pyoracle/sets/guided-free-improvisation-with-markov-oracle>). For the examples, the left channel is always the query, which stays unchanged, and the right channel is the new generated audio either by query-guided (*VMO*) or free-improvisation (*AO*) approach.

After the experiment of using a lead saxophone as the query and the accompaniment recording in *VMO* as target, another experiment, where the roles of the lead and the accompaniment are switched, is performed. Example II is shown here in figure 4.3. The  $L_2$ -norm errors for example II are depicted in figure 4.4. The sound examples for example II are also accessible by the project webpage mentioned above. For example II, similar conclusions to example I based on corresponding plots could be drawn. But with example II, the  $L_2$ -norm error plot in figure 4.4





**Figure 4.3:** Example II. (Upper) Chromagram of the saxophone accompaniment. (Middle) Chromagram of the query-guided lead. (Bottom) Chromagram of the free improvised lead.



**Figure 4.4:** Frame-by-frame  $L_2$ -norm between chromagrams of lead and accompaniment for Example II. The  $\{\mu, \sigma\}$  of  $L_2$ -norm error for guided-synthesis and free improvisation are  $\{0.14, 0.05\}$  and  $\{0.18, 0.06\}$  respectively.

reflects the fact that the query-matching algorithm is able to find the path that minimizes frame-by-frame distances between the query and the target with context constraints. By listening to example II, one can find it clearer that the synthesized lead guided by the accompaniment conforms to the harmonic changes and dominant melodic lines of the accompaniment. Especially around 25 seconds in the result, as the accompaniment made a transition in terms of chord progression used to inform the end of the accompaniment, the guided lead saxophone turned to materials and reshuffling that not been used before to follow the change of chord progression. In general, although aesthetically there is no clear and obvious evidence that using query-guided approach is superior than the previous approach of using unguided machine improvisation, it is still asserted that the materials generated using query-guided approach sound musically meaningful and per-

form a better job conforming to the query used than the ones generated with free improvisation approach. Extra examples of both cases are accessible via the project webpage. One drawback of the concatenative synthesis approach that the guided synthesis has is the artificial layered sounds of the solo saxophone caused by the cross-fade between the overlap and add audio waveforms. A possible remedy is to weight the distance between the current frame and its immediate following frame lower than jumping to frames connected by suffix links. The lowered weights on the next frame will encourage continuous than jumping thus result in the improvised sound having less interruptions due to the jumps. The tradeoff of such remedy is by gaining continuity, the match to target timeseries is sacrificed.

For symbolic query-matching, a MIDI solo is generated by guidance from a MIDI accompaniment. In this case, the accompaniment MIDI is the query while the solo MIDI is the target. The MIDI file is first quantized to sixteenth notes. Next, the pitch classes are extracted from each MIDI file using a frame-by-frame, overlap-add method. For each frame, the notes are transposed to a single octave of 12 notes, and the velocities of each pitch class are summed. The velocities for each pitch class are also summed for any overlapping frames. The analysis frame size and hop sizes used here are 8 and 1 sixteenth notes long, respectively. The final MIDI pitch class feature is a 12-dimensional vector for each time frame similar to the chroma features from audio signals.

After inputting the two MIDI pitch class features into the query-matching algorithm, the recombination path  $P[\text{argmin}(C)]$  is used to index the target MIDI pitch class feature. This allows one to synthesize a new MIDI file with the overlap-add method using a frame size and hop size that match those used in analysis. During this synthesis stage, the overlap-add procedure adds velocities for frames that are not sequential in the original MIDI file. Higher velocity values in the final synthesis indicate notes that are more important in the musical context. For that reason, we use a velocity threshold,  $v_{\text{thresh}}$ , to filter out pitches that have a lower influence on the final musical output. If a note's velocity is less than  $v_{\text{thresh}}$ , the note velocity is set to 0. If a note's

velocity is greater than 127 (the maximum MIDI velocity), then that note’s velocity is clipped to 127. A  $v_{\text{thresh}}$  value set to *maximum velocity of the original solo* – 10 created satisfying results.

For the examples presented here, MIDI files were retrieved from the multi-modal MIREX-like emotion dataset [89]. An example of the first few measures of a MIDI solo guided by a MIDI accompaniment is shown in figure 4.5. The MIDI accompaniment that guided the improvisation is shown on the bottom staff while the improvised solo is on the top staff. The solo MIDI and accompaniment MIDI belong to the same song (John Stewart’s “Daydream Believer”), and inputting the two parts of the same piece into the query-matching algorithm results in a new solo over the original accompaniment. The generated solo uses notes similar to those found in the accompaniment within the same analysis window, as this is a direct result of using the MIDI pitch class feature to guide the solo generation. To quantify how often the original solo and accompaniment combination occur in this query-guided generation, the original solo is compared to the recombination path  $P[\text{argmin}(C)]$  used to index the solo. An original solo and accompaniment combination repetition in the new generation means that a frame in the original solo is equal to a frame in the recombined solo. For this example, there are no repetitions in the newly generated solo.

For a MIDI accompaniment (target) guided by a MIDI solo (query), the MIDI pitch class feature, as described above, is used. During reconstruction, the role of the solo and accompaniment are switched as the recombination path  $P[\text{argmin}(C)]$  is used to index the MIDI accompaniment pitch class feature. For this application, the same MIDI song for both the solo and the accompaniment (Roberta Flack’s “Killing Me Softly”) is used. An example of the first few measures of an accompaniment guided by a MIDI solo is shown in figure 4.6. The solo used to guide the improvisation is in the top staff while the improvised accompaniment is shown in the bottom staff. Because the generated accompaniment is matched with the solo using the MIDI pitch classes, the generated accompaniment has many notes in common with the original solo. For an idea of how novel this generated example is, 0.044% of the original solo and accompaniment

The image shows a musical score for the first few measures of an improvised MIDI solo. The top staff is labeled 'solo (improv)' and contains a melodic line in G major, 4/4 time, starting with a quarter rest followed by a quarter note G, a half note A, and a quarter note B. The bottom staff is labeled 'acc (query)' and shows the original accompaniment, which consists of a steady eighth-note chordal pattern in the right hand and a simple bass line in the left hand.

**Figure 4.5:** First few measures of an improvised MIDI solo guided by a MIDI accompaniment using solo and accompaniment from John Stewart’s “Daydream Believer.” The improvised MIDI solo is on the top staff and the original MIDI accompaniment is shown on the bottom staff.

The image shows a musical score for the first few measures of an improvised MIDI accompaniment. The top staff is labeled 'solo (query)' and contains the original melodic line in F major, 4/4 time, starting with a quarter note F, a quarter note G, a quarter note A, and a quarter note B. The bottom staff is labeled 'acc (improv)' and shows the improvised accompaniment, which features a more complex chordal texture in the right hand and a melodic bass line in the left hand.

**Figure 4.6:** First few measures of an improvised MIDI accompaniment guided by a MIDI solo using solo and accompaniment from Roberta Flack’s “Killing Me Softly.” The top staff is the original MIDI solo while the bottom staff is the improvised MIDI accompaniment.

combination was repeated in this query-guided generation.

### 4.2.3 Examples - Guiding with Rhythmic Content

In the following musical examples, how to guide a tonal piece using a rhythmic piece is described. In the following examples, rhythmic pieces that consist of drum patterns strung together by various jazz drum loop patterns are used. The jazz beat was used to guide the improvisation of three different musical pieces with tonal content:

- Nujabes - “Flowers” (piano cover)

- The Heath Brothers - “Smilin’ Billy Suite Part 1”
- Cole Porter - “Night and Day”

The feature used for both the tonal and rhythmic content is an intermediate calculation in the beat tracking algorithm described in [90]. This feature is referred to as a “beat phase descriptor” as it captures the offset of the first beat from the start of an analysis frame. The feature is calculated given an onset detection function (ODF) and a beat period  $\tau$ , the reciprocal of tempo, in an overlap-add fashion. The ODF is passed through a comb filter consisting of impulses at various offsets from the start of an analysis frame. This comb filter contains all possible shifts of the beat phase for a given  $\tau$ . The resulting beat phase descriptor for a single analysis frame of ODF samples is a vector with length equal to  $\tau$ . In our examples, the ODF is calculated using the spectral flux, the frame rate for the ODF is 11.6 ms, and the comb filtering proceeds in a frame-by-frame analysis with 512 ODF samples and a hop size of 128 ODF samples. An issue that occurs with this feature is that different musical pieces will have different  $\tau$ , and so the feature vectors will be different sizes. The query-matching algorithm requires equal dimensions for features inputs, so only some of the values in the beat phase feature vector are retained. We keep the values at indices  $[\frac{\tau}{8}, \frac{2\tau}{8} \dots, \frac{7\tau}{8}, \tau]$ . The resulting beat phase allows one to infer the beat alignment within an ODF frame.

The beat phase feature for the tonal content is input as the target to the query-matching algorithm while the beat phase feature for the rhythmic piece is input as the query. The beat phase feature guides the tonal improvisation to match the beat phase of the rhythmic piece. Please refer to the online appendix<sup>2</sup> for examples for MIDI and rhythmic-guided improvisations.

---

<sup>2</sup><https://soundcloud.com/pyoracle/sets/mume-acm-special-edition>

## 4.2.4 Related Works

Guided improvisation in music is closely related to the problem of planning and control that includes improvisation capabilities. Control improvisation has also become an area of interest in other domains. Specifying control signals to guide a system into a desired behavior is common in robotics and other dynamic systems, where adding a randomized strategy from examples might add more flexibility to a system [91]. In the case of music, the need for the improvisation to conform to outside constraints, such as partnering with other musicians, is less strictly defined and does not have any critical safety specifications. However, undesired notes can still be quite troublesome, especially during live performances where machines are involved. In [91], monophonic symbolic Jazz melody is generated using *FO* [38, 25, 26] constrained by a specification in the form of another finite state automaton. The finite state constraint automaton penalizing inappropriate transitions during melody generation by encoding chord progression, note progression and beat tatum. *FO* is also the core of the Omax system (IRCAM) [86, 92] and the Mimi system (USC) [93]. Both systems focus on symbolic representations. In [94], an extension of the Omax system, *ImproteK*, is introduced where autonomous improvisations are learned from the human musician accompanied by pre-recorded or live materials. The Mimi system uses visual feedbacks to enable interactions between the human musician and artificial agent [95].

Another approach to this problem is to form the control improvisation problem in constraint satisfaction problem (CSP), as done in [41, 96]. In [41], Markov constraint is introduced and refers to the constraints placed in CSP to enforce Markov properties in the generated sequence from solving the CSP. In [96], meter constraint is introduced in the generation of symbolic melodies from CSP with Marko constraints learned from examples. Other related examples that use time-automata to provide flexible and even improvisatory performances in coordination with a plan are the interactive score projects, such as *Antescofo* [97] and *Virage* [98]. In such cases,

a formal score specification allows the system to modify its performance according to expressive inflection or to a musician, mostly limited to time changes within a tightly predetermined sequence of events. This restriction makes it difficult for the system to navigate the specification in a highly non-linear temporal fashion, i.e. allowing jumps and recombination of events in the way in which VMO is designed.

### 4.3 Structural Improvisation

In this section an alternative machine improvisation mechanism based on the VMO that is an attempt toward structured machine improvisation is proposed, where the artificial musical agent is capable of producing musical content that has both the style of musical surface events from the human input and a sense of structure. The style of musical surface events can be imitated by using the basic machine improvisation algorithm as in algorithm 12 with human input. Before continuing with the details of the proposed mechanism, the assumptions on the “sense of structure” will first be described.

Two common beliefs in music research are 1) that structured sound is what distinguishes music from noise and 2) one of the essential elements that gives rise to the sense of structure is the interaction between expectation and surprise throughout a music piece. From a Markovian point of view, expectation of a time series is closely related to repetition of subsequences. With the repeated suffixes captured by the *VMO*, the concept of surprise is injected in a music piece by randomly jumping between frames constrained by suffix links and introduce the concept of expectation by creating repetitions during the improvisation. To implement the aforementioned hypothesis, the improvisation system generates new musical content based on either the past history of the improvisation itself or the human input. The improvisation is capable of memorizing its own past with the introduction of another *VMO*, *S*, recording the improvisation on the fly. *S* is constructed with the music content generated by the original *VMO*, *P*. During the improvisation,



a switching mechanism is used so that the improvisation either favors itself or the human musical input. When the current improvised phrase has longer repeated sections than that of the human input, the improvisation favors itself,  $S$ , otherwise, the current improvised phrase will favor the human input. The improvisation will also favor the human input when there is a new symbol created in  $P$ . The switching algorithm is a heuristic to balance between introducing new symbols to  $S$  given  $P$  and letting  $S$  have either longer or more repeating sections. Introducing new symbols to  $S$  will increase the first term on the right hand side of equation (2.2) while letting  $S$  have longer or more repeating sections will decrease the second term on the right hand side of equation (2.2). This as a whole, will increase the IR of  $S$ , which is another indication of structuredness. The condition for adding new symbols in  $S$  given  $P$  is straightforward. For the condition toward letting  $S$  have either longer or more repeating sections, the heuristic is to compare the  $\text{lrs}$  of  $S$  and  $P$  at the current navigating index.  $\text{lrs}_P < \text{lrs}_S$  is an indication that the improvisation itself has longer repeating sections at the current index and that it is worthwhile to continue to improvise with itself to create longer phrases and repeating phrases. The algorithm for the alternative improvisation setting is in algorithm 13.

---

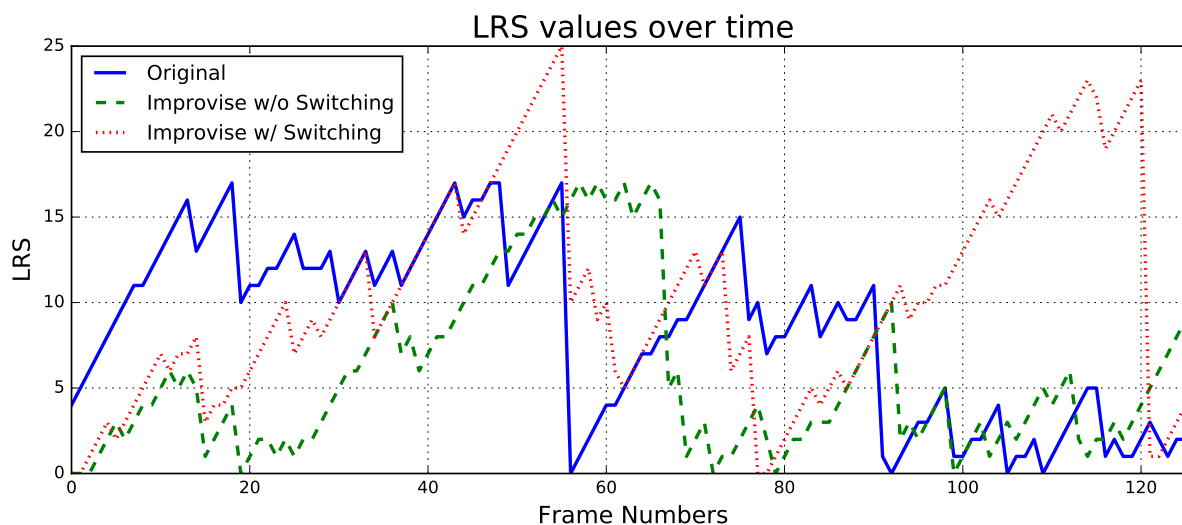
**Algorithm 13** Structured Improvisation

---

**Require:** Human music input

- 1: Create a VMO  $P = p_1 \dots p_T$  with the human music input
  - 2: Create a VMO  $S = s_1 \dots s_N$  with  $N < T$  with the improvisation itself
  - 3: **if**  $\text{lrs}_P[T] < \text{lrs}_S[N]$  and  $|\Sigma|$  is the same from previous update **then**
  - 4:     Improvise with  $S$  by algorithm 12
  - 5: **else**
  - 6:     Improvise with  $P$  by algorithm 12
  - 7: **end if**
- 

The heuristics of the switching mechanism is illustrated in figure 4.7. By comparing the  $\text{lrs}$  values for  $S$  with or without switching, one can observe that the improvisation with the switching mechanism has longer  $\text{lrs}$  values than the one without. Instrumental solos are used to experiment with the alternative machine improvisation setting proposed in this section. The resulting musical outcome does have longer repeated sections of itself, which means that the



**Figure 4.7:** An illustration of how the switching mechanism helps to increase the length of repeated sections of the improvisation. The blue solid line is the *lrs* value for human music input, the green dashed line is the improvised output without switching, and the red dotted line is the improvised output with switching.

results have a better sense of high-level structure than the musical output created by improvisations without the switching mechanism.

## 4.4 Improvisation with VMO-HMM

The use of a *VMO* for improvisation and synthesis is already introduced in [46]. The guided music generation was made possible by specifying a query to recombine the indexed audio or MIDI signal based on the *VMO* suffix structures. The limitation works introduced in previous sections are that the query and the target (*VMO*-indexed) signals have to use the same alphabet, or in other words, the same feature or type of signal. A framework analyzing symbolic music representation (not limited to MIDI) using *VMO*-HMM is proposed in this section, allowing the *VMO* to further expand its generative capabilities across different representations. The most important advancement of the following work is that it allows the user to specify a query signal that uses a different alphabet from the target signal. To be more specific, the user could now

specify a chord label sequence as input to the improvisation system. The system then translates the chord label sequence to pitch class profiles, which is essentially the same representation as the midi-chromagram used in section 4.2.2. Then the translated sequence is used as the query to the *VMO* to generate a new sequence in the same manner as proposed in [46] and previous sections. In the following sections, the proposed framework is applied on jazz music to show its capability.

#### 4.4.1 Jazz chord sequences

Musicologists have tried to capture the essence of jazz chord sequences by applying rules of Classical harmony to understand how basic harmonic structures have been transformed in a jazz composition. One common technique of chord substitution rule can be formalized as a “rewriting rule” which allows transforming a subsequence of chords into another subsequence of chords that introduces diversity, without, in principle, changing the harmonic function of the subsequence. Indeed, although jazz harmony could be considered born from Classical harmony in an evolutionary viewpoint, the harmonic functions of jazz chords seem to be much more complex than those in Classical four-part chorals, because of the underlying combinatorial “game” at play. For example, in classical theory the chord of C major and F# chord are the most “distant” in terms of their tonal context. In jazz however, a C(7) and F#(7) are closely related through sharing a common tri-tone axis, and may be considered interchangeable. Another common example of a distinction between Classical and jazz interpretation of chords is the functional role of C and C7. While in classical harmony a C7 is considered an unstable dominant chord that is expected to resolve to an F, in jazz, C and C7 are often considered equivalent. These examples indicate that the harmonic rules which make sense in classical harmony might not be strictly obeyed in other tonal or modal musical styles. In a VMO-HMM model, the relations between harmonic constructs, captured by the latent variables, depend on the previous note aggregation phase (the feature extraction part described in section 3.3) that is based on surface level note dynamics. The experiments in chapter 4 show that there are two main transition types

between latent states suggesting different tonal relations and chord transitions. One of them is the common jazz operation called the "enrichment" of chords, either viewed as 7, 9, 11 and 13 notes, or as sustained or color notes. These enrichments are often used as special events, and understanding the context of their application is important for allowing targeted and effective use of such harmonic devices. In the analysis conducted in section 3.3 it is found that the same musical notations (such as the  $F^7$  chord in "Now's the time" example) could be split between two clusters, where one (cluster-5 described in section 3.3) of them contains a particularly more rich and embellished set of notes than the other. Accordingly, when a VMO-HMM is used to generate a new chord progression by a random walk on the Markov structure between the latent variables, such alterations, substitutions or enrichments may be controlled as part of the musical meta-composition design.

#### 4.4.2 Random Walks on VMO-HMM

Given different Markov transition matrices from different  $lrs$  values obtained from algorithm 9 described in section 2.2.4.1, it is straight forward to sample latent variable sequences treating each row in the transition matrix as a multinomial distribution conditioned on its previous latent variable. Continuing from the analysis example used in section 3.3, given cluster-0 as the fixed initial latent variable, each next latent variable is drawn randomly given the multinomial distribution conditioned on the current latent variable. After the latent variable sequences are sampled, chord labels are assigned to latent variables based on the clusters shown in figure ?? in the same way as section 3.3. By examining the two sampled examples (figure 4.8), it could be observed that the chord choices and temporal relations of the lower order one are freer than the higher order one. If one focuses only on the root progression of these two example sequences, the 1st-order sequence contains progressions such as  $[I, ii, VI]$ ,  $[I, vi, ii, V]$  and  $[IV, V, I]$ , while the 5th-order one contains mainly the  $[I, vi, ii, V, I, V]$  progression. Based on these observations, the lower order Markov model indeed captures a wider variety than the higher order

| F7 | F7 | F7 | Gm |C7(b9)| F7 | D7 | Gm |C7(b9)| Bb7 | C7 |F7(#11)| C7 |  
 (a) 1st-order

| F7 | D7 | Gm | C7 |F7(#11)| C7 |F7(#11)| F7 | D7 | Gm | C7 |F7(#11)| C7 |  
 (b) 5th-order

**Figure 4.8:** Two sampled 12-bar chord label sequences of different orders Markov transition matrices from the VMO-HMM on the piece “Now’s the time”. It should be noted that the chord labels are inferred by human inspection on the clusters shown in figure 3.12, not the chord labels from figure 3.13.

Markov model but lacks repetitive structures. On the other hand, the higher order Markov model captures more salient harmonic progressions in the music spanning multiple bars. To render actual musical content from the chord label sequences, one simple method is to randomly select a bar containing the midi events from the cluster associated to the latent variable. Due to space limitation the generated scores are not shown here and could be found in the repository<sup>3</sup>. It should be noted that since the random sampling is on the latent variable space, there is no limitation on how the chord labels should be realized. Reshuffling the original musical content is just a convenient way, other generative methods based on chord labels could also be used.

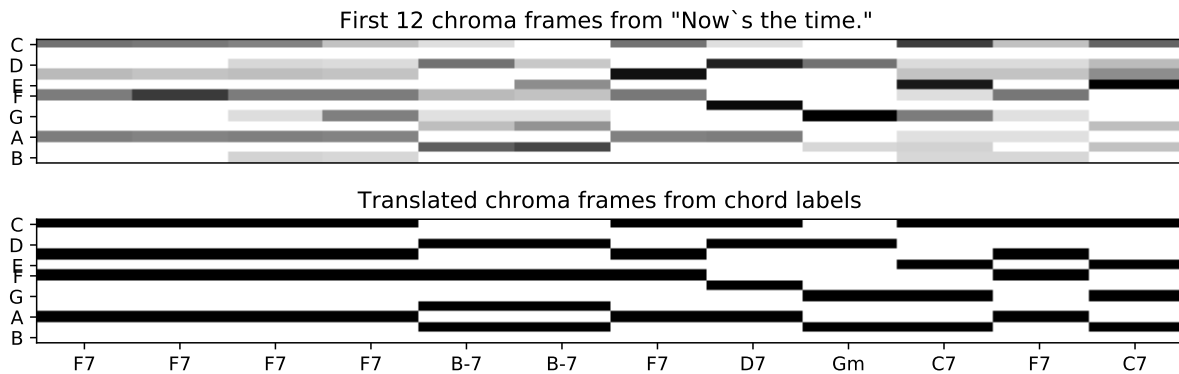
### 4.4.3 Query VMO-HMM by Chord Label Sequence

The other advantage of using the VMO-HMM is that it provides a complete setting for an ordinary HMM Viterbi recognition algorithm [7]. A Viterbi algorithm using *VMO* was proposed in [45] and chapter 2, where the transition probabilities are assumed to be a uniform distribution on the forward links from a frame to possible next frames. In the VMO-HMM setting, the transition probabilities between latent variables are learned from the oracle structure based on the longest repeated suffixes of each frame.

To use the Viterbi algorithm with a VMO-HMM for music content generation, one can specify a chord label sequence similar to the generated sequences in figure 4.8, then translate

<sup>3</sup>[https://github.com/wangsix/markov\\_improvisation](https://github.com/wangsix/markov_improvisation)

the chord label sequence into a chroma (pitch class profile) vector sequence. To translate chord labels to chroma vectors, one can simply use 12-dimensional binary vectors to represent 12-tone-pitch classes. An example of such translation and comparison to the actual chroma sequence is shown in figure 4.9. The translated chroma sequence is then used as the observation in the Viterbi algorithm. To infer the latent variable sequence generating the observations, the emission probability of an observation generated by a latent variable could also be simplified as the cosine distance between the binary pitch class vector (observation) and the centroid (mean or median of the cluster) chroma vector normalized to be a positive-valued vector which sums to 1. The Viterbi algorithm decodes an observation sequence to a latent variable sequence. The decoded latent variable sequence could then be used to generate new musical materials as in section 4.4.2. As a proof of concept using the aforementioned approach generating musical contents with a user specified chord label sequence, the chord labels from the first 12 bars in the reference MusicXml file of “Now’s the time” is used as the input chord label sequence to the Viterbi algorithm to see if the decoded latent variable sequence matches the given chord labels. The goal of this proof of concept is to see if given a version of translation between the chord labels and the pitch class profiles, how well could the Viterbi decoder from a VMO-HMM work. The result of this initial attempt works well since the Viterbi decoder is capable of finding the exact latent variable sequence as the input chord label given the reduced representation from a VMO-HMM. The testing script can also be found in the repository provided above. In figure 4.10, both the query and the decoded chord label sequences are shown. At bar 11, although the input label from the lead sheet specifies F7, but the Viterbi algorithm with the VMO-HMM extracted from the MIDI accompaniment file is able to spell out F7 (#11) given its different context from earlier F7s. This observation confirms that the VMO-HMM is capable of distinguishing similar chord given their pitch classes if they have different context in the music.



**Figure 4.9:** The actual midi-chromagram from the MIDI accompaniment compared to the translated query chromagram (pitch classes) from chord labels in the lead sheet.

| F7 | F7 | F7 | F7 | B $\flat$ 7 | B $\flat$ 7 | F7 | D7 | Gm | C7 | F7 | C7 |

(a) Query Chord Labels

| F7 | F7 | F7 | F7 | B $\flat$ 7 | B $\flat$ 7 | F7 | D7 | Gm | C7 | F7( $\sharp$ 11) | C7 |

(b) Decoded Chord Labels

**Figure 4.10:** The query and decoded chord label sequences. All the chord labels are matched besides bar 11, where the F7 from the lead sheet is identified as F7 ( $\sharp$ 11) given the VMO-HMM.

## 4.5 Discussions

Guided improvisation in music is closely related to the problem of planning and control where flexible/dynamic strategies are desired. Control improvisation has also become an area of interest in other domains. Specifying control signals to guide a system into a desired behavior is common in robotics and other dynamic systems, where adding a randomized strategy or constraining automata from examples to the generative process might add more flexibility to a system [99, 100]. In the case of music, the need for the improvisation to conform to outside constraints, such as partnering with other musicians, is less strictly defined and does not have any critical safety specifications. However, undesired notes can still be quite troublesome, especially during live performances where machines are involved. In the music alignment problem, solutions are proposed to match the same music piece in different media such as audio, midi, score, etc [101], but the solutions are focused on alignment, not creation. Another approach to this problem is introducing constraints to the improvisation system, as explored in [96, 102]. Unlike the probabilistic constraints or control imposed in [100], the constraints imposed in [102] is hard and rule-like. Other related examples that use time-automata to provide flexible and even improvisatory performances in coordination with a plan are the interactive score projects, such as Antescofo [97] and Virage [98]. In such cases, a formal score specification allows the system to modify its performance according to expressive inflection or to a musician, mostly limited to time changes within a tightly predetermined sequence of events. This restriction makes it difficult for the system to navigate the specification in a highly non-linear temporal fashion, i.e. allowing jumps and the recombination of events in the way the VMO structure is designed.

Generating musical structures with mathematical formulations or computational methods has been a recent focus in computer music. In [103], a Markov model is used to generate bar level song structure, while a genetic algorithm is used to generate more detailed musical components in the genre of Electronic Dance Music. In [104], multiple agents that focus on different attributes



compete with each other to form a negotiated decision directing the musical outcome in real-time. In [105], as an effort towards automatic articulation of musical scores into music renditions with expressive performance characteristics, pitch class collections are used as the basic units to decompose a piece into a structured view. Both [103, 105] are off-line systems and work on the symbolic representation of music.

For the application where a tonal piece is guided by rhythmic content, our results could be improved if the analysis frames are synchronized to the downbeats in the music. For each beat-synced analysis frame, the beat phase would be calculated and this feature would describe where significant events occur within a beat frame. This would better synchronize the rhythmic content with the tonal content across downbeats and within each beat measure. The problem with this is that analysis frames for each beat may be of different length because downbeats are not always spaced evenly and reconstruction algorithms would need to be modified. Because the beat phase feature is calculated using evenly spaced ODF samples, the feature would also need to be modified before attempting this extension. Also, it is desirable to not only guide a target signal with rhythmic content but also constrain harmonic or melodic progression so that the musical outcome is natural. To achieve such an effect, a navigation algorithm working with multiple *VMOs* must be designed.

For the structured improvisation proposed in this work with *VMOs*, another possible approach which is not presented here is to query the improvisation itself with pre-recorded materials. The pre-recorded materials are treated as scores or maps for the improvisation. The drawback of the current approach is that the navigation might be trapped in loops, and so an escape mechanism must be formulated. A taboo list containing a recent history of visited states could potentially solve this problem as proposed in [26].

An aspect of musical interaction that has not been addressed in this work is that in musical practice, synchronized entries of a beat, or tightly triggered tutti sections, are a common artistic tool. Allowing the oracle and the improviser to hit notes together after silence, or trigger

notes sequentially in a precisely timed way is a task for future research. Another notion of musical interaction that we have not discussed is the use of harmonic progression / variations / substitutions as a constraint for the oracle. This can be accomplished indirectly in our current system if the harmonic grid or the chord sequence is used as a query. We will consider in the future, the possibility of specifying a chord progression symbolically and providing it as a query to a recording. One problem with using such an approach is that harmonic theory and chroma analysis of audio are difficult to reconcile. To prevent this problem, we will need to either design an intermediate feature that is capable of bridging harmonic progression and chroma, or design a different audio feature that more directly describes harmonic progression.

In general, these issues relate to an even more general problem that we call the “duet problem” that appears when more or less free improvisation happens between two musicians or between a musician and a machine. A related problem in off-line composition using MIDI / symbolic sequences was encountered earlier in composing “Composer Duets”, briefly described in [106]. The idea there was to have the oracle attempt to match a polyphonic pattern from a duet recording. We do not exactly have this type of situation in audio, since usually, we do not have multi-track audio available for oracle training, so that one track could be used as the query while another can be used as improvisation material. In the future, we may experiment with creating recordings where two tracks are produced specifically for the accompaniment task or use multi-channel MIDI files. In this case, guidance of the oracle output on one track will be achieved by querying a second related track.

Chapter 4 is adapted from published materials in “*Guided Music Synthesis with Variable Markov Oracle*”. Wang, Cheng-i. & Dubnov, Shlomo. Tenth Artificial Intelligence and Interactive Digital Entertainment Conference, 2014, “*Machine Improvisation with Variable Markov Oracle: Toward Guided and Structured Improvisation*”. Wang, Cheng-i.; Hsu, Jennifer. & Dubnov, Shlomo. Computers in Entertainment (CIE), ACM, 2016, 14, 4 and “*Context-Aware Hidden Markov Models of Jazz Music with Variable Markov Oracle*”. Wang, Cheng-i. & Dubnov, Shlomo.

5th International Workshop on Musical Metacreation (MUME 2017) at the Eight International Conference on Computational Creativity, ICCC 2017, 2017.

# Appendix A

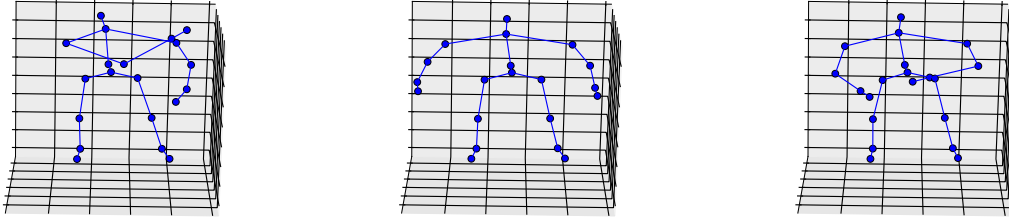
## Human-Gesture Applications

### A.1 3-D Gesture Recognition and Retrieval

The goal for this experiment is to test the performance of algorithm 6. The problem is formalized as follows; given a collection of time series data of the same type (e.g. audio, video, sensory data, etc) categorized into different categories (or types, genres, labels, etc) and a query time series not included in the collection, use the query time series to search over the collection and return multiple matches according to the query. The returned matches are deemed correct if they belongs to the same category as the query.

The MSRC-12 Kinect gesture dataset [107] is chosen for this experiment. This dataset contains 6244 annotated instances of both iconic and metaphoric human full body gestures recorded using XBox Kinect by 30 participants. The sample rate of the 3D joints is 30Hz and each frame of the gesture instance is stored as a 3D 20-joint skeleton. We treat the sequence of skeleton frames as the query time series, with  $R \in \mathbb{R}^{3 \times 20}$ . Snapshots of example skeleton frames are shown in figure A.1. The length of each gesture sequence is between 13 ~ 492 frames, and each participant performed 8 ~ 20 times for each gesture.

The experiment follows the leave-one-out principle; first all the instances are pre-processed



**Figure A.1:** Example frames of 3-D skeletal joints. There are 20 joints, making each frames having a dimension of 60.

**Table A.1:** Number of instances from the subset of the MSRC-12 dataset. These actions were meant for gaming scenarios.

Gesture	Participant					Total Number
	A	B	C	D	E	
Crouch or hide	20	20	21	21	20	102
Shoot with a pistol	20	10	22	9	20	81
Throw an object	9	10	10	20	22	71
Change weapon	19	20	20	20	20	99
Kick to attack	24	10	20	9	20	83
Put on a goggle	10	11	12	10	20	63

to store in *VMO* structures, then each time one instance from the collection is set aside as the query, next we use the skeleton sequence of the query as *R* in algorithm 6 to match all the rest instances in the collection stored as *VMOs*. Algorithm 6 returns the 10 entries with the lowest cost, *C*, for each query. Every instance is used as a query once in this experiment. A smaller subset instead of the full collection is chosen due to the large amount of instances. The subset chosen contains the 6 iconic gestures out of the 12 gestures performed by 5 randomly chosen participants from the full collection. The total number of instances of the subset is 499. A list of the details of the subset is provided in table A.1.

The result of the leave-one-out query-return experiment is listed in table A.2 and table A.3. In table A.2, a retrieved match is considered correct if it belongs to the same category of the query

**Table A.2:** Precision for Kinect skeletal gesture retrieval - Consider type of gesture only - (upper rows) performances using *VMO* and (lower three rows) with other approaches.

<b>Gesture</b>	<b>Precision (%)</b>
Crouch or hide	99.9
Shoot with a pistol	90.6
Throw an object	84.6
Change weapon	98.4
Kick to attack	92.7
Put on a goggle	92.2
<b>Avg.±std.</b>	<b>93.1±5</b>
<b>DTW[108]</b>	<b>82.74</b>
<b>HMM[108]</b>	<b>91.81</b>
<b>Cov3DJ-SVM[109]</b>	<b>93.6</b>

**Table A.3:** Precision for Kinect skeletal gesture retrieval - Both type and participant are considered

<b>Gesture</b>	<b>Participant</b>					<b>Precision (%)</b>
	A	B	C	D	E	
Crouch or hide	99	89.5	87.1	86.2	98.5	92±5.5
Shoot with a pistol	100	85	100	33.3	99.5	83.5±25
Throw an object	63.3	83	81	94.5	86.8	81.7±10
Change weapon	100	98.5	100	88.5	100	97.4±4.4
Kick to attack	100	90	100	51.1	95	87.2±18
Put on a goggle	90	81.8	100	57	100	85.7±15
<b>Avg.±std.</b>	92±13	87.9±5.6	94.6±7.7	68.4±22.5	96.6±4.7	<b>Avg.±std.</b>

for each of the 10 retrievals of the query, then the precision for one type of gesture is calculated as the counts of correct matches divided by the total number of queries from that type of gesture. The result in table A.2 is compared to [109], the state of the art on reported precision on the MSRC-12 dataset so far. The precision achieved using *VMO* query-matching algorithm (93.1%) reached a comparable level to [109] (93.6%) on the subset chosen in this experiment.

In [109], a feature called Cov3DJ is proposed where the covariances between joints are used, then a SVM is trained to recognize testing gestures. SVM with Cov3DJ reaches the state of art performance but since Cov3DJ has to be calculated over all the data points of each gesture, it is not possible for on-line applications. We also compare our results to the HMM and DTW and experiments done in [108] on the MSRC-12 dataset. DTW and HMM are both baseline approaches for time-series query-retrieval experiments. In [108], the feature used for DTW and HMM is similar to ours which is the time series data from the gestures. DTW performs the worst among *VMO*, HMM and itself with 82.75% accuracy. The relatively lower accuracy with DTW in comparison to the other two approaches is caused by the fact that DTW does not allow the query gesture having “jitter” behaviors, since DTW assumes linear time relations between the query and target gestures. HMM has similar performance to *VMO*. The estimation of the number of hidden states in an HMM is found by exhaustively searching over a range of possible values which in a sense is similar to finding the optimal  $\theta$  value for *VMO*, but the EM algorithm for estimating the HMM parameters has to iterate over each gesture for several times until convergence while the construction for *VMO* requires only one pass through each gesture, thus making HMM relatively inefficient comparing to *VMO*.

In table A.3, the criteria for correct match is stricter than the previous experiment in the sense that we only consider a match to be correct if both the gesture and the participant who performed the gesture are the same as the query. From the results in table A.3, the observation is that given the simple frame by frame 3-D skeletal joint data points, the query-matching algorithm not only is able to retrieve the same type of gesture, but is also able to retrieve the gesture that

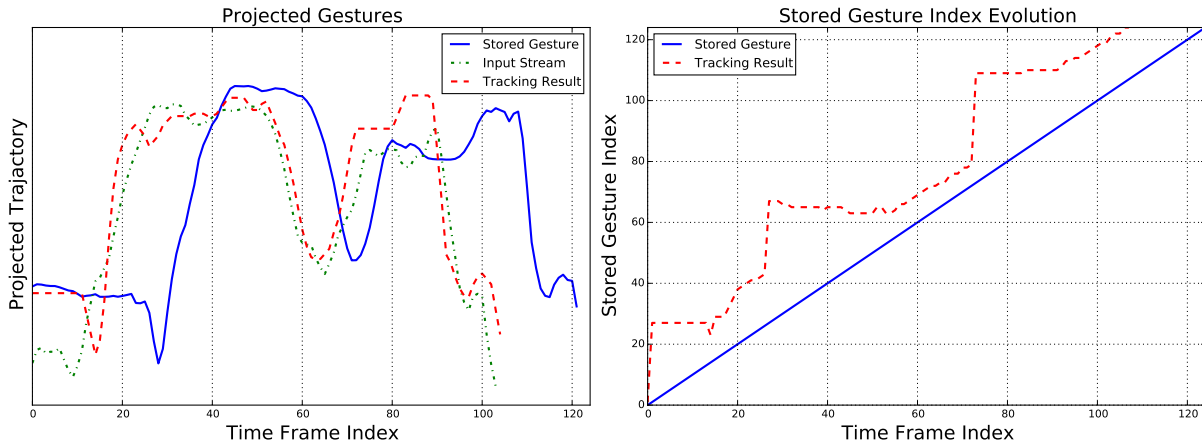
performed by the same participant who did the query gesture. This result implies a possible solution to user identification problem in gaming console or mobile devices, which was not addressed for the MSRC-12 dataset in previous research.

## A.2 Gesture Following

Based on the online query following algorithms proposed in section 2.2.3.2, it is possible to track where in the stored multivariate time series that the newly input observation is closest to. A visualization of such query following is depicted in figure A.2. In figure A.2, the left subfigure shows the result of the gesture following (red dashed line) based on a stored gesture (blue line) given unknown input observations (green dotted line). The right subfigure of figure A.2 shows how the gesture following result (red dashed line) corresponds to its original sequence (blue line) in terms of the matching in time (as of frame indices). The original sequence shown in the figure is part of a longer sequence where multiple gestures sampled from the MSRC-12 dataset were concatenated into one longer sequence to resemble how it would be used in real world performance/gaming environment. This long time series is then projected onto its first principal component for visualization. From the example shown in figure A.2, we observe that the on-line query following algorithm is able to find the correct segment in the stored gesture matching the input observations. Though due to visualization purposes, we omit the rest of the stored gesture (blue line), where the other gestures lie in the original time series, in figure A.2. Also we notice that from time index evolution in the right subfigure of figure A.2, the query following is consistent with the temporal relations of the stored gesture (the red dashed line mostly follows the diagonal with time stretch or compression movements indicated by nearly vertical or horizontal lines).

We combine the aforementioned on-line query following algorithms into an interactive dance/graphics system depicted in figure A.4. We use *VMO* as the mapping interface between

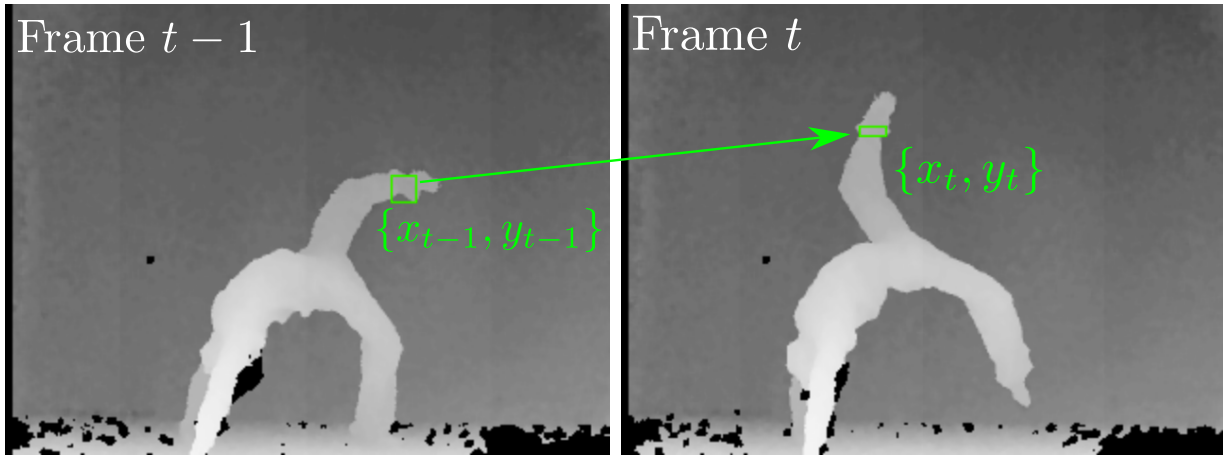




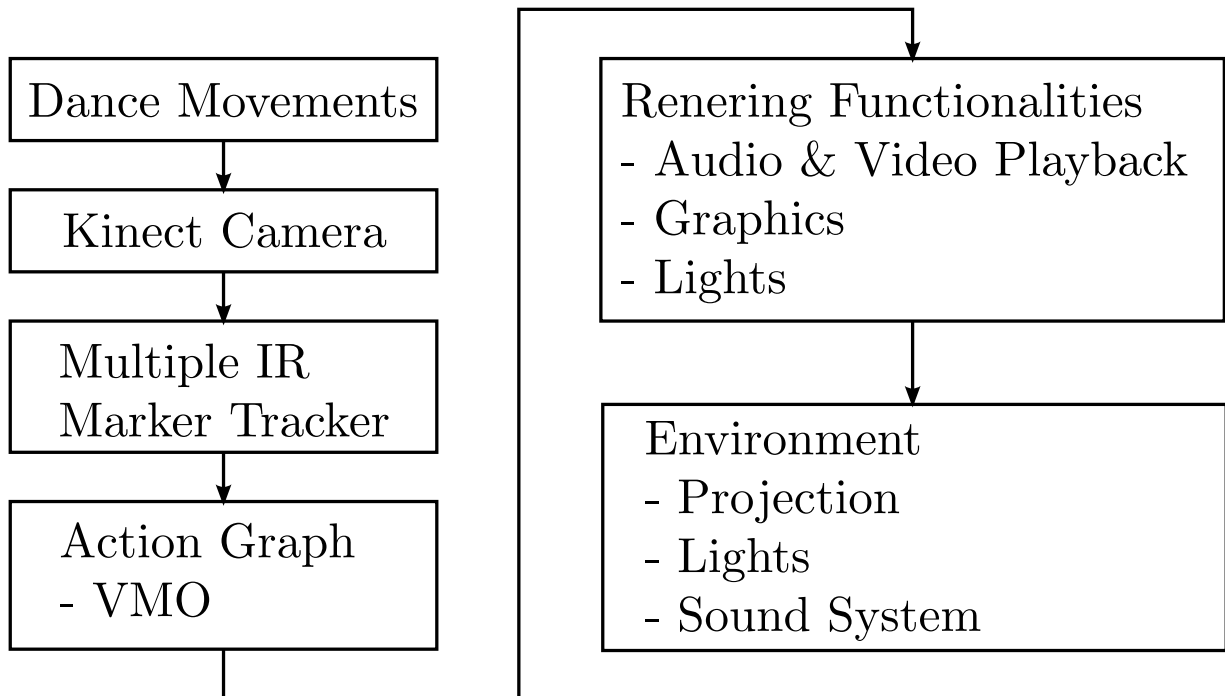
**Figure A.2:** (Left) Projected trajectory of the stored gesture (blue), input stream (green) and gesture following result (red). (Right) Time indices evolution of the gesture following result (red) compared to original gesture (blue).

the input dance movements and graphics/interaction/effects rendered during a performance. We implement *VMO* and the proposed algorithms in C++ within the `OPENFRAMEWORKS` open source environment (<http://openframeworks.cc/>). Snapshots of the *VMO* gesture following working alongside computer-generated graphics in `OPENFRAMEWORKS` are depicted in figure A.5. The dance movements are captured by identifying the infrared reflector tied on the dancer's joints using the Kinect camera. We show an example of the raw depth image along with the identified marker in figure A.3. In figure A.5, the trajectory represents the stored gesture (sequence of 2D points) with different colors indicating mappings to different computer-generated graphics. The bigger circle is the input marker and the green small circle along the trajectory is the current gesture-following position on the stored gesture corresponding to the input marker. In this demo, the system generates different graphics when the gesture following position (green circle) traverses onto different segments of the stored gesture represented by different colors.

For the system proposed in this appendix, we consider a scenario where the rehearsal or practice of the dancer is recorded and stored as *VMO*. During a live performance the system tracks the input dance movements and matches them to the gestures stored in *VMO* via algorithm 7 and 8. For graphics/effects/interaction rendering, the temporal mapping between live dance movements



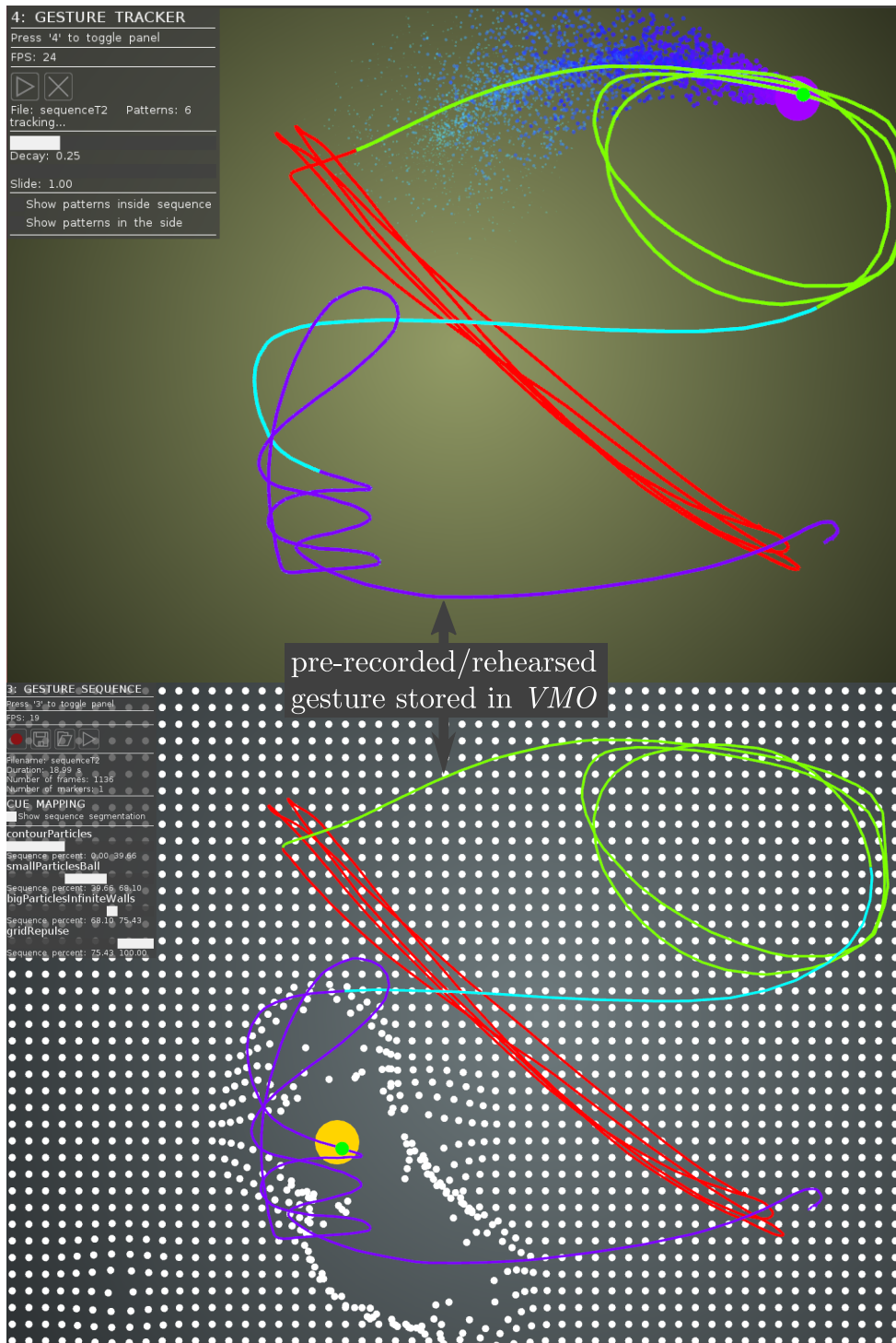
**Figure A.3:** The green rectangles represent input identified markers on the dancer's ankle by processing the infrared images from the Kinect camera.



**Figure A.4:** Diagram of the proposed interactive dance/graphics system using *VMO*. The system would be capable of using guiding or generating multimedia contents with a live query input.

to the stored gestures by gesture following allows the evolvement of the incoming time series *scrubbing* another time series, such as video, audio or rendering function parameters that varies in time. Other details of this interactive dance/graphics system is documented in [110].

Appendix A is adapted from published materials in "*Variable Markov Oracle: A Novel Sequential Data Points Clustering Algorithm with Application to 3D Gesture Query-Matching*". Wang, Cheng-i. & Dubnov, Shlomo. International Symposium on Multimedia, 2014, 215-222 and "*The Variable Markov Oracle: Algorithms for Human Gesture Applications*". Wang, Cheng-i. & Dubnov, Shlomo. IEEE MultiMedia, IEEE, 2015, 22, 52-67.



**Figure A.5:** Snapshots of the OPENFRAMEWORKS application where *VMO* and the gesture following algorithms are embedded. Two different graphics are shown here. See text for details. (Top) Particle emission showing the trace of the input marker along the stored gesture trajectory. (Bottom) Grid particles interact with the input marker.

# Appendix B

## *Memento*

### B.1 Concert Proposal for MuMe 2017

- **Title:** *Memento*
- **Musical and technical creation:** Shlomo Dubnov, Cheng-i Wang and Jaime Arias
- **Conceptual Design:** Shlomo Dubnov
- **Duration:** Variable, but preferably 9-12 minutes
- **Instrumentation:** To be determined with the performers, possibly based on "the shape of distance[5]": 2 flutes, clarinet, viola and percussion. See <https://chambercartel.bandcamp.com/album/the-shape-distance>

#### B.1.1 Description of the work

The proposed performance uses a novel structured improvisation system called VMO-score which is a tool to generate an interactive score to control the improvisation according to larger structures found in an audio recording. "*Memento*" is designed to explore the question

of musical structure by structuring two timelines - one of the reference recording recombined and sequenced by the machine in reverse, and one of the live performer, progressing forward on same music materials. This structure design borrows from similar techniques used in cinema, questioning the relation between order of presentation and the natural arrow of time of the music materials, also hinting at a structure of film bearing a resembling name.

### **B.1.2 Description of the technology behind VMO-Score**

The VMO-Score system [111] takes an audio file as an input and uses the tool *VMO* [45] for improvisation. In addition, *VMO* is also used to do a segmentation analysis of the input [112]. Once the tool has identified possible natural transitions between sections with similar musical content, it translates the musical structure into a Petri net [113] model in order to provide a higher, more intuitive and formal representation of this structure. Therefore, the artist can modify the structure of the Petri net in order to control the improvisation by adding temporal and logical constraints. VMO-Score also generates an interactive score based on the Petri net structure for the inter-media sequencer *i-score* [114]. This tool provides a complete graphical interface for structuring and performing in real-time the improvisation. Such improvisation is carried out by the Max interface called *PyOracle* which is controlled by *i-score* using OSC messages [27]. Musically speaking, the reference audio recording is used as raw material for creating a semi-open musical form for an human-machine improvisation. The tool allows design of improvisation where tighter relations are established between the musician and the machine based on pre-determined music materials, a pre-designed structure, and careful planning of transition times and conditions ahead of the stage performance.

### **B.1.3 Examples of previous performances using VMO-Score**

Examples of improvisation with "parallel" time (unlike the proposed "*Mumento*" design explained above), can be seen on:

- <https://youtu.be/AvAlBux-nM4>
- <https://youtu.be/oT9e6culX40>

### **B.1.4 Novel Technological Elements in the current work**

In the previous performances, the triggers for chaining the improvisation segment and progression and branching of the musical form were initiated by the performer. For reasons of convenience, instead of using pedals or other specific triggers, the musician signaled to the computer operator when and which transitions to follow. In the proposed piece we plan to explore a machine listening element by adding feature extraction and detectors for specific musical materials that would signal a transition to the machine. This change will allow a fully autonomous machine operation, adding to the expressive freedom of the man-machine duo, while maintaining pre-negotiated rules of discourse. Accordingly, the new element we are planning to include is a live machine listening component that will be active during the performance that will control the transitions/form in autonomous manner.

### **B.1.5 Preparing for the MuMe 2017 Concert**

We plan to contact some of the players from Chamber Cartel ensemble (<http://www.chambercartel.com/>) to use some of their musical recordings as the baseline musical material for structuring the new composition. As explained in the section about the *Mumento* structure, the recording, after segmentation, will be re-arranged in a way that reverses the segmental sequence

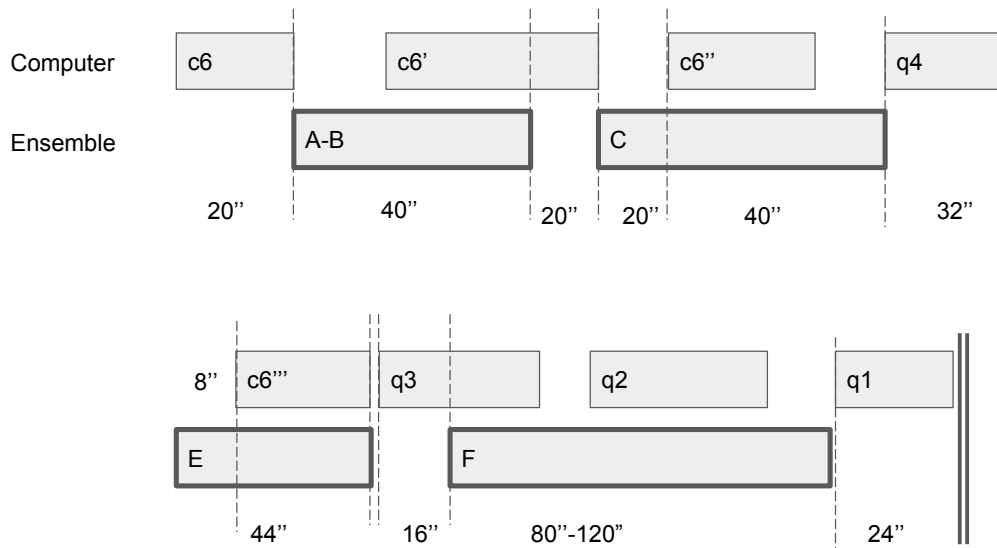
with slight overlap. This kind of design is possible thanks to the segmentation tools and the i-score / petri-net programming flexibility. The ideal situation for designing the piece would be to prepare the improvisation based on an accompaniment sound file that the performers know well. It could be an example from their own composition or prior improvisation, or some sound track that they know well. Such choice of materials allows tighter integration between the computer part and the improviser and better planning of the overall form.

## B.2 Scores and performance plan

The instruction for each musician are shown in figure B.2, B.3, B.4 and B.5. The instruction for each musician is split into 4 parts, with each part containing performance instructions taken from the piece “*Domination of Black*” by Drew Backer. The instructions on each of the 4 parts are based on materials from rehearsal marks **A-B**, **C**, **E** and **F** on the score.

For the computer part, a recording of the “*Domination of Black*” (<https://soundcloud.com/drewbakermusic/domination-of-black>) is segmented using the structural segmentation algorithms proposed in section 3.2.2.1. Two versions of segmentation are extracted, one with chromagram while the other with constant-Q transformed spectrogram (CQT). The segments from the segmentation algorithms are then fed into the basic machine improvisation algorithm, algorithm 12, during the performance to generate new musical materials on the fly. The segments used during the performance are manually selected and ordered reversely to their original order. In figure B.1, the reversely ordered machine improvisation segment are shown. For example, **c6** refers to the 6th segments from segmentation with CQT, **q4** for the 4th segments from the chromagram segmentation, etc. Also the timings for when the ensemble should start, stop and advance to the next part is shown. In the timeline figure it can be seen that while the human ensemble following the instructions are advancing to different parts in the same order as the original score, the computer generated materials are based on segments that are ordered reversely





**Figure B.1:** The timeline of the “*Memento*” piece, indicating when the musicians should start and stop playing, also has information as to which segments are being improvised upon by *VMO*.

to the order in the original score.

The actual performance took place at the MuMe 2017 concert in Atlanta, Georgia, on June 19th, 2017. The venue was the Mammal Gallery. The ensemble was the Chamber Cartel (<http://www.chambercartel.com/>) lead by percussionist Caleb Herron. A video recording of the performance can be found at <https://www.youtube.com/watch?v=xGr5mX7mmIE>.

♩ = 60  
with absolute stillness and focus

Viola

NOT PLAYING FOR FIRST SECTION

(COMPUTER IMPROV. - INSTRUMENTAL W/ COMPUTER - COMPUTER SOLO)

(a) A-B

FREE TO ENTER AND EXIT ON ANY NOTE BUT NO SILENCES, KEEP BLENDING

poco vib. - always blend

*ppp*

col legno tratto

*pp*

ord

*pp*

(EXIT INDIVIDUALLY)

AFTER COMPUTER ENTERS, KEEP PLAYING FOR 10 BARS (~40')

(b) C

TUTTI - ENTER TOGETHER AND KEEP PRECISE COUNT WITH OTHER MUSICIANS

E

(c) E

TUTTI

*p*

*pp*

*p*

col legno tratto

*p*

*pp*

ord

*ppp*

WAIT FOR CONDUCTOR SIGNAL TO MOVE TO THE NEXT STAVE

(d) F

**Figure B.2:** The 4-part instructions for viola. The materials for each part are chosen, gathered, rearranged from corresponding rehearsal marks [A-B, C, E, F] in the original score of “*Domination of Black*”.

♩ = 60  
with absolute stillness and focus

Cello

COMPUTER IMPROV. ~ 20"

SIL.

WAIT FOR CONDUCTOR SIGNAL OR PLAY ONCE AFTER CLARINET AND MARIMBA ALREADY ENTERED TO SIGNAL POSSIBLE ENDING OF SECTION

poco vib. - always blend

ppp

MAY REPEAT IF CONDUCTOR SIGNALS

(a)

FREE TO ENTER AND EXIT ON ANY NOTE BUT NO SILENCES, KEEP BLENDING

ppp

sul D

pp

AFTER COMPUTER ENTERS, KEEP PLAYING FOR 10 BARS (~40)

(EXIT INDIVIDUALLY)

(b)

TUTTI - ENTER TOGETHER AND KEEP PRECISE COUNT WITH OTHER MUSICIANS

pp

(c)

TUTTI

p

pp

p

WAIT FOR CONDUCTOR SIGNAL TO MOVE TO THE NEXT STAVE

col legno tratto

p

ppp

sul A  
ord


ppp

(d)

**Figure B.3:** The 4-part instructions for cello. The materials for each part are chosen, gathered, rearranged from corresponding rehearsal marks [A-B, C, E, F] in the original score of “*Domination of Black*”.

♩ = 60  
with absolute stillness and focus

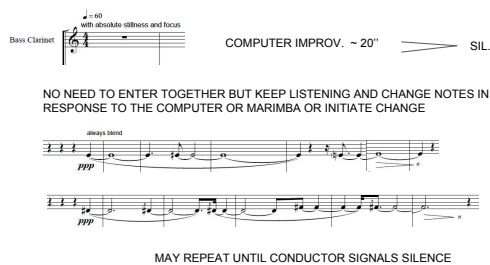
Bass Clarinet

COMPUTER IMPROV. ~ 20"  SIL.

NO NEED TO ENTER TOGETHER BUT KEEP LISTENING AND CHANGE NOTES IN RESPONSE TO THE COMPUTER OR MARIMBA OR INITIATE CHANGE

always blend  
ppp

MAY REPEAT UNTIL CONDUCTOR SIGNALS SILENCE

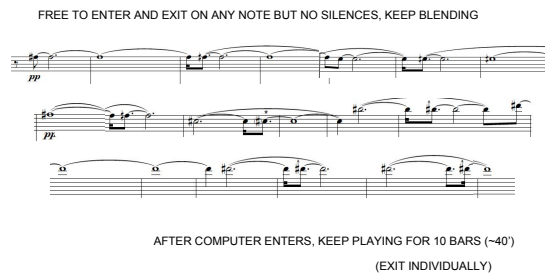


(a)

FREE TO ENTER AND EXIT ON ANY NOTE BUT NO SILENCES, KEEP BLENDING

ppp

AFTER COMPUTER ENTERS, KEEP PLAYING FOR 10 BARS (~40")  
(EXIT INDIVIDUALLY)



(b)

TUTTI - ENTER TOGETHER AND KEEP PRECISE COUNT WITH OTHER MUSICIANS

E

ppp



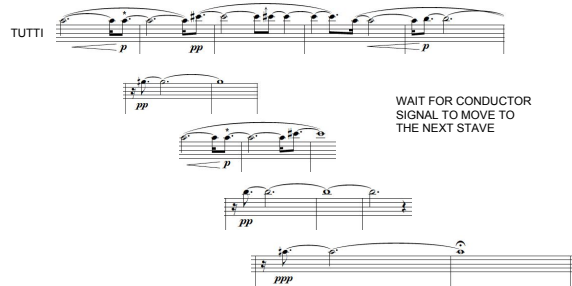
(c)

TUTTI

ppp

WAIT FOR CONDUCTOR SIGNAL TO MOVE TO THE NEXT STAVE

ppp



(d)

**Figure B.4:** The 4-part instructions for bass clarinet. The materials for each part are chosen, gathered, re-arranged from corresponding rehearsal marks [A-B, C, E, F] in the original score of “*Domination of Black*”.

♩ = 60  
with absolute stillness and focus

COMPUTER IMPROV. ~ 20"  $\triangleright$  SIL.

NO NEED TO ENTER TOGETHER BUT KEEP LISTENING AND CHANGE NOTES IN RESPONSE TO THE COMPUTER OR B. CLARINET OR INITIATE CHANGE

*pppp* arco always bend

*pppp*

MAY REPEAT UNTIL CONDUCTOR SIGNALS SILENCE

(a)

FREE TO ENTER AND EXIT ON ANY NOTE BUT NO SILENCES, KEEP BLENDING

*pp*

AFTER COMPUTER ENTERS, KEEP PLAYING FOR 10 BARS (~40")

(EXIT INDIVIDUALLY)

(b)

TUTTI - ENTER TOGETHER AND KEEP PRECISE COUNT WITH OTHER MUSICIANS

E mallet

Crotales

*p* *v. sempre*

take bow

(c)

TUTTI

arco (*v. sempre*)

WAIT FOR CONDUCTOR SIGNAL TO MOVE TO THE NEXT STAVE

to marimba

Marimba arco

*pp* - *p*

*pp*

*ppp*

(d)

**Figure B.5:** The 4-part instructions for percussion. The materials for each part are chosen, gathered, re-arranged from corresponding rehearsal marks [A-B, C, E, F] in the original score of “*Domination of Black*”.

# Acronyms

**AO** *Audio Oracle*. 3, 5–7, 9–11, 16, 62–65, 67, 68, 71, *Glossary*: Audio Oracle

**CRF** Linear Chain Conditional Random Field. 2

**DTW** Dynamic Time Warping. 3

**FO** *Factor Oracle*. 3, 5–11, 16, 28, 62–64, 78, *Glossary*: Factor Oracle

**HMM** Hidden Markov Model. 2

**IR** Information Rate. ix, 5, 15–19, 28, 34, 47

**lrs** length of longest repeated suffixes. 9, 16–18, 28, 31, 34

**PIR** Predictive Information Rate. 4, 5, 17

**sfx** suffix link. 8, 11, 28, 31, *Glossary*: Suffix link

**SSM** Self-similarity matrix. 44–47, 49–51, 55

**VMO** *Variable Markov Oracle*. ix, 3, 6, 7, 10–12, 14–17, 19, 21–23, 25–29, 31, 32, 34, 36–38, 40–43, 45–48, 54–56, 58, 64, 65, 67, 68, 71, 79, 81, 82, 84, 88, *Glossary*: Variable Markov Oracle

# Glossary

**Audio Oracle** A signal extension of the Factor Oracle. 3, 7, 62, 109

**Factor Oracle** A compressed suffix tree on a symbolic sequence. 3, 7, 62, 109

**Suffix link** A back ward pointer pointing back in time to where the longest repeated suffix happend. 8, 31, 109

**Variable Markov Oracle** A compressed suffix tree structure on continuous time-series by combining Factor Oracle and Audio Oracle. ix, 3, 7, 31, 64, 109

# Bibliography

- [1] J. Stambaugh, “Music as a temporal form,” *The Journal of Philosophy*, vol. 61, no. 9, pp. 265–280, 1964.
- [2] P. Kivy, *The fine art of repetition: Essays in the philosophy of music*. Cambridge University Press, 1993.
- [3] E. Campbell, *Boulez, music and philosophy*, vol. 27. Cambridge university press, 2010.
- [4] M. A. Casey, R. Veltkamp, M. Goto, M. Leman, C. Rhodes, and M. Slaney, “Content-based music information retrieval: Current directions and future challenges,” *Proceedings of the IEEE*, vol. 96, no. 4, pp. 668–696, 2008.
- [5] L. Su, C.-C. M. Yeh, J.-Y. Liu, J.-C. Wang, and Y.-H. Yang, “A systematic evaluation of the bag-of-frames representation for music information retrieval,” *IEEE Transactions on Multimedia*, vol. 16, no. 5, pp. 1188–1200, 2014.
- [6] L. R. Rabiner, “A tutorial on hidden markov models and selected applications in speech recognition,” *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.
- [7] A. Sheh and D. P. Ellis, “Chord segmentation and recognition using em-trained hidden markov models,” 2003.
- [8] A. Ozerov, C. Févotte, and M. Charbit, “Factorial scaled hidden markov model for polyphonic audio representation and source separation,” in *Applications of Signal Processing to Audio and Acoustics, 2009. WASPAA’09. IEEE Workshop on*, pp. 121–124, IEEE, 2009.
- [9] J.-J. Aucouturier and M. Sandler, “Segmentation of musical signals using hidden markov models,” *Preprints-Audio Engineering Society*, 2001.
- [10] E. Coviello, A. B. Chan, and G. Lanckriet, “Time series models for semantic music annotation,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 5, pp. 1343–1359, 2011.
- [11] E. M. Schmidt and Y. E. Kim, “Modeling musical emotion dynamics with conditional random fields,” in *Proceedings of the International Society of Music Information Retrieval Conference*, pp. 777–782, Miami (Florida), USA, 2011.



- [12] C. Joder, S. Essid, and G. Richard, “A conditional random field framework for robust and scalable audio-to-score matching,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 8, pp. 2385–2397, 2011.
- [13] M. C. Mozer, “Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multi-scale processing,” *Connection Science*, vol. 6, no. 2-3, pp. 247–280, 1994.
- [14] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, “Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription,” *arXiv preprint arXiv:1206.6392*, 2012.
- [15] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, “Audio chord recognition with recurrent neural networks,” in *Proceedings of the International Society of Music Information Retrieval Conference*, pp. 335–340, Citeseer, 2013.
- [16] S. Sigtia, E. Benetos, and S. Dixon, “An end-to-end neural network for polyphonic piano music transcription,” *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 24, no. 5, pp. 927–939, 2016.
- [17] E. J. Humphrey and J. P. Bello, “Rethinking automatic chord recognition with convolutional neural networks,” in *Machine Learning and Applications (ICMLA), 2012 11th International Conference on*, vol. 2, pp. 357–362, IEEE, 2012.
- [18] K. Ullrich, J. Schlüter, and T. Grill, “Boundary detection in music structure analysis using convolutional neural networks.,” in *Proceedings of the International Society of Music Information Retrieval Conference*, pp. 417–422, 2014.
- [19] K. Choi, G. Fazekas, and M. Sandler, “Automatic tagging using deep convolutional neural networks,” *arXiv preprint arXiv:1606.00298*, 2016.
- [20] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, “Wavenet: A generative model for raw audio,” *arXiv preprint arXiv:1609.03499*, 2016.
- [21] L.-C. Yang, S.-Y. Chou, and Y.-H. Yang, “Midinet: A convolutional generative adversarial network for symbolic-domain music generation,” in *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR2017), Suzhou, China*, 2017.
- [22] M. Cuturi, “Fast global alignment kernels,” in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 929–936, 2011.
- [23] A. Cont, S. Dubnov, and G. Assayag, “Guidage: A fast audio query guided assemblage,” in *Proceedings of International Computer Music Conference (ICMC)*, 2007.
- [24] S. Dubnov, G. Assayag, and A. Cont, “Audio oracle: A new algorithm for fast learning of audio structures,” in *Proceedings of International Computer Music Conference (ICMC)*, 2007.

- [25] G. Assayag and S. Dubnov, "Using factor oracles for machine improvisation," *Soft Computing*, vol. 8, no. 9, pp. 604–610, 2004.
- [26] G. Assayag and G. Bloch, "Navigating the oracle: A heuristic approach," in *International Computer Music Conference*, vol. 7, pp. 405–412, 2007.
- [27] G. Surges and S. Dubnov, "Feature selection and composition using pyoracle," in *Ninth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2013.
- [28] A. Cont, S. Dubnov, and G. Assayag, "On the information geometry of audio streams with applications to similarity computing," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 837–846, 2011.
- [29] S. Dubnov, "Spectral anticipations," *Computer Music Journal*, vol. 30, no. 2, pp. 63–83, 2006.
- [30] S. Abdallah and M. Plumbley, "Information dynamics: patterns of expectation and surprise in the perception of music," *Connection Science*, vol. 21, no. 2-3, pp. 89–117, 2009.
- [31] S. Dubnov, G. Assayag, and A. Cont, "Audio oracle analysis of musical information rate," in *Semantic Computing (ICSC), 2011 Fifth IEEE International Conference on*, pp. 567–571, IEEE, 2011.
- [32] D. B. Huron, *Sweet anticipation: Music and the psychology of expectation*. MIT press, 2006.
- [33] L. B. Meyer, *Emotion and meaning in music*. University of chicago Press, 1956.
- [34] D. Temperley, *Music and probability*. Mit Press, 2007.
- [35] L. B. Meyer, *Music, the arts, and ideas: Patterns and predictions in twentieth-century culture*. University of Chicago Press, 2010.
- [36] J. Ziv and A. Lempel, "Compression of individual sequences via variable-rate coding," *IEEE transactions on Information Theory*, vol. 24, no. 5, pp. 530–536, 1978.
- [37] A. Lefebvre and T. Lecroq, "Compror: on-line lossless data compression with a factor oracle," *Information Processing Letters*, vol. 83, no. 1, pp. 1–6, 2002.
- [38] C. Allauzen, M. Crochemore, and M. Raffinot, "Factor oracle: A new structure for pattern matching," in *SOFSEM99: Theory and Practice of Informatics*, pp. 295–310, Springer, 1999.
- [39] A. Lefebvre, T. Lecroq, and J. Alexandre, "An improved algorithm for finding longest repeats with a modified factor oracle," *Journal of Automata, Languages and Combinatorics*, vol. 8, no. 4, pp. 647–657, 2003.
- [40] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*. John Wiley & Sons, 2012.

- [41] F. Pachet and P. Roy, “Markov constraints: steerable generation of markov sequences,” *Constraints*, vol. 16, no. 2, pp. 148–172, 2011.
- [42] F. Lerdahl and R. S. Jackendoff, *A generative theory of tonal music*. MIT press, 1985.
- [43] H. Schenker, *Free Composition: Volume III of New Musical Theories and Fantasies*, vol. 1. Pendragon Press, 2001.
- [44] M. J. Wainwright and M. I. Jordan, “Graphical models, exponential families, and variational inference,” *Foundations and Trends® in Machine Learning*, vol. 1, no. 1-2, pp. 1–305, 2008.
- [45] C.-i. Wang and S. Dubnov, “The variable markov oracle: Algorithms for human gesture applications,” *IEEE MultiMedia*, vol. 22, no. 4, pp. 52–67, 2015.
- [46] C.-i. Wang, J. Hsu, and S. Dubnov, “Machine improvisation with variable markov oracle: Toward guided and structured improvisation,” *Computers in Entertainment (CIE)*, vol. 14, no. 3, p. 4, 2016.
- [47] T. Collins, “Discovery of repeated themes and sections.”
- [48] B. Janssen, W. B. d. Haas, A. Volk, and P. Kranenburg, “Discovering repeated patterns in music: potentials, challenges, open questions,” in *10th International Symposium on Computer Music Multidisciplinary Research*, Laboratoire de Mécanique et d’Acoustique, 2013.
- [49] B. McFee and D. P. Ellis, “Analyzing song structure with spectral clustering,” in *The 15th International Society for Music Information Retrieval Conference*, pp. 405–410, 2014.
- [50] E. Cambouropoulos, M. Crochemore, C. S. Iliopoulos, M. Mohamed, and M.-F. Sagot, “All maximal-pairs in step–leap representation of melodic sequence,” *Information Sciences*, vol. 177, no. 9, pp. 1954–1962, 2007.
- [51] O. Nieto and M. Farbood, “Perceptual evaluation of automatically extracted musical motives,” in *Proceedings of the 12th International Conference on Music Perception and Cognition*, pp. 723–727, 2012.
- [52] D. Meredith, K. Lemström, and G. A. Wiggins, “Algorithms for discovering repeated patterns in multidimensional representations of polyphonic music,” *Journal of New Music Research*, vol. 31, no. 4, pp. 321–345, 2002.
- [53] T. Collins, A. Arzt, S. Flossmann, and G. Widmer, “SIARCT-CFP: Improving precision and the discovery of inexact musical patterns in point-set representations.” in *Proceedings of the International Society of Music Information Retrieval Conference*, pp. 549–554, 2013.

- [54] T. Collins, S. Böck, F. Krebs, and G. Widmer, “Bridging the audio-symbolic gap: The discovery of repeated note content directly from polyphonic music audio,” in *Audio Engineering Society Conference: 53rd International Conference: Semantic Audio*, Audio Engineering Society, 2014.
- [55] O. Nieto and M. Farbood, “MIREX 2013: Discovering musical patterns using audio structural segmentation techniques,” *Music Information Retrieval Evaluation eXchange, Curitiba, Brazil*, 2013.
- [56] O. Nieto and M. Farbood, “Identifying polyphonic patterns from audio recordings using music segmentation techniques,” in *The 15th International Society for Music Information Retrieval Conference*, 2014.
- [57] J. P. Bello, “Measuring structural similarity in music,” *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 19, no. 7, pp. 2013–2025, 2011.
- [58] G. Tzanetakis, A. Ermolinskyi, and P. Cook, “Pitch histograms in audio and symbolic music information retrieval,” *Journal of New Music Research*, vol. 32, no. 2, pp. 143–152, 2003.
- [59] D. P. Ellis, “Beat tracking by dynamic programming,” *Journal of New Music Research*, vol. 36, no. 1, pp. 51–60, 2007.
- [60] C.-i. Wang and S. Dubnov, “Pattern discovery from audio recordings by variable markov oracle: A music information dynamics approach,” in *Acoustics, Speech, and Signal Processing (ICASSP), 2015 IEEE International Conference on*, IEEE, 2015.
- [61] J. Serra, M. Muller, P. Grosche, and J. L. Arcos, “Unsupervised music structure annotation by time series structure features and segment similarity,” *Multimedia, IEEE Transactions on*, vol. 16, no. 5, pp. 1229–1240, 2014.
- [62] J. Rink, N. Spiro, and N. Gold, “Motive, gesture, and the analysis of performance,” *New Perspectives on Music and Gesture*, pp. 267–292, 2011.
- [63] M. Müller, “Dynamic time warping,” *Information retrieval for music and motion*, pp. 69–84, 2007.
- [64] D. Meredith, “COSIATEC and SIATECCompress: Pattern discovery by geometric compression,” in *International Society for Music Information Retrieval Conference*, 2013.
- [65] O. Nieto and T. Jehan, “Convex non-negative matrix factorization for automatic music structure identification,” in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pp. 236–240, IEEE, 2013.
- [66] R. B. Dannenberg and M. Goto, “Music structure analysis from acoustic signals,” in *Handbook of Signal Processing in Acoustics*, pp. 305–331, Springer, 2008.

- [67] J. Paulus, M. Müller, and A. Klapuri, “State of the art report: Audio-based music structure analysis.,” in *ISMIR*, pp. 625–636, 2010.
- [68] F. Kaiser and T. Sikora, “Music structure discovery in popular music using non-negative matrix factorization.,” in *Proceedings of the International Society of Music Information Retrieval Conference*, pp. 429–434, 2010.
- [69] O. Nieto, *Discovering Structure in Music: Automatic Approaches and Perceptual Evaluations*. PhD thesis, NYU, 2015.
- [70] B. McFee and D. P. Ellis, “Learning to segment songs with ordinal linear discriminant analysis,” in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pp. 5197–5201, IEEE, 2014.
- [71] J. Foote, “Automatic audio segmentation using a measure of audio novelty,” in *Multimedia and Expo, 2000. ICME 2000. 2000 IEEE International Conference on*, vol. 1, pp. 452–455, IEEE, 2000.
- [72] C. Harte, *Towards automatic extraction of harmony information from music signals*. PhD thesis, Department of Electronic Engineering, Queen Mary, University of London, 2010.
- [73] U. Von Luxburg, “A tutorial on spectral clustering,” *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.
- [74] A. Dessen and A. Cont, “An information-geometric approach to real-time audio segmentation,” *Signal Processing Letters, IEEE*, vol. 20, no. 4, pp. 331–334, 2013.
- [75] R. J. Weiss and J. P. Bello, “Unsupervised discovery of temporal structure in music,” *Selected Topics in Signal Processing, IEEE Journal of*, vol. 5, no. 6, pp. 1240–1251, 2011.
- [76] M. Levy and M. Sandler, “Structural segmentation of musical audio by constrained clustering,” *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 16, no. 2, pp. 318–326, 2008.
- [77] R. D. Morris, “Voice-leading spaces,” *Music Theory Spectrum*, vol. 20, no. 2, pp. 175–208, 1998.
- [78] M. V. Mathews, J. E. Miller, F. R. Moore, J. R. Pierce, and J.-C. Risset, *The technology of computer music*. MIT press Cambridge, 1969.
- [79] C. Ames, “Automated composition in retrospect: 1956-1986,” *Leonardo*, vol. 20, pp. 169–185, January 1987.
- [80] A. Eigenfeldt, A. Burnett, and P. Pasquier, “Evaluating musical metacreation in a live performance context,” in *Proceedings of the Third International Conference on Computational Creativity*, pp. 140–144, Citeseer, 2012.

- [81] C. Ariza, “The interrogator as critic: The turing test and the evaluation of generative music systems,” *Computer Music Journal*, vol. 33, no. 2, pp. 48–70, 2009.
- [82] A. Eigenfeldt, O. Bown, P. Pasquier, and A. Martin, “Towards a taxonomy of musical metacreation: Reflections on the first musical metacreation weekend,” in *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment (AIIDE13) Conference, Boston*, 2013.
- [83] S. Dubnov, G. Assayag, O. Lartillot, and G. Bejerano, “Using machine-learning methods for musical style modeling,” *Computer*, vol. 36, no. 10, pp. 73–80, 2003.
- [84] D. Ron, Y. Singer, and N. Tishby, “The power of amnesia: Learning probabilistic automata with variable memory length,” *Machine learning*, vol. 25, no. 2-3, pp. 117–149, 1996.
- [85] G. Assayag, S. Dubnov, and O. Delerue, “Guessing the composer’s mind: Applying universal prediction to musical style,” in *ICMC*, 1999.
- [86] G. Assayag, G. Bloch, M. Chemillier, A. Cont, and S. Dubnov, “Omax brothers: a dynamic yopology of agents for improvization learning,” in *Proceedings of the 1st ACM workshop on Audio and music computing multimedia*, pp. 125–132, ACM, 2006.
- [87] J. Nika, M. Chemillier, and G. Assayag, “Improtek: introducing scenarios into human-computer music improvisation,” *Computers in Entertainment (CIE)*, vol. 14, no. 2, p. 4, 2016.
- [88] C.-i. Wang and S. Dubnov, “Guided music synthesis with variable markov oracle,” in *Tenth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2014.
- [89] R. Panda, R. Malheiro, B. Rocha, A. Oliveira, and R. P. Paiva, “Multi-modal music emotion recognition: A new dataset, methodology and comparative analysis,” in *10th International Symposium on Computer Music Multidisciplinary Research*, 2013.
- [90] M. E. Davies and M. D. Plumbley, “Context-dependent beat tracking of musical audio,” *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 15, no. 3, pp. 1009–1020, 2007.
- [91] A. Donze, S. Libkind, S. A. Seshia, and D. Wessel, “Control improvisation with application to music,” tech. rep., DTIC Document, 2013.
- [92] B. Lévy, “Visualising omax,” *Master II ATIAM. UPMC-IRCAM*, 2009.
- [93] I. Schankler, J. B. Smith, A. R. François, and E. Chew, *Emergent formal structures of factor oracle-driven musical improvisations*. Springer, 2011.
- [94] J. Nika, J. Echeveste, M. Chemillier, and J.-L. Giavitto, “Planning human-computer improvisation,” in *International Computer Music Conference*, p. 330, 2014.

- [95] A. R. François, E. Chew, and D. Thurmond, “Visual feedback in performer-machine interaction for musical improvisation,” in *Proceedings of the 7th international conference on New interfaces for musical expression*, pp. 277–280, ACM, 2007.
- [96] P. Roy and F. Pachet, “Enforcing meter in finite-length markov sequences.” in *Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.
- [97] J. Echeveste, A. Cont, J.-L. Giavitto, and F. Jacquemard, “Operational semantics of a domain specific language for real time musician–computer interaction,” *Discrete Event Dynamic Systems*, vol. 23, no. 4, pp. 343–383, 2013.
- [98] A. Allombert, R. Marczak, M. Desainte-Catherine, P. Baltazar, L. Garnier, and B. Yeti, “Virage: Designing an interactive intermedia sequencer from users requirements and theoretical background,” in *Proceedings of the International Computer Music Conference*, 2010.
- [99] I. Akkaya, D. J. Fremont, R. Valle, A. Donzé, E. A. Lee, and S. A. Seshia, “Control improvisation with probabilistic temporal specifications,” in *Internet-of-Things Design and Implementation (IoTDI), 2016 IEEE First International Conference on*, pp. 187–198, IEEE, 2016.
- [100] R. Valle, A. Donzé, D. J. Fremont, I. Akkaya, S. A. Seshia, A. Freed, and D. Wessel, “Specification mining for machine improvisation with formal specifications,” *Computers in Entertainment (CIE)*, vol. 14, no. 3, p. 6, 2016.
- [101] S. Ewert, M. Müller, and R. B. Dannenberg, “Towards reliable partial music alignments using multiple synchronization strategies,” in *Adaptive Multimedia Retrieval. Understanding Media and Adapting to the User*, pp. 35–48, Springer, 2011.
- [102] A. Papadopoulos, F. Pachet, P. Roy, and J. Sakellariou, “Exact sampling for regular and markov constraints with belief propagation,” in *International Conference on Principles and Practice of Constraint Programming*, pp. 341–350, Springer, 2015.
- [103] A. Eigenfeldt and P. Pasquier, “Evolving structures for electronic dance music,” in *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pp. 319–326, ACM, 2013.
- [104] A. Eigenfeldt, “Generating structure–towards large-scale formal generation,” in *Tenth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2014.
- [105] E. Handelman and A. Sigler, “Artik: Articulation and dynamics from structural pattern analysis,” in *Tenth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2014.
- [106] S. Dubnov and G. Assayag, “Memex and composer duets: computer-aided composition using style mixing,” *Open Music Composers Book*, vol. 2, pp. 53–66, 2013.

- [107] S. Fothergill, H. M. Mentis, P. Kohli, and S. Nowozin, “Instructing people for training gestural interactive systems,” in *CHI* (J. A. Konstan, E. H. Chi, and K. Höök, eds.), pp. 1737–1746, ACM, 2012.
- [108] M. Ramírez-Corona, M. Osorio-Ramos, and E. F. Morales, “A non-temporal approach for gesture recognition using microsoft kinect,” in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, pp. 318–325, Springer, 2013.
- [109] M. E. Hussein, M. Torki, M. A. Gowayyed, and M. El-Saban, “Human action recognition using a temporal hierarchy of covariance descriptors on 3d joint locations,” in *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pp. 2466–2472, AAAI Press, 2013.
- [110] T. Dubnov, Z. Seldess, and S. Dubnov, “Interactive projection for aerial dance using depth sensing camera,” in *IS&T/SPIE Electronic Imaging*, pp. 901202–901202, International Society for Optics and Photonics, 2014.
- [111] J. Arias, M. Desainte-Catherine, and S. Dubnov, “Automatic construction of interactive machine improvisation scenarios from audio recordings,” in *The Fourth International Workshop on Musical Metacreation (MUME 2016)*, 2016.
- [112] C.-i. Wang and G. J. Mysore, “Structural segmentation with the variable markov oracle and boundary adjustment,” *Proceedings of ICASSP 2016, Shanghai.*, 2016.
- [113] T. Murata, “Petri nets: Properties, analysis and applications,” *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, 1989.
- [114] T. De La Hogue, J.-M. Celerier, and P. Baltazar, “Présentation d’un formalisme graphique pour l’écriture de scénarios interactifs,” in *Journées d’Informatique Musicale*, 2016.