UNIVERSITY OF CALIFORNIA, SAN DIEGO

Modeling Musical Anticipation: From the time of music to the music of time

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

in

Music

by

Arshia Cont

Committee in charge:

Shlomo Dubnov, Chair
Alain de Cheveigné
Philippe Manoury
Miller Puckette
Lawrence Saul
David Wessel

2008

The dissertation of Arshia Cont is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

_____

_____

_____
Chair

University of California, San Diego

2008

DEDICATION

تقدیم به آنکه تقدیم می کند

به پدر و مادر عزیزم

غلامحسین کنت و منصوره دانش کاظمی

Dedicated to ones who dedicate...

my parents

Mansoureh Daneshkazemi and Gholamhossen Cont

EPIGRAPH

"I think that the search for a *universal* answer to the questions raised by musical experience will never be completely fulfilled; but we know that a question raised is often more significant than the answer received. Only a reckless spirit, today, would try to give a total explanation of music, but anyone who would never pose the problem is even more reckless."

Remembering the future

LUCIANO BERIO

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

This thesis is the result of almost a decade of struggle between three continents, starting from Tehran, Iran (my hometown), to Virginia (USA), and then back and forth between Paris (France) and San Diego (USA). The main drive during this whole period has been my extreme passion for contemporary music and the science of music. Obliged to commence as an engineer, I have undergone a rather unusual pathway to reach where I stand today, along which the help of many individuals should be acknowledged.

My greatest gratitude goes to my family: My parents Gholamhossein Cont and Mansoureh Daneshkazemi, to whom I dedicate this thesis, exemplify love and sacrifice for me. In a decisive period in my life, where I was desperately seeking ways to combine my two passions for music and engineering sciences, they supported me in every possible way to initiate my migration and continued to do so up to this day. My uncle Masoud Daneshkazemi without whose early paternal support once arrived in US, I could have never arrived where I currently stand. My brother, Rama Cont, who played a crucial role in my professional and personal life in Paris and without whose initial help I would have never found my way there. Finally my whole family in Iran, including my twin sister Mandana, who accompanied me from far during this whole period.

Before the start of this PhD program, I was desperately seeking ways to connect my research to the world of contemporary music. This was initiated by my migration from Iran to US, to continue my studies at VirginiaTech and soon turned out to deceptions as research in music, even today in many of our outstanding academia, is not considered seriously! In spite of this, many individuals helped me to realize this important passage in my life. Legendary figures such as Miller Puckette and David Wessel never ceased to reply to my requests while I was only a desperate undergraduate student. I shall never forget their exemplary modesty and willingness for development of younger generations, which was crucial in my early decisions and surely enough for many others like me. Dr. David A. de Wolf of VirginiaTech's ECE department helped and encouraged me to learn and study music signal processing early on through in-

dependent research programs. After obtaining my bachelors, almost coincidentally I ended up in the ATIAM masters at IRCAM, for which I am grateful to Gilbert Nouno and Gérard Assayag of IRCAM. The intensity and richness of the program, as well as its integration within one of the most important research and production centers in the world for computer music put me in a position which was simply a distant dream since my childhood.

This thesis has been realized as a collaboration between IRCAM and UCSD, and therefore people who should be greeted for it are twice as usual. During my almost two years residency at UCSD's music department I was introduced to the professional world of contemporary music practices and ideas, for which I am grateful to the whole body of music department's faculties and students. To my knowledge, UCSD is among few places in the world to maintain a utopian image of artistic creation and ideas in its music department and I was lucky to be part of it. Two individuals directly and indirectly influenced my thoughts in San Diego: Miller Puckette, deeply influenced my image of professionalism and the field of computer music. His modesty, despite his sculptural figure in the field, his straightforwardness and honesty of expressions, sometimes harsh in spite of himself, helped me clarify my position in the field. And Shlomo Dubnov, with his extreme courage for abstract and difficult problems, has been thoroughly inspiring in the course of this work.

Since I was 15, IRCAM has been a utopian dream place for me and the unique place in the world as a culmination of artistic and scientific collaboration. I am very grateful to the *Real Time Musical Interactions* team for hosting me through several projects during this whole period. An extreme shift in my career occurred when Frank Madlener, the general director of IRCAM, and Alain Jacquinot, director of productions, offered me to collaborate on artistic projects in early 2007. Through these projects I experienced the gap between the world of research, artistic creation, and production, and learned to fill it in through the collaborative environment that IRCAM is famous to provide. I was lucky enough to collaborate with Marco Stroppa, resulting in one of the most important works I have realized up to this date that figures chapters 7 and 8 of

this manuscript. IRCAM brings in a huge body of talented professionals in various field whose presence have in one way or another contributed to this work: Gérard Assayag who generously shared his work for this project, Gilbert Nouno, Olivier Pasquet, Rémy Muller, Nicolas Rasamimanana and more.

Besides formal collaborations, several individuals have provided me the mental force and courage to undertake this rather abstract project: Composer Philippe Manoury, through his never ending pleas for advances in computer music research and his extreme ability for integration of such advances in a compositional setting. Composer Marco Stroppa, who never gives up his utopian image of composer/thinker and engagement in the field of computer music, motivated me intellectually all along my work. And at large, to all composers of music today who maintain our image of *"wayfarers"* and with whom I got the opportunity to work with: Pierre Boulez, Jonathan Harvey, and more. Researcher and artist David Wessel has played a paternal role in the course of this PhD project to whom I am most grateful. Wessel's active role and his openness to issues makes him one of my idols in the field of computer music. Finally, to the whole body of researchers and artists at IRCAM and most importantly to the hidden actors of this scene that maintain the synergy alive: Hugues Vinet, Frank Madlener and Alain Jacquinot.

Last but not least, my greatest gratitude with love, goes to Marie Vandenbussche who accompanied me in every step of this project despite intervals of geographical separations. She has traced every word and bit of work that is represented in this manuscript and I am most in debt to her priceless patience and caring love.

Arshia Cont

Paris, October 2008.

VITA

2003	B.S. Electrical Engineering, Virginia Tech.
2003	B.S. Mathematics, Virginia Tech. *(dual major)*
2004	M.S. Acoustics, Signal Processing and Computer Science Applied to Music (ATIAM). University of Paris 6/Ircam, Paris, France.
2008	Ph.D., Music University of California, San Diego, California.

PUBLICATIONS

A coupled audio/tempo model for real-time alignment of polyphonic audio to music score. Arshia Cont. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2008 (submitted).

Anticipatory Model of Musical Style Imitation using Collaborative and Competitive Reinforcement Learning. Arshia Cont, Shlomo Dubnov and Gerard Assayag. In Anticipatory Behavior in Adaptive Learning Systems, Butz et al. (Ed.), pages 285–306, Springer Verlag, LNAI 4520, June, 2007.

ANTESCOFO: Anticipatory Synchronization and Control of Interactive Parameters in Computer Music. Arshia Cont. In Proceedings of International Computer Music Conference (ICMC), Belfast, Ireland, August 2008.

Audio Oracle: A New Algorithm for Fast Learning of Audio Structures. Shlomo Dubnov, Gerard Assayag and Arshia Cont. In Proceedings of International Computer Music Conference (ICMC), September 2007.

GUIDAGE: A Fast Audio Query Guided Assemblage. Arshia Cont, Shlomo Dubnov and Gerard Assayag. In Proceedings of International Computer Music Conference (ICMC), September 2007.

Realtime Multiple Pitch Observation using Sparse Non-negative Constraints. Arshia Cont. In Proceedings of International Symposium on Music Information Retrieval (ISMIR), October 2006.

Realtime Audio to Score Alignment for Polyphonic Music Instruments Using Sparse Non-negative constraints and Hierarchical HMMs. Arshia Cont. In IEEE International Conference in Acoustics and Speech Signal Processing (ICASSP), May 2006.

ABSTRACT OF THE DISSERTATION

Modeling Musical Anticipation: From the time of music to the music of time

by

Arshia Cont

Doctor of Philosophy in Music

University of California, San Diego, 2008

Shlomo Dubnov, Chair

This thesis studies *musical anticipation*, both as a process and design principle for applications in music information retrieval and computer music. For this study, we reverse the problem of *modeling anticipation* addressed mostly in music cognition literature for the study of musical behavior, to *anticipatory modeling*, a cognitive design principle for modeling artificial systems. We propose anticipatory models and applications concerning three main preoccupations of expectation: *"What to expect?", "How to expect?"* and *"When to expect?"*. For the first question, we introduce a mathematical framework for *music information geometry* combining information theory, differential geometry, and statistical learning, with the aim of representing information content, and gaining access to music structures. The second question is addressed as a machine learning planning problem in an environment, where interactive learning methods are employed on parallel agents to learn anticipatory profiles of actions to be used for decision making. To address the third question, we provide a novel anticipatory design for the problem of synchronizing a live performer to a pre-written music score, leading to *Antescofo*,

a preliminary tool for writing of time and interaction in computer music. Common to the variety of topics presented in this thesis is the anticipatory design concept with the following premises: that an anticipatory design can reduce the structural and computational complexity of modeling, and helps address complex problems in computational aesthetics and most importantly computer music.

Chapter 1

Introduction

The moment we step into a concert hall or start playing music on our favorite stereo (or multi-channel) system, our *expectations* about the piece of music come into play. Today, the role of music expectation in all its forms is widely acknowledged for forming affective behavior related to psychological and even physiological limbic responses in our brain and body. Many musical affects such as the sense of *surprise*, *laughter*, *frisson*, *tension* and more, have their roots in how our *expectations* have been treated with regard to a present musical situation. In other words, expectation implies some sort of mental representation in which our daily musical experience is constantly being examined and updated. These mental representations can come from vast and unrelated number of sources: cultural background, music genres, music schemas, veridical sources and conscious extra-musical attributions. Hanslick (1854) was among the first advocates of an intellectual view of aesthetic pleasure with regard to music where he emphasized the importance of the fulfillment (or not) of listeners' anticipation, with the tradition being continued to this date culminating in works such as that of Huron (2006) on the psychology of music expectation.

The role of musical expectations is not all about emotions and physiological responses but also extends to social and political factors. Adorno (1941) is among the first to warn about the danger of *standardization* of our expectations with regards to

popular music, a point further expanded in a politico-economic stand by Attali stating that "[music] styles and organization are ahead of the rest of the society" and declaring music as both a mirror and prophecy of our society (Attali, 1985, Ch. 1).

The role of expectation pertaining to music is not only limited to the act of listening but also plays a central role in the act of creation, to the point where Meyer (1956) in his seminal book *Emotion and Meaning in Music* drew attention to the importance of expectation, arguing that the principal emotional content of music arises though the composer's choreographing of expectation. Explicit cases of this are beautifully examined and analyzed in Huron's late book *sweet anticipation* (Huron, 2006). On a more musical stance, composer Gérard Grisey in his seminal paper *"Tempus ex Machina: A composer's reflections on musical time"*, puts forward a rather mystical aspect of musical time which he calls the *skin of time*:

> "We left the areas where the composer's action still remained effective (the skeleton of time) to arrive gradually at the point where his actions as becoming more circumspect, more cautious (the flesh of time). This action touches mainly on the areas of investigations by psychoacousticians and sociologists. How does the listener organize and structure the complexity of a sound? How does his memory choose what we perceive? What roles do his culture and musical education play in his choice? In what time does this listener live and breathe? So many questions which I will not try to answer here, since they seem to me to belong more to the sociologist's or psychologist's area of research than the empirical reflections of a composer" (Grisey, 1987).

One of the main goals of this thesis is to provide computational answers to Grisey's pleas inspired by research in music cognition, with a strong aim for approaching a degree of access to the *skin of time* for musical and artistic purposes; hence going from the the time of music to a music of time. To this end, by "time of music" we designate parameters and models useful to describe the temporal flow of music, and by "music of time" we imply a more active and creative process by which the temporal structure is harnessed for musical effect.

Despite its significance, musical expectation has not been given a role worthy of its cognitive importance in existing computational approaches to computer music and

music information retrieval, which mostly favor prediction-driven architectures. This thesis at its inception brought in the idea to study expectation processes in action and provide computational frameworks within which it could find life in sound and music computing systems, whether for analysis, information processing or generation; hence the idea of *modeling musical anticipation* where the term *anticipation* seems to be even more vague than the term expectation.

1.1 Approach

At the onset of this thesis, we are ought to clarify the notion of *anticipation* and present it within a coherent computational context. Literally, anticipation has close ties with both prediction and expectation. In many computational models of music, modeling is mainly on predictions based on gained knowledge from the past and present within an environment. Prediction, or the act of forecasting, is only a subproduct of conscious expectation and not the whole story. Throughout this thesis, anticipation is defined as a marriage of expectation and actions of a cognitive system in its environment. To clarify matters such as this, we start the thesis by a voyage from the world of music cognition centered around the psychology of expectation, to the world of computational design. We clearly define *anticipation* and differentiate its meaning from that of prediction and expectation in section 2.2. In our survey in chapter 2, we study models for *musical anticipation* within music cognition literature where the issue is treated as *natural science*, and shift our view in the following chapter to *anticipatory modeling*, more akin to what Simon (1969) calls *artificial sciences* in that modeling is a design process that addresses cognitive behavior. Through this process, we contemplate on the notion of *modeling* and pertaining to musical anticipation. Studying evidence and models in the cognition literature on musical expectations, we question the legitimacy of an aim to *model* musical anticipation. After this investigation we change direction by focusing on models that underlie anticipatory behavior rather than claiming a universal model for musical anticipation. Within this perspective, anticipation is defined and presented as a

design principle for complex musical systems. This underlies our voyage from *modeling anticipation* to *anticipatory modeling*. We define the latter as a cognitively-inspired design process to achieve complex anticipatory behavior reported in the music cognition literature.

Upon this shift of views, the goal of this thesis is to provide *anticipatory musical models* that address several important and evident aspects of musical anticipation. We study anticipatory modeling from three different perspectives, addressing three main goals of anticipatory behavior, in three different parts of this thesis. In each part we raise a different problem and propose a different design. Each part addresses a different literature with its own history and application range. The unifying factor in all disjoint parts of this thesis would then be *anticipatory modeling*, supporting our central claim that through explicit consideration of anticipation in design models complex behavior emerges with low cost and relatively simple designs.

This thesis inscribes itself in the domains of *Computer Music* research. Computer music, to be simplistic, is an interdisciplinary field and the culmination of research and artistic creation. Its aim is to bring in our knowledge of *science of music*, and other achievements in other fields, to the world of artistic creation and particularly to music with the aid of computers. It would be overly limiting to associate computer music solely with *music technology*. Like any other field of science, music brings in its own complexity and horizons into question that comes into harmony with many problems to be solved within other fields. Therefore, computer music is not a mere application of other engineering findings within the field of music. On the contrary, we strongly believe that the complex nature of music could be a key to solve many problems in other fields such as language and cognition. Our aim, therefore, is to address and provide evidence that our approach can enhance our understanding of several questions in the field of computer music research. An ultimate goal of this thesis is to refine our current views on the complexity of interactive environments for computer music and destined for artistic creation. As a consequence, one of the main concerns of this thesis is *real-time* and *online* systems that can both address cognitive complexity and also find use within the

interactive arts communities. Therefore, all the models and systems presented in this thesis are oriented towards real time use, and run incrementally in time.

1.2 Organization

This thesis is organized into four main parts. Part I constitutes our *modeling investigations* and voyage from *modeling anticipation* to *anticipatory modeling*. We start this part by reviewing evidence and different views of anticipation pertaining to the music cognition and computer music literatures. We confront the two problems of *modeling anticipation* and *anticipatory modeling* separately in chapter 2 and 3. Within this part, we reverse the problem of modeling musical anticipation to that of anticipatory design as a cognitively-inspired design process that exhibits anticipatory behavior reported in the literature. We define *anticipatory modeling* as a design process that explicitly considers anticipatory behavior. Both notions of anticipation and anticipatory modeling are defined throughout the two chapters. Chapter 3 clarifies the main methodologies, claims and design processes that will be explored in the following chapters. Within this framework we emphasize on three main premises of anticipatory modeling as: *Information Access*, *Interactive and Online Learning*, and *Multimodal Interactions*. In what follows, we demonstrate anticipatory musical systems that address three main preoccupations of expectation within three main questions: *"What to expect?"*, *"How to expect?"*, and *"When to expect?"*.

Part II aims at addressing the *"what?"* question of expectation which entails representing and qualifying relevant musical information. We aim at quantification and qualification of the amount of information in a sequence of music signals. With this aim, we present a novel framework based on information geometry of statistical manifolds combining various literatures such as differential geometry, statistical learning and signal processing. We expose the preliminary mathematical framework in chapter 4 and introduce its applications to music or *music information geometry* in chapter 5. In chapter 4, we revisit concepts commonly used in music information retrieval literature such

as divergences, similarity and segmentation from an information geometric standpoint, and provide a framework for quantifying music information on the fly. This theoretical framework provides *access* to musical information and paves the way for two concrete algorithms introduced in chapter 5 entitled *methods of information access*. Therein, we introduce two general algorithms providing access to two different aspects of musical information and examine them within different common applications in computer music and music information retrieval systems: automatic audio structure discovery, concatenative synthesis, and query-by-example over large databases of audio and more.

Part III elaborates our second concern on *"how to expect"*, formulated as a machine learning planning problem. Using methods of information access introduced in the preceding part we introduce an adaptive and interactive learning method that learns goals and behaviors through interactions with a constantly changing environment. The goal of this framework, presented in chapter 6, is to grasp environmental and behavioral regularities and provide them as long-term strategies once used for generation or visualization purposes. We show that the presented algorithm achieves behavioral learning through knowledge-based interactions and distinguishes itself from existing systems by its relative simplicity, computational efficiency, and achieving complex behavior when little data is available.

In part IV, we address the question of *"when to expect"* in the context of the common problem of synchronization of a live performance to a symbolic score in computer music and multimedia art literature. This problem, commonly known as score following or audio to score alignment, has been widely studied in the computer music literature. In chapter 7, we introduce a novel anticipatory design based on coupled concurrent and parallel agents that reduces computational complexity and increases performance precision as shown throughout the chapter. We then use this proposal in chapter 8 and show how the availability of structures and information in an anticipatory architecture enables important first steps in *writing of time and interaction* in computer music. The implementation of the system proposed in chapter 8, called *Antescofo*, has been used in contemporary computer music productions at IRCAM and has seen performances

worldwide.

1.3 Contributions

Contributions of this thesis to the literature can be summarized as follows:

- A formal of definition of *anticipation* destined for computational models of sound and music.
- A formal definition of *anticipatory design* inspired by music cognition and pertaining to music processing systems.
- A computational framework for quantification and qualification of music information and content based on *Music Information Geometry* (chapter 4).
- An online algorithm for incremental clustering and structure discovery of music signals (section 5.1).
- A fast and online algorithm for unit selection over large databases of music based on users' audio query with applications to content retrieval and concatenative synthesis (section 5.2).
- An online adaptive and interactive learning framework achieving anticipatory planning strategies in a constantly changing environment, based on *Active Learning* and without human interventions (chapter 6).
- A computational anticipatory framework for automatic style imitation and improvisation on symbolic music signals (chapter 6) achieving formal long-term behavior.
- An anticipatory design for real-time audio to score alignment featuring coupled audio/tempo agents and capable of decoding real-time position as well as tempo of the performer for polyphonic music signals (chapter 7).

- A preliminary framework and language for writing of time and interaction destined for interactive mixed instrumental and live computer music repertoires (chapter 8).

Part I

From
Modeling Anticipation
to
Anticipatory Modeling

Chapter 2

Modeling Musical Anticipation

At the onset of this chapter, three notions of *prediction*, *expectation* and *anticipation* must be clearly defined in a computational context. Prediction is an act of forecasting. But the other two, even taking linguistic differences into account, have been used interexchangeably in the music cognition literature with much more emphasis and clarity on expectation. Huron (2006) defines expectation as "a form of mental or corporeal belief that some event or class of events is likely to happen in the future" or as a rather general cognitive process, and *anticipation* as a sub-product of expectation when the sense of appraisal for the expected future event is high (Huron, 2006, Ch. 15). Huron's *anticipation* is equivalent to what Bharucha calls *yearning* (Bharucha, 1996) or music theorist Narmour (1990) has termed *implicative*. Other literature in music cognition does not help either. For example, Schmuckler (1997) defines expectation as an "anticipation of upcoming events based on information from past and present".

This ambiguity is not brought up there to question the scholarship of any of the cited works but in an attempt to position ourselves at the onset with the difference and difficulties in defining anticipation in a context useful for computational modeling. But before attempting any definition, we must look at the rich literature of the psychology of musical expectations, underlining key facts and pieces of evidence used throughout this thesis and paving the way for a formal definition of anticipation.

2.1 Psychology of musical expectation

In this section we briefly look at the psychology of musical expectation, presented as a "natural science" approach to the perception and cognition of music as pursued by many researchers to model musical expectation in an effort to explain musical behavior. We begin this review by presenting several areas of experimental research in cognitive musicology with regard to expectation, with an aim of emphasizing the role of expectation in many aspects of music research. In sections 2.1.2 and onwards, we look more closely at some important concepts taken out of the literature which strongly inspire the modeling premises and definitions presented in section 2.2 and used throughout our proposals following part II of this thesis.

2.1.1 Experimental Research Scopes

Research on the psychology of musical expectation is built on both experimental and theoretical frameworks where the latter is most often the fruit of the former. In this thesis we are concerned with theoretical implications of musical expectations in order to motivate design principles that will be introduced in the following chapter. Due to the centrality of musical expectation in the perception and cognition of music, the range of experiments is vast and touches many key problems in music cognition. Here, we briefly introduce some of the experimental setups in music perception literature dealing with the issue to motivate the upcoming theoretical aspects. For more details the reader is referred to (Huron, 2006, Chapter 3).

An important area of research dealing with expectancy is studies on systematic variation in listeners' judgements when exposed to a musical continuum. The main finding of these studies is that expectations are predictable from various music theoretic and perceptual/cognitive principles of pattern classification (Schmuckler, 1990). Another similar research paradigm is processing and encoding of music information among listeners. For example Bharucha and Stoeckig (1986) showed that a target chord is responded to more quickly and accurately following a harmonically related prime

chord, compared to on preceded by a harmonically unrelated chord.

The effect of musical expectation in production and performance of music information has also been a subject of experimental research. For example Carlsen (1981) had listeners sing in response to a continuation of a two-note interval. The time interval between the two notes as well as the response interval were analyzed for their frequency of occurrence and the context interval. Results demonstrate that response intervals vary with context.In another experiment, Schmuckler (1989) undertook experiments on pianists and had them complete different melodic and harmonic-context contours. He found that performers' production mirrored expectancy judgements. One criticism of works done in this field is that the majority of these works study single-event anticipation in a sequence rather than multi-events. Despite this limitation, these experiments underly the importance of musical expectation in many musical activities ranging from performance to listeners' judgments.

Researchers working on musical memory have also contributed to the debates and experiments on music expectation. The majority of research in this field has been dedicated to the importance of tonality in musical memory specially in long-term memory structures. One candidate model is that expectancy and memory are positively correlated; that is, high-expectancy information is better remembered (Bartlett, 1932). Another model is the *Restorff effect* stating that isolating items from background enhances learning and thus leads to faster access and recall in the memory (Schmuckler, 1997).

The scopes of musical expectation in experimental research is vast and what came above is just a glimpse of an ongoing effort in the research community. Besides revealing the importance of expectation in many aspects of music perception, such efforts have helped the emergence of various theories of auditory perception whose key facets are studied in the coming sections.

2.1.2 Auditory Learning

In the music cognition literature, there is still ongoing debate on innate or learned nature of many auditory phenomena. Huron shows that nature does not have this preoccupation. From a biological perspective, there is a clear criterion for which it is best for a behavior to be instinctive and when it is best for the behavior to be learned. The determining factor is the *stability* of the environment. When there is little environmental change, conditions favor instinctive or innate behavior which are usually fast and effective. On the contrary, when the environment changes quickly it is best to learn. Therefore, the difference between instinctive and learned behavior is not that the first is *genetic* and the second *environmental*. Contrary to intuition, learning involves more genetic machinary than do instinctive behaviors and instincts reflect a longer and more profound interaction with the environment than does learning. Therefore, the process of learning is just as much a product of evolution by natural selection as any pre-wired instinct. This evolved capacity to learn is referred to as the *Baldwin Effect* (See Huron, 2006, Ch. 4).

But how does auditory learning occur? Over the past half century, experimental research has shown that we are sensitive to the frequency of various stimuli in our environments. An example of such research has led to the *Hick-Hyman law*, which shows that the reaction-time responses to known and unknown stimuli follow an orderly (logarithmic) law. Said in other words: perception is more efficient for expected stimuli than for unexpected stimuli.

One of the most important discoveries to this end in auditory learning has been that listeners are sensitive to the probabilities (or contingent frequencies) of different sound events and patterns, and that these probabilities are used to form expectations about the future. In other words, auditory learning is shaped by the frequency of occurrence of individual stimuli and groups of stimuli. An important landmark evidence for this is the work of Saffran et al. (1999) at the University of Rochester. In one of their experiments, they constructed small musical vocabularies consisting of 3-note figures.

Using these figures, they constructed a long (seven minute) sequence that consisted of a random selection of six figures. Newborn infant were exposed to a continuous succession of tones for 21 minutes. After this exposure, they were newly exposed to 4 different 3-note sequences and their familiarity ranks were checked (using the head-turning effects). As a result, infants correctly identified three-note sequences they had been exposed to. The conclusion was that listeners were simply cuing on simple *statistical* properties of various tone sequences. Saffran's experiment has been recently replicated by Loui et al. (2006) on adult subjects and using an uncommon tuning system (to avoid veridical and schematic expectations on familiar patterns among adult listeners), leading to similar observations. Similar observations have also been reported for reproduction of rhythmic patterns among westerners matching the actual distribution of such rhythms in western music (Sadakata, 2006).

An important property of auditory learning is that learned mechanisms can be fallible but still useful. In other words, what happens in the brain is a statistical approximation of the outside world and not the thing itself (See Huron, 2006, Ch. 4). This is in parts due to how sounds and music patterns are mentally represented in our brain which is our next topic.

2.1.3 Concurrent and Competitive Representations

Following observations in the previous section, one might ask about the *information contents* of mental representations on which contingencies or statistics are made. The brain does not store sounds. Instead, it interprets, distills and represent sounds[1]. It is suggested that brain uses a combination of several underlying presentations for musical attributes. A good mental representation would be one that captures or approximates some useful organizational property of an animal's actual environment (Huron, 2006, Ch. 7).

[1]This point of view is also supported by the *Enactive* view of cognition whose work is much concentrated on visual perception by their rejection of the *snapshot* photographer view of visual perception in favor of dynamic flows of continuously varying retinal information (Noë, 2004).

But how does the brain know which representation to use? Huron suggests that expectation plays a major role. There is good evidence for a system of rewards and punishments that evaluates the accuracy of our unconscious predictions about the world. Our mental representations are being perpetually tested by their ability to usefully predict ensuing events, suggesting that *competing and concurrent representations* may be the norm in mental functioning (Huron, 2006). This view is strongly supported by the neural Darwinism theory of Edelman (1987). According to this theory, representations compete with each other according to Darwinian principles applied to neural selection. Such neural competition is possible only if more than one representation exists in the brain. In treating different representations and their expectation, each listener will have a distinctive listening history in which some representations have proved more successful than others.

2.1.4 Mental Representations of Expectation

According to Huron, memory does not serve for recall but for *preparation*. He addresses the structure rather than content of mental representations and introduces a taxonomy for auditory memory that constitutes at least four sources of musical expectations as follows (Huron, 2006, Ch. 12):

Veridical Expectation: Episodic Memory is an explicit memory and a sort of autobiographical memory that holds specific historical events from our past. Episodic memory is easily distorted and in fact, the distortion occurs through repeated retelling or recollection. Most importantly, our memories for familiar musical works are episodic memories that have lost most of their autobiographical history while retaining their accuracy. This sense of familiarity or expectation of familiar works is refereed to, by Huron (2006) and Bharucha (1993), as *Veridical expectation.*

Schematic Expectation: This type of expectation is associated with semantic memory; another type of explicit memory which holds only declarative knowledge and is

distinguished from episodic memory by the fact that it does not associate the knowledge to any historical past but as stand-alone knowledge. This kind of memory is most active in first-exposure listening where our past observations and learned schemas are generalized. These sort of auditory generalizations are reminiscent of the learned categories characteristic of semantic memory.

Dynamic Adaptive Expectation: Expectation associated with short-term memory is *Dynamic Adaptive Expectation*. It occurs when events do not conform with expectations that have been formed in the course of listening to the work itself. These expectations are updated in real time especially during exposure to a novel auditory experience such as hearing a musical work for the first time. Patterns of repetition, form, and motivic structure are among musical phenomena that are linked to dynamic expectancy.

Conscious Expectation: All the three types of expectations discussed above are unconscious in origin. Another important class of expectations arise from conscious reflection and prediction. Such explicit knowledge might come from external sources of information (such as program notes) or as part of a listener's musical expertise, or even arise dynamically while listening to a novel musical work. An argument for the last type, and most important for this work, is the perception of musical form during listening.

All these expectation schemes operate concurrently and in parallel. Schematic expectations are omnipresent in all of our listening experiences. When listening to a familiar work, the dynamic-adaptive system remains at work – even though the veridical expectation anticipates exactly what to expect. Similarly, when listening for the first time to an unfamiliar work, the veridical system is constantly searching for a match with familiar works. The veridical system is essential for catching the rare moments of musical quotation or allusion. In short, an anticipatory effect such as *surprise* is a result of various types of interactions among these lower-level components of music expectation cognition. For a thorough discussion see (Huron, 2006, Ch. 12).

2.2 Anticipation Defined

Expectations are more than mere *representations* of our beliefs and their existence are coupled with their consequent *actions* and effects. We saw earlier that expectations entail mental representations, whether partial, accurate or fallible. They are learned through interactions with a stable or unstable surrounding environment, and entertain our very acts of perception (through evoking attention, appraisal or emotions as evoked in (Huron, 2006)). In this schema, it would be simplistic to distinguish any expectancy from its consecutive effects. This is in fact at the core of every experimental study of expectancies (section 2.1.1). In other words, expectations lead to *predictions* which by themselves evoke *actions*, either physiological, mental, or physical, that in return of their outcome affect our beliefs and expectations. We study this *activism* aspect of cognition pertaining to expectation under the term *anticipation*. With this introduction, we provide the following definition for anticipation:

Definition 2.1. *Anticipation* is an *action*, that a system takes as a result of *prediction*, based on current belief or expectations, including actions on its own internal state or belief.

We now refine this definition from different perspectives: that of expectation, enaction and computational design.

2.2.1 Anticipation in view of Expectation

Expectation viewed as a cognitive process is intertwined with both biology and culture (Huron, 2006, Ch. 1). It is a biological adaptation with specialized physiological structures and the capacity to form accurate expectation is important for survival among all species. Culture provides preeminent environments where many expectations are acquired and assessed. When it comes to music, the context for predicting future context is dominated by the latter but does not exclude the former. From a biological perspective, the purpose of expectation is to prepare an organism for the future. It is through this

preparatory physiological process that we experience arousal or attentional responses (or the reverse, reduced or inhibit responsiveness). From a phenomenological perspective, the most interesting property of expectation is that it evokes feelings and emotions. Huron proposes that emotions are evoked by expectation involving five functionally distinct physiological systems that underly his *ITPRA* theory: imagination, tension, prediction, reaction, and appraisal (Huron, 2006, Ch. 1). Each of these systems can evoke responses independently involving both physiological and psychological changes.

The picture to retain here is that mental representations evoking expectations are *adaptive* and in constant *interaction* with the living environment. Expectations imply some sort of mental representation or beliefs adapted to the surrounding environment. These mental representations can be partial and fallible. In all cases, these mental beliefs are in constant *interaction* with the living environment evoking physiological or psychological *actions*. Without these interactions, which reflect stabilities and instabilities of our environment, neither can expectations exist nor can we survive as living beings. Therefore *anticipation* as defined above, not only constitutes the outcome of expectation but also expectations themselves, how they are formed, their contents and whether they exist.

2.2.2 Anticipation in view of Enaction

If expectations are determined by the *actions* coupled with the environment, there is essentially no need to separate the *representation* from the *action*. In other words, *perception* is not a process in the brain but a kind of skillful activity and a mode of exploration of the environment drawing on implicit understanding of environmental regularities as depicted in section 2.1.2. This view is shared by the *Enactive* school of cognition (Varela et al., 1992; Noë, 2004). Such a view of cognition implies that perception is an activity of sensorimotor coupling with the environment, advocates the *marriage* of *action* and *perception* in cognitive studies, and rejects the idea widespread in both philosophy and science that perception is a process in the brain whereby the per-

ceptual system constructs an internal representation of the world. Within this marriage of action and perception, the (internal) representation is coupled with the *sensorimotor actions* which for pure listening takes the form of *conceptual understanding* (as developed in (Noë, 2004, Ch. 6)). Therefore, the perceptual presence in absence of accurate mental representations is assessed by an *access* controlled by patterns of expectational dependence with which we are familiar.

In other words, although our *mental representations* are rooted in empirical observations, we produce them by means of our active cognitive apparatus instead of passively processing the structure of the world. Perception is a kind of skillful activity on the part of an animal as a whole, and constitute active explorations of its environment. Thus representation consists of future potentialities of interaction with an environment. The connection of the enactive school of cognition to our work will be further developed in chapter 3.

2.2.3 Anticipation in view of Computation

The ideas presented above have been exploited in a computational perspective. In fact, one of the main advantages of definition 2.1 is that it prepares the ground for a computational framework where the study of the interaction between a system's belief of itself or its environment with itself becomes possible. The view is shared by an emerging field in artificial intelligence and robotics literature on *anticipatory systems* (Rosen, 1985, for original definition). Consideration of *anticipation* in a computational framework within this definition will be presented and detailed in chapter 3.

Studies of *anticipation* along the lines of definition 2.1 imply explicit consideration and study of expectation. We therefore continue our review of key elements of the expectancy literature helpful for modeling anticipation.

2.3 Models of Musical Expectation

By modeling musical expectation, researchers aim mostly at a better understanding of the neural correlates in the brain and/or to depict and assess a theory regarding one among many aspects of musical expectation. In section 2.1.1 we briefly covered some experiments undertaken on several aspects of music expectation. In this section, we survey several key theories and models with the aim of a better understanding of the governing process of musical expectation.

We survey these models in three distinct groups. The first group consists of researchers mostly emerged from the field of cognitive musicology and music theory whose main goal in modeling is to assess a musical theory with less attention to a generative theory of music expectation. This first group has mostly focused on analysis of music scores with a top-down approach, assessing music theoretical rules that govern some aspects of musical expectation. The second group are researchers who chose a bottom-up approach towards the problem and use artificial learning algorithms, to automatically learn the abstract behavior in listeners' expectations without much a priori knowledge of governing rules. The third group has emerged from the field of information theory whose focus is on a bottom-up approach with a generative design in mind. Neither of the models presented here have yet found applications among composers or computer musicians. A fourth strongly related research direction is the ongoing work on automatic improvisation and style imitation with strong artistic involvements. We leave a thorough review of this last group to part III of this thesis, since the notion of expectation or anticipation is not explicit in that literature despite their strong relevance to the problem.

The survey below is by no means comprehensive. As an example, there is much interesting work done on cross cultural aspects of musical expectation (e.g. see Eerola, 2003) which we do not discuss here. Moreover, the number of theories and models not included in this chapter are many. The main goal of this review is to shed light on the concept of *modeling musical expectation* itself and prepare the main thesis of this work

on *anticipatory modeling*.

2.3.1 Music Theoretic Models

Narmour's Implication-Realization Model

Narmour (1992, 1990) introduced a theory of melody that distinguishes two kinds of melodic situations: those that are *implicative*, evoking a strong sense of predictability, and those that are *non-implicative*. The theory attempts to describe what listeners expect when the musical context is strongly implicative or in other works, how implicative intervals set up expectations for certain realizations to follow. Narmour (1990) proposes that expectations result from both bottom-up and top-down processes. Bottom-up processes are independent of a priori knowledge and include principles relating to the size and direction of a melodic process. Top-down processes incorporate experience as well as the particular history of the piece as it is heard. Narmour's theory inspired a great deal of perceptual experimentation on the subject. It has been shown that the Implication-Realization model of Narmour conform well to listener behavior (Cuddy and Lunny, 1995) and later studies by Schellenberg (1997) and von Hippel (2000) showed that the theory could be simplified to just two principles: pitch proximity and post-skip reversal. On the music theoretic side, Narmour's model was further extended by Lerdahl (2001)'s *Tonal Pitch Space* theory which added stability and mobility factors.

Margulis' Melodic Expectation Model

In Lerdahl's *Tonal Pitch Space* theory, interestingly, some computational frameworks are provided for quantification of musical tension and other perceptual phenomena, whose parameters are deduced through other quantified and structural material defined in (Lerdahl, 2001). Margulis further extended Narmour and Lerdahl's theories in a way that "enables the specification of meaningful connections to listener experience, rather than employ strings of symbols or draw metaphoric relations" (Margulis,

2005). The type of expectation chosen for modeling in Margulis' work is referred to as *deeply schematic* within Bharucha's taxonomy of veridical and schematic expectation. In the view of Huron's taxonomy of expectation structures represented in section 2.1.4, Margulis models the *dynamic adaptive expectation* of melodies based on various pitch information sources. Since Margulis' model is a recent result of the culmination of two (and more) sources of literature regarding expectancy models in the music theoretic sense, we describe an outline of the model for better understanding of our position regarding modeling expectation and anticipation.

Margulis' model considers four main parameters: *stability* (s), *proximity* (p), *direction* (d), *mobility* (m) and their hierarchical implementations. Before hierarchical considerations, all the four parameters are quantitatively defined through numerical tables. For example, the value of s is taken from a table assigning anchoring strength in a tonal context. Proximity (p) captures the intuition that listeners expect subsequent events to be relatively proximate to preceding ones. Direction (d) is a direct outcome of Narmour's theory describing post-skip reversals for large interval melodic moves, assigning continuation or reversal quantitatively. A mobility factor m further overcomes problems with local repetitions where its value is changed from 1 to $2/3$. For all parameters, these quantifications are gathered through interpretation of results of existing experiments and primarily by consulting "intuition to try to trace sensation of tension to the originating expectancies" (Margulis, 2005). Within this framework, Margulis shows that a pitch x is expected to follow a pitch y by an amount z quantified by $z = s \times p \times m + d$ before hierarchical considerations. In order to consider long-term structural dependencies, she makes use of Lerdahl and Jackendoff (1983) rules to obtain hierarchical segmentations of the melodic structure with an additional rule. The idea here is to calculate the expectancy measure introduced above (z) for different hierarchies and consider them all together through a weighting process. The weights ω_i are also assigned through some fixed rules (e.g. note-to-note ratings receive a weight of 15 and no level with time-span longer than 6 seconds is allowed). Given this measure of expectancy, Margulis provides connections with listeners experience summarized in table 2.1 where E_m and E_r are

respectively the amount by which the maximally expected pitch was expected, and the amount by which the actual realization was expected, both described in (Lerdahl, 2001).

Table 2.1: Summary of Margulis (2005) Melodic Expectation Model

Tension Type	Expectancy Source	Evaluator
Surprise	inverse to expectancy rating	$1/\left[\frac{\sum\omega_i[(s_i\times p_i\times m_i)+d_i]}{\sum\omega_i}\right]$
Denial	Proportional to implicative denial	$E_m - E_r$
Expectancy	Most-expected continuation	E_m of next event

2.3.2 Automatic Learning Models

Implicit Learning of Bharucha and Tillmann

While many expectation models rely on a certain theoretically eligible model for music (e.g. tonality), these models are questionable in their universality and partial ignorance of auditory learning principles. This comes partly from the fact that expectation is based on past experience and exposure. In the case of tonal music, despite its complexity, sensitivity to musical structure does not require explicit learning. In other words, musically naive listeners are constantly exposed in everyday life to the regularities underlying the music of their culture and thus, they acquire the knowledge implicitly.

In 1987, Bharucha introduced a neural network system based on a musical hierarchy that would learn tonality and react to unseen situations. The main advantage in using a neural network architecture, besides simulating neuronal system, is that they generalize their knowledge domain. His system showed promising results while being tested on tonal music structures and demonstrated the hierarchical structure of tonal music learning. However, his model was based on music theoretic constraints; neither the connections nor their weights resulted from a learning process. In this respect, his model represented the idealized *end state* of an implicit learning. Tillmann et al. (2000)

took this work a step ahead, by achieving a Self Organizing Map (SOM) architecture that would gain an implicit knowledge of western pitch regularities through a *passive* exposure to musical exemplars. In their experiments, they used a hierarchical SOM in which the first layer is tuned to octave equivalent pitch classes where the second and third layers would learn to specialize in the detection of chords and keys respectively. After learning, they tested the system for similarity ratings, recognition memory, harmonic expectation, perceiving key and modulations, Krumhansl's probe-tone ratings and melodic memory tests (see Tillmann et al., 2000). In each test, they compared their results with the human data and reported similar results.

Bharucha's and Tillmann et al.'s works gave groundbreaking evidence for the implicit learning aspect of auditory learning but did not have aims for computational models useful for production or control of expectation processes for compositional or performance purposes.

Berger's Tonal Expectation Model

Berger and Gang (2000) introduced a computational model for describing the experience of listening as it unfolds in real-time. Their computational model is based on recurrent neural networks (RNN) which are capable of capturing processes which are difficult to formulate by rules and hence, like Bharucha and Tillmann et al. addresses the problem of implicit learning. In addition to this, RNNs are suitable for modeling (short-term) temporal processes. After the learning process, the model's predictions (presented as the listener's expectations) are represented in terms of activation strengths of the input vector element (which consists of vertical pitch classes taken out of a music score). The predictive error of the system also serves as a *degree of realized expectation* which designates the musical affect to be a surprise or not. In (Berger and Gang, 2000), they detail the architecture and tuning of their system and demonstrate results on Haydn's Piano sonata in C major (third movement) along with an analysis of the specific piece, suggesting that the model simulates real-life listening situations.

The interesting point in Berger and Gang's work is in their use of the system's prediction errors to model listening behavior pertaining to musical expectation. Thus, they effectively make use of the fallibility of models actively learning in an environment. Moreover, their use of sequential neural networks provides a means to implicitly model and visualize the influence of several parameters over another (in this specific case, metric inference and functional tonal harmony).

2.3.3 Information Theoretic Models

Information theoretic approaches to music signals has been historically advocated by Moles (1969) and also by Meyer (1956). Both works and their related follow-up studies usually convey measures of complexity or uncertainty in music rather than predictive success of a listening system. Consideration of information theoretic paradigms in the latter sense is relatively new. Amid this young literature, that of Shlomo Dubnov is of particular importance for this work and is further expanded and supplemented in part II of this thesis.

Dubnov's Information Rate Model

In a series of publications, Dubnov has proposed methods that work directly on audio signals without assuming any a priori musical knowledge, and applied information theoretic methods to construct an anticipation measure of spectral observation pertaining to the information structure of music. Dubnov's literature is among the first, to the knowledge of this author, that has explicitly differentiated an anticipation process from that of prediction or expectation.

To capture the information structure of music signals, Dubnov introduced the *Information Rate (IR)* as a transmission process over a noisy time-channel where IR is defined as the relative reduction of uncertainty of the present when considering the past. Denoting a time series $\{x_1, \ldots, x_n\}$ as x_1^n, Dubnov shows that *Information Rate* at time n is equal to the mutual information carried between the past x_1^{n-1} and history of the

signal up to present or x_1^n. With this definition, IR can be interpreted as the amount of information a signal carries into its future. He further showed that given that X is a stationary Gaussian process, IR or $\rho(x)$ can be approximated asymptotically in n using spectral flatness measures of the time signal (Dubnov, Aug. 2004):

$$\rho(x) = -0.5 \times \log(\text{SFM}(x)) \qquad \text{or} \qquad \exp(-2\rho(x)) = \frac{\left[\prod_{i=1}^{N} S(\omega_i)\right]^{\frac{1}{N}}}{\frac{1}{N}\sum_{i=1}^{N} S(\omega_i)} \tag{2.1}$$

where $S(\omega_i)$ is the power spectrum of x_1^n. This measure can be readily estimated over non-Gaussian linear processes expressed as an AR model (See Dubnov, Aug. 2004). He further showed the significance of this measure over music signals and natural sounds in (Dubnov, 2006). However the Gaussian stationary assumption in the signal's generative model makes this difficult to apply to general music signals and structures.

To further relax this stationarity assumption and approach real-world situations, Dubnov (2008) introduced *Model-IR* by assuming a musically plausible hypothesis where the signal is stationary in a finite time-frame under a model θ_k and described by a joint conditional probability $P(x_1, \ldots, x_n|\theta_k)$. This way, the previous framework is augmented by considering *models* that generate signal chunks where the relation of an observation sequence x_1^n to another distribution model defined with parameters θ' can be approximated by

$$P(x_1^n) \approx P(x_1^n|\theta') \int \exp(-nKL(\theta'||\theta))P(\theta)d\theta$$

where $KL(.,.)$ is the *relative entropy* or *Kullback-Leibler divergence* function[2]. Further approximations to calculate the entropy and mutual information of the quasi-stationary time-series lead to the following formulation of IR (See Dubnov, 2008):

$$\rho(x_1^n) \approx \langle \rho_{\theta'}(x_1^n) \rangle_{P(\theta')} + \langle KL(\theta'||\theta^*) \rangle_{P(\theta')} \tag{2.2}$$

where $\langle . \rangle_{p(\theta)}$ indicates averaging following a probability distribution $p(\theta)$. Equation 2.2 gives us two factors for estimation of IR on a quasi-stationary signal: the first factor is

[2]Kullback-Leibler and other information theoretic notions will be explained later in chapter 4. See also (Cover and Thomas, 1991, Ch. 2).

due to the observation (block) being interpreted using a specific model called *Data-IR*, and a second factor situates the present model in relation to other models in the model space, called *model-IR*.

Dubnov (2008) further shows how to estimate this IR measure over multivariate processes and directly from audio signals. One interesting remark in his practical estimations of eq. 2.2 is his attempts to *evade* assumptions of any particular class of probability distributions over models θ_k. To estimate the distinguishability of different model distributions, he uses the method of types (See Cover and Thomas, 1991, Ch. 11) using *marco-frames* or large blocks of signal observations in contrast to the use of *micro-frames* for *Data-IR* calculation.

Dubnov has examined his IR measures on natural sounds and music (including jazz and classical audio) in (Dubnov, 2006) and has applied these methods in application to finding repetitive structures of sound and music in (Dubnov, 2008). A comparison of the measures obtained by IR and listeners's data on affective and structural aspects of music is reported in (Dubnov et al., 2006).

Dubnov's Information Rate models provide a promising framework for decoding of information structures of music with respect to listener's expectations. Following Huron's taxonomy introduced in section 2.1.4, Dubnov's notion of expectation in the reviewed model gets close to *dynamic adaptive expectation* where anticipation is presented and measured as the *action* the system takes (or does not take) as a result of prediction. Dubnov's model suffers mostly on the detection and formation of structures considered for the *model-IR*. To best capture the structural information of an underlying signal, one has to adapt the block and segment sizes to approach the real IR measures where these parameters are likely to change from a piece of music to another. The model does not simulate a real-time listening situation. Despite these facts, Dubnov's IR framework provides a promising framework for capturing dynamic expectancies and also structural information directly correlated to listener's listening behavior. In part II of this thesis, we extend and supplement this theory to better address information content analysis and retrieval.

Abdallah's Information Dynamics Model

Another recent information theoretic approach to music expectation is the work of Abdallah and Plumbley (2007). In their framework, several information measures are presented from a *model-based observer* perspective given a realization of a random process and an adaptively-updated statistical model as the process unfolds in time. Within this structure, expectancy measures can be obtained by measuring information content between three random processes governing information in the *past* (Z), *present* (X), and the *future* (Y). *Belief* over the flow of information can then be represented by some probability of the realization of the event, for example the probability of the unfolding time-series $X = x, Y = y, Z = z$ is represented by $p_{xy|z}(x, y|z)$. Within this framework, expectancy measures can be obtained by applying direct information theory measures to the three processes. For example, a surprise-based measure can be obtained as the negative log-likelihood of the present $X = x$ given its past $Z = z$ or $\mathcal{L}(x|z) \triangleq -\log p_{X|Z}(x|z)$. Similar to Dubnov (2008), they assume that the observer's *model* can vary in time where in this approach, this observation model is represented by some parameter space Θ whose distinction at another time can be measured using the regular Kullback-Leibler divergence between the two distributions with different parameters (See Abdallah and Plumbley, 2007, Sec. 2).

For measure estimations on symbolic music data, Abdallah and Plumbley (2007, Sec. 3) use simple Markov chains for modeling the random process governing the three processes explained above parameterized by a transition matrix $\{a_{ij}\} \in \mathbb{R}^{N\times N}$ encoding temporal transitions between states. Their preliminary assumption in doing so is that the process is stationary and the chain is irreducible in order to compute entropy rates as a function of transition matrix elements alone. For example, the one surprise-based measure discussed above would then be reduced to $\mathcal{L}(S_t = i|S_{t-1} = j) = -\log a_{ij}$ with S_t indicating the temporal state of the system at time t. In order to account for the non-stationarity of real music signals, they assume that the transition matrix is sampled from a fixed probability density whose parameters evolve over time. For this, they choose

a Dirichlet distribution over each column of the transition matrix. The distribution parameters evolve in time using a fixed mapping of the form $\theta_{ij} \rightarrow \beta\theta_{ij}/(\beta + \theta_{ij})$ where β is fixed and set by the authors, and $\theta_i j$s are the parameters of the Dirichlet distribution. They assess their measures on symbolic scores of (straightforward) monophonic minimalist repertoire of Philip Glass and show that the ensemble of their measures correspond to structural and expectancy phenomena that can also be extracted from the music score.

Despite the simple approach, the framework in (Abdallah and Plumbley, 2007) suffers from a strong a priori assumption of the form of the transition matrix and its evolution (a Dirichlet process without assessment of the choice). Note that in Dubnov's *Model-IR* framework, such choice is evaded by a careful and intricate use of the method of types along other estimations. This framework would also suffer in case of high-order or contextual structural dependencies and (most importantly) variations (which are common in most music and less common in music of Glass) due to their choice of Markov chains with simple first-order (or fixed-order) structure. Note that, as mentioned previously, Dubnov also suffers from a similar problem but he cleverly distinguishes between macro-frame observations (destined for his *model-IR*) and micro-frames (for *data-IR*) to partially overcome this drawback in modeling. Another important remark is that Abdallah and Plumbley's framework has been designed for symbolic (MIDI) representations of music. No extension to audio is discussed for this framework.

2.4 *Modeling* Investigations

In the previous section we looked at several key models attempting to explain or exhibit some aspects of musical expectations. To this aim we looked at the literature from three different perspectives. Despite their common goal, these approaches differ drastically in the way they try to achieve expectancy behavior. This is partially due to the fact that they come from different literatures with different philosophical beliefs towards the topic. Another reason is different subjective goals in each approach for modeling

expectation. Without wanting to get deep into the philosophy of modeling a cognitive behavior such as expectation or anticipation, there are several important lessons that arise when such models are to be considered in a computational framework destined for computer music or music information retrieval purposes. In this section, we undertake an investigation on the concept of *modeling* which should pave out the pathway for our proposal of anticipatory modeling.

2.4.1 Imperfect Heuristics and Naive Realism

In section 2.1.2 we noted that auditory learning and consequent mental representations can be fallible. This means that (experienced) listeners are far from perfect in learning to form accurate expectations about music and thus make systematic errors in comprehending the organizational features of music. When we learn from the world, what we learn is selective and imperfect and we are constrained by the problem of induction in general. Along these lines, Huron notes that the biological goal of expectation is different from its musical goal. When our expectations about future are based on faulty heuristics it might lead to potential biological disasters, however, the musical goal of expectation is to evoke a pleasing or compelling emotional dynamic and it does not matter if we form inaccurate expectations about future events.

When it comes to modeling cognitive behavior such as expectation, imperfect heuristics might easily lead researchers to *naive realism*. Huron (2006, Ch. 6) mentions this fact as a sobering point for music theorists. Naive realists consider senses as unbiased windows through the real world. For centuries, music theorists have been looking at music scores with naive realism: considering that the structures seen in notations are the ones we experience, and what is experienced is what is seen in the notation. The majority of models presented in section 2.3.1 suffer from this naive realism.

Huron (2006, in ch. 5) gives intriguing examples of such flaws of naive realism within which the most interesting one is the dilemma of *post-skip reversal* and *melodic regression* as reported in (von Hippel and Huron, 2000). Since at least sixteenth century

music theorists have observed that large intervals tend to be followed by a change of direction. Most theorists therefore have concluded that large intervals tend to be followed by step motion in the opposite direction, referred to as *post-skip reversal*. Through statistical analysis of vocal melodies from four different continents, von Hippel and Huron show that post-skip reversal is rather the result of a less exciting phenomena: regression towards the mean of a melody's tessitura. The results suggest that, in the sampled repertories, patterns such as "gap fill", "registral direction," and "registral return" (Narmour, 1990) which constitute post-skip reversal are mere side effects of constraints on melodic tessitura. More interestingly, they set up a quest to see if any composer's music would comply with the post-skip reversal framework with only one result: Giovanni Palestrina a prominent composer of the 16th century. Palestrina's music exhibit strong evidence of post-skip reversal and beyond a simple regression-to-the-mean framework. The striking fact about this result is that Palestrina is responsible for the promotion of the idea of post-skip reversal in the counterpoint literature.

Another example of naive realism and pertaining to the contemporary music repertoire is the analysis and presentation of *cognitive constraints on compositional systems* by Lerdahl (1988). In this article, the author draws upon several cognitive constraints claimed to drive the relationship between a music composition and the act of listening, presented as relevant concepts from Lerdahl and Jackendoff (1983). He proposes the concept of *musical grammars* as "a limited set of rules that can generate indefinitely large sets of musical events and/or their structural descriptions." and distinguishes between the *compositional* and *listening* grammars. Focusing on the music piece *"Le marteau sans maître"* of Pierre Boulez, as an example of intricate composition of mid-twentieth century serialism, he illustrates the gap between the compositional system and cognized result. He further distinguishes between *natural* and *artificial* grammars and develops a set of 17 "psychologically plausible" constraints on compositional grammars. He develops his reasoning for cognitive opacity of serialism and draws two aesthetic claims, and concluding by the following statement:

"The avant-gardists from Wagner to Boulez thought of music in terms

> of a "progressivist" philosophy of history: a new work achieved value by its supposed role *en route* to a better (or at least more sophisticated) future. My second aesthetic claim in effect rejects this attitude in favour of the older view that music-making should be based on "nature". For the ancients, nature may have resided in the music of the spheres, but for us it lies in the musical mind. I think the music of future will emerge less from twentieth-century progressivist aesthetics than from newly acquired knowledge of the structure of musical perception and cognition" (Lerdahl, 1988).

As bitter as it sounds, Lerdahl is probably right about our cognitive constraints in perceiving musical structures but we will surely be wrong if we assume that all structures serve an aesthetic purpose, or that all structures serve some psychological function. In music theory, naive realism is evident in two assumptions: that the structures we see in notated music are the ones we experience, and that the structures we experience can be seen in the notation. Anyone who believes that *subjective perception* is the same as *objective reality* is going towards the path of naive realism (Huron, 2006, p. 371). These repercussions of naive realism are at the heart of the analysis and conclusions of Lerdahl. On the other hand, after more than half a century from early avant-garde experiments, there still exists many counterexamples to constraint rules of Lerdahl that have passed their virtue of history and inscribed themselves in our musical heritage. Early music of the composer Gyorgy Ligeti is just one example, and computer music at large is another; both going outwards from Lerdahl's conception of "nature". Once again, our attitude towards such analysis is not all sober. Whether there are constraints that must be taken into account is not our question. But in formulating ideas, one must not mistake ignorance for imagination. Such theoretical failure should not deter theorists from forming or offering new theories. Empirical failures simply imply a more general cast for the development of our understanding of human cognition. To this author's listening abilities, Schoenberg, Boulez and Mozart wielded the same psychological tools of expectations despite significant difference in resulted works. The fact that some listeners are able to internalize counter-organizations (such as the ones Lerdahl observes) and form repertoire-specific expectation is actually a hopeful sign for the future development

of music.

2.4.2 Over-intellectualization of the intellect

Is the legitimate exercise of understanding a deliberate act of bringing things under rules? The philosopher Ludwig Wittgenstein elaborates the issue within the realm of *rule-following* (Wittgenstein, 1973, Aphorisms $185 - 201$). For Wittgenstein understanding is akin to ability. Understanding a concept is having a skill. One way to exercise the relevant conceptual skills is in explicit deliberative judgment; but that is not the only way. According to Wittgenstein it is psychologism to hold that actions can be rule governed only if the governing rule is explicitly stated in mind. Following a rule is only one of the different ways for rules to govern what we do. An expert no longer has any need for such explicit reliance on the rule. He has learned how to act in accordance with the rule without any need to consult the rule in thought. That does not mean that the behavior is no longer rule-governed either.

It is to overintellectualize the workings of the intellect to suppose that every exercise of understanding requires a deliberate act of compilation of an explicitly formulated rule (Noe, October 2005). Such an overintellectualized conception of the intellect leaves out the possibility that intellectual skills themselves may admit of expertise and effortless exercise.

In light of these considerations, we tend to reject the widespread idea of reducing expectancy behavior to rules as a result of naive realism of cognitive observations or over-intellectualization of induction. These criticisms could draw a sober image of findings pertained to current music theoretic approaches to musical expectations. However, one can easily question the degree of quantifications and formulated rules for expectations presented by Margulis (2005) and summarized in section 2.3.1: Is table 2.1 (on page 23) there to lay out a rule-model governing our melodic expectancies? How do we follow them? Whence the standards which decide if a rule is followed correctly? Are they in the mind, along with a mental representation of the rule? Following Wittgen-

stein, the very formulation of the questions as legitimate questions with coherent content should be put to test:

> "... no course of action could be determined by a rule, because every course of action can be made out to accord with the rule. The answer was: if everything can be made out to accord with the rule, then it can also be made out to conflict with it. And so there would be neither accord nor conflict." (Wittgenstein, 1973, Aphorism 201)

Therefore, similar to the post-skip reversal dilemma, studies on expectation such as the ones presented in section 2.3.1 could reveal part of the *story* but not on its entirety. One could imagine that such behavior or considerations could emerge from our environmental effects and could be partial and fallible representations common in some listeners. It is therefore a fallacy to consider these facts as *dispositions* of our cognitive apparatus.

2.4.3 Scientific pluralism

In Summer 2004 Marvin Minsky and Aaron Sloman, pioneers of Artificial Intelligence (AI), organized a symposium in St. Thomas, U.S. Virgin Islands, to discuss designs of architectures for human-level intelligence. In their report, they criticize current trends in research where the principal goal of AI is forgotten and instead researchers have developed special techniques that can deal with small-scaled engineering problems (Minsky et al., 2004). They indicate that we must develop ways to combine the advantages of multiple methods to represent knowledge, multiple ways to make inferences, and multiple ways to learn. They call for a "need for synthesis in Modern AI" where we should not seek a single unified theory to build a machine that is resourceful enough to have human-like intelligence. The central idea behind their architecture is that the source of human resourcefulness and robustness is the diversity of our cognitive processes: "we have many ways to solve every kind of problem so that when we get stuck using one method of solution, we can rapidly switch to another." In their proposed architecture, based on Minsky's *Emotion Machine* (Minsky, 2006), when the system encounters a problem, it first uses some knowledge about problem-types to select some *way-to-think*

that might work. Minsky describes ways-to-think as configurations of agents within the mind that dispose it towards using certain styles of representation, collections of commonsense knowledge, strategies for reasoning and all the other aspects that go into a particular "cognitive style." However, any particular such approach is likely to fail in various ways. Then, if certain *critic* agents notice specific ways in which that approach has failed, they either suggest strategies to adapt that approach, or suggest alternative ways-to-think.

For the purpose of this thesis, we are not interested in the entirety of the proposed architecture but their abstract view on modeling cognitive behavior and from an AI standpoint. It is interesting to see how Minsky et al.'s views coincide with what we have identified as key elements of psychology of music expectations in section 2.1. Their *ways-to-think* are the mental representations as depicted by Huron and reviewed in section 2.1.4 that work competitively and concurrently with the predictive power as a *guide* or *critic* over their collaboration as shown in section 2.1.3.

The main point here is that not only the field of cognitive modeling but also the engineering part of AI is asking for scientific plurality towards modeling and conceptual design, a point which will be at the core of this thesis and elaborated in part III.

2.5 Summary

With the aim of clarifying our stance towards cognitive concepts of *expectation* and *anticipation*, we reviewed key beliefs in the psychology of musical expectations that would constitute the core of any model that regards expectation. For this review, we relied on Huron's book in (Huron, 2006) and pinpoint important modeling factors for addressing anticipatory behavior that can be summarized as follows:

- The determinant factor for learning auditory phenomena is their stability in the environment.
- Listeners are sensitive to the frequency of appearance of events and sound pat-

terns in their surrounding environment, providing strong evidence for statistical nature of auditory learning.

- Exposure to the environment gives rise to *expectations* as *mental representations*.
- Listeners appear to code sound and sound patterns in concurrent and fallible representations.
- Representations are differentially favored depending on their predictive success and the unique path of the individual, hinting at the reinforcement and rewarding effect of expectations in learning and interaction.

We formally defined *anticipation* as an *action* that a system takes as a result of prediction using its belief or expectations including actions on its own internal state or belief. We examined this view from three different perspectives: that of psychology of musical expectations, computational frameworks and enaction. We showed conceptually that study of anticipation in such a framework is not separate from that of expectation and in the conceptual sense, entails the latter. We also hinted at the consistency of that framework with computational approaches to anticipatory systems which will be the topic of the coming chapter.

We continued our exploration by studying current key models of musical expectation from three different literatures: music theory, automatic learning and information theory. Given our observations in the music cognition literature, we favored the latter two approaches. This was followed by a contemplative discussion on the concept of *modeling* and its flaws where we favor more *objective* methodologies such as automatic learning and information theoretic approaches to music theoretic ones that can risk naive realism and over-intellectualization of the intellect. We also maintained that despite these flaws the mentioned music theoretic methods still underlie important aspects of our cognitive behavior that must be considered in a pluralistic design along with other views in order to achieve complex cognitive behavior. We ended our chapter by reviewing recent proposals and criticism in the *AI* literature in favor of scientific pluralism

that also underly our view of the psychology of musical expectations.

Through our criticism of *modeling*, it would be hard to ask what would *modeling anticipation* entail? How could it be done or whether it can be done? These questions would be the core of the next chapter.

Chapter 3

Anticipatory Modeling

What does *modeling anticipation* entail? Is there a single and unified model that can describe anticipatory behavior? Whence come the principles unifying such framework? Would that be in the mind along with mental representations of expectations? This chapter is our attempt to clarify these questions and position ourselves in their regards.

To address these questions and once again, we question their coherency and legitimacy. In the previous chapter, namely in section 2.4, we raised concerns about the very nature of modeling a cognitive behavior such as expectation and anticipation. We concluded that an attempt to formulate a single and universal *model* of musical anticipation would easily retreat to naive realism.

In the literature, there is much emphasis on cognitive models based on *causality* or *reactive frameworks*. In such a view, action is the result of a belief based on the past and present. This reactive paradigm is universal in the sense that given any mode of system behavior which can be described sufficiently accurately, there is a purely reactive system which exhibits precisely this behavior. In other words any system behavior can be simulated by a purely reactive system but this does not mean that the reactive paradigm is completely adequate for all scientific explanations. A pure *reactive* view of the world undergoes the same naive realism discussed in section 2.4.1 between the

representation of the things and the *things-themselves*. When it comes to modeling cognitive behavior such as expectation or anticipation, the same argument holds: Given that we have in our possession an accurate form of interaction between cognitive systems and their environments, a purely reactive paradigm might be able to explain and simulate such interactions. However, such attempts would became extremely approximative due to two main reasons: The fact that all forms of cognitive interactions can in no way be transcribed, formalized, or assumed as dispositions. And the fact that even at the disposition of such pure reactive framework, it would not necessarily provide a scientific explanation of the phenomena itself and nor does it extrapolate to previously unknown situations. These considerations led Robert Rosen and researchers in AI and later cognitive sciences to conclude that behaviors are not simply reactive but rather *anticipatory*, and attempt to develop a computing framework for anticipatory behavior called *anticipatory modeling*.

3.1 Anticipatory Computing

As mentioned in chapter 2, the term *anticipation* is often used in the literature to stress a simple lookahead into the future whereas the most important and often overlooked characteristic of anticipation is the impact of this lookahead on actual behavior. We proceeded by distinguishing our definition of the term in section 2.2. The basic premise in our consideration is that expectations are not only useful for predicting the future but also alter our behavior or our behavioral biases and predispositions according to predictions or expectations. To distinguish anticipation in regards to definition 2.1 (page 17) from other aspects of cognition, we focus on *Anticipatory Behavior* defined as a process, or behavior, that does not only depend on past and present but also on predictions, expectations, or beliefs about the future (Butz et al., 2003a).

Anticipatory computing has started as an attempt to uncover commonalities often overlooked in considerations of anticipatory processes (in animals and human behavior) and offer useful conceptualizations and interconnections between research dis-

ciplines in cognitive systems research. Robert Rosen puts forward one of the first definitions of an anticipatory system:

Definition 3.1 (Anticipatory System)**.** A system containing a predictive model of its environment, which allows it to change state at an instant in accord with the model's predictions pertaining to a later instant.

In Rosen's original definition (Rosen, 1985, p. 330), anticipatory systems contain predictive models *of themselves* as well as their environments. Inherent to this definition is the fact that the *representation* of the environmental states and internal states of the system are distinct and the system contains an *internal representation* of the outside world to constitute artificial behavior. Following our enactive view of cognition (section 2.2.2) we reject the dissociation of the *internal representation* of a system from the environment itself. An internal representation of a computational system does not constitute a detailed snapshot view of its environment, nor is it dissociated from its actions; but is representing the dynamic flow of continuously varying information of the environment itself (Noë, 2004, Ch. 1)[1].An important consequence of such approach is to lessen the representational burden of the system.

We define *anticipatory modeling* as the design process for anticipatory systems. In contrast to *modeling anticipation*, anticipatory modeling does not attempt to provide a universal model or framework for anticipatory behavior, but rather to provide models that anticipate. It considers anticipation as the fore-front design concept in cognitive systems to achieve complex real-life behavior. Throughout this thesis, we adopt this last view for our goal of achieving and accessing anticipatory musical systems. We simply shift our approach from modeling musical anticipation to anticipatory modeling for computer music by explicit consideration of anticipatory interactions, often overlooked in system design, as the main concern of modeling.

[1] An example of such an approach is the *animate vision* program of Ballard (1991, 2002), that bases his visual recognition system on gaze control mechanisms and adaptive learning from an environment. In this approach, instead of building up internal and detailed maps of the environment, the system *acts* upon the desired goal at the onset using its available skills, and adapts itself continuously through series of perceptions and actions in the environment and in real time.

Rosen's definition of anticipatory systems, has inspired various fields of research such as experimental psychology, theoretical biology, physics and economy and has led to an emerging literature on *Anticipatory Computing*. Recently, attempts have been made in artificial intelligence to integrate anticipatory mechanisms into artificial learning systems in various frameworks, leading to the creation of an interdisciplinary international workgroup named *Anticipatory Behavior in Adaptive Learning Systems* or *ABiALS* in short. Their work has culminated to two collections (Butz et al., 2003c, 2007). The new field has attracted AI researchers as well as cognitive neuroscientists, and psychologists and created debates among philosophers (e.g. Nadin, 2004; Glasersfeld, 1998). Therefore, the scope of research in *ABiALS* varies dramatically from experimental psychology to logic and engineering AI applications as well as cognitive neuroscience and pure AI. The efforts of *ABiALS* has led to a mutual understanding and transfer of concepts from experimental psychology to engineering applications which have found life in various fields such as robotics and visual processing.

We review the common frameworks and approaches that constitute the design philosophy of anticipatory systems for this thesis.

3.2 General Modeling Framework

Following definitions 3.1 and 2.1, we aim at providing a general framework for anticipatory modeling in the current section. Such a framework should directly address *learning* for capturing stabilities of the system's external environment interactively. Learning thus constitutes artificial beliefs of the system about conditions and consequent actions in an environment, as well as the dynamics of the environment itself. One of the frequently used frameworks for such problems in artificial intelligence is the *Markov Decision Process (MDP)*. One of the main attractions of the MDP framework is its flexibility and extensibility for different design problems as reviewed below.

From this point on through this thesis, we are officially in the realm of *artificial design*. Up to this point and mostly through chapter 2, we looked at modeling

anticipation in an effort to explain musical behavior which could be thought of as a *natural science*. Anticipatory modeling is more akin to what Herbert Simon would call an *artificial science* where modeling is a design process that addresses musical behavior (Simon, 1969). Through this change of view, many terminologies that we have visited so far within the realms of cognitive psychology and in connection to living humans and animals, would change their scope of meaning to that of artificial systems aiming at modeling such behaviors. Therefore, from this point on, words such as "belief", "mental representations", and "observations" reflect their respective functionalities within artificial systems with their limitations, distinct from their usage within experimental psychology.

To clarify matters further, anticipatory modeling does not mean that we have access to the future itself, but to a *belief* over the states of the system in future. Moreover, the interactive nature of learning in anticipatory systems strongly requires *online* systems where data transaction with an environment is done incrementally with no access to the future itself. Therefore the word "future" in the following sections, chapters and captions refer respectively to the belief of a system or mental configurations of the system towards predicted future states.

3.2.1 Markov Decision Process Framework

Markov Decision Processes (MDPs) provide a simple, precise and relatively neutral way of talking about a learning or planning agent interacting with its environment to achieve a goal. As such, MDPs are starting to provide a bridge to biological efforts to understand the mind (Sutton, 1997) and serve as a conceptual tool contributing to a common understanding of intelligence in animals and machines. More importantly for this work, they constitute the basic tool for anticipatory modeling motivated by psychological evidence on animal behavior (See Butz et al., 2003b, pp. $87 - 90$).

A *Markov Decision Process* comprises an *agent* and its *environment*, interacting at all times. At each interaction cycle, the agent perceives the state of the environment s_t,

and selects an action a_t. In response to actions and in the next time step, the environment changes to a new state s_{t+1} and emits a scalar reward r_{t+1}. In a *finite MDP* the states and actions are chosen from finite sets. In this case the environment is characterized by arbitrary probabilities (also called transition probabilities) $P(s, s', a)$ and expected rewards $R(s, s', a)$, for each possible transition from a state s to a next state s' given an action a.

The MDP framework is abstract and flexible, allowing it be applied to different problems and in different ways. We make use of this flexibility of MDPs for the two concrete applications presented in parts IV and III of this thesis. For example, the time steps need not refer to fixed intervals of real time (as is the case in parts IV); they can refer to arbitrary successive stages of decision making and acting as is the case in an improvisation setting between two musicians (parts III). Actions can be low-level controls such as time distributions of a musical event (part IV) or high-level decisions such as generation of musical phrases (part III). States of the system can be determined by low-level sensations from the environment such as continuous real time audio observations for an accompaniment system (part IV, or they can be high-level such as symbolic descriptions of music in an improvisation system (part III). Actions might be totally mental or computational. For example, some actions might control what an agent chooses to generate (part III), or how it focuses its attention in decision making (part IV).

Extensions of MDP frameworks can be done in a way to reduce the extended problem back to an MDP. For example, in MDPs the dynamics of the environment are assumed stationary and Markov, which limit applications to complex problems such as modeling musical interactions. The most prominent extension of MDPs to the non-Markov case is the classical approach to *partially observable MDPs (POMDPs)*, in which the state is not fully observable on each time step and its probability is estimated according to an observation probability distribution conditioned on inputs from the environment. It turns out that this probability distribution can itself be treated as the state of the whole process and then all the classical MDP methods can be applied, albeit in a larger and more complex state space (Murphy, 2000; Sutton, 1997). Another example of

such extension, is hierarchical and modular approaches to decision-making and action generation using low-level MDPs (e.g. Littman, 1994; Rohanimanesh and Mahadevan, 2002).

In this thesis, we will constantly revisit the stationary and Markov assumptions of MDPs and in different manners. In part II we provide a state-space representation (the *Audio Oracle*) that preserves the state-action representation but extends the basic Markov assumption to larger context. This is used and further expanded to a hierarchical framework for automatic generation and planning problems in music of part III. In part IV, we extend the Markov architecture to semi-Markov occupancies and define an inference framework conform to MDP learning and decision making.

3.2.2 Interactive Learning in an Environment

In an MDP framework, the agent implements a mapping from states to probabilities of selecting possible actions. This mapping is called the agent's *policy*, denoted π_t, where $\pi_t(s, a)$ is the probability that $a_t = a$ if $s_t = s$. The agent's goal is to maximize the total amount of reward it receives over the long run. MDPs were originally studied under the assumption that the dynamics of the environment (or $P(s, s', a)$ and $R(s, s', a)$) are completely known.

An anticipatory system is by definition in constant interaction with its environment in order to form *expectations* and *act* according to past and present environmental contexts and also *beliefs* about the future. The policy π_t of a regular MDP therefore represents current beliefs that the agent has acquired within its environment. Besides this implicit state model, an anticipatory agent is equipped with a *predictive model* M^p which specifies how the state model changes dependent on *possible* actions geared towards future. Hence, M^p describes an action-dependent probability distribution of future environmental states. A common goal of any anticipatory system is then to learn an optimal *behavioral policy* to determine how the system decides *what* to do, *which* actions to execute and possibly *when* to execute it. This behavioral policy might depend

on current sensory input, or generated predictions, the state model or directly on internal states, or moreover a combination thereof depending on the nature of the problem at large.

Besides learning optimal policies at each interaction, we do not assume that the environmental dynamics are known to the system. This requires the system to obtain these dynamics through interaction episodes with the environment and adapt its belief to the ongoing context. Specifically within an MDP framework, this entails to acquiring $P(s, s', a)$ and $R(s, s', a)$ adaptively through interactions with the environment. The transition probabilities $P(s, s', a)$ constitute both the structures of the environment in terms of information content, and the values pertaining to each state-action transaction; where both should be ideally adapted online. The reward structure $R(s, s', a)$ on the other hand, serve as the feedback of the environment to a previous state-action transaction and is used extensively for belief updates of the system.

Within this framework, learning is then to acquire optimal policies and environmental dynamics through interactions with the outside environment. Learning can be incorporated by allowing modifications of any of the above components over time. Learning should also be *interactive* and *online* to undergo adaptive behavior. Therefore, the choice of architecture for interaction and learning would determine the type of anticipatory behavior of the system. Besides the problem of learning policies that is common to all MDP frameworks, environmental dynamics can be acquired artificially using different methods. But before reviewing different learning techniques common to anticipatory modeling, we review several common anticipatory behaviors modeled in the literature and used throughout this thesis.

3.3 Distinctions of Anticipatory Behavior

Following a generic MDP framework as above, Butz et al. (2003b) distinguish between four different types of anticipatory behavior mostly destined for *animat* (i.e. artificial animal) design literature. To be consistent with this literature, we reiterate

these findings here and mostly focus on their design schemes. Some of these schemas will be later considered for our various designs in parts IV and III of this thesis.

3.3.1 Implicit Anticipation

In most existing designs that consider sensory input and action outputs, predictions about the future are not explicitly considered for influencing the system's behavioral decision making. This way, the sensory input is directly mapped onto an action decision possibly combined with internal state information to form some inductive inference. Therefore, in this simple scheme the prediction model M^p and its role for influencing current decision-making is simply ignored. However, such systems can be considered as *prediction-driven* systems in the sense that the action-decision is the result of an implicit prediction based on current state-action information. Therefore in implicit anticipatory behavior there is no explicit knowledge about possible future states but it is anticipatory in the sense that the behavioral architecture is predicted to be effective. The basic structure of an implicit anticipatory architecture is shown in figure 3.1.

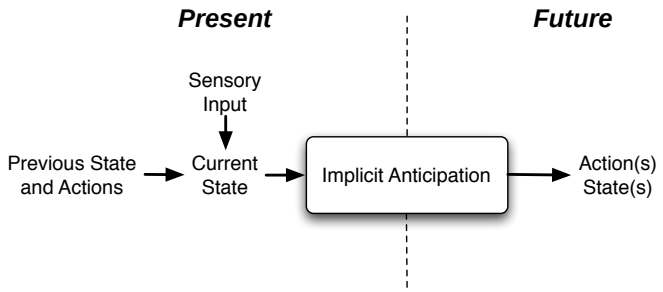


Figure 3.1: Implicit anticipatory behavior architecture

It is interesting to note that even pure reactive systems are still implicit anticipatory systems in the sense that the behavioral programs in their codes are implicitly anticipated to work in their context. A genetic code, for example, is implicitly predicted (by evolution) to result in successful survival and reproduction. Many existing engineering systems that make use of POMDP, HMMs and variants enter this category. A typical

example of such approach in the literature are current audio to score alignment or score following systems (see chapter 7).

3.3.2 Payoff Anticipation

If a system considers predictions of possible actions in terms of their *payoff* values without any consideration for the arriving states in the environment, it is a *Payoff Anticipatory System*. In such systems, the predictions estimate the benefit of each possible action and bias action decision making accordingly. State predictions have no influence on the system in this schema. Figure 3.2 sketches the schematic architecture for payoff anticipation.

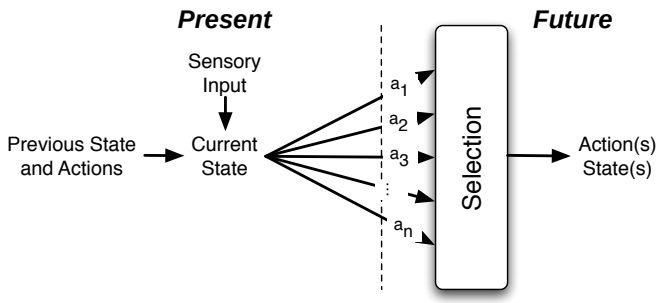


Figure 3.2: Payoff anticipatory behavior architecture

Typical examples of payoff anticipatory systems in computer music literature are current approaches to automatic improvisation or style imitation systems where generation is commonly based on selecting the best action among all possible action given the current state of the system and upon some contextual value for each action (see chapter 6).

3.3.3 Sensorial Anticipation

Explicit prediction of future states of a system does not need to directly influence the behavior of a system but its sensory processing (or observation handling in computational terms). In this scheme, called *sensorial anticipation*, the prediction of

future states and thus the prediction of future stimuli influences stimulus processing. This scheme requires an explicit predictive model M^p of the underlying processes. This anticipatory mechanism is shown in figure 3.3

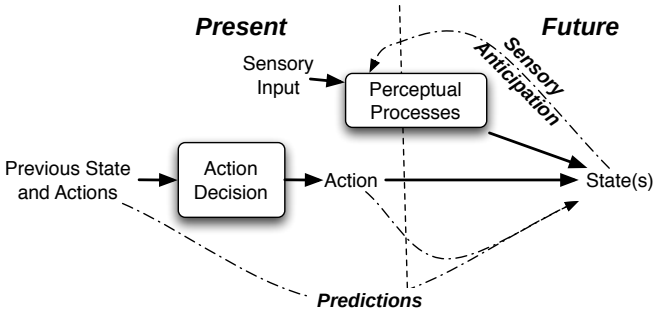


Figure 3.3: Sensorial anticipatory behavior architecture

This anticipatory behavior is strongly related to preparatory attention in psychology and results in a predisposition of processing sensory input. This biased sensory processing could then influence the actual behavior or affect learning in an indirect manner.

Typical examples of sensorial anticipatory schemes in computer music literature are multinomial processing systems where the predictions on one data source (e.g. tempo, gesture data) affects the sensory processing or observation module of another source (e.g. audio onset, pitch). Later in chapter 7, we propose a model along these lines.

3.3.4 State Anticipation

In the three sketches above, the role of expectation has been implicit in one way or another. The most explicit way of considering anticipation, where the system's behavior is influenced by explicit future state representations is called *State Anticipation*. Whereas sensorial anticipation indirectly affects learning and sensory processing, in state anticipation this influence is direct. The basic property here is that predictions about, or simply representations of, future states influence actual action decision. Fig-

ure 3.4 sketches this architecture.

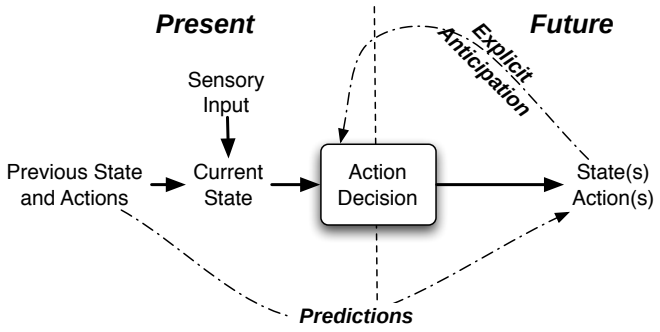


Figure 3.4: State anticipatory behavior architecture

Many existing computer music frameworks can benefit by the extension of their designs to a state anticipatory system. Automatic improvisation and style imitation systems are obvious examples where an explicit representation, access and manipulation of predictive models and interactions within would improve the generative results and control aspects of their design. This will be the main concern of part III of this thesis.

3.4 Learning Approaches

Anticipatory modeling is heavily based on on-line learning of system parameters and internal models through interaction with an outside environment. The type of desired anticipatory behavior and also the goals of learning determine the design and choice of learning approaches. Due to this diversity, the nature of learning algorithms used and proposed in this literature are vast. Here we summarize the basic concepts of some of the mainstream machine learning approaches that conform to anticipatory computing literature. Learning in anticipatory models is usually concerned with two main issues: Learning and updating predictive models and dynamics of the environment, and learning a behavioral policy for decision making.

3.4.1 Reinforcement Learning

The Reinforcement Learning (RL) framework is a considerable abstraction of the problem of goal-directed learning from interaction. It proposes that whatever the details of the sensory, memory, and control apparatus, and whatever objective one is trying to achieve, any problem of learning goal-directed behavior can be reduced to three signals passing back and forth between an agent and its environment: one signal to represent the choices made by the agent (the actions), one signal to represent the basis on which the choices are made (environmental states), and one signal to define the agent's goal (the rewards) (Sutton and Barto, 1998). In the standard reinforcement learning model, an agent is connected to its environment via perception and action.

Reinforcement learning is primarily concerned with how to obtain the *optimal behavioral policy* when such a model is not known in advance. The agent must interact with its environment directly to obtain information which, by means of an appropriate algorithm, can be processed to produce an optimal policy. Within this view, RL techniques can be categorized into two main groups: *Model Based* methods learn or update *predictive models* (MDP or else) first and then proceed with policy optimization, and *Model-free methods* which emphasize on learning policies with no explicit knowledge requirement of the expected immediate rewards of the state-transition probabilities or model topologies. To this respect, *model-free RL* agents are close to *payoff anticipatory* agents where there is no explicit predictive model despite the estimated action-payoffs. In this framework, eventhough the system does not learn a representation of the actual sensory consequences of an action, it can compare available action choices based on the payoff predictions. *Model-Based RL* however, can address the other three distinct anticipatory mechanisms and specifically state anticipation. The *Dyna* architecture of Sutton (1991) is probably the most widely used and one of the first frameworks of this kind where the agent learns a model of its environment in addition to policies. The basic premise of the *Dyna* architecture is its generic form and flexibility for extensions. Several anticipatory mechanisms can be reached via this approach for example by bias-

ing the decision making agent towards the exploration of unknown regions or accessing specific expectancy patterns in the (memory) model. The design strategy for learning both the representation and policies are determinant of the type of anticipatory behavior.

RL techniques are mostly studied within *Markov Decision Process* framework. However, extensions exist for *Semi-Markov Decision Processes* where each state has an explicit waiting time, and for *Partially Observable Markov Decision Processes (POMDP)* where the assumption of complete observability of the environment is relaxed. Traditional RL techniques store state-action values and policies in tabular form which makes them difficult to scale up to larger problems. To overcome this curse of dimensionality, various methods such as function approximation, hierarchical learning, collaborative learning and more have been proposed in the literature. For an extensive overview of RL techniques see (Kaelbling et al., 1996). We will come back to reinforcement learning in part III of this thesis.

3.4.2 Learning Classifier Systems

Genetic Algorithms (GA) constitute another literature suitable for solving reinforcement learning problems. In general, they search in the space of behaviors in order to find one that performs well in the environment through genetic reproductions and fitness or survival functions. However, their consideration in an interactive environment where the action's behavior is highly dependent upon its interaction with an outside environment is limited due to the very nature of genetic programming. These considerations led John Henry Holland (the father of genetic algorithms) to work on Learning Classifier Systems (LCS) where he proposed a model of a cognitive system that is able to learn using both reinforcement learning processes and genetic algorithms (Holland and Reitman, 1978). Reinforcement values in LCSs are stored in a set (or population) of condition-action rules (the classifiers). The learning mechanism of the population of classifiers and the classifier structure is usually accomplished by the means of a genetic algorithm. With this respect, a LCS is a rule-based reinforcement learning system

enhanced with the capability to generalize what it learns through genetic algorithms. There has been several extensions to the original LCS proposal with one specific to Anticipatory Learning Classifier Systems (ACS) (Butz and Goldberg, 2003).

Due to the fact that learning classifier systems in general assume rule-based structures and reasoning, and following our extensive discussions in section 2.4, they will not be considered for this work.

3.5 Modeling Implications

We now draw several important implications of Anticipatory Modeling as they will constitute the core of the coming chapters and the scientific contributions of this thesis.

3.5.1 Information as Available

The first premise in any anticipatory design is the representation or encoding of incoming information into useful representations and enable fast and easy access to these dynamic memory structures. In an anticipatory system, as noted earlier, the system is in constant interaction with an outside environment to update its current belief. This belief constitutes the knowledge of the system of its environment. Therefore the form of representation of this belief is crucial. In an interactive system that unfolds in time, this knowledge representation also constitutes the *memory* of the system with regards to its past and to the past of its interactions. More importantly it constitutes a special style of *access* of the system to the world and in our case takes the form of representations of the world outside. We noted in chapter 2 that such mental representations in human cognition are not necessarily accurate and can be fallible. By analogy in computational modeling, the percept of the system in absence of accurate representations is assessed by *access* to the information controlled by patterns of expectational dependence. We strongly emphasize on *accessibility* of pertinent information from the environment to

the system. During learning and adaptation, comparing predictions with actual environmental consequences, it is important for such a system to have acquainted information as available and present whenever needed. To take this matter to the extreme, achieving expectation or anticipation is not a matter of dividing attention to information and not a matter of attention at all. It is a matter of having *skillful access* to the associated information whenever required. We emphasize again the importance of information availability as a consequence of its representation by reiterating section 2.2.2 with respect to our goal of providing musical anticipatory models: If *anticipation* is the very act and consequence of prediction as in definition 3.1, there is no need to separate the *representation* from the *action* as one encompasses the other. Therefore representation of information and its availability are at the core of any anticipatory system design.

With this respect, we dedicate the part II of this thesis, and the start of our design exploration, to the problem of encoding and representing musical information with a high emphasis on the problem of information accessibility controlled by patterns of expectancy.

3.5.2 Interactive and on-line Learning

This thesis is en effort to extent computational models for music from *reactive frameworks* to *anticipatory frameworks*. For computational models of cognitive phenomena, one could plausibly design reactive systems with satisfactory results given that every possible interactions of the living phenomena with its environment are at our disposal in terms of syntactic or declarative laws to be integrated into a reactive design. We showed at the onset of this chapter that such an approach would be immediately reduced to approximations and would result to shortcomings on generalization of its knowledge-domain to unknown situations. To overcome this, we introduced *anticipatory modeling* where constant adaptation and learning through an environment are at its very core.

As we mentioned in chapter 2, anticipation in humans is the act of predicting, reacting and self-adapting in a constantly changing environment and results from the sta-

bilities and instabilities of the environment we live in. By analogy in computational design, learning and adaptation are at the core of an anticipatory system, and like real-life situations and unlike many engineering approximations, this learning should be *on-line*, meaning that it should accept information incrementally as they arrive and in real time; and *interactive*, meaning that it is the result of a constant interaction with an outside environment for comparing and examining predictions with actual consequences. In modeling anticipatory musical frameworks, learning comes to play for obtaining new representations and updating existing ones, and also for decision making purposes through interaction with the environment.

Due to these facts, all the learning algorithms and concepts represented hereafter are incremental and adapted for real time use. Part II of this thesis is concentrated on the aspect of learning and updating representations, and part IV and III will concentrate on two different cases of decision making using anticipatory modeling.

3.5.3 Multimodal Interaction and Modeling

In section 2.1.3 of chapter 2 we showed evidence in the literature that anticipatory behavior is the result of concurrent representations of the same environment evaluated perpetually by their ability to usefully predict ensuing events. This means that several representations are at work to encode the same event that might correspond to the same or several information sources in the environment. Moreover, the competitiveness of these representations requires an amount of interaction during learning among representations. Computationally speaking, these considerations lead to a *multimodal interaction* framework between the ensuing agents and the environment, as well as among the agents themselves. This multimodal interactivity in its term requires specific anticipatory models in-between agents. Such multimodal interactions are connected to the very nature of musical interactions and any interactive music system. An example of such multimodal interaction is the act of listening and following a music score: From an auditory point of view, when we listen and follow a music score we might be following

exact pitch information, pitch contours, registral pitch information, or time informations such as the rhythmic structure, or pivotal points in the score, or a combination thereof; or to make it even more realistic, a mental representation such as temporal rhythmic structures might come to the help of (or prevention of) another fallible representation such as pitch. A problem such as this is one that can be directly addressed using an anticipatory framework.

Two case-studies for multimodal interaction and modeling will be presented in parts IV and III of this thesis.

Part II

What to Expect

Chapter 4

Music Information Geometry

What do we actually expect? We saw in chapter 2 that expectations imply some sort of mental representation of the stabilities of the outside environment. This in return implies finding ways to discover and represent *relevant* information in an environment within a system's architecture in order to form expectational beliefs. On another level, in section 3.5.1 we identified *Information as available* as one of the most important premises of anticipatory modeling. It constitutes the form of representation of audio and music information for memory of the system, behavioral learning, and content access in anticipatory interactions. This part of our thesis is thus dedicated to the problem of decoding and representation of *relevant* musical information with an aim of integration into anticipatory musical system. In this chapter, we focus on a theoretical framework that enables decoding and representation of musical information, and in the next chapter we elaborate the problem of information access.

4.1 General Discussions

All music information retrieval and computer music systems deal one way or another with the information content of music signals, their transformations, or extraction of models or parameters from this information. A common question that many such

systems ask at their front-end is what information is presented in the signal and to what relevancy? Despite its wide use in the literature, classical information theory as put forward by Shannon (1948) has few answers to the representation and fidelity concerns. As an example, entropy is commonly used to assess a signal's *uncertainty* but signals have uncertainty regardless of whether they have relevant information or not (Sinanović and Johnson, 2007). When the issue of music complexity is considered, matters come worst. Pressing shows the inconsistency of basic information theory measures of complexity with regards to the structure of musical patterns (Pressing, 1999). An alternative choice for quantifying information is by using mutual information which quantifies how closely the probability functions of a system's input and output agrees. Previously in section 2.3.3 we reviewed two proposals for measuring structural regularities in music based on mutual information. Rate distortion theory and information bottleneck methods also use this property to assess how well signals encode information mostly with regards to signal compression (Cover and Thomas, 1991; Tishby et al., 1999).

On these lines, it is quite interesting to look back at some historical notes as the field of information theory emerged: Warren Weaver, in his introduction to Shannon's classic paper, states that while Shannon's work was the ultimate mathematical theory of communication, it did not touch the entire realm of information processing notions that require analysis. Furthermore he developed communication problems on three grounds: *technical*, *semantic*, and *influential* (Shannon and Weaver, 1949; Sinanović and Johnson, 2007). In this chapter, we draw a framework to address the third aspect of information processing with regards to music signal: the influential aspect of information as the effectiveness of music information as it unfolds in time.

On another basis, one of the main goals of music information retrieval (MIR) systems is to characterize music information in terms of their *similarity* structures. Most MIR techniques also rely on geometric concepts for building classifiers in supervised problems (genre classification, artist identification, query by example etc.) or clustering data in unsupervised settings (audio search engines, structure discovery etc.). The considered data sets and their underlying spaces are usually *not* metric spaces, however

in most typical MIR applications a feature-based similarity matrix is computed over the data which is used as a front-end for further processing. The *metric* used for each approach is usually varied among researchers and applications, and include common metrics such as Euclidean, Mahalanobis and probabilistic distances between feature-based representations but without any explicit consideration for the underlying geometric space and whether it constitutes a metric space or not. In this chapter, we provide a new view of *similarity* based on influential aspect of information.

Most existing computational music systems bear difficulties when the temporal aspect of music comes into play and undergo extensive approximation (or elimination) of temporal resolution for representing music content. For example, a common method in most MIR systems destined for classification or clustering is the *bag of features* method where the time-series information is considered out-of-time (as put in a bag) and thereafter the relevant information is searched, learned or processed. The framework introduced in this chapter explicitly model the temporality of music information. This assumption constitutes the only a priori information involved in our framework.

In this chapter, we seek a comprehensive framework that allows us to quantify, process and represent information contained in music signals and structures. This topic brings in concepts from various literatures: music signal processing, differential geometry, information theory and machine learning. By this combination, we aim at investigating the natural geometric structures of families of probability distributions over music signals that implicitly represent the ongoing information structure of the signal over time. Within this framework, music information arrives in discrete framed windows over time and occupy statistical *points* in an information manifold. By translating concepts such as *similarity*, *clustering*, and *metric ball trees* in an information geometric framework, we are able to reduce the complexity of several important information retrieval problems as a consequence of geometrical relations and specially affine duality of information geometric spaces. To achieve this goal, we revisit common concepts in music information retrieval (MIR) literature such as *similarity* in an information geometric context and provide a music information processing framework within this realm

called *Music Information Geometry*. The geometric structure proposed here is based on general *Bregman divergences* (Bregman, 1967) and their geometric consequences which will be introduced shortly. Besides the geometric intuition of the provided information theoretic framework, this approach has important modeling consequences when considered within the more general exponential family of probability distributions on the incoming signals, among which the affine duality is of most importance for this work.

We start the chapter by building up the mathematical preliminaries vital for the construction of our method and introduce the general notion of information geometry (section 4.2.1), elements of Bregman geometry (section 4.2.2) and exponential distributions (section 4.2.3). The introduced frameworks along with provided examples, are essential for a thorough understanding of the music information geometry framework introduced thereafter. This mathematical framework is tightly related to (Amari and Nagaoka, 2000; Nielsen et al., 2007) whereas the machine learning aspects come close to (Banerjee et al., 2005; Cayton, 2008). Upon the introduction of the mathematical structures, we introduce a framework for *Music Information Geometry* that builds upon the *Information Rate* measures of Dubnov (2008) (see review in section 2.3.3 on page 25) and aims at expanding and improving the *Model IR* measures. We show that our mathematical framework would naturally bring in important aspects of information measures introduced by Dubnov (2008) and extends the methods by introducing novel access to information structures of music. In section 4.4 we show how our mathematical struct would lead to a similarity metric space and provide an information theoretic definition of *similarity*. Finally, section 4.5 details our methodology for incremental model formations in an ongoing music signal with the consequence of segmenting an audio stream in real-time to chunks of quasi-stationary structures.

4.2 Preliminaries

In this section, we introduce the basic mathematical constructs that form the basis of our methodology for music information geometry. We start by introducing basic

concepts of *information geometry* (Amari and Nagaoka, 2000) and move on to Bregman divergences and their geometric properties and continue on by introducing exponential families and their behavior in a Bregman geometry. Many of the concepts and theorems here are replicated from their cited references and without proof (except for some key properties) and the reader is referred to the provided references for the proof of each theorem. Some sections are enhanced by *examples* demonstrating the underlying theorem in action. Thorough understanding of the given examples are important as they build up the mathematical construct that constitute our music information geometry framework.

4.2.1 Information Geometry of Statistical Structures

Let us consider a family of probability distributions specified by a vector parameter $p(x, \boldsymbol{\xi})$ where $\boldsymbol{\xi}$ is a vector constituting the model parameters of the probability distribution. This set can be regarded as a manifold under certain regularity conditions where $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_n)$ would be its coordinate system. A manifold, in short, is an abstract mathematical space in which every point has a neighborhood which resembles a regular Euclidean space but the global structure may be more complicated. By defining probability distributions on a manifold, each *point* in our space would then refer to a realization of a family of probability distribution. The manifold has a natural geometrical structure if the following two invariance principles hold:

1. Geometrical structure is invariant under whichever coordinate system (or parameters) used to specify the distributions.

2. Geometrical structure is invariant under rescaling of the random variable x.

A Riemannian manifold in short is a real differentiable manifold $\mathcal{S}$ where each tangent space is equipped with an inner product g in a manner which varies *smoothly* from point to point. Amari and Nagaoka (2000) show that the manifold of statistical structures has a unique Riemannian metric which is given by the Fisher information measure. For the two conditions above to hold, one needs to define proper *affine connec-*

tions that can in return define various geometric notions such as angle, length of curves, areas (or volumes), curvatures, gradient of functions and divergence of vector fields. In differential geometry, an affine connection is a geometrical property which *connects* nearby tangent spaces allowing tangent vector fields to be differentiated as functions on the manifold. With an affine connection, a path between two points of the manifold establishes a linear transformation (*parallel*) between the tangent spaces at those points. Amari and Nagaoka (2000) introduced α-connection (Δ_α), a canonical form of connection over an information manifold that further possesses a dual structure (Δ_α^*) which is discussed later. Given this, an information manifold can be defined by the Fisher Information as the inner product and also by the type of connection (and consequently its dual because of the extreme computational interest thereof) as $(\mathcal{S}, g, \Delta_\alpha, \Delta_\alpha^*)$.

To complete the geometric tools in this context, one needs a *distance like* measure between two points (or probability distributions in our context). Such measures, called *divergences*, provide the directed (asymmetric) difference of two probability density functions in the infinite-dimensional functional space, or two points in a finite-dimensional vector space that defines parameters of a statistical model. A typical example of such divergence is the widely-used *Kullback-Leibler distance*. Furthermore, Eguchi (1983) shows that a well defined divergence D would also lead to a general construction of the dualistic structure $(\mathcal{S}, g, \Delta^{(D)}, \Delta^{(D^*)})$ and hence allowing the geometric structure to be defined directly by the Fisher Information and the given divergence (instead of the affine connection). There is a tight relation between divergences and affine-connections on a statistical manifold; however, this is not a one-to-one relationship as there usually exists infinitely many divergences corresponding to a dualistic manifold structure. Recently, Zhang (2004) introduced a canonical form of affine connection where α-connection would be one of its special cases, and can deduce many types of divergence functions which are in common use in engineering applications, including the well-known *Bregman Divergence* family (Bregman, 1967). Given these findings, and within the framework introduced by Zhang (2004), we can easily assume a geometrical structure over probability manifolds $\mathcal{S}$ using Fisher Information and Bregman

Divergences.

Throughout this section, we assume that a system under measurement generates families of probability distributions within a dual information manifold $(\mathcal{S}, g, \Delta^{(D)}, \Delta^{(D^*)})$ where its geometric properties are induced by employing *Bregman Divergences*. Also, from hereon the term *point* is used in an information geometric sense and thus represents a family of probability distributions that belongs to a probability simplex $\mathcal{X} \in \mathbb{R}^d$. As another convention, vector mathematical constructs are notated using boldface characters in contrast to scalar constructs, therefore $\boldsymbol{p}$ is a vector and p is a scalar. We now introduce the Bregman divergence family and their geometric structures.

4.2.2 Elements of Bregman Geometry

In this section we define *Bregman divergences* and investigate their geometrical properties such as duality, Bregman balls, and centroids, and provide important theorems and examples.

Bregman Divergences

Definition 4.1 (Bregman (1967); Nielsen et al. (2007))**.** For any two points $\boldsymbol{p}$ and $\boldsymbol{q}$ of $\mathcal{X} \in \mathbb{R}^d$, the Bregman Divergence $D_F(.,.) : \mathcal{X} \to \mathbb{R}$ of $\boldsymbol{p}$ to $\boldsymbol{q}$ associated to a strictly convex and differentiable function F (also called *generator function*) is defined as:

$$D_F(\boldsymbol{p}, \boldsymbol{q}) = F(\boldsymbol{p}) - F(\boldsymbol{q}) - \langle \nabla F(\boldsymbol{q}), \boldsymbol{p} - \boldsymbol{q} \rangle \tag{4.1}$$

where $\nabla F = \left[\frac{\partial F}{\partial x_1}, \ldots, \frac{\partial F}{\partial x_d}\right]$ denotes the gradient operator and $\langle \boldsymbol{p}, \boldsymbol{q} \rangle$ the inner or dot product.

One can interpret the Bregman divergnce as the distance between a function and its first-order taylor expansion. In particular, $D_F(\boldsymbol{p}, \boldsymbol{q})$ is the difference between $F(\boldsymbol{p})$ and the linear approximation of $F(\boldsymbol{p})$ centered at $\boldsymbol{q}$.

The most interesting point about Bregman family of divergence is that they can generate many of the common *distances* in the literature. Several examples follow:

Example 1 (Euclidean Distance)**.** For $x \in \mathbb{R}$ in $F(x) = x^2$, eq. 4.1 would lead to $D_F = (p-q)^2$ or the Euclidian norm.

Example 2 (Kullback-Leibler Distance)**.** For $\boldsymbol{x} \in d-$Simplex (or $\sum_{j=1}^{d} x_j = 1$) and $F(x) = \sum_{j=1}^{d} x_j \log x_j$ or the negative entropy, D_F would amount to the famous Kullback-Leibler divergence:

$$D_F(\boldsymbol{p}, \boldsymbol{q}) = \sum_{j=1}^{d} p_j \log(\frac{p_j}{q_j}) = KL(\boldsymbol{p}||\boldsymbol{q}) \tag{4.2}$$

Example 3 (Itakura-Saito Distance)**.** If $\mathcal{X}$ is the power spectrum of $\boldsymbol{x}$, and $F(\mathcal{X}) = \frac{1}{2\pi}\int \log(\mathcal{X}(e^{j\theta})d\theta$ which is convex on $\mathcal{X}$, then the associated Bregman divergence between two power spectrums $\boldsymbol{p}$ and $\boldsymbol{q}$ is the Itakura-Saito distance widely used in signal processing applications:

$$D_F(\boldsymbol{p}, \boldsymbol{q}) = \frac{1}{2\pi}\int_{-\pi}^{\pi} \left(-\log\left(\frac{p(e^{(j\theta)})}{q(e^{(j\theta)})}\right) + \frac{p(e^{(j\theta)})}{q(e^{(j\theta)})} - 1 \right) d\theta \tag{4.3}$$

Many other important divergence functions commonly used in the literature can also be produced using Bregman divergences. Therefore, a common mathematical framework based on Bregman divergences can have important impact on many engineering applications. Basic properties of Bregman divergence are as follows (Banerjee et al., 2005):

Property 4.1 (Non-negativity)**.** *The strict convexity of the generator function F implies non-negativity with $D_F(\boldsymbol{p}, \boldsymbol{q}) = 0$ iff $\boldsymbol{p} = \boldsymbol{q}$.*

Property 4.2 (Convexity)**.** $D_F(\boldsymbol{p}, \boldsymbol{q})$ *is convex in its* first *argument $\boldsymbol{p}$ but not necessarily on the second.*

Property 4.3 (Linearity)**.** *Bregman divergence is a linear operator i.e. for any two strictly convex and differentiable functions F_1 and F_2 defined on $\mathcal{X}$ and $\lambda \geq 0$:*

$$D_{F_1+\lambda F_2}(\boldsymbol{p}, \boldsymbol{q}) = D_{F_1}(\boldsymbol{p}, \boldsymbol{q}) + \lambda D_{F_2}(\boldsymbol{p}, \boldsymbol{q})$$

Property 4.4 (Invariance)**.** *With $\boldsymbol{a} \in \mathbb{R}^d$ and $b \in \mathbb{R}$ given $G(x) = F(x) + \langle \boldsymbol{a}, \boldsymbol{x} \rangle + b$ strictly convex and differentiable on $\mathcal{X}$ we have $D_F(\boldsymbol{p}, \boldsymbol{q}) = D_G(\boldsymbol{p}, \boldsymbol{q})$.*

Property 4.5 (Generalized Pythagoras)**.** *For any triple p, q, r of points in $\mathcal{X}$, we have:*

$$D_F(\mathbf{p}, \mathbf{q}) + D_F(\mathbf{q}, \mathbf{r}) = D_F(\mathbf{p}, \mathbf{r}) + \langle \mathbf{p} - \mathbf{q}, \nabla F(\mathbf{r}) - \nabla F(\mathbf{q}) \rangle \tag{4.4}$$

Dual Structure

One of most important properties of Bregman divergences, due to the strict convexity of F, is their Legendre dual divergence. Let F be a strictly convex and differentiable real-valued function on $\mathcal{X}$. The Legendre transformation makes use of the duality relationship between points and lines to associate to F a *convex conjugate* function $F^* : \mathbb{R}^d \to \mathbb{R}$ given by (Rockafellar, 1970):

$$F^*(\boldsymbol{y}) = \sup_{x \in \mathcal{X}} [\langle \boldsymbol{y}, \boldsymbol{x} \rangle - F(\boldsymbol{x})] \tag{4.5}$$

The supremum is reached at the *unique* point where $\boldsymbol{y} = \nabla F(\boldsymbol{x})$. It can be shown that F^* is also convex (Nielsen et al., 2007). From now on, we denote the dual point of $\boldsymbol{x}$ as $\boldsymbol{x}' = \nabla F(\boldsymbol{x})$. Due to strict convexity of F, its gradient as well as the inverse of gradient are well defined. Then the convex conjugate F^* is the function $\mathcal{X}' \subset \mathbb{R}^d \to \mathbb{R}$ defined by:

$$F^*(\boldsymbol{x}') = \langle \boldsymbol{x}, \boldsymbol{x}' \rangle - F(\boldsymbol{x})$$

We can now introduce the important dual divergence property of Bregman divergences (Notice the inversion of $\boldsymbol{p}$ and $\boldsymbol{q}$ orders in the dual version):

Property 4.6 (Nielsen et al. (2007))**.** *D_{F^*} is also a Bregman divergence called the Legendre dual divergence of D_F and we have:*

$$D_F(\boldsymbol{p}, \boldsymbol{q}) = F(\boldsymbol{p}) + F^*(\boldsymbol{q}) - < \boldsymbol{p}, \boldsymbol{q'} >= D_{F^*}(\boldsymbol{q'}, \boldsymbol{p'})$$

In order to obtain D_{F^*} one would need to obtain F^* from F as $F^* = \int \nabla^{-1} F$. This integral does not necessarily have a closed-form solution, however the duality formulation of Bregman divergences would become significant when used in conjunction with exponential family of distributions and in problems with probabilistic frameworks where going back and forth between the natural parameter space and its convex conjugate space is plausible (see section 4.2.3).

Bregman Balls

In analogy to Euclidean geometry, we can define a *Bregman ball* (or sphere). Due to the assymetric nature of Bregman divergences, a Bregman ball can be defined as two counterparts which are *right-type* or *left-type*. This dualistic definition also holds for all other geometric definitions in our framework.

Definition 4.2. A Bregman ball of right-type centered at $\boldsymbol{\mu}_k$ with radius R_k is defined as:

$$B_r(\boldsymbol{\mu}_k, R_k) = \{\boldsymbol{x} \in \mathcal{X} : D_F(\boldsymbol{x}, \boldsymbol{\mu}_k) \leq R_k\} \tag{4.6}$$

Similarly, the Bregman ball of left-type $B_\ell(\boldsymbol{\mu}_k, R_k)$ is defined by inverting the divergence relationship in eq. 4.6 to $D_F(\boldsymbol{\mu}_k, \boldsymbol{x})$.

Bregman Centroids

Given a cluster of points, we are often interested in finding the *best single representative* of the cluster. This representative point is the *centroid* defined as the optimizer of the minimum average distance for the entire set of points in the cluster. The following theorem significantly states that the right-type centroid of a set of points $\mathcal{X} = \{x_i\}_{i=1}^n$ is independent of the choice of Bregman divergence. We provide its proof due to its crucial importance.

Theorem 4.1 (Banerjee et al. (2005))**.** *Let X be a random variable that takes values in $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^n \subset \mathbb{R}^d$. Given a Bregman divergence D_F, the* right type *centroid of $\mathcal{X}$*

defined as

$$\boldsymbol{c}_R^F = \operatorname*{argmin}_{\boldsymbol{c}} \sum_{i=1}^{n} \frac{1}{n} D_F(\boldsymbol{x}_i, \boldsymbol{c}) \tag{4.7}$$

is unique, independent of F and coincides with center of mass $\boldsymbol{\mu} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_i$.

Proof. The minimization above is equivalent to minimizing the function

$$J_F(\boldsymbol{s}) = \sum_{i=1}^{n} \frac{1}{n} D_F(\boldsymbol{x}_i, \boldsymbol{s}).$$

Assuming that $\boldsymbol{\mu} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_i$ we can see:

$$\begin{aligned}
J_F(\boldsymbol{s}) - J_F(\boldsymbol{\mu}) &= \sum_{i=1}^{n} \frac{1}{n} D_F(\boldsymbol{x}_i, \boldsymbol{s}) - \sum_{i=1}^{n} \frac{1}{n} D_F(\boldsymbol{x}_i, \boldsymbol{\mu}) \\
&= F(\boldsymbol{\mu}) - F(\boldsymbol{s}) - \langle \sum_{i=1}^{n} \frac{1}{n} \boldsymbol{x}_i - \boldsymbol{s}, \nabla F(\boldsymbol{s}) \rangle + \langle \sum_{i=1}^{n} \frac{1}{n} \boldsymbol{x}_i - \boldsymbol{\mu}, \nabla F(\boldsymbol{\mu}) \rangle \\
&= F(\boldsymbol{\mu}) - F(\boldsymbol{s}) - \langle \boldsymbol{\mu} - \boldsymbol{s}, \nabla F(\boldsymbol{s}) \rangle \\
&= D_F(\boldsymbol{\mu}, \boldsymbol{s}) \geq 0
\end{aligned}$$

where equality is reached only if $\boldsymbol{s} = \boldsymbol{\mu}$ due to strict convexity of F (See property 4.2). Therefore, $\boldsymbol{\mu}$ is the unique minimizer of J_F. □

Once again, due to general asymmetry of Bregman divergences, we can define a *left-type* centroid by reversing the order of computation in eq. 4.7. In order to obtain the left centroid c_L^F, we can now avoid exhaustive computation and easily use the dualistic structure of our information manifold. Combining theorem 4.1 and property 4.6 we can easily obtain:

$$c_L^F = (\nabla F)^{-1}(\sum i = 1^n \nabla F(x_i)) = (\nabla F)^{-1}(c_R^{F^*}(\mathcal{X}')) \tag{4.8}$$

stating simply that to obtain the left-type centroid we calculate the right-type centroid in the dual manifold $\mathcal{X}'$ using theorem 4.1 and convert it back to the original space given that $(\nabla F)^{-1}$ exists.

Bregman Information

Let $\boldsymbol{X}$ be a random variable following a probability ν that takes values in $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^n \subset \mathbb{R}^d$. Let $\boldsymbol{\mu} = E_\nu[X]$. Then the *Bregman Information* of X is defined as (Banerjee et al., 2005):

$$I_F(X) = E_\nu[D_F(X, \boldsymbol{\mu})] = \sum_{i=1}^{n} \nu_i D_F(\boldsymbol{x_i}, \boldsymbol{\mu}) \tag{4.9}$$

Example 4 (Variance)**.** If $\boldsymbol{X} \subset \mathbb{R}^d$ with uniform probability measure $\nu_i = \frac{1}{n}$, the Bregman information with squared Euclidean distance as Bregman divergence is just the variance of samples $\boldsymbol{x}_i \in \boldsymbol{X}$.

Example 5 (Mutual information)**.** The *mutual information* $I(\boldsymbol{X}; \boldsymbol{Y})$ between two discrete random variables $\boldsymbol{X}$ and $\boldsymbol{Y}$ with joint distribution $p(\boldsymbol{X}, \boldsymbol{Y})$ is given by:

$$I(\boldsymbol{X}; \boldsymbol{Y}) = \sum p(\boldsymbol{x}_i) KL(p(\boldsymbol{Y}|\boldsymbol{x}_i), p(\boldsymbol{Y}))$$

which can be expressed in terms of a random variable $\boldsymbol{Z}_x$ that takes values in the set of probability distributions $\mathcal{Z}_x = \{p(\boldsymbol{Y}|\boldsymbol{x}_i)\}_{i=1}^n$ following the probability measure $\nu_i = p(\boldsymbol{x}_i)$ and using Bregman information,

$$I(\boldsymbol{X}; \boldsymbol{Y}) = \sum_{i=1}^{n} \nu_i D_F(p(\boldsymbol{Y}|\boldsymbol{x}_i), \boldsymbol{\mu}) = I_F(\boldsymbol{Z}_x) \tag{4.10}$$

where $\boldsymbol{\mu} = E_\nu[p(\boldsymbol{Y}|\boldsymbol{x}_i) = p(\boldsymbol{Y})]$ and D_F is the Kullback-Leibler divergence.

4.2.3 Exponential Family of Distributions

Among different distribution families, the exponential family of probability distributions are of special importance and have found their way in many pattern recognition and engineering applications. A canonical definition of exponential family distributions is as follows:

$$p(x|\boldsymbol{\theta}) = \exp[< \boldsymbol{\theta}, \boldsymbol{f}(x) > - F(\boldsymbol{\theta}) + C(x)] \tag{4.11}$$

where $f(x)$ is the *sufficient statistics* and $\boldsymbol{\theta} \in \mathcal{X}$ represents the *natural parameters*. F is called the *cumulant function* or the *log partition* function. F fully characterizes the exponential family while the term $C(x)$ ensures density normalization. Since the sum (or integral) of a probability density function adds to one, it is easy to show that,

$$F(\boldsymbol{\theta}) = \log \int_x \exp\left[< \boldsymbol{\theta}, f(\boldsymbol{x}) > + C(x)\right]$$

showing that F *fully* characterizes the exponential family. It can be seen that many of the commonly used distribution families (such as normal, multinomial, Bernoulli etc.) can be generated by proper choice of *natural parameters* and *sufficient statistics* (Banerjee et al., 2005). As a final definition, we call the expectation of X with respect to $p(\boldsymbol{x}; \boldsymbol{\theta})$ the *expectation parameter* given by:

$$\boldsymbol{\mu} = \boldsymbol{\mu}(\boldsymbol{\theta}) = \int \boldsymbol{x}\ p(\boldsymbol{x}; \boldsymbol{\theta})\ d\boldsymbol{x}$$

Duality of natural and expectation parameters

It can be shown (Amari and Nagaoka, 2000) that the expectation and natural parameters of exponential families of distributions have a *one-to-one* correspondence with each other and span spaces that exhibit a dual relationship as outlined in section 4.2.2. To begin, note that:

$$\nabla F(\boldsymbol{\theta}) = \left[\int_x f(x)\ \exp\left\{< \boldsymbol{\theta}, f(x) > - F(\boldsymbol{\theta}) + C(x)\right\} dx\right] = \boldsymbol{\mu}$$

meaning that the expectation parameter $\boldsymbol{\mu}$ is the image of the natural parameter $\boldsymbol{\theta}$ under the gradient mapping ∇F. Now similar to eq. 4.5, we define the conjugate of F as

$$F^*(\boldsymbol{\mu}) = \sup_{\boldsymbol{\theta} \in \Theta} \{(\boldsymbol{\mu}, \boldsymbol{\theta}) - F(\boldsymbol{\theta})\}. \tag{4.12}$$

Due to the convexity of both F and F^*, they would be duals of each other and the following important one-to-one mapping holds between the two spaces:

$$\boldsymbol{\mu}(\boldsymbol{\theta}) = \nabla F(\boldsymbol{\theta}) \qquad \text{and} \qquad \boldsymbol{\theta}(\boldsymbol{\mu}) = \nabla F^*(\boldsymbol{\mu}) \tag{4.13}$$

4.2.4 Bregman Geometry and Exponential distributions

With the introduction above, we are now in a position to consider the geometrical properties and statistical consequences of a statistical manifold defined on exponential probability distributions using Bregman divergences. Such a statistical manifold is defined on a d-simplex $\mathcal{S}$ where each *point* is a discrete probability distribution $p(x;\boldsymbol{\theta})$ belonging to the *same* exponential family that are in return fully characterized by their *log partition function* F as seen in section 4.2.3. In this framework, the geometry is defined on the natural parameter space $\boldsymbol{\Theta}$.

Bijection with regular Bregman divergences

A natural question to ask at this point is: *What family of Bregman divergence to choose for a given family of exponential distribution?* The answer to this question lies in the important property of *bijection* between exponential families and Bregman divergences as shown and proved by Banerjee, Merugu, Dhillon, and Ghosh (2005). This property simply means that every regular exponential family corresponds to a *unique* and distinct Bregman divergence and vice versa – leading to a one-to-one mapping between the two. Due to the importance of this fact and for completeness we reiterate the main aspect of the theory here:

Theorem 4.2 (Banerjee et al. (2005))**.** *Let $p(\boldsymbol{x};\boldsymbol{\theta})$ be the probability density function of a regular exponential family of distribution with F as its associated log partition function. Let F^* be the conjugate function of F. Let $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ be the natural parameter and $\boldsymbol{\mu}$ be the corresponding expectation parameter. Then $p(\boldsymbol{x};\boldsymbol{\theta})$ can be uniquely expressed as*

$$p(\boldsymbol{x};\boldsymbol{\theta}) = \exp(-D_{F^*}(\boldsymbol{x},\boldsymbol{\mu}))b_{F^*}(\boldsymbol{x}) \tag{4.14}$$

where $b_{F^*}(\boldsymbol{x})$ is a uniquely determined function.

According to this theorem, the Legendre duality between F and F^* ensures that no two exponential families correspond to the same Bregman divergence i.e. the mapping is one-to-one. A similar theorem by Banerjee et al. (2005) assures the existence

of a regular exponential family corresponding to every choice of Bregman divergence (which is not detailed here since in our framework we are more interested in the one way relationship shown in the above theorem). These findings are crucial for our framework. They simply state that assuming a family of exponential family over our data will directly provide us with a Bregman divergence that constructs an information manifold over the geometry of the data and we can take advantage of all the theorems stated so far.

Example 6 (Multinomial Distribution)**.** A widely used exponential family is the *Multinomial distribution*:

$$p(\boldsymbol{x};\boldsymbol{q}) = \frac{N!}{\prod_{j=1}^{d} x_j!} \prod_{j=1}^{d} q_j^{x_j}$$

where $x_j \in \mathbb{Z}_+$ are frequencies of events, $\sum_{j=1}^{d} x_j = N$ and $q_j \geq 0$ are probabilities of events that sum up to 1. Below, we show that $p(\boldsymbol{x};\boldsymbol{q})$ can be expressed as the density of an exponential distribution in $\boldsymbol{x} = \{\ x_j\}_{j=1}^{d-1}$ with natural parameter $\boldsymbol{\theta} = \{\log \frac{q_j}{q_d}\}_{j=1}^{d-1}$ and cumulant function $F(\theta) = -N\log(q_d) = N\log(1 + \sum_{j=1}^{d-1} e^{\theta_j})$:

$$\begin{aligned}
p(\boldsymbol{x};\boldsymbol{q}) &= \exp\left(\sum_{j=1}^{d} x_j \log q_j\right) \frac{N!}{\prod_{j=1}^{d} x_j!} \\
&= \exp\left(\sum_{j=1}^{d-1} x_j \log q_j + x_d \log q_d\right) p_0(\boldsymbol{x}) \\
&= \exp\left(\sum_{j=1}^{d-1} x_j \log \frac{q_j}{q_d} + N \log q_d\right) p_0(\boldsymbol{x}) \\
&= \exp\left(\langle \boldsymbol{x}, \boldsymbol{\theta}\rangle - N \log\left(\sum_{j=1}^{d} \frac{q_j}{q_d}\right)\right) p_0(\boldsymbol{x}) \\
&= \exp\left(\langle \boldsymbol{x}, \boldsymbol{\theta}\rangle - N \log\left(1 + \sum_{j=1}^{d-1} e^{\theta_j}\right)\right) p_0(\boldsymbol{x}) \\
&= \exp(\langle \boldsymbol{x}, \boldsymbol{\theta}\rangle - F(\boldsymbol{\theta})) p_0(\boldsymbol{x})
\end{aligned}$$

with $p_0(\boldsymbol{x})$ independent of $\boldsymbol{\theta}$ and the third passage is obtained by using $x_d = N - \sum_{j=1}^{d} x_j$. The expectation parameter can then be easily computed as:

$$\boldsymbol{\mu} = \nabla F(\boldsymbol{\theta}) = [N q_j]_{j=1}^{d-1}$$

The Legendre dual F^* of F is

$$\begin{aligned} F^*(\boldsymbol{\mu}) &= \langle \boldsymbol{\mu}, \boldsymbol{\theta} \rangle - F(\boldsymbol{\theta}) \\ &= \sum_{j=1}^{d} N q_j \log q_j = N \sum_{j=1}^{d} \left(\frac{\mu_j}{N}\right) \log \left(\frac{\mu_j}{N}\right) \end{aligned}$$

Note that this F^* is a constant multiple of negative entropy and thus it is not surprising that the Bregman divergence bijection of the Multinomial distribution (according to theorem 4.2) would be constant multiple of the KL divergence (see example 2):

$$\begin{aligned} D_{F^*}(\boldsymbol{x}, \boldsymbol{\mu}) &= F^*(\boldsymbol{x}) - F^*(\boldsymbol{\mu}) - \langle \boldsymbol{x} - \boldsymbol{\mu}, \nabla F^*(\boldsymbol{\mu}) \rangle \\ &= N \sum_{j=1}^{d} \frac{x_j}{N} \log \left(\frac{x_j/N}{\mu_j/N} \right) \end{aligned}$$

The above derivations for Multinomial distribution will be later used in this and the following chapters. Due to this importance the summary of the above derivations for Multinomial manifold along some additional properties are provided in appendix A.1 on page 269.

Maximum Likelihood

A common goal of many statistical learning methods and applications is to find the best natural parameters given certain a priori constraints. More specifically, given a set of points or data in $\mathbb{R}^d$ we need to find the maximum likelihood distribution. The following proposition discusses this issue using what we have observed so far:

Proposition 4.1 (Maximum-Likelihood)**.** *The problem of finding the maximum likelihood on natural parameter space is equivalent to the right-type centroid following theorem 4.1, and thus equal to the sample mean and independent of the choice of Bregman divergence.*

Proof. Maximum-likelihood problem is often studied using the log-likelihood or simply the $\log$ of the likelihood formulation. In this problem setting we are interested in finding the optimum $\boldsymbol{\theta}$ for a set of independent random variables in the set $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^{N}$.

Using theorem 4.2, minimizing the negative log-likelihood is the same as minimizing the corresponding expected Bregman divergence of the exponential family distribution, or for an exponential family distribution $p_F(\boldsymbol{x};\boldsymbol{\theta})$ with log-partition function F,

$$\min\left[-\log\sum_{\boldsymbol{x}_i\in\mathcal{X}} p_F(\boldsymbol{x}_i;\boldsymbol{\theta})\right] \equiv \min\sum_{\boldsymbol{x}_i\in\mathcal{X}} D_{F^*}(\boldsymbol{x}_i,\boldsymbol{\mu}).$$

Now, following theorem 4.1 this minimization amounts to the optimal distribution that has $\boldsymbol{\mu} = \boldsymbol{E}[\mathcal{X}]$ as the expectation parameter. In other words, the problem of maximum-likelihood on natural parameter space $\boldsymbol{\Theta}$ is reduced to the $\boldsymbol{\theta}$ that corresponds to $c_R^{F^*} = \boldsymbol{\mu} = \nabla F^*(\boldsymbol{\theta})$ or $\boldsymbol{\theta} = \nabla^{-1}F^*(c_R^{F^*})$ (according to eq. 4.13). □

Note that estimation of maximum likelihood on natural parameter space by itself might lead to complex convex optimization problems which in this formulation is reduced to a sample mean calculation in the conjugate space thanks to bijection of exponential family distributions with Bregman divergences as well as the dualistic mapping between expectation parameter and natural parameter spaces.

Generalized Pythagoras theorem

Using our dualistic notation, we can rewrite eq. 4.4 of the generalized pythagoras formulation in property 4.5 as:

$$D_F(\mathbf{p},\mathbf{q}) + D_F(\mathbf{q},\mathbf{r}) = D_F(\mathbf{p},\mathbf{r}) + \langle \mathbf{p}-\mathbf{q}, \boldsymbol{r}'-\boldsymbol{q}'\rangle \tag{4.15}$$

where the second term on the right $\langle \mathbf{p}-\mathbf{q}, \boldsymbol{r}'-\boldsymbol{q}'\rangle$ has now a direct geometrical interpretation: It corresponds to the *angle* between the *geodesic* (equivalent of lines in Riemannian manifolds) connecting $\boldsymbol{p}$ to $\boldsymbol{q}$ and the geodesic connecting $\boldsymbol{r}'$ to $\boldsymbol{q}'$ in the conjugate space. Naturally, this term vanishes whenever the two geodesics are orthogonal to each other. In general, if $\boldsymbol{q}$ is the *projection* of $\boldsymbol{r}$ onto the simplex of $\boldsymbol{p}$ (denoted here as $\mathcal{X}$) then the right-most term above becomes negative and we gain the triangle-

inequality[1]. This projection is also equivalent to:

$$\boldsymbol{q} = \underset{x \in \mathcal{X}}{\operatorname{argmin}}\, D_F(\boldsymbol{x}, \boldsymbol{r})$$

which in return is also equivalent to the maximum likelihood formulation in proposition 4.1 or theorem 4.1. This observation will become significant when later on we consider Bregman divergences for similarity distances.

4.3 Music Information Geometry

The main goal of this work is to provide a theoretical framework within which we are able to capture information qualities and effectiveness. Using the tools introduced in the previous sections, we aim at providing a framework for *Music Information Geometry* where information geometries are constructed on-the-fly and their geometrical properties reveal important remarks on their information qualities and effectiveness.

4.3.1 Methodology

Our framework is strongly based on timed structure of music signals. A music signals is represented as sampled waveforms in time. In most signal processing front-ends, the signal is represented as overlapping windows of the time-domain signals as a vector $\boldsymbol{X}_{t_i}$ where t_i is time (in seconds) of the window center. For simplicity of notation, we drop the i index hereafter and use $\boldsymbol{X}_t$ instead where $t \in \mathbb{N}$. A common representation in most signal processing approaches to music and audio is the frequency distribution of the time-domain signal $\boldsymbol{X}_t$ as $\boldsymbol{S}_t(\omega)$. This can be obtained by various methods such as Fourier transforms, wavelet transforms etc. or a feature-space derived from such representation (or even directly from the time-domain signal). Figure 4.1 shows a simple diagram of such a signal processing front-end for a simple Fourier transform.

[1] If $\mathcal{X}$ is an *affine set*, then this term vanishes and gives rise to an equality (See Banerjee et al., 2005, Appendix A).

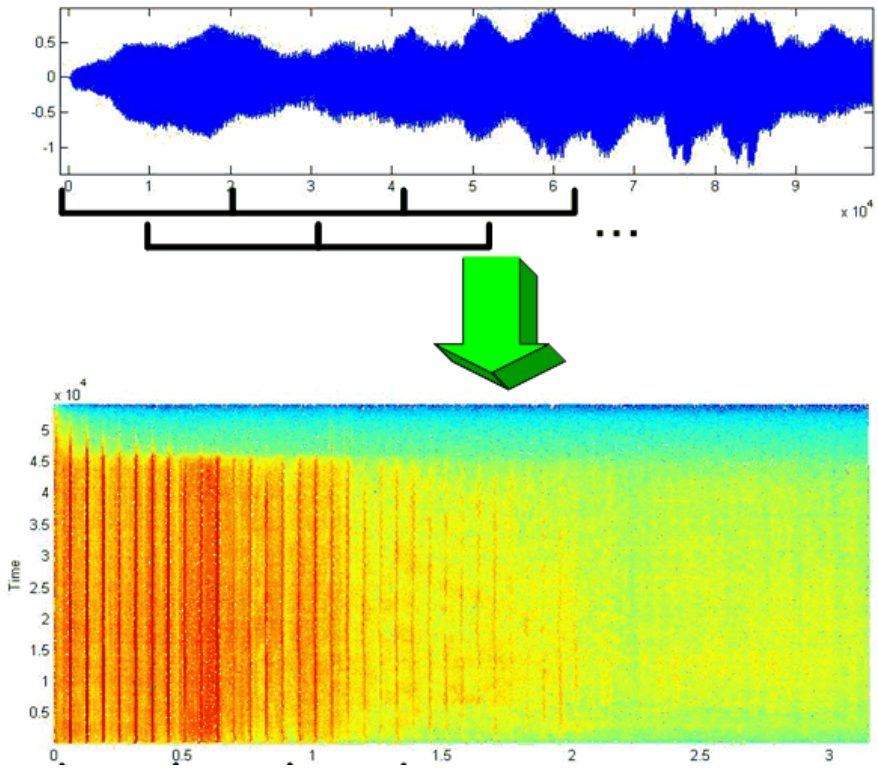


Figure 4.1: Signal Processing front-end

Given a front-end representation, we are interested in the information geometric structure of the statistical manifold created by $\boldsymbol{S}_t(\omega)$. The information manifold in this framework is described by $(\mathcal{S}, g, \Delta^D, \Delta^{D^*})$ as a dual information manifold using Bregman divergences (see section 4.2.1).

Evidently, an information geometric framework captures the information provided in the representational front-end. The framework that will be introduced shortly is independent of the representational scheme provided to the system and can be adapted using mathematical constructs if needed. The only assumption made on the representation is that it can be generated using one of the general families of exponential distributions. The choice of the parametric exponential family depends on the nature of the problem and can be considered as a priori information and, as stated earlier, provides choices among most of the distributions commonly used in many engineering applications. For example, one might prefer using mixtures of normal distributions on a specific

set of features for modeling applications of musical genre classification. For our general framework, we avoid a priori assumptions and try to be as general as possible by considering a form of representation as close as possible to the frequency spectrum of music and audio. Without loss of generality, we choose our representational front-end as a running Constant-Q power spectrum representation of the audio signal geared on a logarithmic musical scale (corresponding roughly to notes on a piano) for $\boldsymbol{S}_t(\omega)$ (Purwins et al., 2001).

To begin our study, we adopt Dubnov's distinction between (Dubnov, 2008) *Data Information Rate* and *Model Information Rate* where the first assumes a certain model over the signal and the second situates the on-going signal in relation to other models in the learned model space.

4.3.2 Data IR

We start our exploration by immediately proving Dubnov's Data-IR formulation in equation 2.1 (Dubnov, Aug. 2004) using the information geometry tools provided so far. The following theorem shows that this measure is equal to a special case of the Bregman Information as defined in section 4.2.2. As a consequence, this theorem shows that our information geometric framework would inherit automatically the nice information measure properties of Dubnov's measure such as its relevancy to listeners' expectations (Dubnov et al., 2006) and surprisal structures of natural sounds (Dubnov, 2006).

Theorem 4.3. *The Bregman Information corresponding to Itakura-Saito divergence over a time-series with corresponding power spectra* $\mathcal{S} = \{\boldsymbol{S}_i(\omega)\}_{i=1}^n$ *and uniform probability measure* ν *is equal to Dubnov Data-IR measure.*

Proof.

$$
\begin{aligned}
I_\phi(\mathcal{S}) &= \sum_{i=1}^{n} \nu_i D_F(\boldsymbol{S}_i, \hat{S}) \\
&= \sum_{i=1}^{n} \frac{1}{2\pi} \int_{-\pi}^{\pi} \left[-\log\left(\frac{\boldsymbol{S}_i(e^{j\omega})}{\hat{\boldsymbol{S}}(e^{j\omega})} \right) + \left(\frac{\boldsymbol{S}_i(e^{j\omega}}{\hat{\boldsymbol{S}}(e^{j\omega})} \right) - 1 \right] d\omega \\
&= -\frac{1}{2\pi} \int_{-\pi}^{\pi} \sum_{i=1}^{n} \nu_i \log\left(\frac{\boldsymbol{S}_i(e^{j\omega}}{\hat{\boldsymbol{S}}(e^{j\omega})} \right) d\omega \\
&= -\frac{1}{2\pi} \int_{-\pi}^{\pi} \log\left(\frac{\prod_{i=1}^{n} \boldsymbol{S}_i(e^{i\omega})}{\left[\frac{1}{n} \sum_{i=1}^{n} \boldsymbol{S}_i(e^{j\omega}) \right]^n} \right) d\omega
\end{aligned}
$$

where D_F is the Itakura-Saito divergence from eq. 4.3. The last equation above is simply the spectral flatness measure of the power spectra for n framed samples of data as in equation 2.1 and thus directly equals to *Data-IR* as put forward in (Dubnov, Aug. 2004).

□

4.3.3 Model IR

On the same lines as Dubnov (2008), to provide a more realistic framework and to relax the stationary assumption of *Data-IR*, we can assume a musically plausible hypothesis where the signal is stationary in a finite time-frame under a model $\boldsymbol{\theta}_k$ and described through $P(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n | \boldsymbol{\theta}_k)$. To assess this improvement and without loss of generality, we assume that $\mathcal{S}$ is generated by a regular exponential family of distributions $p_F(\boldsymbol{x}; \boldsymbol{\theta})$ with natural parameter space $\boldsymbol{\theta} \in \Theta$, expectation parameters $\boldsymbol{\mu}$, and log-partition function F. Given the bijection between regular Bregman divergences and exponential distributions (theorem 4.2), such assumption would automatically provide us with the associated Bregman divergence and its dual and all the nice tools and properties introduced in sections 4.2.2 through 4.2.4.

The regular exponential family chosen to represent our statistical manifold is the *multinomial distribution* whose Bregman divergence and dual maps have previously

been derived in example 6. Multinomial distribution is an exponential family that is commonly used in image and sound processing over histogram features (such as a normalized frequency distribution) without any dimensionality reduction ($N-1$ degree of freedom for the case $|\boldsymbol{S}_t| = N$), practically equal to the directly observed representation. Therefore, in this framework, input signals are represented as *Multinomial distributions* which are in fact normalized frequency distributions of the power spectrum $\boldsymbol{S}_t$ of the input time-domain signal vector $\boldsymbol{X}_t$. Appendix A.1 summarizes basic properties of Multinomial manifolds and provide the conversion to and back from it as needed throughout this and following chapters.

Within this framework, we assume that a continuous-time music signal constitutes *models* as quasi-stationary subsets of the continuous audio signal described by a regular exponential family in a dual Bregman geometry. In other words, *models* in our framework correspond to *stable* subsets of environmental information input to the system. By explicit consideration of incremental nature of the continuous signal (as a time-series), we now define a geometric framework to *form* these models *on-the-fly* using an information geometric framework. We start by an information geometric definition of *models*.

Definition 4.3 (Models)**.** Given a dual structure manifold $(\mathcal{S}, g, \Delta^D, \Delta^{D^*})$ derived on a regular exponential family formed on data-stream X_k, a *model* θ_i consist of a set $\mathcal{X}_i = \{\boldsymbol{x}_k | k \in \mathcal{N}, \mathcal{N} \subset \mathbb{N}\}$ that forms a ***Bregman Ball*** $B_r(\boldsymbol{\mu}_i, R_i)$ with center $\boldsymbol{\mu}_i$ and radius R_i.

From theorem 4.1, we know that for a set of points $\mathcal{X}_i$, the centroid of the Bregman ball above is simply the sample mean. Moreover, once a model is formed with its corresponding Bregman ball, the equivalence of an incoming point x_t to the model can be easily checked using the ball's radius and the Bregman divergence of the framework or $\boldsymbol{x}_t \in B_r(\boldsymbol{\mu}_k, R_k)$ if $D_F(\boldsymbol{x}_t, \boldsymbol{\mu}_k) \leq R_k$ where the Bregman ball and divergence are defined as the regular Bregman ball corresponding to the Multinomial distribution as derived in example 6 (see also appendix A.1).

A natural question to ask at this point is: How do we learn or form these models over data? If the environment emitting information is *fixed* and *atemporal*, then the problem of learning optimal models over data is equivalent to classical clustering and variants of hard-clustering for Bregman divergences as proposed in (Banerjee et al., 2005; Teboulle, 2007). However, due to the *continuous-timed* nature of music signals, and our interest in on-line discovery of information structures, we need to find ways to *incrementally* learn and form models as the music information unfolds in time. Before we introduce our incremental clustering method, we need to contemplate on the notion of *similarity* as often discussed in the literature, within our framework.

4.4 From Divergence to Similarity Metric

One of the main goals of our information geometric framework is to provide a mathematically sound framework where divergences can be used as close as possible to the notion of *similarity metrics*. In this section we study the loose notion of "degree of similarity" used in pattern recognition literature within an information theoretic framework. We then attempt to connect this notion to the notion of divergence in our information geometric framework.

In the field of Music Information Retrieval, Jonathan Foote has much been credited for promoting and using self-similarity measures for music and audio (Foote, 1997). The MIR literature on database search, structure discovery, query-based retrieval and many more, rely on Foote's general notion of *similarity* as a basis to compare and deduct music structures. As mentioned earlier in section 4.1, most of these methods have the underlying assumption that the employed similarity measures provide *metric spaces* while they are not. For example, using a simple Euclidean distance between audio feature spaces does not assure the triangle inequality (see below) and therefore equivalence of imaginary structs A and C while A and B, and B and C are similar, is not guaranteed. In this section, we try to formulate a general information-theoretic notion of *similarity* and discuss the necessary conditions for a metric similarity space

and study the realization of such notions in our information geometric framework.

We provide a different information geometric notion of similarity. Our theoretical view of subject is similar to (Sinanović and Johnson, 2007). Rather than analyzing signals for their relevant information content, we conceptually consider controlled changes of the relevant and/or irrelevant information and determine how well the signals encode this information change. We quantify the effectiveness of the information representation by calculating how different are the signals corresponding to the two information states. Because the signals can have arbitrary forms, usual choices for assessing signal difference like mean-squared error make little sense. Instead we rely on distance measures that quantify difference between the signals' probabilistic descriptions. The *abstract entities* of signals in our framework are *models* represented by the symbol $\boldsymbol{\theta}$ as discussed before. In this context, the symbol represents the *meaning* of the information that signal $\mathcal{X}$ encodes and is a member of a larger set of models.

Definition 4.4 (Similarity)**.** Two signals (or models) $\boldsymbol{\theta}_0, \boldsymbol{\theta}_1 \in \mathcal{X}$ are assumed to be *similar* if the information gain by passing from one representation to other is zero or minimal; quantified by $d_X(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1) < \epsilon$ which depends not on the signal itself, but on the probability functions $p_X(x; \boldsymbol{\theta}_0)$ and $p_X(\boldsymbol{x}; \boldsymbol{\theta}_1)$.

Now let us look deeper into the distance metric above. In all pattern recognition applications based on a notion of similarity, a similarity *metric* is employed for clustering or classification. Let Ω be a nonempty set and $\mathbb{R}^+$ be the set of non-negative real numbers. A *distance* or *metric* function on Ω is a function $d : \Omega \times \Omega \rightarrow \mathbb{R}^+$ if it satisfies the metric (in)equalities (Cilibrasi and Vitanyi, 2005):

Property 4.7. $d(\boldsymbol{x}, \boldsymbol{y}) = 0$ *iff* $\boldsymbol{x} = \boldsymbol{y}$

Property 4.8 (Symmetry)**.** $d(\boldsymbol{x}, \boldsymbol{y}) = d(\boldsymbol{y}, \boldsymbol{x})$

Property 4.9 (Triangle Inequality)**.** $d(\boldsymbol{x}, \boldsymbol{y}) \leq d(\boldsymbol{x}, \boldsymbol{z}) + d(\boldsymbol{z}, \boldsymbol{y})$

A geometric framework with the distance $d(.,.)$ as outlined above, can correctly cluster its domain into *equivalence classes* inherent within the definition of d. Our main

goal in employing an information geometric framework is to provide a framework where the definition of *divergences* on a statistical manifold would come as close as possible to a *metric* based on the our definition of similarity above. Considering our methodology described in section 4.3.1, we propose to use Bregman divergences over a dual statistical manifold of exponential family functions describing a signal $\mathcal{X}$ with the notion of *models* interchanged with the *symbols* in definition 4.4. It is important at this point to note that Bregman divergences in general are *not* metrics since properties 4.8 and 4.9 do not generally hold while property 4.7 is equivalent to basic Bergman property 4.1 as shown in section 4.2.2. Despite this inconsistency, our framework with its use of Bregman divergence is built on *models* rather than *points* where each model constitutes an optimal representation of a cluster of *points*. This fact allows us to approach the two missing properties in the following subsections.

4.4.1 Symmetrized Bregman Divergences

We noted earlier that Bregman divergences are not necessarily symmetric. For these assymetric Bregman divergences, a symmetrized Bregman centroid can be defined by the following optimization problem on the set $\mathcal{P} = \{\boldsymbol{p}_i\}_{i=1}^n \subset \mathcal{X}$:

$$\boldsymbol{c}^F = \underset{\boldsymbol{c}\in\mathcal{X}}{\operatorname{argmin}} \sum_{i=1}^{n} \frac{D_F(\boldsymbol{c}, \boldsymbol{p}_i) + D_F(\boldsymbol{p}_i, \boldsymbol{c})}{2} \tag{4.16}$$

Nielsen and Nock (2007) show that this optimization problem can be simplified to a constant-size system relying on the right-type and left-type centroids defined previously. Their approach generalizes that of Veldhuis and Klabbers (Jan 2003) on symmetrizing Kullback-Leibler divergence employing convex optimization to solve for the optimal c^F. They provide a new algorithm by extending and simplifying the former approach by using duality and introducing a simple geodesic-walk dichotomic approximation algorithm. Moreover, their approach is well adapted to problem settings where an exponential distribution with its associated Bregman divergence are assumed. We give a full account of their algorithm in appendix A.2.

Note that to solve eq. 4.16 an optimization scheme is inevitable in contrary to most literature that define (for example) symmetrized KL divergences as arithmetic or normalized geometric mean of the left-type and right-type centroids. Both approaches of Veldhuis and Klabbers (Jan 2003) and Nielsen and Nock (2007) empirically prove this remark on image and audio processing applications. For our framework, we adopt the geodesic-walk algorithm of Nielsen and Nock (2007) to solve for an optimal symmetric Bregman ball, extending our methodology defined previously to symmetrized Bregman divergences. Therefore, from hereon any occurrence of the word *centroid* inherently refers to *symmetrized centroids* according to eq. 4.16.

Other authors have different approaches in symmetrizing different Bregman divergences but besides the work of Nielsen and Nock (2007), we are aware of no generic framework for symmetrizing Bregman divergences in general. To cite a few, Johnson and Sinanović (2001) attempted an alternative symmetrization of KL divergence by introducing the *resistor-average* distance, via a harmonic mean (instead of the arithmetic mean above). Our reasoning for choosing the method proposed by Nielsen and Nock is based on two grounds: First, alternative formulations are based on intuitive averaging rather than formulating the problem as an optimization problem, and second, the formulation of eq. 4.16 comes close to the famous *J-divergence* that has been widely applied to pattern recognition and engineering problems.

4.4.2 Triangle Inequality

The triangle inequality does not generally hold for Bregman divergences. However, we saw in section 4.2.4 that for three points $\boldsymbol{x}$, $\boldsymbol{y}$, and $\boldsymbol{z}$, we *can* have

$$D_F(\mathbf{x}, \mathbf{y}) \geq D_F(\mathbf{x}, \mathbf{z}) + D_F(\mathbf{z}, \mathbf{y})$$

if and only if $\boldsymbol{z} = \operatorname{argmin}_{\boldsymbol{q} \in \mathcal{X}} D_F(\boldsymbol{q}, \boldsymbol{y})$ when $\mathcal{X}$ is a convex set containing $\boldsymbol{x}$. In a special case where $\boldsymbol{z}$ is the Bregman centroid over a set of point $\{\boldsymbol{x}_i\}_{i=1}^N$ in the convex-set $\mathcal{X}$, the mentioned assumption holds and consequently we have the triangle inequality. It turns out that the domain of the log-partition function associated to Multinomial

distribution for our framework (see appendix A.1) is simply $\mathbb{R}^{d-1}$ which is convex by definition. We will see in the next section that during the incremental model formation process, whenever the triangular inequality is needed to assess equivalent classes, it is being used within the described situation. It is worthy to note that any future extension of our methods to other classes of probability distributions should strongly consider and solve this issue.

4.5 Incremental Model Formations

Using the information geometric framework introduced in section 4.3.1 and in the context of data-stream analysis (such as music signals), we can devise a simple scheme to incrementally segment the incoming signals to quasi-stationary chunks defined by a radius R. Besides its direct geometric interpretation, the Bregman ball radius R defines the maximum *information gain* around a mean μ_k that a *model* (or ball) contains through the given Bregman divergence. To assess the formation of Bregman balls on continuous data-streams, we assume that this information gain for a given model is a *càdlàg*[2] function in time. This assumption is a direct consequence of our initial assumption that the signal is stationary in a finite time-frame under a model $\boldsymbol{\theta}_k$ and the *flatness* of our information geometry defined through dual Bregman divergences of exponential families. It also conforms to the intuitive nature of musical information often characterized by distinct events with an information *onset* implying a discontinuity with regards to the past. This way and in the ideal case, the information gain upon the arrival of each point is stable (or continuous) within R for an ongoing model $\boldsymbol{\theta}_k$ until a jump is seen at time t indicating a candidate for a new model, and continuous again after t within the new model θ_{k+1} until a new model emerges later on.

In an ideal situation, the set of points in a specific model would be a continuous subset of indexes in $\mathbb{N}$. However, a realistic system should consider outliers which come from observation noise, environmental noise, silence and more. We call such inputs

[2] right-continuous with left limits

non-events, in contrast to (musical) events that are quasi-stationary over a finite-time window and define them as follows:

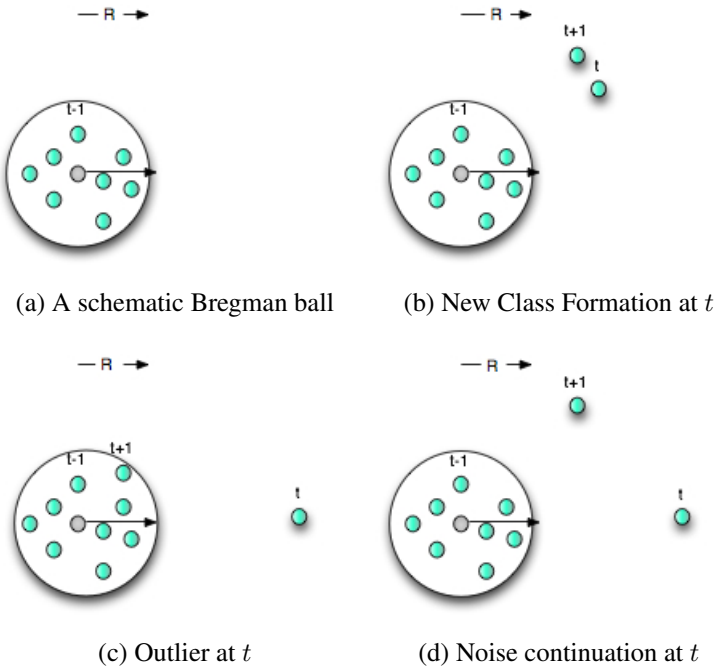
Definition 4.5 (Non-events)**.** In a data-stream $X_t = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t\}$, a *non-event* is determined by a non-stationary continuation in *information gain* defined by R with respect to the last formed model and a new model candidate. In other words, in a data-stream analysis with the latest model in $B_r(\boldsymbol{\mu}_k, R)$, $\boldsymbol{x}_t$ is considered a non-event if one of the following conditions hold:

$$\begin{cases} D_F(\boldsymbol{x}_t, \boldsymbol{\mu}_k) > R \\ D_F(\boldsymbol{x}_{t+1}, \boldsymbol{\mu}_k) \leq R \end{cases} \quad \text{or} \quad \begin{cases} D_F(\boldsymbol{x}_t, \boldsymbol{\mu}_k) > R \\ D_F(\boldsymbol{x}_{t+1}, \boldsymbol{\mu}_k) \geq R \\ D_F(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t) > R \end{cases} \tag{4.17}$$

Figure 4.2 demonstrates hypothetical diagrams showing three main situations that can occur upon arrival of information at time t and $t+1$. These diagrams are hypothetical since Bregman balls are not necessarily symmetrical and do not demonstrate a real ball-shape as in figure 4.2a. Figures 4.2c and 4.2d respectively correspond to the left and right group of equation 4.17 and represent two *non-events* as outliers and noise-continuation.

Using this simple segmentation scheme, we can decide whether the information arriving at time-frame t belongs to a previously formed model k by examining $D_F(\boldsymbol{x}_t, \boldsymbol{\mu}_k)$ against the fixed information gain R or whether it would be a non-event as defined above. From this simple scheme it is clear that the decision is made with a lag time of one analysis frame.

Running this simple algorithm on audio streams would result into segmentation of data $\mathcal{X} \in \mathbb{R}^D$ into separate Bregman ball regions represented by their double-sided centers $\mathcal{M} \in \mathbb{R}^D$ and $|\mathcal{M}| \ll |\mathcal{X}|$, that are characterized by their information stability within an information gain of R from their respective centers. The segmentation result is therefore dependent on the choice of R. Eventhough our goal here is not to provide a meaningful segmentation of audio signals, the results of this segmentation are quite significant and meaningful.



(a) A schematic Bregman ball (b) New Class Formation at t

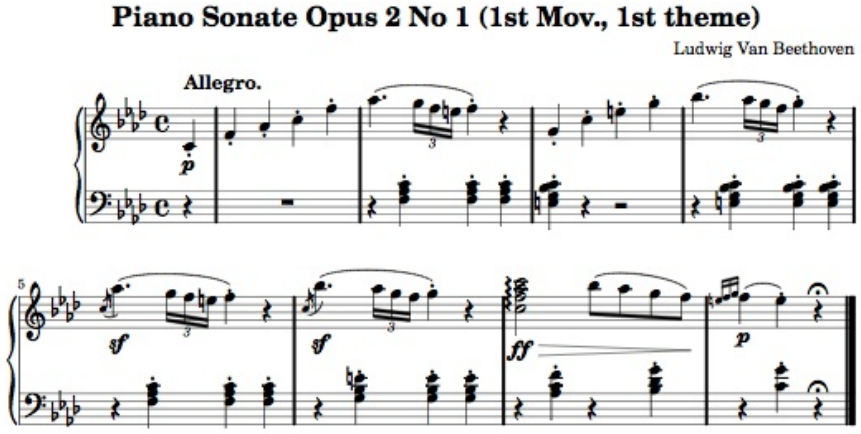
(c) Outlier at t (d) Noise continuation at t

Figure 4.2: Schematic diagrams for incremental class formations using imaginary Bregman balls. Points correspond to continuous-time input vectors.

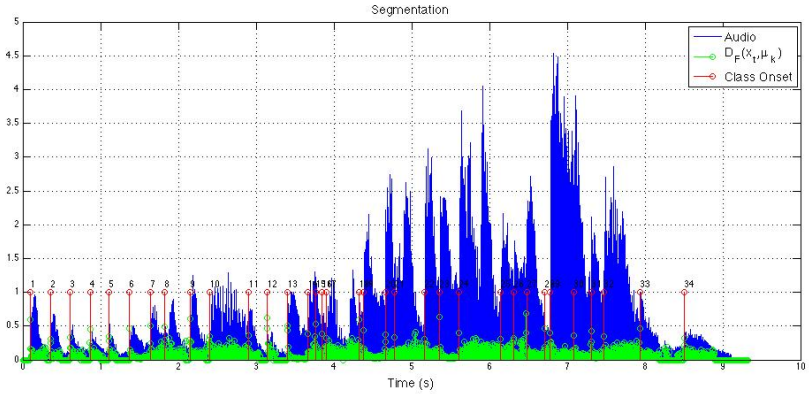
Example 7 (Real-time Music Segmentation)**.** Using the framework introduced in section 4.3.1, i.e. using normalized frequency distributions transfered to multinomial family of exponential distribution, and applying the simple algorithm above on the Bregman divergence of Multinomial distribution (see example 6 and appendix A.1) which is the Kullback-Leibler divergence, we can obtain real-time segmentation of audio streams corresponding to quasi-stationary *models*. Figure 4.3 shows the result of an old audio recording of the first theme of Beethoven's first sonata[3] (with the music score shown in figure 4.3a) for two different information gain or Bregman ball radius as thresholds of the segmentation.

In figure 4.3, the absolute value of the audio amplitude along with on-line values for class memberships or $D_F(x_t, \mu_k)$ and model onsets are represented over time. It can be noticed easily that the first 6 segments in both figures 4.3b and 4.3c correspond to the

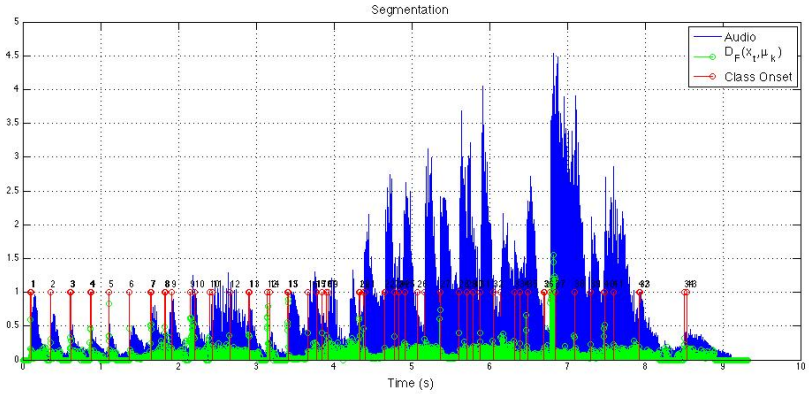
[3]Performed by Friedrich Gulda and published by DECCA-Philips Records, Catalog No. $00289 - 475 - 6835$, Original recording $1950 - 58$.



(a) Music Score of audio test



(b) Segmentation results with $R = 0.3$



(c) Segmentation results with $R = 0.15$

Figure 4.3: Segmentation results on the first theme of Beethoven's first sonata, performed by Friedrich Gulda (1950-58) with different R (information gain) thresholds.

first six music events in the music score of figure 4.3a.

It should be mentioned that the final *goal* of incremental model formation of our Music Information Geometry framework is *not* to obtain a segmentation that corresponds to musical elements of a music score. The fact that such a phenomena is observed in this example is a simple byproduct of using the right representational framework where the quasi-stationary model chunks correspond roughly to notes and chords that reconstruct the musical work.

4.6 Discussions

In this chapter we presented a framework for *Music Information Geometry* that emphasizes on the *influential* aspect of information to decode and represent music information for further quantifications and qualifications. Our framework is based on information geometry of statistical structures. In this framework, we assumed that music information arrives instantaneously and on-the-fly to the system as *points* on a manifold of exponential probability distribution and whose geometrical structures are defined by their associated Bregman divergence. The key theorem in allowing such assumption is the bijection between Bregman divergences and exponential probability distributions described earlier. Within this framework, we revisited Dubnov's *Information Rate* theory for music signals. We provided an alternative proof of his *Data-IR* and extended his *Model-IR* concept where models are defined as quasi-stationary chunks of audio describable through a Bregman ball with a fixed radius indicating the *information gain* or *extent*, and varying ball center within each model. We then provided a simple incremental algorithm for model formations in a music signals. This framework allows us to *access* relevant information within music streams which will be the topic of the next chapter.

In an ideal framework, the *information radii* should be varying much like the ball centers in the manifold. The basic intuition here is that two *models* might correspond to different extent of information and this extent of information is not necessarily fixed.

In our incremental model formation algorithm, we basically detect model instantiation by looking an instantaneous jumps in information novelty, indicated by comparing the membership of the newly arrived information to the last detected model. This was simply realized by using a threshold over the information gain. In an ideal framework, to allow varying information radii, one would prefer looking directly for *jumps* using dynamic stochastic models. This issue should be strongly considered for future expansion of the proposed methods.

We conclude this chapter by an important remark regarding the validity of the presented framework with regards to a common practice in the signal processing literature. It is commonly accepted among researchers to use Kullback-Leibler divergence as a metric among distributions and in some applications among raw frequency spectrums. This latter use has become lately common and has been experimentally validated by many researchers and for various applications (See for example Stylianou and Syrdal, 2001). Besides experimental assertions, we clearly showed in this chapter that Kullback-Leibler divergence is the canonical divergence associated to a statistical manifold of Multinomial distribution. When using KL on normalized spectrum analysis, we inherently consider them as Multinomial distributions. Therefore, this experimental fact can be theoretically explained through the bijection of Bregman divergences and exponential distribution manifolds of Multinomial family of exponential distributions. In general, KL can be used on any representation of signals that comes close to *histograms*. Therefore it is not surprising that another common use of KL in engineering literature is in image recognition and analysis on image histogram features.

Chapter 5

Methods of Information Access

In this chapter we visit the problem of Information Access and from two different view points: *representation* and *retrieval*. For the first, we seek a framework capable of representing temporal regularities of music signals and using the music information geometric structures described in the previous chapter. Such representational scheme should give fast access to relevant music entities in long sequences of music and should quantify to some extent the degree of relevancy of each entity with respect to the overall structure of music information. For the second, we present a retrieval algorithm over learned structures of music signals that allows fast access to relevant chunks of a system's memory given a query of audio signals. This method, proposed in section 5.2, extends traditional query-by-example systems in a way that results can be partial and could also correspond to a concatenation or re-assemblage of different chunks of the search domain.

5.1 Incremental Clustering and Structure Discovery

In the previous chapter, we introduced our framework's model formation as segmentation of data-stream into Bregman Balls with a fixed radius indicating the information gain relative to the double-sided centroid of the ball points. As a natural extension,

one would like to find equivalent segments (or Bregman balls) in the past as the stream unfolds in time and eventually based on these equivalencies obtain information on the overall structure of the stream in terms of regularities and repetitions. In other words, we are interested in obtaining an on-line procedure that can quickly access equivalent models that have occurred in the past and if possible, find the longest sequence of models in the past or a combination thereof that comes as close as possible to the ongoing data-stream. The first problem is usually referred to as clustering and the second as structure discovery. We are particularly interested in a fast method that can address both in a single shot.

The method we propose in this paper is based on a finite-state automata used for string matching and DNA sequences called *Factor Oracle* (Allauzen et al., 1999). Factor Oracles (FO) have also been successfully applied to symbolic music sequences for automatic improvisation and style imitations (Assayag and Dubnov, 2004) and will be revisited again in part III of this thesis. An extension of Factor Oracle for continuous data-streams first appeared in (Dubnov et al., 2007) called *Audio Oracle*. We base our model on this extension and further enhance it with our information geometric tools introduced in chapter 4.

5.1.1 Related Works

We start by introducing related works both on the application (audio structure discovery) and machine learning sides.

Audio Structure Discovery

There has been several attempts in the literature towards audio and music structure discovery algorithms directly from audio signals. Audio structure learning relies heavily in most present implementations on signal segmentation and can be viewed from two main perspectives: *model-based* approaches that incorporate certain musical or cognitive knowledge into the system in order to obtain structure boundaries, and *model-free*

approaches where the structure is learned directly from the audio itself without any incorporation of a priori knowledge of musical structures (Ong, 2007). For this work, we are interested in a model-free approach without incorporating any a priori rule-based or musicological knowledge. Among various model-free methods that have been proposed, we also focus our attention on how the temporal structure of an audio stream is derived given a set of audio features, whether it be a simple similarity matrix (Foote, 2000) or combinations of different audio features (Peeters, 2004; Rauber et al., 2002), without much concern for the audio representation itself (type of audio features used etc.).

Chai (2005) has proposed *dynamic programming* to perform music pattern matching for finding repetitions in music and later discovering the structure of music. The dynamic programming scheme provides a score matrix that uses a normalized Euclidean distance between two multi-dimensional feature vectors. In computing the score matrix, the author uses a finite-window over time. Actual matching alignment occurs by backtracking over the score matrix and repetitions can be detected by finding local minima. In another approach, Peeters (2004) uses Hidden Markov Models (HMM) and a multi-pass approach combining segmentation and structure discovery together. In each pass, different time-order similarity measures are run over audio feature vectors where the results estimate the number of classes and states for a K-means clustering that provides early parameters for learning of an ergodic HMM using Baum-Welch algorithm. In other approaches authors have attempted to learn audio structures by direct clustering methods such as k-means (Logan and Chu, 2000), Singular-Value Decomposition (Foote and Cooper, 2003) and more. Note that the mentioned methods are neither *on-line* nor incremental, and make use of future information to deduct temporal structures of underlying audio representation.

In this section, we present an *on-line* model-free algorithm for audio structure discovery that is capable of retrieving and representing long-term dependencies in the signal without using any ad-hoc windowing or Markov order parameterization.

Nearest-Neighbour and Tree Structures

On the machine learning side, our method is comparable in its aims to research on nearest neighbor (NN) clustering methods where given new data, its placement in a tree structure representing the structured data is searched. These methods are usually quite exhaustive in nature but because of tremendous practical and theoretical implications thereof in machine learning and many retrieval schemes, an extensive amount of research has been devoted to reduce the computational cost and complexity of finding NNs. KD-trees (Freidman et al., 1977) is among one of the earliest and most popular data structures for NN retrieval schemes. In such algorithms, a tree structure defines a hierarchical space partition where each node represents an axis-aligned rectangle. The search algorithm is then a simple branch and bound exploration of the tree. Metric balls (as proposed in Omohundro, 1989) extended these methods to metric spaces by using metric balls in place of rectangles. These search methods use the triangle inequality (see property 4.9) to prune out nodes and seem to scale with dimensionality better than simple KD-trees.

In many applications, an exact NN structure and search method is not required and a close equivalence would be good enough. The structures learned through tree structures would also seem too exhaustive both in computation and complexity during search and structure learning. Due to these practical considerations many researchers have considered *approximate* NN search methods, leading to significant breakthroughs such as in (Liu et al., 2005) where metric balls can overlap.

In this work, we are interested in a data structure where retrieval and bucketing can be done as fast as possible and ideally in linear time and space. Moreover, we would like this representation to be able to reveal regularities in the data-stream in terms of repetitions and recombinations to grasp an on-line structure of the overall signal. Our method is similar in its goals to that of the *approximate* NN literature on metric balls but uses a significantly different approach using on-line learning of state-space structures where the notion of tree structures are inherent but accessible. We introduce

our proposed method in the following sections. We first start by formally defining the structure of *Audio Oracle* and its structural implications and then move on to algorithmic construction and learning of the oracle itself.

5.1.2 *Audio Oracle* Data Structure

Audio Oracle is a fast automaton learning procedure that was motivated by a similar technique used for fast indexing of symbolic data such as text called *Factor Oracle* (Allauzen et al., 1999). In this section, we study the basic structure and properties of Factor Oracles as they are basically the same as in Audio Oracles. A sequence of symbols $S = \sigma_1, \sigma_2, ...\sigma_n$ in a Factor Oracle P is learned as a state-space diagram, whose states are indexed by $0, 1, \ldots, n$. There is always a transition arrow (called the *factor link*) labelled by symbol σ_i going from state $i-1$ to state $i, 1 < i < n$. Therefore, navigating a FO from state 0 to n would result in generating the original sequence S. Depending on the structure of S, other arrows will be added to the automaton: Transitions directed from state i to state j ($i < j$) that belong to the set of factor links and are labelled by symbol σ_k, are denoted by $\delta(i, \sigma_k) = j$. And some transitions that are directed *backwards*, going from a state i to a state j $(i > j)$ called *suffix links*, bearing no label and denoted by $S_P(i) = j$.

The factor and suffix links created during FO learning have direct structural interpretations. A *factor link* going from state i to j ($i < j$, labeled with σ_ℓ) indicates that a (variable length) history of symbols immediately before i is a common *prefix* of the sequence of symbols leading to j, or in other words $\{\sigma_{i-k} \ldots \sigma_i\} \cup \sigma_\ell = \{\sigma_{j-k-1} \ldots \sigma_j\}$. A *Suffix link* in oracle P that points from state m to an earlier state k or $S_P(m) = k$, models a factor automaton in a sense that states m and k would *share* the longest *suffix* (or history). In other words, a suffix link goes from i to j if and only if the longest repeated suffix of $s[1 \ldots i]$ is recognized in j. Thus, suffix links connect repeated patterns of S. Moreover, the length of each repeating factor pointed to by a suffix link can be computed in linear time and denoted as $lrs(i)$ for each state i (Lefebvre and Lecroq,

2000). The following simple example should give an intuitive view of FO structure:

Example 8. The Factor Oracle constructed for the text string `abbbaab` is shown in figure 5.1 with transition arrows shown as regular lines and suffix links as dashed. By following forward transitions and starting at state 0 one can generate factors such as `bbb` or `abb`. Repeated factors such as `ab` are connected through suffix links. The suffix link from state 4 to 3 indicates the longest common suffix between the two states (bb in this case), as well as the suffix link from state 7 to 2 (indicating ab). The factor link labeled with a from state 3 to 5 indicates that bb is a common prefix the two states etc.

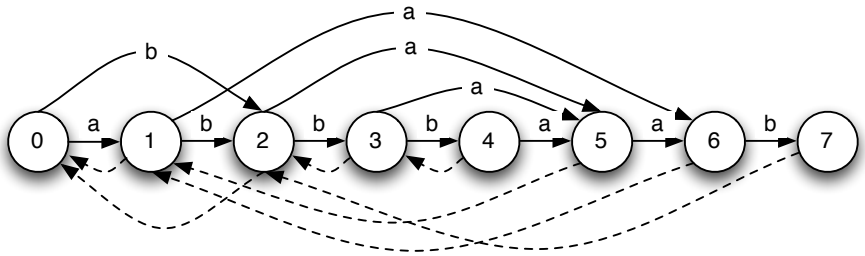
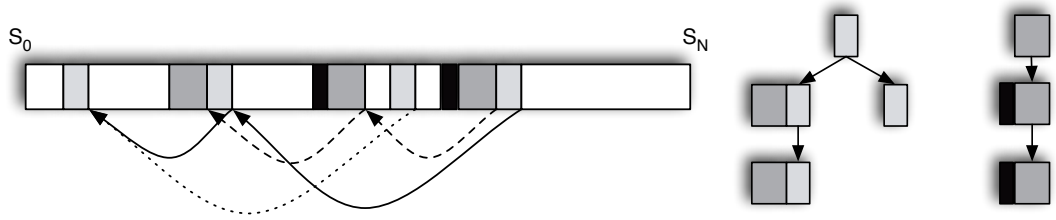


Figure 5.1: The Factor oracle for string `abbbaab`.

The fact that *suffix links* point to the *longest* repeated suffix between two states make them particularly attractive for our purpose. Figure 5.2a shows schematically how maximum length repeated factors are interconnected by suffix links. The thickness of the lines represents the length of the repeated factor. This length is computed at no additional cost by the oracle algorithm, and we will see later that it provides a very important clue in revealing the information structure. Assayag and Bloch (2007) also show that following each suffix link from the head of a Factor Oracle structure to the very beginning provides a forest of disjoint tree structures whose roots are the smallest and leftmost patterns appearing in the trees. A fundamental property of these *Suffix Link Trees (SLT)* is that the pattern at each node is a suffix of the patterns associated to its descendants. This way, *SLT*s capture all the redundancy organization inside the learned sequence. Figure 5.2b shows two of the three suffix link trees that come directly out of the suffix structure of figure 5.2a.



(a) Suffix Structure Diagram (b) Suffix Link Tree Diagram

Figure 5.2: The Suffix structure and Suffix Link forest of disjoint trees.

Probably the most significant property of FO structures is their *inherent* variable-length dependency structure. As mentioned in section 5.1.1, many audio structure discovery algorithms suffer from modeling long-term dependency structures and use hand-assigned parameters such as time-windows over history, fixed n-gram or Markov orders in order to grab long-term regularities common in music information. As seen above, Audio Oracle inherently handles long-term structures and moreover provides disjoint tree structures that reveal patterns of regularities during the life of an audio stream.

An extension of FO to continuous audio signals was first introduced in (Dubnov, Assayag, and Cont, 2007) where each audio analysis frame would constitute a state and the equivalence relationship was replaced by an Euclidean norm with a threshold as metric between different states. Despite significant results, the fact that the Euclidean norm does not constitute a similarity metrics in the chosen domains resulted in discontinuities in the results and loss of long-term structural dependencies.

In this work, we extend this approach to our music information geometric framework that explicitly provides a framework for using derived divergences of a statistical manifold as similarity metrics as shown in chapter 4. Along the methodology of section 4.3.1, we present Audio Oracle (AO) within two paradigms: The first considers a metric ball space where each state constitutes a Bregman balls over pre-segmented *models* as introduced in section 4.5 and represented by their double-sided centroid. This setup, referred to as *Model-AO*, is quite useful for structure discovery over long (and realistic) audio streams as the resulting state-space is sparse and correspond to global

structure of the audio stream. In the second, called *Data-AO*, the entities entering the system are *points* over a statistical manifold (rather than pre-segmented *models*). The main idea behind this separation is that for some applications a fine-grain and detailed structure is desired which implies a *data-based* rather than *model-based* access to information. A typical case of such application is studying the structure of constantly varying spectrum sounds (such as natural sounds). *Points* can be directly derived from audio observations once the exponential distribution manifold is defined by the user. Such transition is possible by direct conversion of normalized audio feature representations (which after normalization can simulate a probability mass function) to our defined exponential probability distribution. For our chosen framework of Multinomial distribution this conversion is direct (see appendix A.1 for derivations). This conversion is necessary to obtain the correct metric space as discussed in section 4.4.

5.1.3 *Audio Oracle* Learning and Construction

The oracle is learned on-line and incrementally. The learning and update algorithm represented here is an extension of the symbolic FO algorithm in (Allauzen et al., 1999) to the continuous audio domain. The important factor that allows such passage is the construction of FO over a manifold where similarity measures can be considered (approximately and closely enough) as metric spaces thus allowing the notion of equivalence classes in the signal domain. The algorithms presented here are thus based on a pre-defined music information geometry as defined in chapter 4, where the predisposed exponential distribution provides a symmetrized Bregman divergence $D_F(.,.)$ used here as a similarity metric.

The algorithms for forming *Model-AO* and *Data-AO* are the same and differ only on the type of input entities. For *Model-AO*, the entering entities are *models* formed incrementally as described in section 4.5 represented by their double-sided centroids $\boldsymbol{\mu}_i \in \mathbb{R}^d$, and for *Data-AO* entities are *points* of the defined manifold. For each new entering entity, a new state i is added to the sequence and an arrow from state $i-1$ to i

is created with label $\boldsymbol{\mu}_i$ (the Bregman ball's double-sided centroid). The algorithm then updates the transition structure by iteratively following the previously learned structure backwards through available factor links and suffix links in order to create new ones according to their similarities.

Algorithms 5.1 and 5.2 demonstrate psuedo-codes for Audio Oracle construction. During the online construction, the algorithm accepts Bregman balls centroids as *models* or *vector points* as entities $\boldsymbol{\sigma}_i$ coming from an exponential distribution manifold supplied with its associated Bregman divergence up to time t^i, and incrementally updates the Audio Oracle. Algorithm 5.1 shows the main online construction algorithm. Algorithm 5.1 calls the function `Add-Model` described in algorithm 5.2 which up-

Algorithm 5.1 On-line construction of *Audio Oracle*

Require: Audio entities as a stream $\boldsymbol{\Sigma} = \boldsymbol{\sigma}_1\boldsymbol{\sigma}_2 \cdots \boldsymbol{\sigma}_N$
1: Create an oracle P with one single state 0
2: $S_P(0) \leftarrow -1$
3: **for** $i = 0$ to N **do**
4: $\quad Oracle(P = p_1 \cdots p_i) \leftarrow$ Add-Frame $(Oracle(P = p_1 \cdots p_{i-1}), \boldsymbol{\sigma}_i)$
5: **end for**
6: **return** Oracle $(P = p_1 \cdots p_N)$

dates the audio oracle structure using the latest entity. Similarity between two *entities* is assured by using a small threshold ϵ checking for *closeness* in terms of information between two model centroids using the provided Bregman diveregnce. This algorithm traces the Suffix Link Trees backward and follows their forward transitions to find equivalent models to the new incoming ball. It returns new transition and suffix links as well as the length of the longest repeating suffix for each state (or `LRS`) by calling the function `Compute-LRS` as described in algorithm 5.3.

The `Compute-LRS` function of algorithm 5.3 is another incremental algorithm that traces back the suffix link tree structures in search of continuations that can constitute the longest repeating sequence in the present time that has occurred in the past.

Upon the construction of an Audio Oracle, the overall structure consists of forward arrows $\delta(i)$, suffix links $S_p(i)$ and the longest repeating suffix length corresponding

Algorithm 5.2 `Add-Model` function: Incremental update of *Audio Oracle*

Require: Oracle $P = p_1 \cdots p_m$ with suffix links in $S_P(i)$ and transitions in $\delta(i)$ for each state p_i; and new vector entity $\boldsymbol{\sigma}$

Create a new state $m+1$
Create a new transition from m to $m+1$, $\delta(m) = m+1$
$k \leftarrow S_P(m)$ and $\pi_1 \leftarrow S_P(m)$
while $k > -1$ **do**
  Set $I = \{i \in \delta(k) | D_F(\boldsymbol{\sigma}_{\delta(k)}, \boldsymbol{\sigma}) < \epsilon\}$
  if $I == \emptyset$ **then**
    Create a transition from state k to $m+1$; $\delta(k) = \delta(k) \cup \{m+1\}$
    $k \leftarrow S_P(k)$
    $\pi_1 = k$
  end if
end while
if $k = -1$ (no suffix exists) **then**
  $S_P(m+1) \leftarrow 0$
else
  $S_P(m+1) \leftarrow$ where leads the *best* transition (min. distance) from k
end if
$lrs(m+1) \leftarrow$ Compute-LRS(P, pi_1)
return Oracle $P = p_1 \cdots p_m \sigma$

Algorithm 5.3 `Compute-LRS` function: Longest Repeating Sequence length calculation of *Audio Oracle*

Require: Oracle $P = p_1 \cdots p_{m+1}$ with suffix links in $S_P(i)$, transitions in $\delta(i)$, previous LRS in $lrs(i)$ and π_1

$\pi_2 \leftarrow S_P(m+1) - 1$
if $\pi_2 == S_P(\pi_1)$ **then**
  return $lrs(\pi_1) + 1$
else if $\pi_2 \in \sigma(S_P(\pi_1))$ **then**
  return $lrs(m) + 1$
else
  while $S_P(\pi_2) \neq S_P(\pi_1)$ **do**
    $\pi_2 \leftarrow S_P(\pi_2)$
  end while
  return $\max(lrs(\pi_1), lrs(\pi_2)) + 1$
end if

to each state in $lrs(i)$. Each state of the Audio Oracle would then refer either to a Bregman ball for *Model-AO* or individual vector points for the *Data-AO*. In the case of the former, the models are formed incrementally over time and consists of the ball's center $\boldsymbol{\mu}_k$ and a set of (implicit) points in the original data-stream that can be accessed by their time-tags $\boldsymbol{t}^k = \{t_1^k, t_2^k, \ldots, t_N^k\}$ with N indicating the duration of the corresponding *model* in terms of analysis frames and the value of t_1^k as the onset time of the model in the data-stream as shown previously in section 4.5.

5.1.4 Sample Results

Given a learned Audio Oracle structure of a music data-stream, the suffix links would reveal the repeating structure of the ongoing signal and their corresponding longest repeating sequence length or lrs. These two simple measures can reveal the structural regularities of music directly from audio signals, which is crucial in many applications of music information retrieval. Below we look at sample results for both *Model-AO* and *Data-AO*.

Model-AO Results

Figure 5.3 shows the learned AO structure over Beethven's first piano sonata as performed by Friedrich Gulda (recorded between $1950-1958$). The three subplots show the audio waveform, the suffix structure (as pointing the present time (in seconds) to a past time), and the length of repeating sequence lrs associated to each state respectively. The suffix (or SFX) subplot should be read as follow: A time t on the x-axis would send a pointer back to a time t' $(t' < t)$ indicating the longest common suffix (in terms of similarity) between a factor at time t and t'. The corresponding value for t on the lrs subplot would reveal the length of the detected longest sequence (in terms of number of states) associated to that state-time.

A close look at the the suffix and LRS structures would reveal a striking relation: between time $50s$ and $100s$, the suffix links are systematically pointing to times $0s$ to

$50s$. Consequently, we see monotonically increasing lrs behavior in the same interval. Such a behavior is the result of *exact pattern repetitions*. A close look at the symbolic music score of the piece or simply listening to the audio excerpts reveals the nature of such behavior: The audio excerpt corresponding to the time interval $[50s, 100s]$ is an exact repetition of the *theme* of the sonata as indicated in the score by Beethoven. Relations such as this, but in smaller scale, are numerous in a piece of music. Another example in the same figure are audio subclips at time interval $[150s, 160s]$ and $[180s, 200s]$ which are also repeating the first theme with variations (belonging respectively to the *development* and *recapitulation* sections of a classical sonata form), revealed also in the lrs structure.

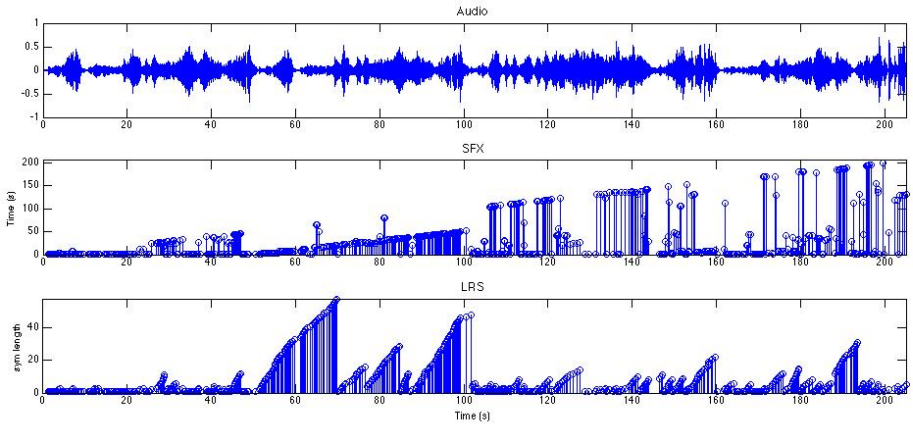


Figure 5.3: Incrementally learned Audio Oracle structure in Beethoven's Piano Sonate 1-movement 1, interpreted by Friedrich Gulda.

Figure 5.4 shows another example on Beethoven's first piano sonata, 3^{rd} movement played by the same performer and taken from the same recording. In this example, we have provided the original structure extracted from simple musicological analysis of the original music score in terms of structure blocks labeled by `A`, `B` and `C`. As can be seen in the figure, this particular piece goes through various structural repetitions and recombinations which are mostly captured by the Audio Oracle structure. We should note that this analysis is being done on an audio recording of a *human performance* of

the piece of music and therefore *repetitions* are never *exact*. Nevertheless, the Audio Oracle structure still reveals long term dependencies in the overall data-stream.

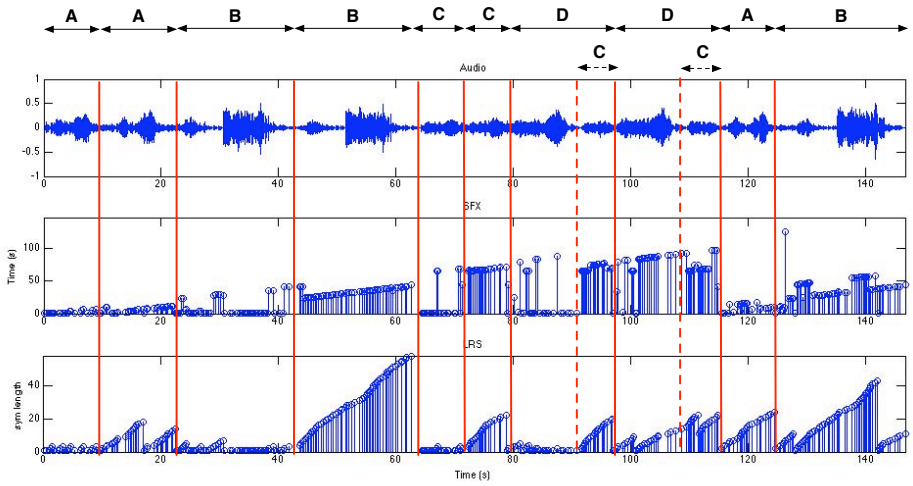


Figure 5.4: Incrementally learned Audio Oracle structure along with the segmented structure in terms of blocks from the orignal symbolic music score – Beethoven's Piano Sonate 1-movement 3, interpreted by Friedrich Gulda.

The *Audio Oracle* structures for both examples above were calculated using the framework presented in section 4.3.1, and real-time simulation of Audio Oracle learning on Constant-Q spectrum of audio with an analysis window of approximately 32 *milli-seconds* and an overlap factor of 2. The information radius for *Incremental Model Formation* algorithm of section 4.5 is set to 0.3 and ϵ of Audio Oracle learning to 0.1 for both examples. The analysis in figure 5.3 leads to approximately 13000 analysis frames mapped onto 650 models and states in the oracle. The example of figure 5.4 leads to 9500 analysis frames and 440 learned models and states. Note that all the algorithms proposed here are incremental and do not access previous analysis frames in the memory. They only store models (centroids along their indexes) as data arrives. Audio oracle leaning, as mentioned before, has linear complexity in time and space making it feasible for real-time calculation and storage of long audio structures.

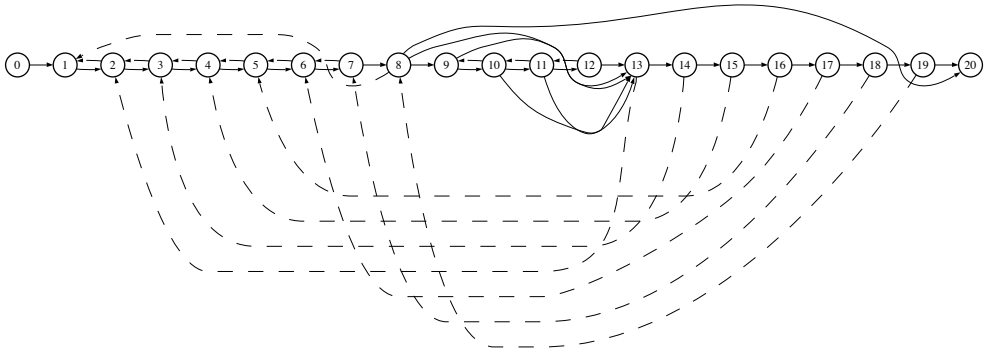
***Data-AO* results**

As mentioned previously, Audio Oracle can also be used in a *data* setting (i.e. without model formations and directly on data streams projected on a manifold). Figure 5.5 shows the *Data Audio Oracle* calculated over a natural bird utterance with a natural repetition. The learned Audio Oracle structure is shown in figure 5.5a where each state has a one-to-one correspondence with an analysis frame (based on Mel-Frequency Cepstrum as shown in figure 5.5c). The suffix links in figure 5.5a are represented as dashed-lines and correspond to detected repetitions of the second occurrence of the bird's utterance. For this analysis a time-window of $92ms$ with overlap factor of 2 is being used along for calculation of Mel-frequency Cepstrum Coefficients as the front-end for the Multinomial manifold, and using $\epsilon = 0.3$ during *Data-AO* learning.

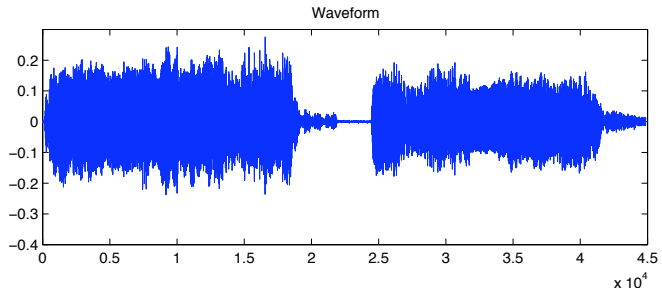
5.1.5 Discussions

In this section, we presented the *Audio Oracle* algorithm, as an unsupervised and incremental live algorithm for audio structure discovery. Audio Oracle was built on top of the Music Information Geometry framework introduced in chapter 4 which ensures that the divergences used in the algorithm can be considered as a metric space among other utilities it provides. We further showed that the AO structure inherently gives access to a forest of disjoint tree structures that represent various regularity patterns in a stream of audio.

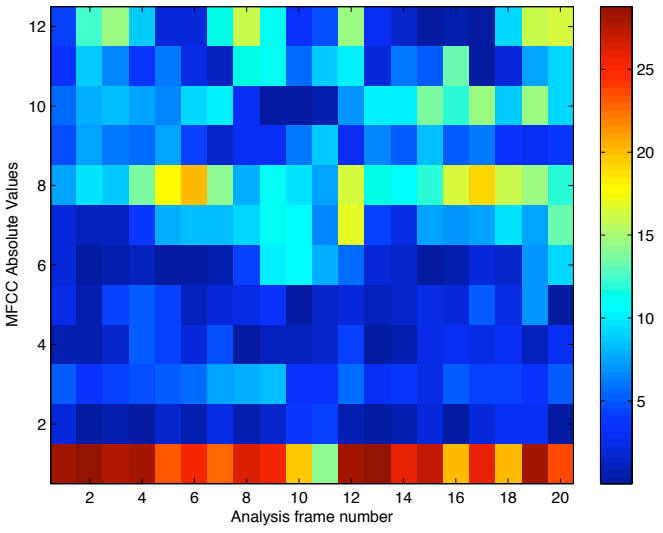
We presented Audio Oracle in two versions: *Model-AO* where the state-space structure is learned upon learned *models* providing sparse and compact representations of long audio. Our results show that learned *Model-AO*s show regularity structures that can also be observed either through listening or by analyzing the music score. Note that the algorithm does not assume any a priori musical knowledge. In the second version, we presented *Data-AO* with an aim of accessing fine-grain micro-structures of continuous audio (without any pre-segmentation), showing some results on natural sounds. *Data-AO* will be sufficiently exploited in the next section for retrieval applications.



(a) Learned Audio Oracle



(b) Sample Audio Waveform – two natural bird utterances



(c) Mel-Frequency Cepstral Coefficients features

Figure 5.5: *Data Audio Oracle* sample on natural bird utterances.

The Audio Oracle's computational complexity is linear in time and space. Note that while traditional self-similarity metrics make use of the whole spectrum of an audio stream against each frame to calculate self-similarity matrices, in Audio Oracle and at each step the algorithm only calculate the distance of the incoming frame to a limited and sparse set of previous entities which depends solely the structural regularities of the audio stream seen up to that point.

Audio Oracle enables *fast access* to audio content based on their structural similarities. These two facts alone, make Audio Oracle very attractive as a front-end for many music information retrieval applications and probably as a replacement for traditional self-similarity measures. This idea should be strongly pursued in future research.

5.2 *Guidage*: Fast Query-Based Information Retrieval

In the previous section we showed that learned *Audio Oracle* structures give fast access to forests of disjoint tree structures where each tree structure models variable-length patterns of regularities over the whole audio stream pertained to a specific audio content. Having easy access to these patterns, a natural question to ask is whether we can use this property in a retrieval framework and bring to the front the subclips (or combinations thereof) from the memory using a simple query. Also the *Suffix Link* and *lrs* information associated with the running audio can reveal to what extent a newly arrived chunk of audio is *novel* given *its own* past. Furthermore, a natural question to ask is whether we can obtain a similar measure of information relevancy of an audio chunk but pertained to an external set of music data rather than the running audio itself.

The two questions raised above share the same technological concerns and answers to them touch many applications in computer music and music information retrieval (MIR) literatures. The common grounds in both questions are *audio query* and *concatenative synthesis* systems, while their scopes extend from audio matching, query-

by-example, audio-content retrieval, and audio search engines as MIR systems to unit selection, concatenative texture synthesis, automatic improvisation and style imitation systems in computer music as well as access to information aspect of audio, external to the stream itself. In this section, we address the two concerns within one framework and examine results within retrieval applications of the proposed method. Due to the extent of the topic, we begin our discussion by looking at the scope of this research and a general discussion of the proposed method.

5.2.1 Research Scope

Quite often in signal processing applications, whether for search or synthesis purposes, we are interested in selecting the most proper *unit* given a *context*. This problem is often referred to as *unit selection* in concatenative synthesis literature and is also the main concern of information retrieval over sequential data (such as video and music). In this section, we propose a method called *Guidage* for efficient retrieval of sequential information given a query as input. An applicative advantage of *Guidage* is its capability to reassemble subclips or chunks in the search target wherever appropriate in order to reconstruct the query if exact repetition of the query is not available in the database. Therefore, *Guidage* aims at directly solving the unit selection problem common in concatenative synthesis and query search problems.

Guidage is a unique combination of two main technologies - concatenative synthesis and audio query - used together in order to achieve audio content-based control over sound synthesis as well as revealing the relevancy of external data to an audio query. The system uses a sound source (a query) to create new content from a variety of sounds over a large database of target sounds. The unique property of the system is its capability to match variable length clips of the query input to new audio materials, which are used for recombination and concatenation from the target database. The uniqueness of our approach comes from the fact that the search domain of the algorithm is *not* the audio domain itself but a compact state-representation of its internal structure

to better handle context-sensitivity and temporal morphologies of sounds. *Audio Oracle* structure (in its two forms as presented in the previous sections) constitute the search domain of our algorithm and acts as meta data. The proposed search procedure is a fast dynamic programming algorithm specifically designed for Audio Oracle structures that is capable of matching the query by concatenating factors or subclips in the target audio. The resulting audio material preserves naturalness and captures temporal morphologies of the input query due to its capability to find longer phrases and flexibility of the matching criteria, as explained hereafter.

Guidage automatically identifies segments in the target sounds that match parts of the query. Segments in the query that do not pass a user-defined matching threshold in the target sound are left silent, resulting in a collection of possible excerpts from the target database, matching to different portions of the query that are obtainable using a simple resynthesis algorithm. In other words, the search algorithm *guides* the resynthesis engine to parts of the target audio pertaining to the query audio and user-defined search parameters, and allows *reassembly* of these factors to replicate the given query. Therefore, on the applicative side *Guidage* can be looked within two perspectives: An algorithm for *guiding* the memory to relevant places in the memory pertained to a query (hence the origin of the name *Guidage*). This aspect of the algorithm is quite important for interactive learning setups such as active learning or reinforcement learning algorithms and will be revisited in part III of this thesis. The second aspect is its capability to *re-assemble* the target audio based on guided paths to replicate all or parts of the original query. This aspect of the proposed method can find immediate applications in any system where *unit selection* is of great importance: concatenative synthesis, search engines and more.

The work presented here touches upon several existing techniques for sound manipulations: sound texture synthesis, and audio mosaicing on the synthesis side; and audio matching, query by audio and audio structure discovery on the analysis side. We therefore begin in section 5.2.2 by reviewing the most relevant works in the literature in these fields as well as their comparison to the proposed methods. Section 5.2.3 provides

the general framework of the proposed algorithm, followed by a discussion and definition of the meta-data used by the algorithm in section 5.2.4 as today's large-scale audio data necessitates. In section 5.2.5 we define the proposed search algorithm *Guidage* that uses Audio Oracle structures to find occurences or partial occurences of a given audio query within search target(s). The resynthesis engine is briefly described in section 5.2.6. We then examine the results of the system within two application frameworks in section 5.2.7: Primarily as information retrieval on long music data using *Model-AO* as meta-data and second for fast retrieval of sound textures on large databases of audio.

5.2.2 Related Works

Concatenative Approaches to Sound Synthesis

Concatenative approaches to sound synthesis undergo various definitions and approaches for treatment of sound recording using re-assemblage of an audio recording or an external audio corpus. Sound texture synthesis systems (Lu et al., 2004; Hoskinson and Pai, 2001) try to create new versions of sounds, mostly from natural sources, that would be similar to the original source without repeating or looping it. The synthesis engine within this group usually uses short sound *grains*, or time-frequency representations, including wavelet multi-scale analysis, to create new sounds that have similar statistical properties as the original source. *Mosaicing* usually refers to combination of larger chunks of sound in a "creative" manner, drawing upon re-mix cultures and other ways of composing by assembling samples (Zils and Pachet, 2001; Lazier and Cook, 2003). Both of the above might have different levels of control over the generative process, from uncontrolled manner of creating texture variants to constrained selection designed to match a compositional design. Concatenative synthesis usually refers to the use of recordings for audio synthesis driven by non-audio data, such as note sequences (MIDI) for music or text in the case of speech (Schwarz, 2007). This method mostly deals with finding the right recorded sounds in order to create a required pitch or phonetic sequence, trying to obtain an optimal tradeoff between the length of recopied clips

and the amount of editing operations (pitch shifting, time stretching and etc.). Creative applications of concatenative synthesis usually undergo feature matching on local or short-term time scales (Sturm, 2004). Other creative applications include granular synthesis, and iterated non-linear functions, to mention a few.

One of the main drawbacks of most of the creative applications proposed using these methods is the lack of intuitive control over the produced contents. Some of the existing systems (Schwarz, 2007; Lazier and Cook, 2003) provide parametric control interfaces and in the case of the second, an audio content retrieval scheme for accessing target specific contents. In this work we present a new and fast way of controlling audio assemblage by explicitly taking the audio structure and its temporal morphologies into account by considering *Audio Oracle* structures as meta-data for the system. The search algorithm then takes into account this structure and guides the synthesis engine towards sequences within the search target that can replicate the given structure as query. In *Guidage*, the level of match or precision versus tolerance in similiarity between the query source and the candidate targets is controlled by a threshold parameter corresponding to an *information tolerance* as a result of applying the algorithm on the basis of music information geometry and Audio Oracle structures presented in chapter 4 and section 5.1.

Audio Matching

Within the audio matching and query-by-example literature, perhaps the most relevant system to our work is the *Freesound project*[1] with major focus on short samples (with mean duration of 3.25 seconds). Within the *Freesound project* there is a content-based search capability that describes the microsound structure using audio feature vectors using a Nearest Neighbor search. On top of the audio structure match is an ontology based on an English lexical model that accesses labels associated with each sample and collected using collaborative filtering that enhances semantical capabilities

[1] http://www.freesound.org

of the results (Cano, 2007).

A more relevant framework to our work is the system described by Casey (2005). In this approach, spectral dynamics are modeled using a 40-state hidden Markov model with parameters inferred by machine learning over a large corpus of training audio data. Time dependence between audio features is modeled by a first-order Markov chain process where each state is considered as a generator for feature vectors through a multi-dimensional Gaussian kernel. In Casey's terminology, these states are called *acoustic lexemes*. Matching can be achieved using the learned HMM and n-gram models to achieve longer sequences where the author considers n-grams of orders from 1 to 8 for the specific applications described in (Casey, 2005).

Due to representational advantages of Audio Oracle structures, *Guidage* does not undergo any time-dependency limitation as is often the case with n-gram approaches and there is no limit to the time extent of the samples used during applications. Moreover, AO structures enable fast access to content-base tree structures that are more efficient than often exhaustive Nearest Neighbor search algorithms (see section 5.1.1 for a discussion on this issue).

5.2.3 General Framework

The work presented here is a rather simple dynamic programming search algorithm. But the premise of its success lies within two important assumption: First, that the musical information is represented and lies within a Music Information Geometric framework as proposed in chapter 4. And second, that its search domain (audio information) is represented not as the audio information itself but its structural information. Therefore, *Audio Oracle* structures are used as meta-data of audio information during search applications as will be discussed in detail in section 5.2.4.

The *Guidage* search algorithm detailed in section 5.2.5 takes an audio input as query. It is a fast dynamic programming algorithm that browses all the audio files in a given database (or their associated metadata) to achieve highest reconstruction of

the query audio by concatenating factors or subclips within each audio in the search database. More precisely, it provides paths over each *Audio Oracle* structure as a combination of its inherent *suffix links* and *factor links* to achieve the longest possible and coherent (in terms of temporality) similar structural sequence to the query audio. With this respect, the algorithm can work in the two domains of *Audio Oracle* as discussed in section 5.1.3: *Model-AO* for large audio files assuring scalability of search results, and *Data-AO* assuring access to microscopic structural variations of audio information. We present both aspects within applicative frameworks in section 5.2.7.

The general schema of the *Guidage* algorithm is modular in the sense that the user can specify the search criteria that is used in both meta data construction and search procedure, and moreover a threshold parameter can control the degree of structure similarity that the user desires to assess during resynthesis. Since Audio Oracle is independent of the input vector presented to the algorithm, the modularity amounts to the choice of the user in terms of what representation she desires to use as input to the algorithm as far as correct conversion exist to assure a music information geometry. Using Audio Oracle structures as meta data would allow us to learn and store Audio Oracles for different sets of audio features that can be easily recalled and changed in the search user interface and utility of the system. To demonstrate results, we have chosen to experiment with a limited set of audio features and combinations thereof. In practice, the choice of signal processing front-end is left to users and researchers interested in expanding the applicative framework of *Guidage*.

5.2.4 Search Domain and Meta Data

When dealing with large data bases of audio, and in comparison to retrieval from small sound sources, two problematic issues arise: *Scalability* of data and *access* to a wide and disparate amount of data. The issue of scalability is of high importance at our time due to the exponential trend of media data size and specially music data. This exponential growth in turn affects maintenance and technological life of any system that

deals with meta-data containing such information. Within this context, de Cheveigné defines scalability as the conjunction of two properties: arbitrary resolution, and convertibility between resolutions (de Cheveigné, 2002). Scalability of data becomes even more important in the context of search algorithms such as ours and specific to music information because of high degree of variability of musical entities of the same piece of music represented within different contexts (different performances, styles, instrumentation etc.). The issue of scalable data is not independent from the issue of data *access*. Even for an algorithmically fast search methods (that is close to linear time) in the context of a huge audio file, accessing furthest points in the data sequence arbitrarily might become an important burden. The ideal solution for such situations is to gain access to extreme data points using some inherent structural and contextual information.

In our proposal, we address both issues discussed above by replacing the audio information by their *Audio Oracle* representations as the search domain of the algorithm. Each *model* or state in a *Model-AO* structure represents a stable audio chunk where its temporal resolution can be variable during retrieval applications and thus scalable. With this respect, we separate two issues related to scalability of data: *temporal scalability* that refers to degrees of difference on time-span of information content, and *representational scalability* pertaining to variability of representations of the same audio phenomena. *Audio Oracles* address the first but neglige the second. This is rather a property of *Audio Oracle* than a shortcoming. *Audio Oracles* do not *represent* the content but the *influential aspect* of the data as it unfolds in time and hence, the structural content of a specific representational scheme would not naturally be the same on another scheme. The important factor when using *Audio Oracle* instead of audio itself with this respect is its sparse and compact representation of flow of information content. If the temporal resolution of such content is modified in an alternative representation of the same audio flow, its *information content* should ideally stay the same to be recognizable by both humans and machines as the same phenomena. Also the structural representation of audio in *Audio Oracle* provides fast access to extreme sequential placements based on variable-length content similarities within the data. This way any search algorithm

can instantaneously access two extremes in the same flow if they undergo structural similarities as represented by arrows in an *Audio Oracle* representation.

In our experiment, we store learned Audio Oracle structures, along with their corresponding audio features and analysis information in meta data information using the Sound Description Interchangeable Format (SDIF) (Schwarz and Wright, 2000). Besides its universal accessibility and interchangeable format, SDIF allows fast and efficient access to any desired set of analysis frames during execution. Moreover, SDIF allows personalized type-definitions which are used here to describe Audio Oracle structures, along parameters that are used during audio re-assemblage once the search algorithm has finished its task.

5.2.5 *Guidage* Algorithm

We define the problem context for our proposed search algorithm as follows: Given an audio query and a search target audio file, find an assemblage of sub-clips within the target audio file that can replicate the audio query. In other words, we are aiming at reconstructing a new audio similar to the query by concatenating sub-clips of the target audio file. The search criteria is defined by the user and corresponds to the type of audio feature set (and its corresponding *Audio Oracle* representation of targets) used for analysis and similarity comparison.

The search algorithm proposed here is based on Dynamic Programming, an algorithm paradigm in which a problem is solved by identifying a collection of subproblems and tackling them one by one, smallest first. Dynamic Programming uses the "answers" to small problems to help figure out larger ones, until the whole of them is solved. In the context of our problem, the "small" problem set amounts to finding audio chunks (or audio analysis frames in this case) in the search target audio file, that are *similar* to a corresponding chunk in the query *and* can be considered the *longest possible path* and a *continuation* of the previously obtained chained based on the Audio Oracle structure of the target audio. We call this step of the procedure the *forward pass*. The "larger"

problem, then, becomes finding the *best path* among all recognized that best meets the search criteria when all the small-set problems are solved. This step is referred to as *backward* procedure.

As the algorithm is based on *Audio Oracle* meta-data, it has two possible usage based on the type of *Model* or *Data* Audio Oracle being used for meta-data. In both usages, the algorithm rests the same but the definition of sound entities entering the system differ as described shortly. We refer to these two practices of the algorithm as *Model-Guidage* and *Data Guidage*.

To describe the algorithm in a formal manner, we use the following notations: Query Audio is represented as $\mathcal{Q} = \{\boldsymbol{Q}_1, \boldsymbol{Q}_2, \ldots, \boldsymbol{Q}_N\}$ where each $\boldsymbol{Q}_i$ is either the (user-specified) feature description of the i^{th} time-domain analysis window converted to a Multinomial distribution to represent points on a statistical manifold for *Data Guidage* usage, or *models* as a result of *Model-AO* representation of the query audio. Similarly, the search target audio is represented as either its data or model AO vectors in $\mathcal{Y} = \{\boldsymbol{Y}_1, \boldsymbol{Y}_2, \ldots, \boldsymbol{Y}_M\}$ and also by its corresponding Audio Oracle structure data, $\delta(i)$ for *factor links* and $S_P(i)$ for *suffix links* as before where $i \leq M$. The algorithm assigns similarity using a user-defined threshold ϵ used on the symmetrized Bregman divergence (defined previously in chapter 4) associated to the representation and within the previously discussed approximate similarity metric space.

The basic functionality of the *forward pass* of *Guidage* is as follows: At the very onset, the algorithm identifies similar states in $\mathcal{Y}$ as initial candidates for $\boldsymbol{Q}_1$ using Bregman divergence and the given threshold ϵ. This is the only place throughout the whole algorithm where a single state is tested against the whole target domain. This procedure results into a set of state-index candidates in $\mathcal{Y}$ denoted as $\mathcal{I}_1^1$. From this point onwards, the goal of the algorithm is to find the *longest* and *best* possible *continuation* by branching within the Audio Oracle structure of the target. Continuity is assured by following the structural similarities and spectral morphologies of the target Audio Oracle. To this end, every candidate state in $\mathcal{I}_1^1$ is considered as a *parent* and the similarity of its *children* are tested against the next pass in the query or $\boldsymbol{Q}_2$. The *children* of a

state in Audio Oracle are states that either immediately follow the parent through factor links or follow a *neighbor* of the candidate on the *Suffix Link Tree* where the candidate lives. This relationship is represented schematically in figure 5.6. Testing $\boldsymbol{Q}_2$ against the children of $\mathcal{I}_1^1$ for similarity would lead to $\mathcal{I}_2^1$. This process will continue for each $\boldsymbol{Q}_i$ until there is no possible continuation after $\mathcal{I}_k^1$. At this point, either k is equal to N (or size of the query) meaning that we have browsed the whole query space, or $k < N$ meaning that the ensembles $\mathcal{I}^1 = \{\mathcal{I}_1^1, \ldots, \mathcal{I}_k^1\}$ provides various paths to reconstruct the audio chunk corresponding to the sequence $\boldsymbol{Q}_1 \ldots \boldsymbol{Q}_k$. In the case of the second ($k < N$), the algorithm then restarts but this time at $\boldsymbol{Q}_{k+1}$ (instead of $\boldsymbol{Q}_1$) until a second set $\mathcal{I}^{k+1}$ is obtained and continue this schema until the whole query space is browsed. Algorithm 5.4 shows the steps taken during the forward pass of *Guidage*.

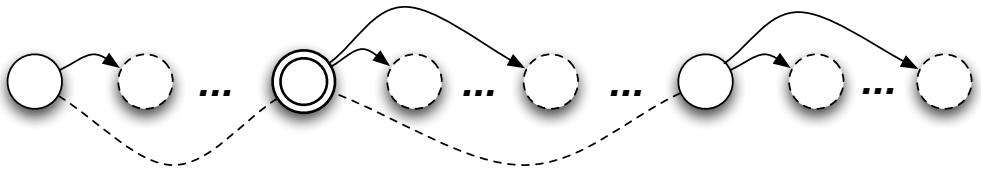


Figure 5.6: *Audio Oracle* Parent/Children structure diagram. The parent is represented as a double-lined state and adopted children with dashed-lines. Directed edges represent *Factor Links* and dashed lines the *Suffix Links* as undirected edges.

Algorithm 5.4 returns the structure $\mathcal{I}$, with N member sets equal to the number of entities in the query. Within $\mathcal{I}$ each group of subsets $\mathcal{I}_1^k \ldots \mathcal{I}_\ell^k$ correspond to a *tree* structure determining a path on the Audio Oracle structure of the target ($\mathcal{Y}$) that can re constitute the audio chunk $\boldsymbol{Q}_k \ldots \boldsymbol{Q}_{k+\ell}$. This tree structure is the result of combining factor link and suffix link continuations on the target oracle. In the case where the whole query can not be constructed in one pass, $\mathcal{I}$ would contain several tree structures whose concatenation can reconstruct relative parts of the query where parts can also be empty (or silent). Therefore, each tree structure correspond to several and many complete or partial paths that can reconstruct parts or the whole query.

Having $\mathcal{I}$ from algorithm 5.4, it suffices to find the *best paths* among each tree

Algorithm 5.4 Forward pass of `Guidage`

Require: Query Audio Oracle centroids in $\mathcal{Q} = \{\boldsymbol{Q}_1, \boldsymbol{Q}_2, \ldots, \boldsymbol{Q}_N\}$, Search Target Audio Oracle P given by centroids $Y = \{Y_1, Y_2, \ldots, Y_M\}$, factor links $\delta(i)$ and suffix links $S_P(i)$.

Initialization: $k \leftarrow 1$
while $k < N$ **do**
  Initialize the search space as $\mathcal{I}_1^k = \left\{ j | D_F(\boldsymbol{Y}_j, \boldsymbol{Q}_k) \leq \epsilon \right\}$
  $i \leftarrow k + 1$
  while $\mathcal{I}_{i-1}^k \neq \emptyset$ **do**
    Set $\mathcal{I}_i^k = \emptyset$
    for Each state index j in $\mathcal{I}_{i-1}^k$ **do**
      Set $\mathcal{C}_j$ to the index of *children* of state j
      Obtain $\mathcal{I}_i^k$ where

$$\mathcal{I}_i^k = \mathcal{I}_i^k \cup \left\{ h | \, h \in \mathcal{C}_j, \, D_F(\boldsymbol{Y}_h, \boldsymbol{Q}_i) \leq \epsilon \right\}$$

    end for
    $i \leftarrow i + 1$
  end while
  if $\mathcal{I}_i^k = \emptyset$ **then**
    $k \leftarrow i$
  end if
end while
return $\mathcal{I}$ structures

structure to reconstruct the query using the target. This amounts to a simple backtracking and branching procedure using indexes in sets $\mathcal{I}_i^k$ following the Audio Oracle structure of the target and the scores of each transition which is equal to the similarity obtained during the forward pass using the Bregman divergence. Once the best path is obtained, we can easily reconstruct the audio using a simple concatenative resynthesis as described below.

5.2.6 Resynthesis

Once a reconstruction path is obtained from *Guidage*, resynthesis of the search results amounts to a simple concatenation of the corresponding audio to each state designated by the path. If the result corresponds to the *Data Guidage* framework (i.e. using *Guidage* on *Data-AO*s), then the resynthesis is a simple windowed overlap-add algorithm that assures phase continuity of the reconstruction and in accordance with the signal processing front-end used for the music information geometry framework (see section 4.3.1). In the case of *Model-Guidage*, each state in the path corresponds to a *model* $\boldsymbol{Y}_j$ on the target oracle related to another model $\boldsymbol{Q}_h$ through similarity and continuity. Note that each *model* in a *model-AO* by itself corresponds to a set of continuous *points* with time indexes indicating analysis time-windows in the original audio. The number of points within each model is usually different. In a case where model $\boldsymbol{Y}_j$ corresponds to k_y points, $\boldsymbol{Q}_h$ to k_q points, and $k_q \leq k_y$, the described concatenative procedure can reconstruct $\boldsymbol{Q}_h$ by using the first k_q original points from the target audio. Note that since each *model* in a *model-AO* represents quasi-stationary chunks of audio in terms of information content, textual continuity is more or less assured and controlled by the threshold parameter used during AO construction. In the case where $k_q > k_y$, the number of points needed to construct the model in the query exceed the ones given by the model in the target. Here, the quasi-stationarity of models in AO comes to help by simply recycling the points in the target model to reconstruct the associated query chunk and assuring phase continuity.

5.2.7 Sample Applications and Results

Results of *Guidage* can be best studied in application frameworks. In this section we demonstrate these results within several immediate retrieval application frameworks: One on *Model Guidage* and two on *Data Guidage*. The *Model Guidage* framework allows fast and high-level retrieval over large audio files and using large musical queries whereas *Data Guidage* can be used for texture retrieval and access to micro-structures of sound and audio. The *Model Guidage* experiment is presented here to provide visualization of results of *Guidage* whereas the other experiments study the applicability of the algorithm within two sample applications. In one application, an audio query is given by the user as well as a pointer to a search target folder, and some search parameters. The application then runs *Guidage* over the whole database and a Graphical User Interface (GUI) demonstrates ranked search results and visualizes different parameters. In the second application, an audio query is searched within *one* given target audio, where the aim is to reassemble various occurrences of the query in the target file and access the micro-structure level re-assemblage.

Model-Guidage Sample Results

Here, we examine the result of *Guidage* in a *Model-AO* set up where *Model-AO* is used as meta-data for both query and target audio. To make our observations straightforward, we demonstrate a rather simple problem and study the result structure of *Guidage*. We run *Guidage* on Beethoven's 1^{st} Piano Sonata (first movement) with the audio query as the first theme of the same piece. We previously demonstrated the *Model-AO* of both: The query *models* were presented in figure 4.3 on page 86, and the *Model-AO* of the whole piece was shown in figure 5.3 of page 100. The incremental model formation algorithm of section 4.5 leads to 37 models ($\boldsymbol{Q}_i$s) for the query constituting around 10 seconds of audio, and the *Model-AO* of the whole piece has around 650 states or models ($\boldsymbol{Y}_i$s) on over 3.5 minutes of audio. This setup allows us to interpret the tree structure as the intermediate result of *Guidage* and easily interprete their musical

relevance since the *best path* in this case is really a *self-similarity* test. The algorithm returns the results within less than 2 seconds when run in the MATLAB programming environment and on a $2.33GHz$ Intel Core laptop.

Figure 5.7 on page 119 shows the result of *Guidage* on the described setup by visualizing the *tree structure*[2] of $\mathcal{I}$ and by showing the reconstructed audio on the top, corresponding to the best path on the tree-graph highlighted by light-gray states. The tree structure should be read from left to right as the arrows indicate. This experiment was done using a similarity threshold of $\epsilon = 0.3$ for *Guidage*. While the *best path* (highlighted as light-gray nodes) correctly identifies the *original* audio chunk (states 1 through 37), there are several alternative paths that can be chosen to this optimal path that are musically important to us. Some of the minor paths have been omitted from this graph to help readability. Parallel to the main path, there is an alternative path consisting of continuous states 169 to 202. These states correspond to the *repetition* of this theme after the exposition of the sonata form as was discussed in section 5.1.4. A third path consists of states 460 to 511 that more or less maintain their continuity throughout the tree. This path corresponds to the third repetition of the theme (with variations) in the beginning of the *reprisal* of the sonata form. Other sub-paths correspond to partial constructions and reappearance of parts of the theme throughout the whole sonata specially during the *development* section. Another interesting observation in this example is the explosion of state choices at particular time-levels of the tree. At time-level 11 (that here corresponds to light-gray state 11), there are many state candidates as possible continuations representing that particular model. A close look at the corresponding *model* shows that it belongs to a particular *cadential chord* in the theme that re-appears in various places throughout the piece as an important stylistic element, and hence explains the explosion of state choices. The same phenomena can be observed for the very last state that represents another *cadential chord*.

Every application of *Guidage*, whether in *Model* or *Data* mode, would result into a tree structure similar to figure 5.7. Besides the importance of the depth of the tree

[2]Using *GraphViz* (Ellson et al., 2003).

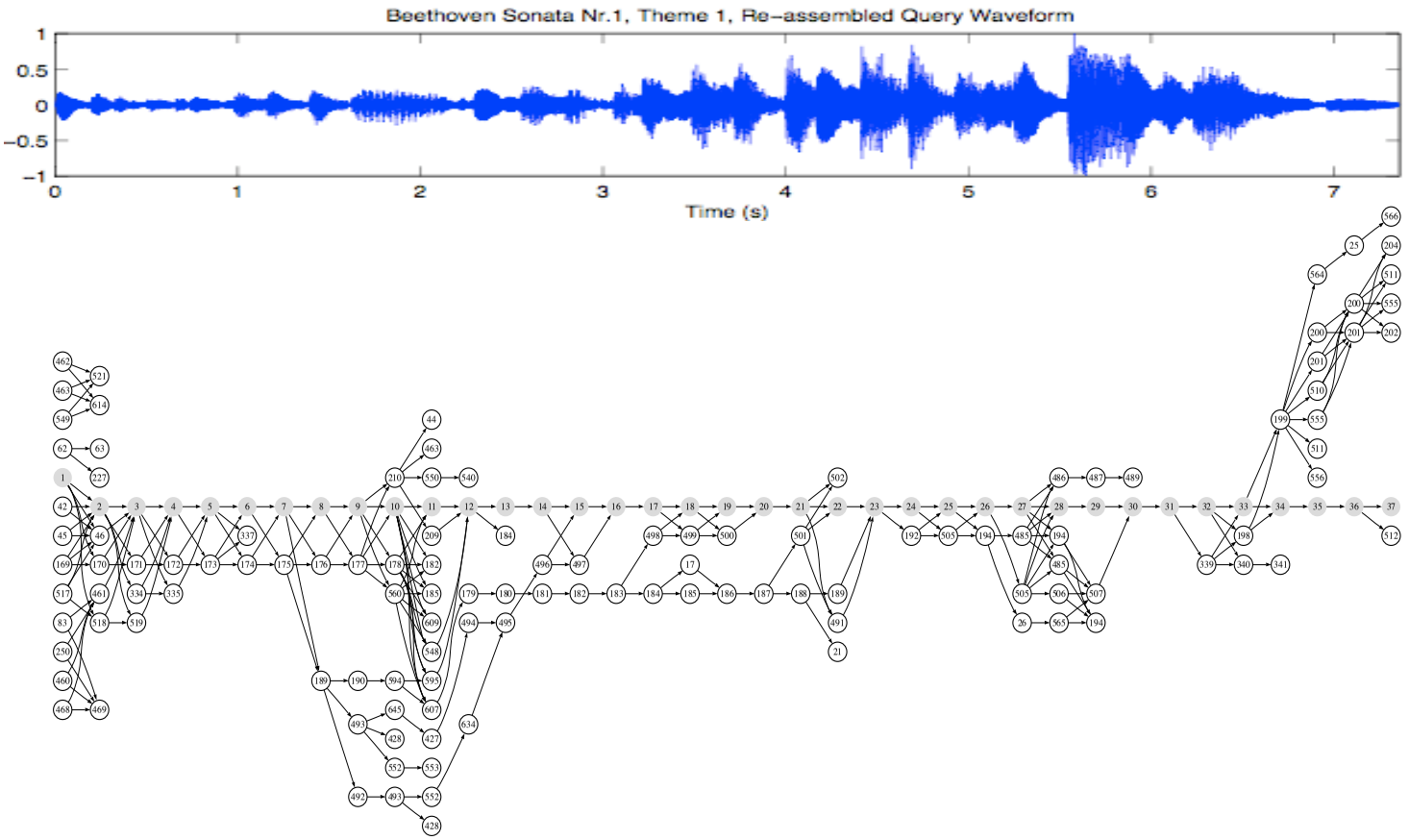


Figure 5.7: *Model-Guidage* sample results on Beethoven's Piano Sonata Nr.1-Mvt.1, using the first theme as audio query – Showing the reconstructed audio and the tree structure of the *forward pass* with *best path* highlighted in light-gray.

(indicating the length of the longest continuation) and the best construction path, all the information contained in the tree structures reveal structural similarities throughout the whole search space of the target. Therefore, *Guidage* provides access to macro-level as well as micro-level structures of data. This property of *Guidage* makes it a strong candidate for many application settings. Below we will look at some sample retrieval applications.

Data Guidage: Query over an Audio Database

In this section, we demonstrate *Data Guidage* for query retrieval over databases of audio. Within this problem, we are aiming at macro-structure and surface results of *Guidage* (i.e. the *best reconstruction* and search depth) with less emphasis on micro-structures and details of the tree structures. To this aim, we represent results within an application framework implemented on a Graphical User Interface (GUI) to visualize results, and control input query, database folder and the threshold ϵ parameter. The data (query and database) presented to the system are in *Data-AO* format, corresponding to audio analysis frames converted to a Multinomial manifold. The database is presented to the application through *Data AO* metadata in SDIF format. The front interface of the GUI is shown in figure 5.8.

This GUI is designed as follow: At the right of figure 5.8, there are three panels that separate different functionalities of the application. The first and most important is the *search panel* where the user can freely choose the audio query file and a folder as the search target. In the next step, the user can choose the search criteria that she intends to perform the algorithm on from a pop-up menu. The list of search criteria is either (1) loaded automatically from the metadata in case the chosen query is in SDIF format, or (2) is the list of available features to the system (MFCC, ΔMFCC, pitch), also open to further expansions. Another parameter that can be controlled by the user is the similarity threshold or ϵ used in algorithm 5.4. Pressing the *Search* button then performs the algorithm over the whole database assigned by the user. Once the results

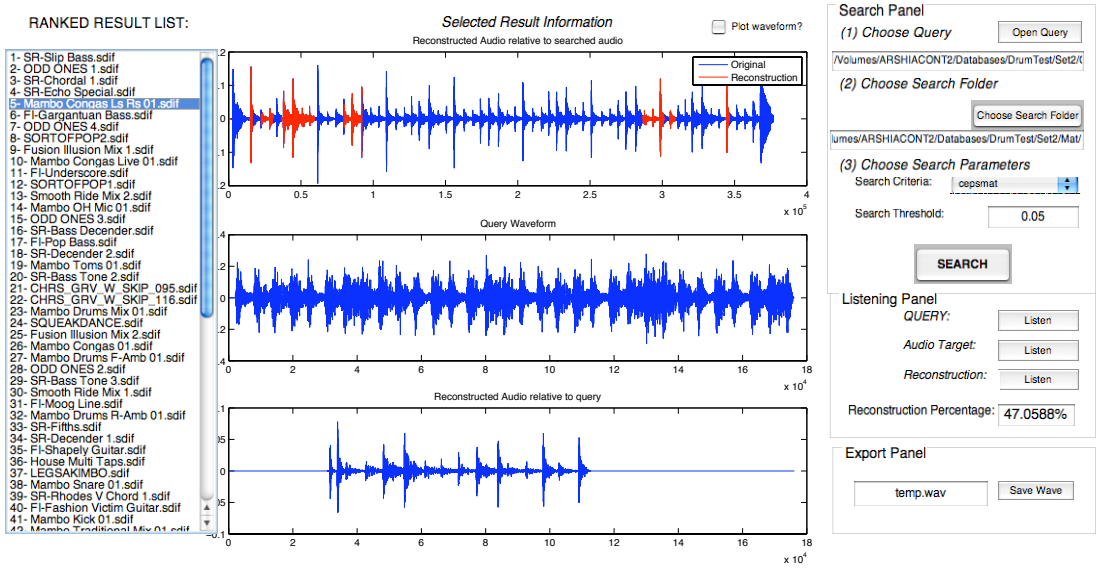


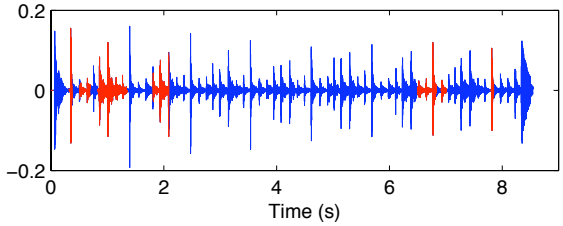
Figure 5.8: GUI for Audio Query over an Audio Database

are obtained, the result-box at the left of the GUI demonstrates a ranked list of sound files in the database according to the *reconstruction percentage* with regards to the audio query. The best reconstruction for each file is thus the *longest* sequence that can be reassembled through resynthesis to achieve similar factors to the audio query. Choosing each of the ranked results in the result-box reproduces three figures in the middle of the GUI for visualization and better browsing of the semantical contents that has been found during the procedure. The top figure shows the chosen target audio waveform where the found factors within the audio are highlighted in red. The middle figure shows the query audio waveform and the bottom one shows the concatenative synthesis of the highlighted factors in the first figure relative to the audio query. A *Listening Panel* respectively allows listening to the three described waveforms and an additional *Export Panel* allows exporting the resynthesis, if desired by the user, to an external audio file.

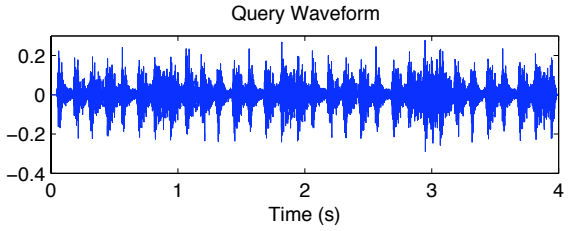
To demonstrate the performance of this simple application, we conduct two experiments on two sets of sounds and different queries. All the sounds reproduced here and more sample results on other data sets are available for audition on the internet[3]. The

[3]http://cosmal.ucsd.edu/arshia/index.php?n=Main.Guidage

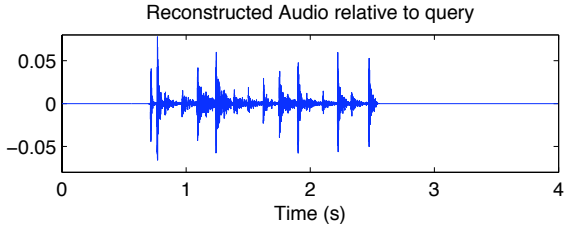
first set corresponds to a database of musical loops and rhythms out of realistic recording sessions taken from the *Kontakt Sound Library*[4]. The collection of loops chosen for the search session amounts to approximately $200Mb$ of disk space, 140 audio files with mean duration of 7 seconds. The query used for this demonstration is an *African drum sequence* that lasts approximately 4 seconds and non-existant in the search database. Figure 5.9 shows the 3^{rd} ranked results, corresponding to 47.06% reconstruction and the produced waveforms out of the GUI described earlier. The search criteria used for this query session are the MFCC feature vectors with an similarity threshold of 0.3.



(a) Target Audio Waveform and Found Factors (highlighted in red/gray)



(b) Query Audio Waveform



(c) Synthesized Best result relative to query

Figure 5.9: *Data Guidage* query retrieval sample result on a drum-loop database

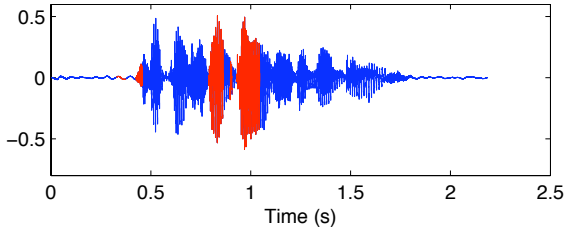
Figure 5.9c shows the resynthesized result audio as the best reassembly of the

[4]http://www.native-instruments.com/

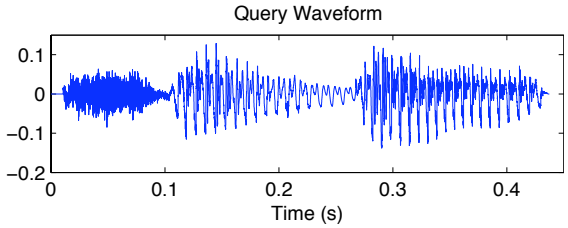
factors in the target audio (figure 5.9a) to achieve a replication of the query waveform (figure 5.9b). As mentioned earlier, the silences in figure 5.9c correspond to factors where no match has been found. Highlighted (red/gray) waveforms in figure 5.9a correspond to factors of the target which are used for concatenative synthesis and indexed results of *Guidage* to achieve the waveform in figure 5.9c. Comparing figure 5.9c and 5.9b, we can see that the algorithm has used factors corresponding to sound objects in figure 5.9a to achieve a similar rhythmic pattern as in figure 5.9b. Listening to the synthesized audio (on the provided URL) also reveals the timbral similarity between the query and result - another reason why this sample has appeared among the top 10 results in a search session over 140 sound files.

The second experiment-set corresponds to speech audio files. This experimental database corresponds to 10 recordings of theatre actors saying the same phrase in French with different durations, intonations and emotional intentions taken from *Ircam Expressivity Corpus* database (Beller et al., 2008). The repeated phrase in French is: `C'est un soldat à cheveux gris.` The query is taken out of one of the sound files and corresponds to the pronunciation of the word `Soldat` in French. The aim of the search algorithm here is therefore to find the occurrences of the word `Soldat` in different phrases despite their variances *or* to reconstruct the same word by concatenating different phonemes where linear reconstruction is not possible. The search criteria used for this query session is a mixture of MFCC feature vectors and their first derivative (ΔMFCC). This choice of audio feature is common among speech recognition systems. For this sample result, we demonstrate the 9^{th} ranked result among 10, thus towards the worst, in figure 5.10. The interest in showing "worse" results is to demonstrate the behavior of the re-assemblage.

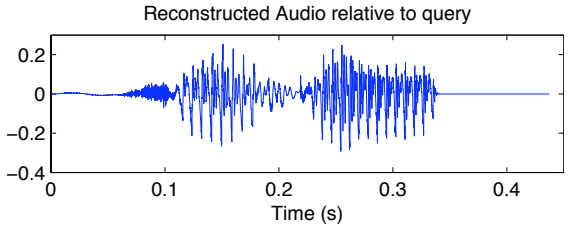
Similar to figure 5.9, the three figures within correspond to the query waveform of the word utterance `Soldat` (figure 5.10b), the target phrase recording and the used factors during resynthesis (figure 5.10a) and the resynthesized result in figure 5.10c. The remarkable result here is that since the intonation and duration of the target audio is quite different from the original source, the algorithm has reinforced the reconstruction of the



(a) Target Audio Waveform and Found Factors



(b) Query Audio Waveform



(c) Synthesized Best result relative to query

Figure 5.10: *Data Guidage* query retrieval sample result on a speech database

query using different phoneme sequences than the original. More precisely, factors used during reconstruction (highlighted waveforms in figure 5.10a) correspond to the /s/ sound in the pronunciation of the word C'est (in French) in the beginning of the phrase (and not the /s/ sound in Soldat) and partial factors of the spoken word Soldat as seen in the sub-figure to achieve reconstruction. This observation simply reveals that among all the tree structures and paths within them that can reconstruct the first occurrence of the sound /s/, the longest one is the one belonging to that of the pronunciation of the word C'est (in French).

Data Guidage: Query within Micro Audio Structure

The sample application described in the previous section uses only one result path among all that is found by the algorithm described in section 5.2.5. Recall that the tree structure found by *Guidage* provides many possible paths and sub-paths that might lead to other concatenations of the target audio entities. In some applications, users might be interested to focus on the microscopic search results rather than the longest path result that is represented above. Here, we demonstrate this aspect of the *Guidage* by focusing on one audio query and *one* search target, and visualize *all* the possible results obtained by *Guidage*. Once again, the user has the ability to control search and audio analysis parameters to a fine-scale control, allowing access to query guided grains in the target audio.

Figure 5.11 and 5.12 show two snapshots of the application run on the same query/target pair, but showing the first and second results respectively. Here again, as before, a *Search panel* allows the user to choose the audio query and target files as well as search parameters and analysis parameters (in case an audio file is chosen instead of pre-formated SDIF meta-data). Clicking on the *Search* button lists all the results in the *result-box* on the left and choosing each result (represented by the reconstruction percentage), visualizes the results in two corresponding figures in the GUI. Here, the top figure corresponds to the target audio waveform, again, where highlighted waveforms

correspond to the found factors used during concatenation. The lower figure visualizes the resynthesized waveform relative to the query waveform (not shown here). The query used for this demonstration is the same *African drum* sequence used in our previous experience and the search target audio is a *Live Mambo Congas* audio sequence. Audition of results is possible over the internet[5].

Figure 5.11 shows the *first* result that corresponds to the highest reconstruction percentage (24.7% here). This is the typical "best" result used in the audio query over database application. As before, the resynthesized audio has the same timeline as the query; showing that in figure 5.11, the reassembled sequence corresponds to the original query audio that appears approximately between samples $40K - 60K$ or time window $0.9s$ through $1.4s$. A close look at the resynthesized audio and audition of it suggests that the reconstructed portion imitates the rhythmic structure of the query.

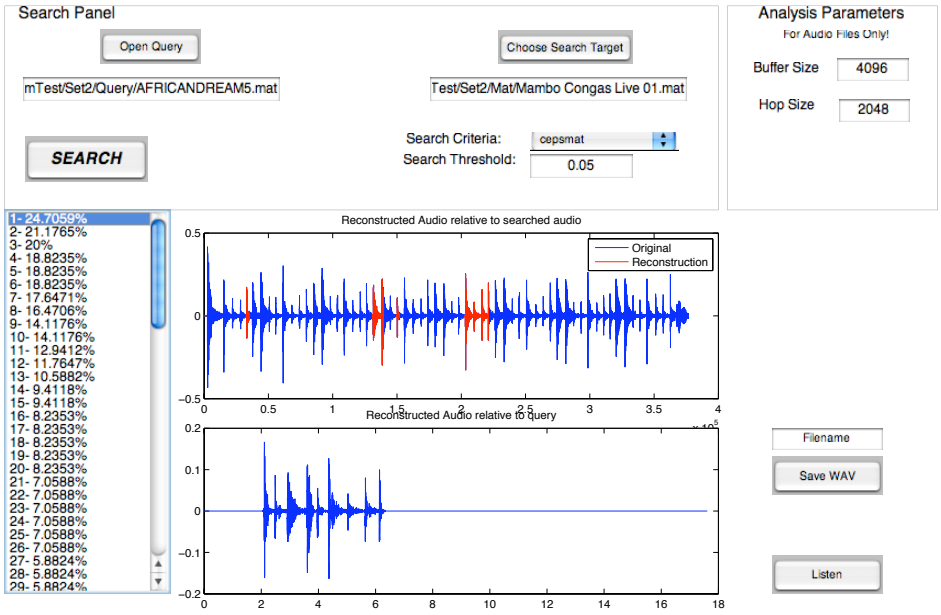


Figure 5.11: *Data Guidage* GUI for Micro Audio Query – First result.

Figure 5.12 shows the 2nd result in the list chosen by the user. This result corresponds to 21.1% reconstruction. What is remarkable here is that comparing both vi-

[5]http://cosmal.ucsd.edu/arshia/index.php?n=Main.Guidage

sualized subfigures in figures 5.11 and 5.12, it is clear that (1) the factors used during re-assemblage and results of *Guidage* are different. And (2) the resynthesized sequences correspond to two different timelines of the original audio query (in figure 5.12 between $3 - 4$ seconds).

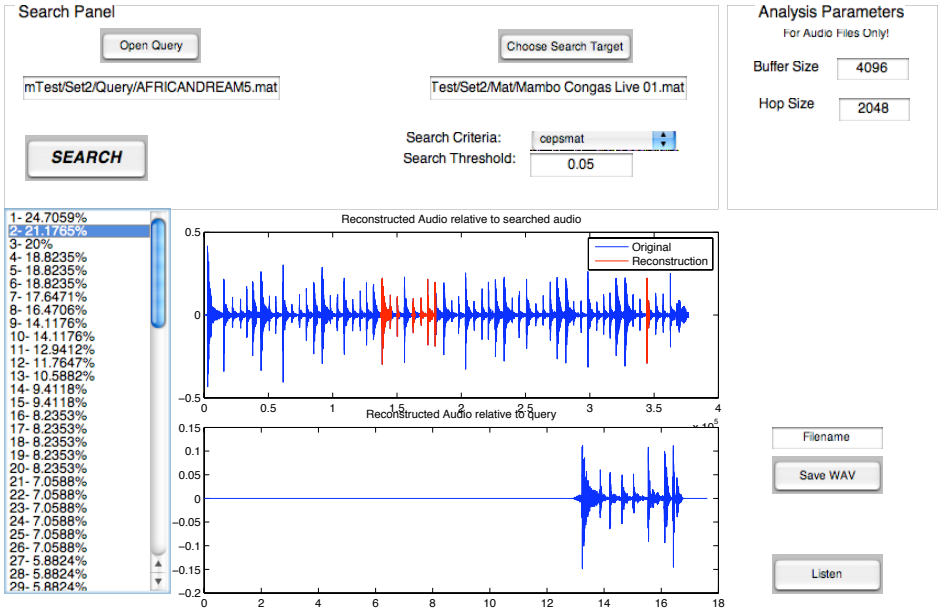


Figure 5.12: *Data Guidage* GUI for Micro Audio Query – Second result.

These samples along with previous shown experiments in this section should reveal the important advantages of using *Guidage* as well as its aims in retrieval: Fast access to information structures using external queries both within microscopic and macroscopic levels of information.

5.2.8 Discussions

In this section, we addressed the question of accessing external data structures and revealing information relevancy using an audio query. We introduced *Guidage* as query guided audio assemblage and provided an algorithm for fast retrieval of audio-content and resynthesis of retrieved factors as a re-assembly of the target audio. We presented the algorithm within two frameworks pertained to previously discussed *Audio*

Oracle structures and discussed its aims for *temporal scalability* and *access* to huge collections of data structures. The results of the algorithm were demonstrated within three experiments. We showed how *Guidage* finds all possible recurrences of sub-factors of the query by recollection and recombinations of structural chunks of a target audio structure. We then demonstrated its performance within two sample retrieval applications emphasizing on the fact that the algorithm accesses both microscopic and macroscopic levels of structured information. Note that in all shown experiments, and as before, the algorithm does not have any a priori knowledge of musical structures, and everything is done through unsupervised and fast discovery of sub-factors which would maintain a perceptual continuity in the final result. This perceptual continuity is a consequence of choosing to navigate on the audio structure rather than the audio itself for search purposes. With this choice the temporal morphology of sounds is explicitly put into account in our retrieval framework. This was possible thanks to the music information geometry framework that favors the influential aspect of information.

We mostly presented *Guidage* in an information retrieval framework. But let us emphasize at this point that the most important factor of *Guidage* is in enabling fast *access* to (disjoint) audio-content and structural information. This factor is essential for any ancitipatory system that is in constant interaction with its environment to update mental representations or create new ones.

Speed performance of *Guidage* depends on the degree of semantic similarity between the query and search targets. For the experiment described in section 5.2.7 on a corpus of 140 audio files with total size of $200Mb$, *Guidage* performs in approximately 20 seconds using `MATLAB` and a $2.3Ghz$ Mac-Intel machine. In another audio query over database experiment on natural bird sounds with 244 audio files and over $600Mb$ of data, *Guidage* performs in less than 25 seconds. This suggests some degree of *computational scalability* of the proposed algorithm when dealing with large corpuses of audio.

Part III

How to Expect

Chapter 6

Adaptive and Interactive Learning in an Environment

In the two previous chapters we proposed methods to represent some aspects of music information and showed how they can form mental representation about the underlying structures of information flow within an environment. We also showed various forms of access to this information that demonstrate regularities of information content and proposed two algorithms. In this chapter we take one step further and show *how* this information can affect the behavior of living agents in a constantly changing environment or *how* these representations could lead to actions based on some expectation about the environment. Therefore, from this chapter on we are formally in the process of *Anticipatory Design* of musical systems.

In chapters 2 and 3 we emphasized the importance of *learning* both in cognitive aspects of auditory learning pertaining to expectations and computational design. Leaning in this context is *adaptive* and *online* and in constant interaction with a constantly changing environment. The goal of learning is thus to grasp adaptive *anticipatory behavior* (as defined in chapter 3) and use them latently in decision making situations. These considerations led us to the definition of *Anticipatory Systems* in section 3.1 whereafter we studied adaptive learning frameworks, different modeling approaches to

anticipatory design and learning, and drew the premises of anticipatory design in section 3.5. The first premise, *Information Availability*, was the topic of part II and the two remainings, *Interactive and on-line learning* and *Multimodal Interaction and Modeling*, are the main concerns of this and the following parts of the thesis.

In this chapter we attempt to address the following questions: *How* can we formulate an adaptive system with regards to a changing environment? *How* can such a system learn adaptive behaviors from environmental regularities that are useful for future decision making? and *How* can the system update its mental beliefs faced to constant changes in the environment and in its own behavior?

6.1 Introduction

We introduce a first attempt towards modeling interactive intelligent musical systems with regards to the psychology of musical expectations. For modeling these constraints, we use anticipatory systems where several accounts of musical anticipation are explicitly modeled. We formulate our approach as a *long-term planning* problem between multiple-agents representing the system's belief on distinct mental representations of its environment and in constant interaction with the environment. The design and considerations for such approach is constrained and inspired by cognitive aspects of musical expectations presented in chapter 2 and follow anticipatory design concept introduced in chapter 3. We claim that such cognitive modeling of music would constitute complex musical behavior such as long-term planning and generation of learned long-term formal shapes. We will also show that the anticipatory approach greatly reduces the dimensions of learning and allows acceptable performance when little data is available.

Among existing literature, the one that come closest to the topic of this chapter is the literature on *statistical models of music generation* with applications to automatic style imitation and improvisation. We review this literature in section 6.2. Where most existing approaches are based on *prediction* on learned *context models* with no explicit

consideration for *planning strategies* for action decision making, our approach directly addresses the problem of *learning strategies* and separate this important issue from that of context learning and prediction without excluding them from the learning equations. We furthermore expand the idea of predictive models to explicit anticipatory models as discussed in chapter 3.

We formulate the problem of *planning* in an anticipatory and adaptive learning framework and provide solutions pertaining to cognitive foundations of musical expectations. Based on this framework we propose an architecture that features reinforcement and active learning as an interactive module between the system and an outside environment. To this end, we divide the problem of statistical modeling of music into three subproblems: (1) The problem of *memory* or mental representations of music data for a system, capable of forming *contexts* and providing access to structural aspects of music information wherever appropriate. (2) The problem of *interaction* with an environment that *guides* learning, behavior and updates memories or mental representations. And (3) that of *learning* adaptively and interactively in an environment for planning and behavior learning.

After discussing the state-of-the-art in section 6.2, we provide a general overview of our framework's architecture in section 6.3 and proceed by detailing the three main design components of the framework, leading to a detailed description of our architecture in section 6.6. The interactive learning framework that is introduced in section 6.5 features an *active learning* framework for *memory-based* active learning with a *competitive and collaborative* norm of learning between agents. We conclude this chapter by examining the performance of the proposed framework within several experiments and discussing the computational complexity of the proposed framework.

Throughout this chapter, we content ourselves with *symbolic* representations of music signals rather than audio signals to simplify demonstration of concepts and examples in our design. As it will become clear shortly, the representational scheme used for this purpose is simply the symbolic version of the modules presented before in part II of this thesis. Thus, in the proposed framework passing from symbolic to signal represen-

tations in application simply amounts to replacing the former with the music information geometry framework presented earlier. We will evoke this issue later in section 6.7.

6.2 Background on Stochastic Music Modeling

Earlier works on style modeling employed information theoretical methods inspired by universal prediction. In many respects, these works build upon a long musical tradition of statistical modeling that began in the 1950s with Hiller and Isaacson's "Illiac Suite" (Hiller and Isaacson, 1959) and Xenakis using Markov chains and stochastic processes for musical composition (Xenakis, 1971). These early hopes were soon replaced by frustration as many of the models following this literature were shown incapable of producing even simple well-formed melodies and led to the shift of research to hand-crafted algorithms for style imitation such as the ones in (Cope, 2001; Biles, 2003; gabriel Ganascia et al., 1999). While such ad-hoc approaches have proven to be *successful* in terms of generative results, they are heavily biased towards their developers and do not attempt to provide any scientific explanations towards the complexity of musical behavior (Conklin, 2003). However since 1990s and mostly due to success of statistical approaches to speech and language, the empirical and statistical research in music modeling has regained momentum and created an interesting burst of new methods and literatures that we will review in this section.

Despite differences in modeling and approach, all the models presented hereafter have the following premise in common: They are based on *predictive models* out of *learned contexts* from various musical attributes. In this sense, following our terminology of chapter 3, these systems are *Implicit Anticipatory Systems* (see section 3.3.1 on page 46) where the role of the predictive model as a feedback to enhance current decision making is ignored.

Within these three aspects, the *memory modeling* approach is explicit and of great importance among all approaches. Despite this coherency among all approaches, the problems of *learning*, *planning* and *interaction* have been addressed interchangeably

and inherently in each design. *Learning* in all approaches is reduced to formation of memory models or learning associations thereof to form *contexts*. The problem of *planning* is inherent in their different approaches to *generation* (or deciding on *actions*) over time. And consideration of *interaction* (if any) with external agents have not enjoyed its importance in most designs. We look at different approaches to *learning* and *planning* in sections 6.2.2 and 6.2.3, and after discussing the common interest in memory and context models.

6.2.1 Memory Models

Probably one of the main reasons that music statistical modeling literature has emerged lately compared to parallel achievements in speech or language is inherent in the natural complexity of musical information that bypasses that of speech or language. The complexity of musical representations has its roots in two basic properties of music information: Music information has a natural componential and sequential structure. While sequential models have been extensively studied in the literature, componential or multiple-attribute models still remain a challenge due to complexity and explosion in the number of free parameters of the system. Therefore, a significant challenge faced with music signals arises from the need to simultaneously and instantaneously represent and process many attributes of music information. The ability (or inability) of a system to handle this level of musical complexity can be revealed by studying its means of musical representations or memory models both for storage and learning. In other words, the musical representation or memory models in a system determine the level of expressivity that can be achieved by the system, and also at the onset could draw difficulties or advantages for the problem of machine learning over such representations. The second important property of music information modeling is the inherent *long* temporal dependencies of music signals. Standard approaches to speech and language processing such as regular Markov models and n-gram models are usually sufficient to model the short-time context dependencies in the speech or language domains but fail or give

approximate results when applied to music for context modeling.

We compare different memory models used and proposed in the literature for statistical models of music outlined above. We undertake this comparison by analytically looking at each model's complexity and its modality of interaction across attributes which in terms, determine its power of (musical) expressivity. We will be looking at *cross-alphabets* (Dubnov et al., 2003; Assayag and Dubnov, 2004; Pachet, 2002), *multiple-viewpoints* (Conklin and Witten, 1995) and mixed memory *Factorial Markov models* (Saul and Jordan, 1999).

In order to compare these approaches, we use a toy example demonstrated in Figure 6.1 containing the first measure of J.S. Bach's *two-part invention No.* 5 (Book II). The music score in figure 6.1 is parsed between note onsets to obtain distinct events through time as demonstrated. In this chapter, we consider discrete MIDI signals as our representational front-end for model comparisons and for their wide use and recognition among different systems. For the sake of simplicity, we only represent three most important attributes, namely pitch, harmonic interval (with regard to a base pitch) and quantized beat duration of each parsed event as shown in table 6.1. This table represents 15 vector time series $\boldsymbol{I}_t$ corresponding to 15 parsed events in figure 6.1 where each event has three components (i_t^μ). Let k denote the number of components for each vector I_t and n_ℓs denote the dictionary size for each attribute i_t^ℓ.

Table 6.1: Music attributes for distinct events of parsed score in Figure 6.1

Event Number I_t	$\boldsymbol{I}_1$	$\boldsymbol{I}_2$	$\boldsymbol{I}_3$	$\boldsymbol{I}_4$	$\boldsymbol{I}_5$	$\boldsymbol{I}_6$	$\boldsymbol{I}_7$	$\boldsymbol{I}_8$	$\boldsymbol{I}_9$	$\boldsymbol{I}_{10}$	$\boldsymbol{I}_{11}$	$\boldsymbol{I}_{12}$	$\boldsymbol{I}_{13}$	$\boldsymbol{I}_{14}$	$\boldsymbol{I}_{15}$
MIDI Pitch (i_t^1)	0	51	63	62	63	0	65	0	67	67	63	68	68	58	60
Harmonic Interval	0	0	0	0	24	0	0	0	4	5	0	8	6	0	0
Duration (i_t^3)	1	1	1	1	1	1	1	2	1	1	1	1	1	1	1

Cross alphabets

The simplest model used so far for musical representation is *cross-alphabet* where a symbol (or dictionary element) in the system is represented as a vector of mul-

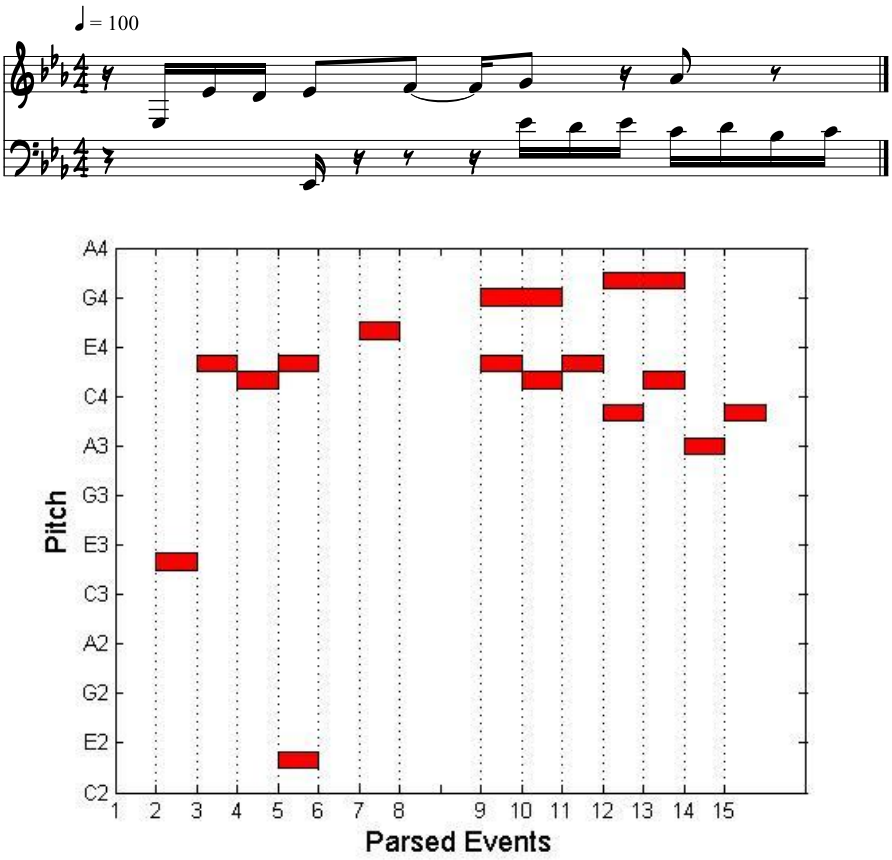


Figure 6.1: *Toy Example* for comparing musical representation approaches – Parsed pianoroll presentation (bottom) for the first measure of J.S. Bach's *two-part Invention No.5*, Book II (top)

tiple attributes. Therefore cross-alphabet models are computationally cheap but fail to model interactions among components. To overcome this shortcoming, researchers have considered heuristic membership functions to obtain contextual dependencies in-between vector components mostly taken from musicological findings pertained to a certain style of music (jazz, classical etc.). Such heuristics might make the system dependent upon the style of music being considered or reduce generalization capabilities. Moreover, as the number of components (or dimensions) increase this representation becomes less informative of the underlying structure since each alphabet is considered independent from another. For example, two harmonically related symbols would not be considered equal in a cross-alphabet representation (if all their individual components are not equal) unless some harmonic equivalence relationship functions are introduced into the system (e.g. Assayag and Dubnov, 2004; Pachet, 2002).

In our toy example each symbol of the alphabet is a unique 3-dimensional vector. Hence, for this specific example there are 15 alphabets since none of them is being reused despite considerable amount of modal interactions among components and high autocorrelations within each component.

Multiple viewpoints

Multiple viewpoints model (Conklin and Witten, 1995) is obtained by deriving individual expert models for each musical attribute and then combining the results obtained from each model in every possible way. This means that a multiple viewpoint representation of our toy example in table 6.1 would have three additional rows for two-dimensional representations of <*pitch, harmonic interval*>, <*pitch, duration*>, and <*duration, harmonic interval*>, and another additional row consisting of a 3-dimensional representation of the three main elements. At this stage, the model's *context* is constructed and then each individual expert model is trained over music data to learn specific musical behaviors.

Multiple viewpoint models are more expressive than cross-alphabet models since

by combining models we allow modal interactions among components. Moreover, the system can reach parts of the hypothesis space that the individual models would not be able to reach. However, the context space is obviously too large and hence, learning requires huge repertoire of music for training data to generate few musical phrases (Conklin and Witten, 1995). In our toy example, with 9 distinct pitches, 6 distinct harmonic intervals and 2 quantized durations, the state-space of the multiple-viewpoint approach leads to $9 + 6 + 2 + 54 + 18 + 12 + 108 = 209$ elements; simply too many for this short toy example.

Factorial Markov Models

Mixed Memory models are geared to situations where combinatorial structure of state space leads to an explosion of the number of free parameters. Factorial Markov models attempt to model the coupling between components in a compact way. To obtain a compact representation, we assume that components at each time t are conditionally independent given the previous vector event at $t-1$, or

$$P(\boldsymbol{I}_t|\boldsymbol{I}_{t-1}) = \prod_{\nu=1}^{k} P(i_t^\nu|\boldsymbol{I}_{t-1})$$

and that the conditional probabilities $P(i_t^\nu|\boldsymbol{I}_{t-1})$ can be expressed as a weighted sum of "cross-transition" matrices,

$$P(i_t^\nu|\boldsymbol{I}_{t-1}) = \sum_{\mu=1}^{k} \phi^\nu(\mu) a^{\nu\mu}(i_t^\nu|i_{t-1}^\mu) \tag{6.1}$$

where $\phi^\nu(\mu)$s are positive numbers that satisfy $\sum_\mu \phi^\nu(\mu) = 1$ and measure the amount of correlation between the different components of the time series. A non-zero $\phi^\nu(\mu)$ means that all the components at *one time step* influence the νth component at the next. The parameters $a^{\nu\mu}(i'|i)$ are $n \times n$ are transition matrices and provide a compact way to parameterize these influences (Saul and Jordan, 1999).

The number of free parameters in eq. 6.1 is therefore upper-bounded by $O(k^2n^2)$

(where n denote $\max(n_i)$[1]) and the state-space size is $\prod_i n_i$. In our toy example the state-space size of the system would then be $9 \times 6 \times 2 = 108$.

6.2.2 Approaches to Statistical Learning

As stated earlier, in most existing systems statistical learning is reduced to learning *contexts* whose predictions could create acceptable actions for a style imitation system. Herein we briefly look at these statistical approaches and study considerations for *interaction* within each proposal. We leave a discussion on planning strategies (or action-decision) to the next subsection.

The most prevalent type of statistical model encountered for music are *predictive* models based on *contextual information* implying general Markov models (Conklin, 2003). Universal prediction methods improved upon the limited memory capabilities of Markov models by creating context dictionaries from compression algorithms, specifically using the Lempel-Ziv incremental parsing (Ziv and Lempel, 1978), and employing probability assignment according to Feder et al. (1992). Music improvisation is then accomplished by performing a random walk on the phrase dictionary with appropriate probabilistic drawing among possible continuations (Dubnov et al., 1998, 2003; Pachet, 2002). Later experiments explored Probabilistic Suffix Tree (PST) (Ron et al., 1996), and more recently in (Assayag and Dubnov, 2004) using Factor Oracle (FO) (Allauzen et al., 1999). These works achieved credible musical results in terms of style imitation. Some informal testing suggested that people could not distinguish between real and computer improvisation for a short period of time (Pachet, 2006). These works are important for showing that major aspects of music can be captured without explicit coding of musical rules or knowledge.

The inference and learning structures for Multiple Viewpoint Models (section 6.2.1) can be categorized as *Ensemble Learning* algorithms and have had multiple manifesta-

[1]In the original Factorial Markov models paper, authors assume that the dictionary sizes are all the same and equal to n (Saul and Jordan, 1999). For the sake of comparison we drop this assumption but keep n as defined above to obtain the coarse definition in equation 6.1.

tions (Pearce et al., 2004; Conklin and Witten, 1995). One advantage of this type of modeling is the explicit consideration of long-term dependencies during learning by combining viewpoint predictions for long-term and short-term models (Pearce et al., 2004). Due to the explosion of parameters, results of learning are hard to visualize and assess. The generation results for these systems are usually few monophonic bars out of learning on an entire database of music (e.g. all Bach chorals); and hence not quite useful for an interactive framework.

Despite the explicit componential representation of Factorial Markov Models, the correlation factors $\phi^{\nu}(\mu)$ model only *one step* dependencies and lack considerations for long-term behavior, essential in computational models of music. Correspondingly, authors use this method to analyze correlations between different voices in componential music time series without considering generation (Saul and Jordan, 1999). However, as seen previously, these models are strong candidates for a comparative study of representational efficiency.

6.2.3 Approaches to Planning and Interaction

Planning refers to strategies that an agent undertakes for action-decision in a given musical situation or context. It is also directly related, in a musical context, to the concept of *interaction* when faced with a constantly moving context (such as music) or in consideration of live and interactive algorithms with an external flow of music information (such as performing with a musician). A major drawback of the above methods is their lack of responsiveness to changes in musical situations that occur during performance, such as dependence of musical choices on *musical form* or changes in *interaction* between players during improvisation (or simply environmental interactions). In all the systems reviewed above the problem of planning is considered inherently and within the generative approaches of each method. Following (Conklin, 2003), we can categorize these systems according to their interactive considerations and planning into two subgroups below:

Random walk methods

Once a *context model* is constructed and learned from the environment, the simplest strategy for generating actions would be to sample a random event from the contextual distributions of events at that stage. Within this framework the more expressive the context model, the better are the generated actions according to the context. This planning method has been employed by many and mostly by real-time music improvisation systems that require fast and immediate system response (Dubnov et al., 1998, 2003; Assayag and Dubnov, 2004; Pachet, 2002).

In general, these methods are destined to produce actions with *overall high probabilities* and are hence *greedy*. Despite the practicality of this method, the musicality of results can be severely questioned since expectancy frameworks do not entail simply to most frequented actions. This is where *interaction* comes to the help of such approaches by considering a constantly moving context and examining the random walk method within an evolving memory model being updated as new information arrives from the environment. While *interaction* in this context is rather implicit, authors of all mentioned systems have considered basic heuristics to enhance the random walk strategy introduced above. For example Assayag and Dubnov (2004) consider equivalence membership functions between cross-alphabets in their memory model to further guide the random-walk and also by explicit accounts of the memory-model structure under consideration. Another example is the system in (Pachet, 2002) for PST based improvisation where interaction is introduced by combining the Markovian probability with a fitness function influencing prediction probabilities according to an ongoing musical context, albeit no consideration for planning or adaptive behavior.

Also, these methods generally suffer from the curse of dimensionality when more intricate musical representations are to be considered. For this reasons, they mostly use cross-alphabets as described above with membership heuristic functions to enhance the musical expressivitiy of their systems pertained to multi-dimensionality of musical information.

Pattern-based sampling methods

The random walk strategies discussed above provide a single-step action at each stage of generation in hope that the learned context models take care of the variable length context dependencies desired in music generation. In general, to take account of the diversity to exhibit creativity, a *long-term* planning strategy is desired that goes beyond a single-step decision making, even in presence of strongly structured learned contexts. Conklin and Witten (1995) present their ensemble learning *multiple-viewpoint* model in a pattern-based sampling framework by considering longer time dependencies (or patterns) within their (exhaustive) context model during decision making. In a later approach to the same problem, Pearce et al. (2004) combine both short-term and long-term viewpoint predictions to achieve more coherent pattern-based sampling.

Despite improvements, the employed models suffer from fixed-range or n-gram models used for pattern retrieval (where parameters are usually fixed by the authors). Moreover, criticisms of random-walk approach employed by Conklin (2003) applies as well to these systems where *planning* is reduced to exploiting the most frequent patterns in the context. In our anticipatory framework, introduced hereafter, planning not only considers pattern-based structures but also subsequent memory cells whose anticipated future paths would generate an expected pattern without necessarily exploiting the pattern itself.

The majority of literature on pattern extraction methods of music signals for automatic generation and improvisation belong to systems following *rule-based* methods for planning and improvisation. These approaches are generally inspired from musicological rules governing a certain style of music (e.g. Rolland and Ganascia, 2000). Another example is the system of Cope (2001) that uses pattern matching techniques to extract recurrent structures in music scores pertaining to a certain style of music or composer and using rule-based methods to recombine these pattern to achieve coherent formal structures during generation.

6.3 General Discussions

The complexity of an improvisation process is beyond one's imagination where the desired complexity of results makes modeling impossible in terms of computed structures and hand-crafted logics. Some cognitive science researchers (Pressing, 1987; Johnson-Laird, 1991) have pointed out the difficulties in *formalizing* complex tasks such as music improvisation as problem solving rules and patterns. In particular, Johnson-Laird questions the pattern-based methods commonly used for many automatic jazz improvisation systems as follows:

> "A common misconception about improvisation is that it depends on acquiring a repertoire of motifs – 'licks' as they used to be called by musicians – which are then strung together one after the other to form an improvisation, suitably modified to meet the exigencies of the harmonic sequence. There are even books containing sets of 'licks' to be committed to memory to aid the process" (Johnson-Laird, 1991, p. 292).

Following Johnson-Laird, the "success" of pattern-based methods does not provide a scientific explanation of the underlying musical complexity of the improvisation process, neither does it lead to frameworks for achieving such complexity, and nor does it undermine the importance of musical patterns.

In our conception and following Simon (1969), the complexity of a structure is not the result of the complexity of the underlying *system* but due to the complexity of its environment. Hence, an architecture that aims at modeling the complexity of an improvisation system must be *adaptive* and demonstrate *intelligence* for learning and adopting interactions between the agents and the environments when desired. Therefore, music complexity is left to the environment (instead of the system) and complex behavior is achieved through *machine learning* once interaction is being placed with a complex environment. The framework presented here is a first attempt in modeling such interactions.

Statistical approaches seem to capture only part of the complex music information, but an essential one, needed for computer generations of music sequences, i.e.

successfully modeling a relatively short term stylistics of the musical *surface*. Even though variable markov length and universal methods improve upon the finite length Markov approach, they are still insufficient for modeling the true complexity and flexibility of music models. It is well known from musical practice that design of musical form, even in situations such as free improvisation, require certain degree of *planning*, with possibility to judge or assign rewards to different musical choices, regardless of them being *correct* in terms of musically syntactical or stylistic rules.

All of the systems reviewed in the previous section are based on predictions out of a learned context. In this work, we extend this view by considering *musical anticipation* and in accord with the psychology of musical expectation. Anticipation, as seen in section 2.2, is different from both prediction and expectation. As a recall, *Anticipation* in our context is the mental realization of possible predicted actions and their effect on the perception and learning of the world at an instant in time. Hence, anticipation can be regarded as a marriage of actions and expectations, addressing directly the problem of *planning* in our context. In this framework, an anticipatory system is in constant interaction with an outside environment for which, it possesses an internal predictive model. In an anticipatory system, action decisions are based on future predictions as well as past inference. It simulates adaptive frameworks in the light of different behaviors occurring in interaction between the system with itself and/or its environment.

Our proposal is built on top of the simple model proposed by Assayag and Dubnov (2004), by augmenting the memory models to accept more complex representational schemes and by introducing an anticipatory framework that addresses planning strategy and interaction through active and adaptive learning with an outside environment. The proposed system in this chapter is both a *payoff anticipatory system* and *state anticipation system* (see section 3.3). It is a state anticipatory system because of explicit use of prediction and anticipation during both learning and decision making. It is also a payoff anticipatory system because of the selective behavior caused by the collaborative and competitive learning and generation discussed later in section 6.5.

With this introduction, it would be natural to consider a *reinforcement learning (RL)* architecture (see section 3.4.1) for our anticipatory framework as is often the case with the *ABiALS* literature (Butz et al., 2003c, 2007). In our proposal, we slightly modify this view by introducing a more efficient paradigm of learning called *Active Learning*.

6.4 Active Learning Architecture

The field of *machine learning* provides various methods for learning regularities in an environment and extrapolating the acquired knowledge to unseen patterns and situations in the real world. These algorithms can be generally viewed within two perspectives: *supervised algorithms* where a set of labeled data (in terms of action-cause) is available and used to learn concepts, and *unsupervised learning* where manual labels of data are not available and the algorithm automatically discovers desired concepts (an example is the *Audio Oracle* structure discovery algorithm in chapter 5). Besides these two mainstream categories, there are situations in which unlabeled data is abundant and labeling is expensive, but the learner *itself* can choose examples to learn from. This type of learning is called *Active Learning* (Monteleoni, 2006). An early landmark research on this topic is the selective sampling scheme of Cohn, Atlas, and Ladner (1994) which became the main inspiration for many subsequent works in the field including this work.

Getting back to our goal of modeling interactive and adaptive learning musical systems, there are two main reasons for considering Active Learning (AL) frameworks: First, in our discussion in section 2.2.2 on *enactive* view of music cognition we emphasizes the role of *sensory-motor* engagement in musical experience. The enactive view of cognition and the link between perception and action (Noë, 2004) is dominated by concerns with visual experience. If for the sake of simplicity and coherence of this presentation we set aside the visual and gestural aspects of a music performance, the sense of an auditory sensory-motor knowledge becomes less obvious than for its visual counterpart. The point here is that perceptual experience is *active* and thoughtful. Following Wessel (2006), one can imagine an auditory version of the classic perception action link

experiments by Held and Hein (1963) where it is shown that that a kitten with a passive exploration experiment of an environment has considerable perceptual impairment compared to the one who was actively transacting with the environment. In the context of an anticipatory learning music system, interaction is nothing but an active exploration and exploitation of the constantly changing environment where the sensory-motor knowledge takes the form of *conceptual understanding* (Noë, 2004, Ch. 6). The second and more technical motivation behind this choice, is the fact that in an interactive setting and for an agnostic system such as ours with no human intervention, the *learner* should have the ability to influence and select its own training data through interaction with a constantly changing environment; hence the general definition of the *active learning* problem. Therefore, learning is an act of auto-organisation in the face of a constantly changing environment to capture relevant expectancy beliefs.

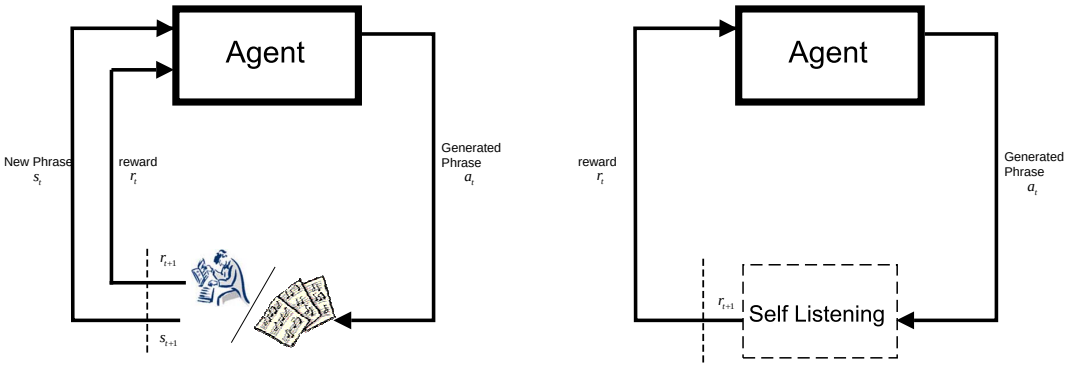
The Active Learning algorithm presented here has close ties to the Reinforcement Learning (RL) paradigm and is a continuation of a previously presented work using RL algorithms in (Cont et al., 2007a). The reinforcement learning problem is meant to be a straightforward framing of the problem of learning from interaction to achieve a goal. The learner and decision-maker is called the *agent*. The thing it interacts with, comprising everything outside the agent, is called the *environment*. These interact continually, the agent selecting actions and the environment responding to those actions and presenting new situations to the agent. The environment also gives rise to rewards, special numerical values that the agent tries to maximize over time. This way, the model or agent is interactive in the sense that the model can change through time according to reinforcement signals sent by its environment. Any RL problem can be reduced to three signals passing back and forth between an agent and its environment: one signal to represent the choices made by the agent (the actions), one signal to represent the basis on which the choices are made (the states), and one signal to define the agent's goal (the rewards) (Sutton and Barto, 1998). Learning in RL is performed by simulating episodes of state-action pairs and updating the policies in order to obtain a maximum reward towards a defined goal. Note that while the simulation episodes during learning

of RL algorithms is a way to solve the burden of unsupervised learning, it does not help the agents to explore the environment actively and would usually require sufficiently large amount of time so that all (relevant) states are visited and updated accordingly. In our *Active Learning* architecture, we adopt the general RL framework but replace RL's reward signals by *guides* that actively explore the existing knowledge domain and help to direct learning updates to *relevant states* in the memory pertaining to the current environmental context and providing numerical values to their information relevancy. In an Active Learning framework exploration is enhanced by guiding the learning agents to the relevant states in the memory given a context where changes are most probable. In this section, we draw the general Active Learning framework of our method.

An Active Learning framework is thus in constant interaction with an ensuing environment. Within our framework, we propose two modes of interaction as demonstrated in Figure 6.2. In the first, referred to simply as *Interaction mode*, the system is interacting with an outside environment (a human performer for live machine improvisation, music score(s) for style imitation, etc.) and occurs when external information is being passed to the system. During the second mode, called *self listening mode*, the system is in the *generation* or *action* phase and is interacting with itself. During the *interaction mode* of figure 6.2a, new environmental information is also presented to the system which help update memory models of the system.

Our proposed Active Learning architecture resembles a model-based RL *dyna* architecture (see section 3.4.1) augmented to multiple collaborative and competitive agents. Each agent in our framework models a separate *mental representation* of the environment and is a learner and decision-maker by its own. Each agent has an internal memory-model of the environment and adapts itself based on new musical information and rewards it receives at each interaction. The proposed framework consists of three main modules each with independent but coherent tasks as follows:

1. *Memory Models:* Constitute state-space representations of the past knowledge observed by the system and enable access to previous state-action paths when-



(a) Interaction Mode (b) Self Listening Mode

Figure 6.2: *Active Learning*'s Modes of Interaction diagrams

ever desired by active learning or generation modules.

2. *Active Selection:* Responsible for transactions between the environment and the improvisation agent. In other words, guiding the agent after each interaction to *relevant* state-action pairs stored in previously learned models based on the current environmental context and assigning reward values.

3. *Anticipatory Learner:* At each interaction with the environment, anticipatory values corresponding to each state-action pair (in each model) is learned through a competitive, concurrent and memory-based learning.

Algorithm 6.1 shows a simplified version of the main interactive cycle of the Active Learning algorithm. This architecture uses learning cycles with multiple competitive and concurrent agents. Upon the arrival of environmental or interactive sequences, the system prepares for learning by (1) calculating *immediate rewards* for stored states in system's memory and (2) selecting relevant states for the anticipatory learning procedure. For this preparatory procedure, responsible for interactivity of the system, we adopt the *Guidage* algorithm presented before in section 5.2 and discussed further hereafter. The memory models used for our architecture are based on *Audio Oracles* as described in section 5.1. In order to use these models in the context of reinforcement

learning, we will view them as a particular case of so called Markov Decision Processes, described in section 6.4.1.

Algorithm 6.1 `Active Learning` Interactive Cycle

Require: At each time t: previously learned models (AO_{t-1}), new environmental sequence $A^t = \{A_1, A_2, \cdots, A_N\}$

1: Obtain active states and guides using `Active Selection` on A^t and AO_{t-1}.
2: **if** we are in *Interaction Mode*, **then**
3: Update Models through learning (AOs)
4: **end if**
5: Perform Competitive, Concurrent and Memory-based Anticipatory Learning

At each interactive cycle between agents and the environment, two instances of *learning* occurs: One for updating memory models (or mental representations) of each agent, and another instance for learning the optimal planning *policy* for collaborative and competitive decision making of each agent. The *policies* learned over action-decisions of each agent use explicit *ancitipatory learning* which assigns *anticipatory values* to each action evaluating their significance in an infinite future horizon given the current environmental and internal context. The anticipatory learning module will be discussed in section 6.5. For the remainder of this section we review *Audio Oracles* and *Guidage* algorithms presented earlier in view of our Active Learning architecture.

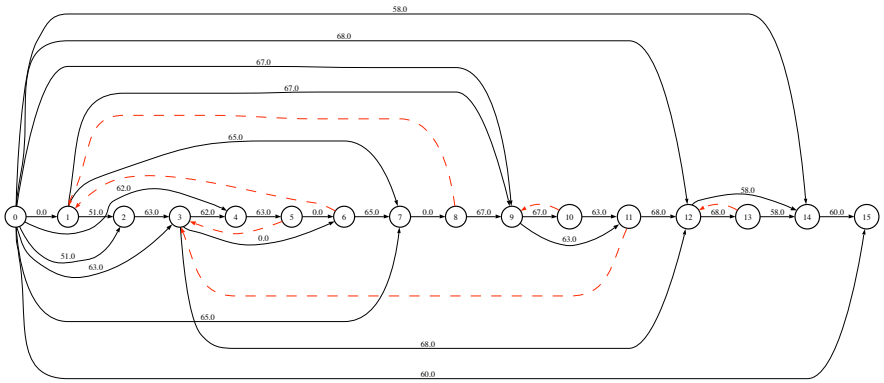
6.4.1 *Audio Oracles* for Memory Models

Representation of musical sequences in our system serves as musical memory, mental representation of music signals and internal models of the agents. Furthermore, they provide methods of access to short-term or long-term context whenever required by the system. A single music signal has multiple attributes and each attribute is a candidate for an individual mental representation which *collaborates* and *competes* with others during actions and decision making. The configuration of representation between these agents have important consequences for information access and also dimensionality of learning algorithms. Following our discussions on the psychology of musical expectation in section 2.1.3, each agent in our system is independent of the other agents in

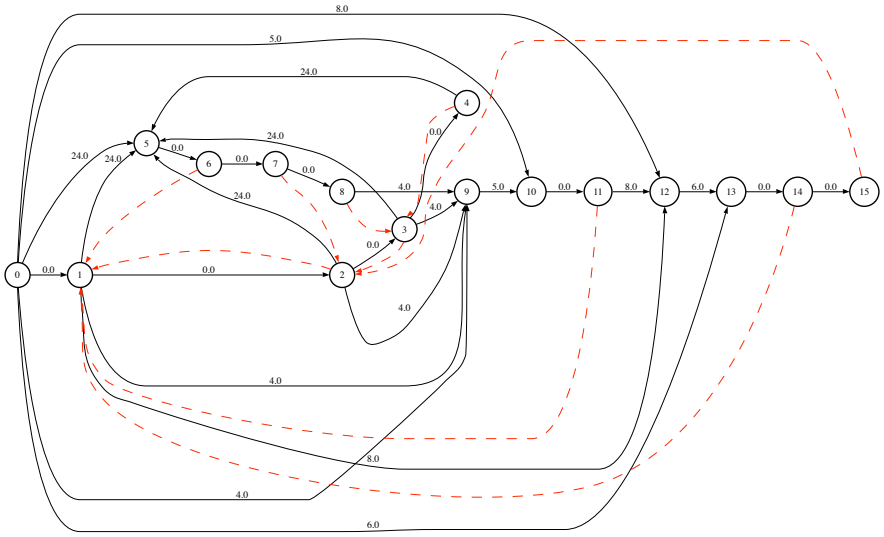
terms of memory models, representing parallel aspects of the same information source, but compete and collaborate during decision making according to their ability to usefully predict the environment. This collaboration and competition is handled by the *anticipatory learning* module discussed in section 6.5. This feature is of great importance since it reduces the dimensionality of the system during learning, allowing it to learn efficiently when small data is available.

The representational framework chosen for this aim should automatically learn the regularities in the environment pertained to its input attribute (and independent from the nature of the attribute) and also provide fast access to these learned contexts during policy learning and interactions with the environment. We discussed this issue previously in part II of this thesis and provided the *Audio Oracle* algorithm for both learning and accessing music information in chapter 5. Therefore, for our interactive *dyna* architecture of this chapter we adopt the same method but by employing several in parallel to represent different aspects of the same information. *Audio Oracles* are independent from the nature of the input data and incremental in nature. The input sequence can be either audio signals (as was the case throughout chapter 5) or symbolic data. For the latter, it only suffices to replace all the divergence similarity thresholding in all mentioned algorithms by a simple equivalence relationship between two symbols. This symbolic version of the oracle, in conformance with previous uses, is referred to as *Factor Oracle* (FO). As an example, figure 6.3 shows three instances of FO construction over sequential data of table 6.1 for the toy example of figure 6.1, representing three attributes of the same sequential data.

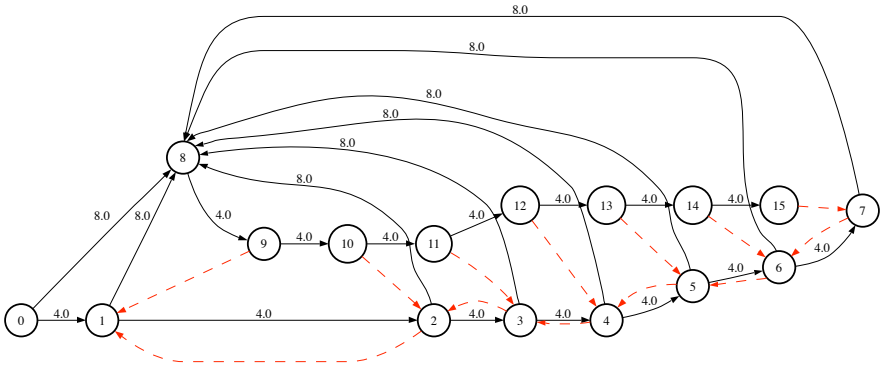
Besides their power of structure discovery and fast access, Factor and Audio Oracles' states can be considered as emitting *actions* which are either symbols or data associated to the factor link or the state itself. This brings in an important property of FO (and consequently AO) for this work which is their power of *generation*. Navigating the oracle and starting in any place, following forward transitions generates a sequence of labeling symbols that are repetitions of portions of the learned sequence; following one suffix link followed by a forward transition generates an alternative path in the sequence,



(a) Pitch FO



(b) Harmonic Interval FO



(c) Duration FO

Figure 6.3: Learned Factor Oracles over pitch, harmonic interval and duration feature sequences in table 6.1. Each node represents a state, each solid line a *factor link* and dashed line a *suffix link*.

creating a recombination based on a shared suffix between the current state and the state pointed at by the suffix link. In addition to completeness and incremental behavior of this model, the best suffix is known at the minimal cost of just following one pointer. By following more than one suffix link before a forward jump or by reducing the number of successive factor link steps, we make the generated variant less resemblant to the original. This property is the main reason for the success of FO based methods for automatic and real-time improvisation systems (Assayag and Dubnov, 2004) even by employing simple random-walk strategies as stated earlier.

The fact that Audio and Factor oracles can be defined by actions associated to states allow us to formally define their structure similar to Markov Decision Processes that are widely used for policy learning. Oracles can then be expressed by *state-action pairs* $(s_i, a_i) : \ s_i \in \mathcal{S}, a_i \in \mathcal{A}$ where $\mathcal{S}$ and $\mathcal{A}$ are sets of all states and actions, and *factor links* as deterministic transition functions $T : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$. Later on, we will associate *anticipatory values* to each state-action pair for better handling of long-term planning behavior.

6.4.2 *Guidage* for Active Selection

One of the most frequent approaches to *Active Learning* is *selective sampling* originally introduced by Cohn et al. (1994) where the learner receives unlabeled data, and request certain labels to be revealed (by an external agent) and hopefully the most informative ones. In our framework, besides gaining computational efficiency, selecting appropriate and relevant states within running memory models (or Audio Oracles) during an interactive cycle would help evade the rather exhaustive trial-and-error learning episodes of traditional reinforcement learning frameworks by focusing directly on states in the memory where *change* in the anticipatory value is most probable. Besides selecting sampling of relevant states, at each interactive cycle and during both modes of interaction (figure 6.2), the agents should be numerically *rewarded* upon their previous performance either indirectly as the result of new environmental information that arrives

into the system, or directly by examining the system's own performance regarding its current context during the self-listening mode.

Therefore, an *active selection* method for our framework should ideally provide each (parallel) memory model with a measure of *information relevancy* and serve as *guides* to these relevant states during policy learning. This issue was the main motivation and point of departure for presenting the *Guidage* algorithms in section 5.2. *Guidage* was used to search for relevant states in an AO structure given an external information chunk providing the depth or longest achieved reconstruction for each sequence of states found as its results (see section 5.2.7 and discussions therein). Therefore, we can easily integrate *Guidage* into our interactive framework for guiding the learning module to relevant states in memory models of each agent and also assigning a reward measure as the longest achieved reconstruction of environmental information within learned context models of memory structures.

Besides their computational attractions, selective sampling methods might have the danger of being too *greedy*, disfavoring many states systematically and leading to unwanted behavioral learning. In other words, *Guidage* in a long run might favor *most seen* patterns in memory models during interaction over rarely seen ones. While this feature is conform to our cognitive observations in chapter 2 that expectations are build on stabilities and regularities of the surrounding environment, it might become troublesome especially during a generation phase by over-emphasizing or over-fitting of a pattern during learning. We overcome this issue by two important considerations: First, the reward signals for the *self-listening mode* of interaction have *negative* signs. In other words, the rewards for the *interaction mode* correspond to a *psychological attention* towards appropriate parts of the memory; and rewards for the *self-listening mode* correspond to a *preventive* anticipation scheme. This means that while interacting with a live musician or sequential score, the system needs to be attentive to input sequences and during self-listening it needs to be preventive so that it would not generate the same (optimal) path over and over. And second, our anticipatory learning (for both modes of interaction) extensively makes use of data structures inherent in its memory models

to update not only selected states by *Guidage* but also appropriate contexts that lead to these states in the current memory. This issue is discussed next.

6.5 Anticipatory Learning

The *Guidage* procedure described earlier, provides *immediate rewards* for undertaking different actions for each state in stored memory models depicted as $r(s_i, a_i)$ where $r : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$. While these values can provide instant continuations for a generative process given a context, they simply fail to maintain a coherent context-based structure in longer time spans. In order to maintain long-term coherency during generation either a human-operator or an external learning agent is needed to optimize the temporal context of the generated sequence. The idea behind *Anticipatory learning* is to further enhance the knowledge of the system by blending the ongoing immediate guides with their values in an infinite future horizon.

Given the memory models (Audio or Factor Oracles) presented in section 6.4.1 as state-space decision processes and guides from active selection (or *Guidage*), Anticipatory Learning procedure aims at finding policies as a mapping probability $Q(s, a) : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$. This way the policy can be represented as a matrix Q which stores values for each possible state-action pair in each memory model.

The policy Q can be learned using the regular Q-learning algorithm frequently used in Reinforcement Learning literature. In a simple single-agent RL problem and at each interaction time t, the policy Q is learned through simple value iteration updates for each state-action pair through learning *episodes* that are basically simulations of the system ornamented by updates. These simulation episodes in our context are similar to a *rehearsing musician*, bringing to the front relevant past expectations and performing them over and over until satisfied. During each step of an episodes of Q-learning, the agent simulates a one-step prediction $(s_t, a_t) \rightarrow s_{t+1}$ and updates the policy for state-action pair (s_t, a_t) by the following factors:

1. An *infinite horizon predicted reward* factor $R(s)$, different from the immediate reward $r(.,.)$, where the idea comes from our interest in the impact of future predictions on the current state of the system. This value is defined as:

$$R(s_t) = \sum r(s_t, a_t) + \gamma r(s_{t+1}, a_{t+1}) + \cdots + \gamma^m r(s_{t+m}, a_{t+m}) + \cdots \quad (6.2)$$

   where γ is a discount factor and less than one. Equation 6.2 simply simulates *future* behavior by an ϵ-greedy algorithm (or random-walk) over current belief in Q and estimate the predicted reward than the system would gain by stepping in state s_t.

2. A *Time Difference* update as $(\max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t))$,

leading to the simple Q-learning algorithm (Sutton and Barto, 1998):

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[R(s_t) + \gamma \cdot \max_{a'}(Q(s_{t+1}, a')) - Q(s_t, a_t) \right] \quad (6.3)$$

where α is a fixed *learning rate*. It is shown that if these training episodes are done in sufficiently large number of times, spanning all relevant states, the behavior policy Q converges to the optimal behavior π at each interaction cycle.

In our *Active Learning* framework we adopt the Q-learning for policy learning of our *Dyna* architecture, except that instead of leaving episodes run *free* simulations of system behavior we *guide* the simulations to sampled states as a result of *Active Selection*. We also extend this simple framework by (1) integrating collaborative and competitive learning between multiple (parallel) agents that contribute to the same environment, and (2) by integrating the data structures during learning to achieve memory-base learning.

6.5.1 Competitive and Collaborative learning

As discussed previously in section 2.1.3, different mental representations of music work in a collaborative and competitive manner based on their predictive power to make decisions. This can be seen as kind of a *model selection* where learning uses all the

agents' policies available and chooses the best one for each episode. This winning policy would then become the *behavior policy* with its policy followed during that episode and other agents being influenced by the actions and environmental reactions from and to that agent.

At the beginning of each episode, the system selects one agent using the Boltzmann distribution in equation 6.4, with positive parameter β_{sel}, and M as the total number of agents or attributes. Low β_{sel} causes equi-probable selection of all modules and vice versa. This way, a behavior policy π^{beh} is selected *competitively* at the beginning of each episode based on the value of the initial state s_0 among all policies π^i as demonstrated in equation 6.4. A similar competitive scheme has been previously employed in (Singh et al., 1994) in the context of Partially Observable Markovian Decision problems.

$$Pr(i|s_0) = \frac{\exp(\beta_{sel} \sum_k Q^i(s_0, a_k))}{\sum_{j=1}^{M} \exp(\beta_{sel} \sum_r Q^j(s_0, a_r))} \quad , \quad \pi^{beh} = \underset{i}{\operatorname{argmax}}\, Pr(i|s_0) \quad (6.4)$$

During each learning episode, the agents would be following the behavior policy. For update of π^{beh} itself, we would use the regular Q-learning algorithm of eq. 6.3 described above but in order to learn other policies π^i, we should find a way to compensate the mismatch between the target policy π^i and the behavior policy π^{beh}. Uchibe and Doya (2004) use an *importance sampling* method for this compensation and demonstrate the implementation over several RL algorithms. Adopting their approach, during each update of π^i when following π^{Beh} we use the compensation factor

$$IS = \frac{Q^i(s_m, a_m)}{Q^{Beh}(s, a)}$$

during Q-learning as depicted in eq. 6.5, where (s_m, a_m) is a *mapped* state-action pair of (s, a) in behavior policy to the oracle of attribute i. As before α is the learning rate, and $R(.)$ as defined in eq. 6.2.

$$Q^i(s_m, a_m) = Q^i(s_m, a_m) + \alpha \left[R(s_m) + \gamma \cdot IS \cdot \max_{a'}(Q^i(s_m, a')) - Q^i(s_m, a_m) \right] \quad (6.5)$$

This scheme defines the *collaborative* aspect of interactive learning. For example, during a learning episode, pitch attribute can become the behavior policy Q^{beh} and during that whole episode the system follows the pitch policy for simulations, and other attributes' policies $Q^i(.,.)$ will be influenced by the behavior of the pitch policy as shown in equation 6.5.

6.5.2 Memory-based Learning

In the Q-learning algorithm above, state-action pairs are updated during each episode through an ϵ-greedy algorithm on previously learned policies and using updated rewards. This procedure updates one state-action pair at a time. In an ideal music learning system, each immediate change should also evoke context-related states already stored in the memory. In general, we want to go back in the memory from any state whose value has changed. When performing update, the value of some states may have changed a lot while others rest intact, suggesting that the predecessor pairs of those who have changed a lot are more likely to change. So it is natural to prioritize the backups according to measures of their urgency and perform them in order of priority. This is the idea behind *prioritized sweeping* (Moore and Atkeson, 1993) embedded in our learning. In this memory-based learning, priorities are assigned to each state according to the magnitude of their *update* and chosen by a threshold θ. This priority measure is demonstrated in equation 6.6 for a current state s and predicted next state s'.

$$p \leftarrow |R(s) + \gamma \max_{a'}(Q^{Beh}(s', a')) - Q^{Beh}(s, a)| \tag{6.6}$$

The ensemble of state-actions pairs whose priorities are identified as significant or $p > \theta$ constitute a *priority queue* list $\mathcal{PQ}$. The priority queue list $\mathcal{PQ}$ acts as a *stack memory* with a *Last-In-First-Out* mechanism where *pulling* the first element out takes it out of the stack. During prioritized sweeping the state-action pairs in $\mathcal{PQ}$ are traced *backwards* on the memory model's state-space structure to undertake more updates and obtain more priority states recursively.

6.6 Active Learning Algorithm

With all the components of our Active Learning architecture visited, we can now put everything within one framework and detail the previously presented Active Learning interactive cycle algorithm on page 149. Algorithm 6.2 shows the resulting algorithm. This algorithm shows the learning procedure within one interaction cycle between the environment and the system (the *interaction mode*), or within the system itself (*self-listening* mode). This framework brings in previous beliefs of the agents along previous interactions from memory models AO^i and anticipatory policies Q^i for each of the parallel agents into the learning equations. The *guides* are obtained *before* updating memory models and separately for each memory model. The algorithm then chooses the *behavior policy* competitively among all available models, and undertake collaborative memory-based learning as described in the previous section.

Within each interaction cycle, the learning is done over N fixed number of episodes (line 14 onwards of algorithm 6.2) and specifically on *Active States* as a result of our selective sampling method of *Guidage*. Moreover, to avoid over-fitting we devised a *prioritized sweeping* where each active state is traced back in the memory structure and added to the *priority queue* stack $\mathcal{PQ}$ for update considerations if its priority measure is important. This scheme assures a compromise between exploration and exploitation during learning. Moreover, instead of undertaking learning for each model separately (which is very costly for a high-dimensional interactive setting), we devised a *collaborative learning* scheme where corresponding state-action maps of the behavior policy to other policies are *influenced* by the updates of the winning behavior at each episode and through importance sampling as discussed in section 6.5.1.

These considerations greatly reduce computation time and complexity of our algorithm compared to its RL equivalences, and also to most systems discussed in section 6.2. Below we will give an account of model complexity of our algorithm in comparison to previously discussed methods.

Algorithm 6.2 `Active Learning` Interactive Cycle (detailed)

Require: At time t: Previously learned memory models AO^i_{t-1} for all models i, Previously learned policies $Q^i(s,a)$, New sequence A^t from environment.

Initialize *priority queue* $\mathcal{PQ} = \emptyset$
for All models i **do**
  $\mathcal{I}^i \leftarrow Guidage(AO^i_{t-1}, A^t)$ for all models i (algorithm 5.4)
  Update *immediate reward* $r^i(.,.)$ by reconstruction lengths in $\mathcal{I}^i$
end for
if We are in *Interaction Mode* (figure 6.2) **then**
  $\forall i$, update $AO^i_{t-1} \rightarrow AO^i_t$ with the new sequence A^t (algorithm 5.2)
end if
Choose the *behavior policy* Q^{beh} (eq. 6.4)
$\mathcal{PQ} \leftarrow$ Active States in $\mathcal{I}^{beh}$
$s \leftarrow$ current state
$a_{beh} \leftarrow \epsilon$-greedy over $Q^{beh}(s,.)$
Execute action $a_{beh} \rightarrow s'$, and cumulate future rewards $R(s)$ (eq. 6.2).
for N number of trials **do**
  $p \leftarrow |R(s) + \gamma \max_{a'}(Q^{beh}(s',a')) - Q^{beh}(s,a)|$
  if $p > \theta$ **then**
    insert (s,a) into $PQueue$ with priority p
  end if
  while While $\mathcal{PQ} \neq \emptyset$ **do**
    $(s,a) \leftarrow$ `PULL`$(\mathcal{PQ})$
    Execute $a \rightarrow s'$, and cumulated future rewards $R(s)$
    for for all model policies Q^i **do**
      if $Q^i \neq Q^{beh}$ map $(s,a) \in AO^{beh}_t \rightarrow (s_m, a_m) \in AO^i_t$
      if (s_m, a_m) exists, do *Q-learning* with *importance sampling* (Eq. 6.5)
    end for
    for all $(\hat{s}, \hat{a})$ predicted to lead to s **do**
      $\hat{R} \leftarrow$ predicted reward on Q^{beh}
      $p \leftarrow |\hat{r} + \gamma \max_a(Q^{beh}(s,a)) - Q^{beh}(\hat{s},\hat{a})|$
      if $p > \theta$ **then**
        insert $(\hat{s},\hat{a})$ into $\mathcal{PQ}$ with priority p.
      end if
    end for
  end while
end for
return Learned Behavior Policies Q^i

6.6.1 Model Complexity

In the architecture introduced above, because of the concurrent and competitive multiple agent structure, each component or attribute is modeled separately and the state-space size increases linearly with *time* as $k \times T$ coming down to 45 for the toy example of figure 6.1 (compare to 209 for *Multiple Viewpoints* of (Conklin and Witten, 1995), and 108 for Factorial Markov Models of (Saul and Jordan, 1999) as shown in section 6.2.1). Modal interaction is not considered by directed graphs between components but rather by influence of each attribute on others through an *Importance Sampling* procedure in equation 6.5 as a result of collaboration and competition between agents. This choice was also motivated by cognitive foundations of music expectation rather than mere simplicity. The complexity of the system depends linearly on T, n_is (or number of *alphabets* in each memory model i) and an adaptive environmental factor. This is because the arrows of the state-space model are inferred on-line and are dependent on the context being added to previous stored knowledge. We could say that our learning has an upper-bound complexity of $O(nkT)$ where $n = \max(n_i)$, but is practically sparser than this upper bound.

The fact that T is a factor of both state-space size and complexity has advantages and shortcomings. The main advantage of this structure is that it easily enables us to access long-term memory and calculate long-term dependencies, induce structures, and go back and forth in the memory at ease. However, one might say that as T grows, models such as Factorial Markov would win over the proposed model in terms of complexity since n_i would not change too much after some time T. This shortcoming is partially compensated by considering the phenomena of *finite memory process*. A finite memory process in our framework is one that, given an oracle with N states and an external sequence A^t, can generate the sequence through a finite window of its history without using all its states (Martin et al., 2004). More formally, this means that there exist a non-negative number m such that the set $\{s_n \in AO : n \in \mathbb{N} \text{ and } n \in [N-m, N]\}$ would suffice for regeneration of a new sequence A^t. This notion is also supported by the fact

that music data in general is highly repetitive (Huron, 2006). The parameter m is usually dependent on the style of music but since our framework adaptively searches for relevant context, we keep it sufficiently large (≈ 200 states) to cover most of the knowledge domain and to evade over-fitting of earlier memory states in large sequences.

Besides reducing representational complexity. As will be seen shortly, the proposed framework does not need an extensive amount of data for training. Due to the adaptive nature of selective sampling used during learning, it is basically sensitive to any relevant context and propagates relevant information throughout all models whenever appropriate. We demonstrate these advantages in the following section through different experiments.

6.7 Results and Experiments

In this section we explore the results and use of the anticipatory *Active Learning* framework presented previously. The premise of the framework is its use in *interactive* and multi-dimensional settings. Interaction is obtained through continuous information exchange with an outside environment. The environment in this setting is anything that lives outside the system itself, and can be a human operator, music information from live musicians or even sequential data from a given music score. Through these interactions, the algorithm builds memory models over the data and assigns anticipatory values to each state-action pertaining to the ongoing context of the interaction.

We demonstrate the performance of our framework in two steps: First, we set two simple experiments to demonstrate the effect of learning through simple interaction in order to evaluate belief propagation in between interactions and see the agents in *action* (or in generation) as a result of policy learning. We term these basic experiments *knowledge-base interactions* since in each case an outside signal containing external knowledge is influencing an already established knowledge within existing memory models. After understanding the basic premises of this interactive learning, we move on to a more complex musical setting in section 6.7.2 for automatic style imitation or

improvisation.

6.7.1 Knowledge-Based Interactions

We demonstrate single-shots of interactive cycles of our Active Learning framework where interaction between the system and the environment is directly controlled. Our main goal here is to study the behavior change of system after each interactive cycle. Therefore, we limit ourselves for this setup to simple monophonic melodies. We further assume that the system disposes some *previous knowledge* of some music phrases. We then interact with this knowledge by feeding the system external information and hence simulating a learning cycles and study the results.

For the two experiments presented here, we consider an excerpt of J.S. Bach's solo violin line from Aria Number 39 of *Saint Matthew's Passion* as *a priori* knowledge-base of the system. The score of the considered fragment is demonstrated in figure 6.4. We choose a representational scheme similar to the toy example in section 6.4.1. Features are extracted out of parsed MIDI signals which are chosen to be <*pitch, duration (beats), pitch contour,* Δ *duration*> for this experiment. After feature extraction, the score of figure 6.4 becomes a 136×4 table (similar to table 6.1 for the toy example) with 136 events in time. Similar to section 6.4.1, each of the four musical attributes are then translated into four Oracles constituting the memory models of four competitive and collaborative agents. Note that the nature of chosen features can be anything as long as they are discrete sequences and the choices above are just for demonstration purposes[2]. The policies of the four agents Q^i are also initialized to an equal ϵ value for each state-action pair.

At this stage, we are ready to simulate interactive cycles of Active Learning by hand-feeding patterns into the system. In the two experiments that follow, we will be interacting with the system through different reinforcement attributes. In a real setting, the entry of the system consist of multiple-attribute music information. However, for the

[2]Original MIDI files used for and generated out of these experiments are accessible in the following URL: http://cosmal.ucsd.edu/arshia/index.php?n=Main.Quals

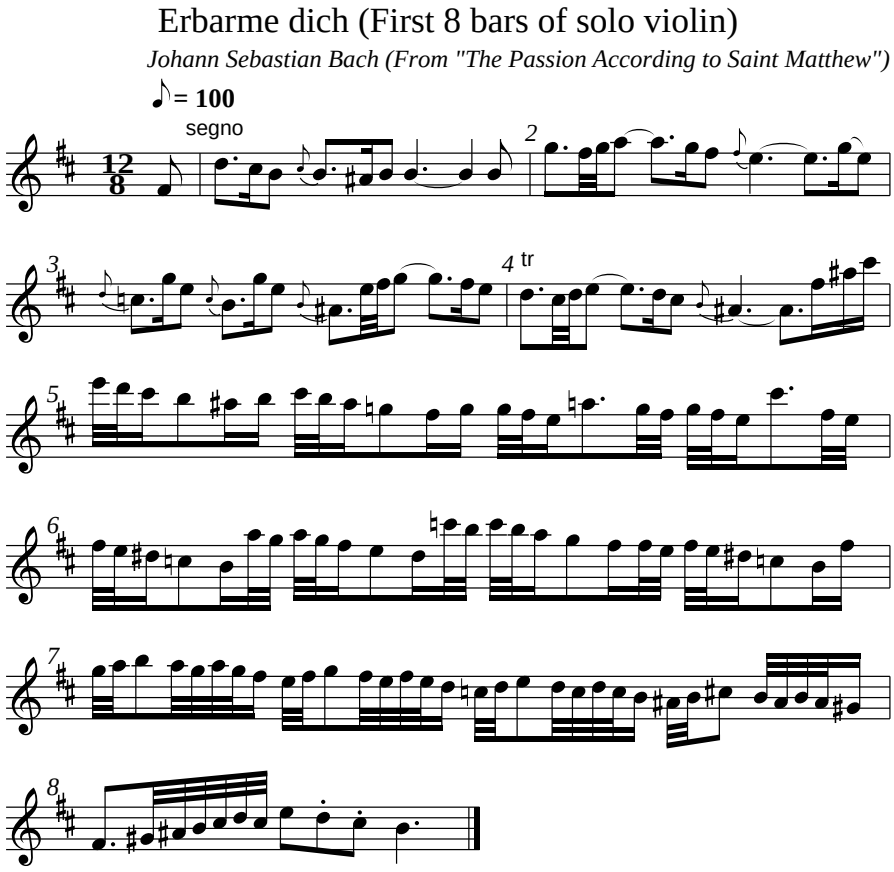


Figure 6.4: Score excerpts of *Erbarme Dich*, Aria No. 39 from Bach's *Saint Matthew's Passion* – solo violin part.

sake of observation simplicity in these two experiments we focus on specific attributes (by skipping model selection) and concentrate on anticipatory learning of algorithm 6.2 and within the self-listening mode (i.e. not updating models).

Experiment 1: Reinforcing patterns

For our first simple experiment, we interact with the system by showing it a *relative pitch pattern* as depicted in figure 6.5. This pattern is actually the pitch contour of the first 6 notes of the second theme in measure 2 of the score of figure 6.4. This pattern is fed into the system as the environmental sequence $A^t = \{-1, 1, 2, -2, -1\}$. Since this pattern is only concerned with one particular component (pitch contour), this attribute becomes the *behavior policy* of Active Learning algorithm and other policies are influenced by this behavior through *collaboration* between agents.

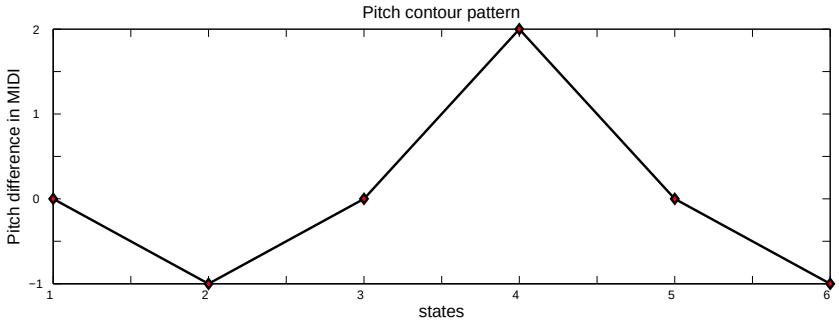


Figure 6.5: Pitch contour pattern used in Experiment 1.

Using the four available memory models and this entry, algorithm 6.2 undertakes learning and updates the four Q^i policies for each agent. For this learning session, we set $N = 20$, $\gamma = 0.8$, and $\alpha = 0.1$ as learning parameters of algorithm 6.2. Figure 6.6 shows a snapshot view of learned anticipatory values for all state-action pairs for the *pitch* policy. The *actions* on the y-axis of this figure are sorted MIDI pitch numbers and the x-axis indexes are 136 existing states in the corresponding Factor Oracle. The gray-scale values indicate the magnitude of learned values for each state-action pair or $|Q(s, a)|$.

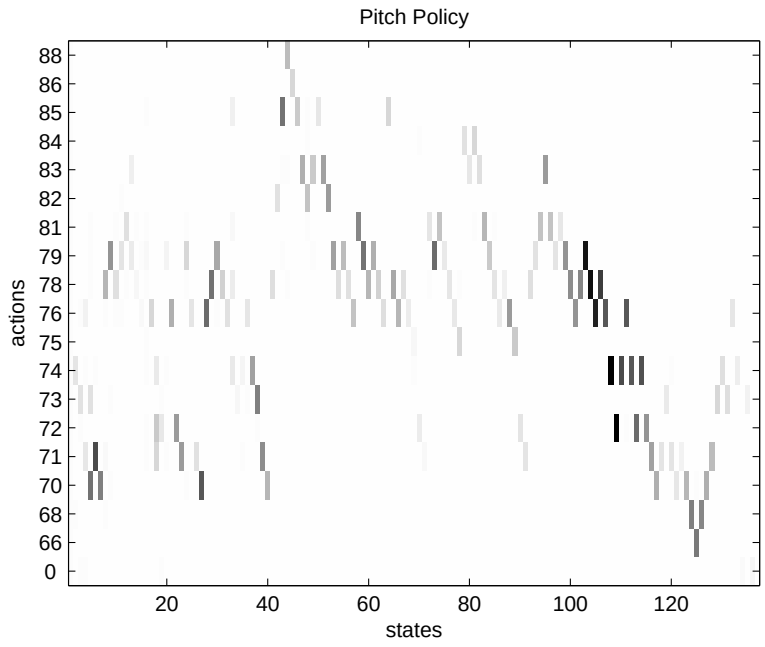


Figure 6.6: Q matrix for pitch agent after reinforcement of pattern in figure 6.5 using Active Learning.

Although the `pitch contour` agent is used during learning as *behavior policy*, figure 6.6 show a different agent's learned policy to emphasize the *influential* aspect of collaborative learning. We also chose the `pitch` agent for visualization since the contours are detectable by the eye. Following the gray scaled values, some instances of the original reinforced pattern as well as subsets of the pattern are clearly identifiable on the learned policy. But there are other rather irrelevant state-action paths which are also emphasized during learning. These mediocre valued policies are mainly due to our memory based anticipatory learning and correspond to paths whose *future* actions would *guide* to the reinforced pattern. Such actions are valuable because their *anticipated* outcome in the long-run would increase the overall reward structure. This simple fact distinguishes our approach from string-matching or pattern-recognition algorithms which do not usually consider anticipated outcomes of actions. The policies learned at each interaction cycle demonstrate *anticipatory profiles* of different components through

time.

To better understand the effects described above, we force the system to generate 100 events using random-walk over the learned policies. For this random-walk generation, we choose a *behavior policy* on which random-walk is undertaken and at each generation step map the behavior policy state-action sequence to subsequent agents to generate full multiple-attribute music events. We call this simple generative scheme *collaborative generation*. Also, to make our improvisation setting more explorative (and avoiding repetitions) we undertake this generation in five episodes (20 generation steps each) and negatively reinforce back the newly generated signals into the system using our Active Learning framework in between episodes. Recall that this generative mode of interaction is referred to as *self-listening* mode (figure 6.2). The generated score is demonstrated in figure 6.7. A close look at this generated score highlights the dominancy of the pitch pattern that was reinforced during learning.

The goal of generation here is to show the influential aspect of information during our Active Learning framework and not the artistic output. A simple comparison of the score in figure 6.7 and that of figure 6.4 highlights two striking facts: The generation results are consistent with the reinforced pattern and knowledge added to the system and also they *extrapolate* the original knowledge domain (e.g. new pitches appear that do not exist in the original score) in a consistent manner with the acquired knowledge and even when little data is available. We will get back to these points in section 6.7.2.

Experiment 2: Accumulating and blending Knowledge

Now imagine that we want to keep the previously acquired knowledge and blend-in new and different sort of knowledge into the system hoping to maintain what has been learned previously. In other words, we are interested in the *accumulation* of knowledge within different interactive cycles. The Active Learning framework is actually made for this kind of interaction through time.

For this experiment, we choose the rhythmic pattern in figure 6.8 as a different

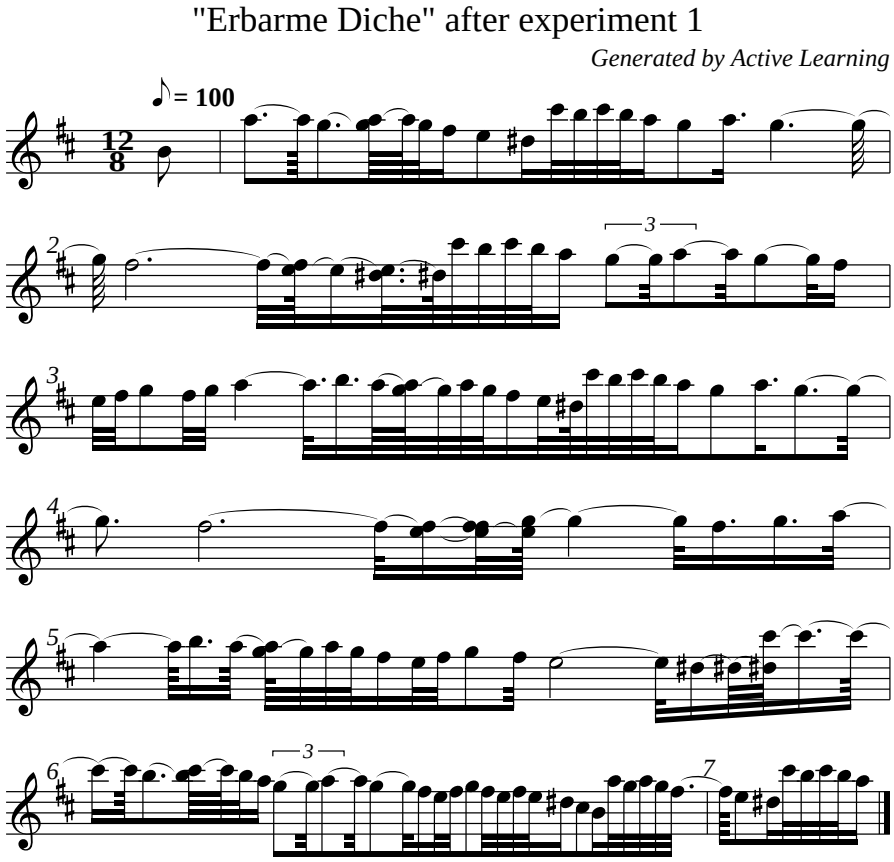


Figure 6.7: Generation of 100 events after Experiment No.1

source of knowledge than the one used in the previous experiment. This pattern also appears first at the beginning of measure 5 in the score of figure 6.4.

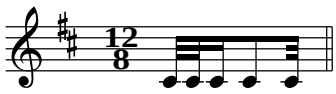


Figure 6.8: Rhythmic pattern used in experiment 2.

After the introduction of this pattern, it is obvious that either the duration or its derivative should be chosen as behavior policy during learning. Choosing the second will take into account all possible rhythmic modulations available in the memory but we choose the duration itself since this example is quite short and consistent with rhythm. As before, we choose the pitch policy to show the result of learning in figure 6.9.

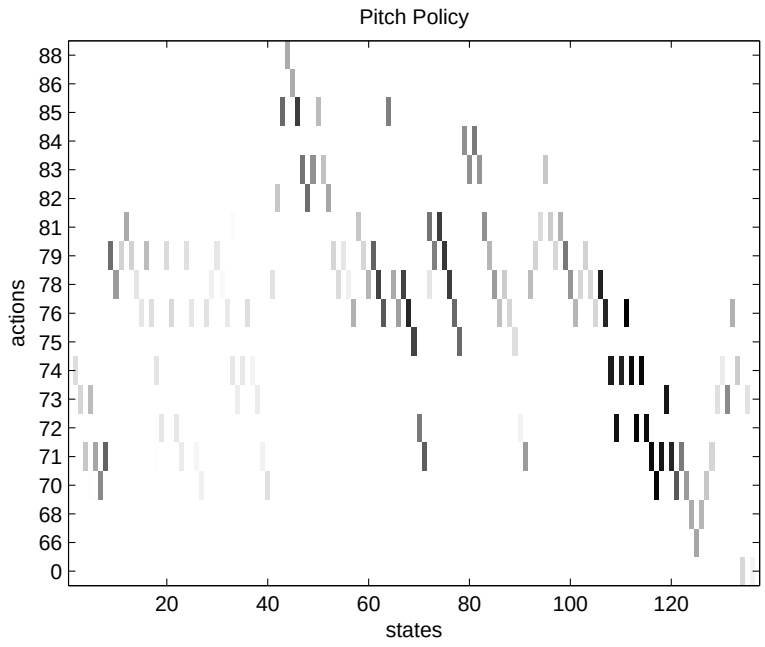


Figure 6.9: Q matrix for pitch agent after reinforcement of pattern in Figure 6.8 using Active Learning.

A visual comparison of figures 6.6 and 6.9 reveals that in the second, some middle state values have been emphasized while the overall structure of previously learned policy are maintained. Perhaps a better explanation would be to look at what the system would generate over time using this new episode of learning. A sample result is demonstrated in figure 6.10. A close look at the score reveals two dominant structures: The first occupying almost half of the score until the end of measure 3 is the result of our recent reinforcement of the rhythmic pattern with pitch contour flavors of the previously learned knowledge. The second structure starting from measure 4 towards the end has the same pattern as in figure 6.5.

6.7.2 Anticipatory Style Imitation and Automatic Improvisation

In both experiments above, we saw how our anticipatory learner handles pattern reinforcements, and accumulation and blending of knowledge on simple signals. The

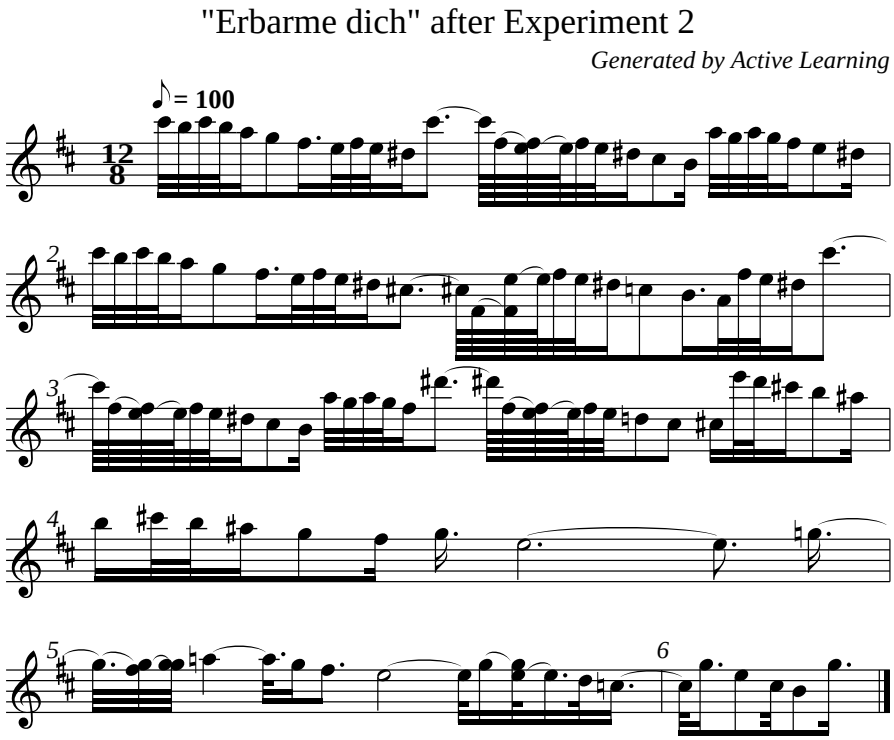


Figure 6.10: Generation of 100 events after Experiment No.2

main promise of this chapter is to devise an anticipatory framework that can simulate complex musical behavior with no a priori knowledge and by direct interactive learning through an environment. Here, we demonstrate our system in applications to automatic music improvisation and style imitation as a direct way to showcase complex musical behavior. Musical style modeling consists of building a computational representation of the musical data that captures important stylistic features. Considering symbolic representations of musical data, such as musical notes, we are interested in constructing a mathematical model such as a set of stochastic rules, that would allow creation of new musical improvisations by means of intuitive interaction between human musicians or music scores and a machine.

We devise this experiment as follows: The system is trained on a specific piece of music by sequentially interacting with the score using full functionalities of algorithm 6.2 and our Active Learning framework. After training, the system turns into a generative phase and generate a *musical piece* by composing and also interacting with its own behavior.

For the representational front-end of this experiment, we hand-picked 3 different attributes (pitch, harmonic interval and quantized duration in beats) along with their first difference, hence a total of 6. Upon the arrival of a MIDI sequence it is quantized, cut into polyphonic "slices" at note onset/offset positions (figure 6.1), and then different attributes are extracted out of each slice. Slice durations are represented as multiples of the smallest significant time interval that a musical event would occupy during a piece (referred to as *tatum*). This representational scheme is similar to the toy example that resulted in table 6.1 except that 3 additional rows corresponding to the derivatives of those are added in. This constitutes our experimental mental representations or agents of the system. During training, interaction cycles are simulated by feeding the music score sequentially into the system. By the time these interaction cycles cover all of the music score, memory models and policies are learned and ready for generation.

There are many ways to generate or improvise once the policies for each attribute are available. We represent one simple solution using the proposed architecture. At

this stage, the system would be in the *self listening mode*. The agent would generate *phrases* of fixed length by undertaking a random-walk on a behavior policy (obtained from the previous interaction). When following a behavior attribute, the system needs to *map* the behavior state-action pairs to other agents in order to produce a complete music event. For this, we first check and see whether there are any common transitions between original attributes and, if not, we would follow the policy for their derivative behavior. Once a phrase is generated, the system runs an interactive learning cycle in *self listening mode* but with *guides* considered as *preventive* or *negative* reinforcement. This generative scheme was also used in the previous experiments and can be regarded as a *collaborative* generation.

For this experiment we demonstrate results for style imitation on a polyphonic piece of music: *Two-part Invention* No.3 by J.S. Bach. The training phase was run in *interaction mode* with a sliding window of 50 events with no overlaps over the original MIDI score. After this training phase, the system entered *self listening* mode where it generates sequences of 20 events and reinforces itself until termination. Parameters used for this session were $\alpha = 0.1$ (in eq. 6.5), $\gamma = 0.8$ (in eq. 6.2), $\theta = 0.5$ for prioritized sweeping threshold, and $\epsilon = 0.1$ for the $epsilon$-greedy selection or random-walk on state-action pairs. Number of episodes (N) simulated during each Active Learning phase was 20. The generated score is shown in figure 6.11 for 240 sequential events where the original score has 348. For this generation, the *pitch* behavior *won* all generation episodes and direct mappings of *duration* and *harmonic* agents have been achieved 76% and 83% in total respectively leaving the rest for their derivative agents.

While both voices follow a polyphonic structure, there are some formal musicological structures that can be observed in the generated score. Globally, there are *phrase* boundaries in measures 4 and 11 which clearly segment the score into three formal sections. Measures 1 to 4 demonstrate some sort of exposition of musical materials which are expanded in measures 7 to the end with a transition phase in measure 5 and 6 ending at a week cadence on `G` (a fourth in the given key). There are several thematic elements which are reused and expanded. For example, the repeated `D` notes appearing in mea-

Improvisation Session after learning on Invention No.3 by J.S.Bach

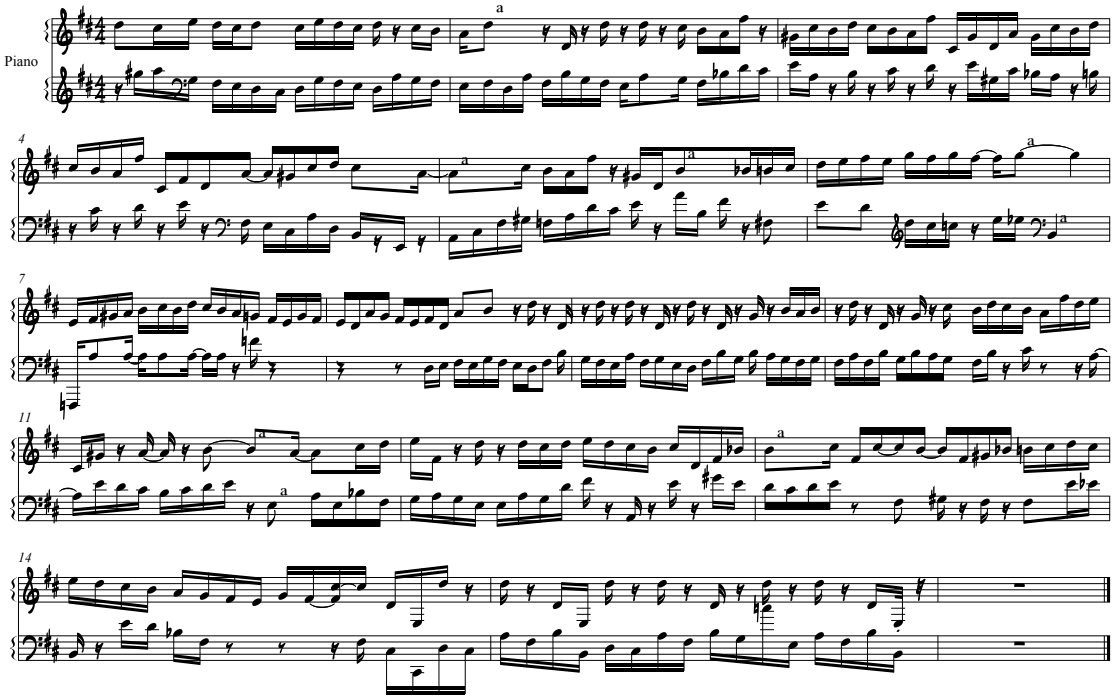


Figure 6.11: Style imitation sample result

sures 2 appear several times in the score notably in measure 7 as low A with a shift in register and harmony in measures 9 and 15. More importantly, these elements or their variants can be found in the original score of Bach.

Figure 6.12 shows the pitch-harmony space of both the original MIDI and the generated score. This figure suggests that the collaborative and competitive architecture has resulted in combination of attributes that do not exist in the original domain.

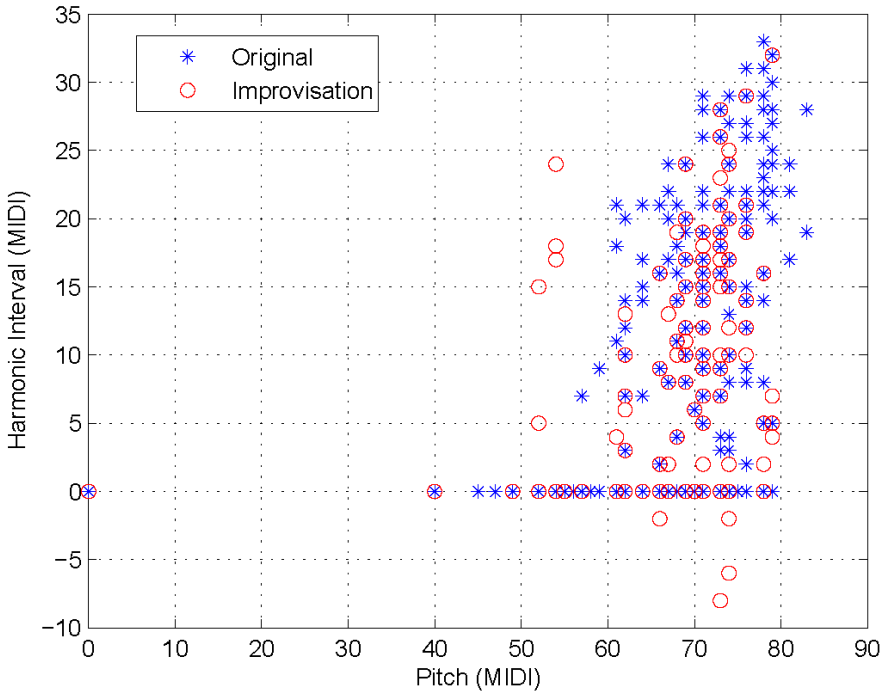


Figure 6.12: Improvisation Space vs. Original Space

Note that these results are obtained by an agnostic system with no a priori knowledge of the musical structures and no human intervention during learning and generation. The observed results suggest:

- Some degree of *formal* long-term planning behavior obtained during training and exercised during generation.
- Achieving anticipatory behavior and learning even in presence of little training

data (compared to existing literature).

- Complex behavior within a relatively simple statistical framework as a result of adaptive and interactive learning with an environment

6.8 Discussions

In this chapter we aimed at addressing *how* to build and act upon mental expectations through a simple anticipatory behavior learning framework. In our proposed framework, an anticipatory learning system is in constant interaction with an outside environment, updating its memory, beliefs and planning strategies according to the environmental context at each interaction cycle. Our aim was to implement adaptive learning by putting forth the notion of anticipation in learning rather than mere prediction.

Computationally speaking, our system features an agnostic view of its environment where all beliefs and behavioral actions are learned by observing the ongoing environment through continuous interactions. We showed that such a framework is capable of constituting complex musical behavior without a necessarily complex model as a result of anticipatory adaptations and demonstrated results for style imitation and automatic improvisation. We also showed that the system, through its modular and adaptive memory models and thanks to the competitive and collaborative learning framework, is capable of behavior learning when little data is available and generates new context not necessarily available in the hypothesis domain. These facts make the system a strong candidate for considerations as an interactive computer music system.

On the cognitive side, the framework presented in this chapter aims at modeling short-term veridical expectations as well as dynamic expectations as defined in section 2.1.4. We also showed within two experiments how the system brings in previous knowledge and blends them in with new beliefs. In order to achieve other types of expectancies such as schematic or long-term veridical expectations, one would need to bring in *compact* representations of previous knowledge preferably in syntactic forms

into new experience. The idea here is to deduct latent belief structures from current memory models and store them for future use and access. This idea will be strongly pursued for future research.

We also showed the proposed framework in application to automatic improvisation and knowledge-base interactions. We mentioned these results as experiments rather than paradigms of computer music. The two experiments mentioned in section 6.7.1 are suggestive of a *knowledge-based* interaction paradigm for the field of generative computer music. On the artistic considerations, they come close to the idea of *controlled or composed improvisation* (Chabot et al., 1986; Dubnov and Assayag, 2005) and *stylistic re-injections* (Assayag et al., 2006a) for computer assisted composition. This promising view of integrating knowledge, interaction and composition should be further pursued in collaboration with artists and composers.

Part IV

When to Expect

Chapter 7

Anticipatory Synchronization

In parts II and III of this thesis we roughly addressed concerns on *What to expect* and *How to expect* and proposed methods to grasp part of the answer to each question. We showed how anticipatory modeling can help reveal the influential aspects of information and provide access to structures over time. We then introduced an interactive and adaptive active learning framework that help *action decisions* in an anticipatory generative setting. In this chapter we turn our attention to the *When* or *timing* aspect of musical expectation. The temporality of music events has in fact been a core of all the models and methods presented so far but the musical time itself has not still been the center of attention up to this chapter.

Since music is a timed signal by nature the issue is vast and covers many different fields in the literature. In this chapter we focus on the problem of live *synchronization* which is at the core of many musical activities such as music performance and production and subject of extensive research in interactive music systems and cognitive studies of musical rhythm. Perhaps a better target for this chapter would be the problem of real-time tempo detection of music signals. Despite our interest, automatic tempo detection is a whole field of research by itself. We therefore reduce the problem to that of *synchronization* in a live performance situation and with a live musician where the computer plays the role of a musician. Moreover, the problem of live synchronization is

crucial for many interactive music systems and paves the way for musical considerations of anticipatory systems.

In this chapter we present a different approach to anticipatory modeling for the problem of live synchronization. We show that our approach provides access to temporal structures and parameters of music performance in a real-time setting and simplifies design and computational complexity of the system thanks to anticipatory modeling. Our approach has important consequences for interactive music systems which we leave for chapter 8 and focus here on the technical design and performance of the synchronization problem. In this chapter, we study the most basic and studied form of synchronization in a musical setting: live audio to score alignment which is also commonly referred to as *score following* or *automatic accompaniment* in the research literature.

7.1 Introduction

Audio to score alignment and automatic accompaniment has a long tradition of research dating back to 1983. The initial and still the main motivation behind score following is the live synchronization between a computer with a music score and a human performer playing the same score with her musical instrument. This can also be extended to a live computer accompaniment with a human performer, for example the computer assuming the performance of the orchestral accompaniment while the human performs the solo part. Another musical motivation is for new music repertoire and mostly for live electronic performances where the computer assumes the performance of a live electronic score that should be synchronous to the human performer in a realtime performance situation. This synchronism is crucial for live electronic music repertoire since they mostly use live score-specific processing on the audio input from the human performer. The overall picture is to bring the computer into the performance as an intelligent and well-trained musician capable of imitating the same reactions and strategies during music performance that human performer(s) would undertake in a similar situation. In recent years, automatic audio to score alignment systems have become

popular for a variety of other applications such as Query-by-Humming (McNab et al., 1996), intelligent audio editors (Dannenberg, 2007) and as a front-end for many music information retrieval systems.

Score alignment begins with a *symbolic* score and an audio input of the same piece of music where during the alignment process many musical parameters pertaining to expressivity can also be extracted. The audio input can be a recording of a performance or can be entering the system as live, leading to *off-line* versus *live* audio to score alignment algorithms. At this point, we would like to draw a line between two different views of *live* versions: A live score follower can be *real-time* but not necessarily *on-line*; thus allowing some latency in detection. For example, the accompaniment system in (Raphael, 2006) is *real-time* but allows latency in reporting results with an acceptable time-resolution for music performance situations.

In this chapter, we present a real-time and on-line audio to score alignment systems that is also capable of decoding the live *tempo* (or musical pace) of the performance; a parameter which is highly responsible for expressive variations in different realizations of the same piece. In its design and in comparison to existing systems, the proposed system encompasses two coupled audio and tempo agents that collaborate and compete in order to achieve synchrony. Collaboration is achieved through the feedback of prediction of one agent into the other. The inference engine features a hidden hybrid markov/semi-markov model that explicitly models events and their temporalities in the music score. The tempo agent features a self-sustained oscillator based on Large and Jones (1999), adopted here in a stochastic framework for audio processing. The novelty of the proposed design are two folds: (1) Direct access to temporal structures in music thanks to the coupled audio/tempo models within a *unique* inference technique, and (2) an anticipatory model handling multimodal interactions between agents in different time-scales and featuring an adaptive learning framework, liberating the system from any need for off-line training. The system gets as input a score representation and live audio input. The output of the system are the event indexes and real-time tempo with no need for external training. In practice, the presented system is capable of suc-

cessfully decoding polyphonic music signals and has been featured in several concert performances world-wide.

Our consideration of anticipatory models for the problem of live synchronization requires an understanding of the state-of-the-arts design of score following techniques, cognitive and computational models of time as well as important contemplations on time models as input from the music community. We therefore begin this chapter by an overview of the literature within these topics in section 7.2.We introduce the general architecture of our proposal in section 7.3 and provide the general inference framework thereafter in section 7.4. Sections 7.5, 7.6, and 7.7 detail the two agents and their observation models. We evaluate the system's performance in various situations in section 7.8 followed by discussions and conclusions on the proposed approach.

7.2 Background

7.2.1 Score Following Research

Score following research was first introduced independently by Dannenberg (1984) and Vercoe (1984). Due to computational limitations at the time, both systems relied on symbolic MIDI inputs rather than raw audio. The problem would then be reduced to *string matching* between the symbolic input stream and the score sequence in real-time. The problem becomes more complicated when expressive variations of the performer, either temporal or event-specific, comes into play. Underlying systems must have the tolerance to deal with these variations as well as human or eventually observation errors in order to keep exact synchrony with the human performer. All these factors made the problem appealing even on the symbolic level.

In the early 90s with the advent of faster computers, direct use of audio input instead of symbolic data became possible, allowing musicians to use their original instruments. In this new framework, the symbolic level of the score is not directly observable anymore and is practically *hidden* from the system. Early attempts at this

period focused on monophonic pitch detectors on the front-end, providing pitch information to the matching algorithm under consideration and thus, doubling the problem of tolerance with the introduction of pitch detection uncertainly which is an interesting problem by itself. An example of this is the system introduced by Puckette and Lippe (1992) and widely used in computer music community. By mid-90s and parallel to developments in the speech recognition community, stochastic approaches were introduced by Grubb and Dannenberg (1997) and Raphael (1999b), with the latter based on Hidden Markov Models and statistical observations from live audio input. Raphael's approach was further developed by himself and other authors leading to many variants (for example in (Orio and Déchelle, 2001; Cont, 2006)). In a more recent development, Raphael has introduced a polyphonic alignment system where a tempo model based on dynamic Bayesian networks computes the tempo of the musician based on the posterior data from the previously introduced Hidden Markov Model (Raphael, 2006).

It is interesting to note that Vercoe's initial MIDI *Synthetic Performer* had explicit interaction with the deducted tempo of the live performer (Vercoe, 1984)[1]. With the move to audio systems using pitch detection algorithms *tempo* was temporarily forgotten focusing on string matching techniques and back again with Grubb and Dannenberg (1997) where the pitch observations used in the probabilistic model can influence the running tempo by comparing the elapsed time and the idealized score tempo. Perhaps the most elaborate and explicit time model that has been used belongs to Raphael in (Raphael, 2006). Raphael's design has two (cascaded) stages for decoding of score position and tempo. The first, comprising Hidden Markov Models deducted from the score responsible for decoding the position in the score (called the *listener*) and the second, an elaborate Bayesian network which takes this information to deduct the smooth tempo during the performance. Notably, Raphael uses this tempo in interaction with his accompaniment system to adapt the time-duration of the accompanying section using phase-vocoding techniques.

In this chapter, we propose an anticipatory model for the problem of score fol-

[1] See the historical video at http://www.youtube.com/watch?v=vOYky8MmrEU

lowing. In our approach, tempo and audio decoding are not separate problems but are rather tied and coupled together within an anticipatory framework. We will show later that this approach reduces design complexity, computation and increases robustness and control of the system. Besides these technical issues, one of the main goals of our design is to make use of such interactive systems for accessing, controlling and eventually scripting electronic events directly using our model. For this reason, a thorough understanding of the concept of *time* in music and its modeling consequences is crucial.

7.2.2 Cognitive Foundations of Musical Time

The perception of musical metric structure over time is not merely an analysis of rhythmic content, rather it shapes an *active listening* strategy in which the listener's expectations about future events can play a role as important as the musical events themselves. Therefore, any model for timing synchronisation of musical events should consider the hypothesis that the temporal structure of listeners' expectations is a dynamic structure. A primary function of such structure is *attentional* which allows *anticipation* of future events, enabling perceptual targeting, and coordination of action with musical events.

The above considerations led Large and Jones (1999) to propose a model of meter perception where they assume a small set of internal oscillations operating at periods that are approximates to those of hierarchical metrical level. Each oscillator used in the model is *self-sustained* in the sense that once activated, it can persist even after simulation ceases or changes in significant ways. Meter perception is then achieved by different oscillators competing for activation through mutual inhibition. Their model also has the ability to entrain to incoming rhythmic signals. This model has been tested and verified largely in different experiments with human subjects (Large and Jones, 1999). The tempo model of our framework, introduced in section 7.5, is an adoption of the internal oscillator of Large and Jones in a stochastic and continuous audio framework.

On another level, research on brain organization for music processing show

strong evidence for dissociation of pitch and temporal processing of music (Peretz and Zatorre, 2005). This evidence suggests that these two dimensions involve the operation of separable neural subsystems, and thus can be treated in parallel in a computational framework.

7.2.3 Compositional Foundations of Time

In most computational music systems that deal with models of time, modeling concepts are inherited from long studied models available from speech recognition or biological sequence literatures. Despite their success, these approaches bring in departing hypothesis and approximations that are not that of musical structures; thus, leading to imperfect generalization of the underlying design. We believe that for the specific problem of music information processing such as ours, the best source of inspiration is the music literature and contemplations thereof on the problem which is the result of centuries of human intellect. There is no single reference regarding *time* in the arts and music. Every composer or artist has dealt with the problem one way or another. We will not go through all aspects of time and neither do we claim providing a universal model of time. Here, we briefly introduce the main sources of inspirations for our modeling, mostly gathered from music theoreticians and composers' contemplations on the "writing of time" (Ircam, 2006). These inputs will be later gathered and furthered synthesized in our considerations for writing of time and interaction of chapter 8.

An ideal score following application should inherently possess the grammar used in the score of the music under consideration. Looking at a traditional western notated music score, the simplest way to imagine this is a set of discrete sequential note and silence events that occupy a certain amount of relative duration in time. The relative notion of musical time is one of the main sources of musical expressivity which is usually guided by a *tempo* often represented in beats-per-minute (BPM). Remaining within the realms of western classical music notation, we provide two important insights on temporality of musical events with direct consequences on our model.

Temporal vs. Atemporal

An *Atemporal* (or *out-of-time*) event corresponds to an object that possess its own internal temporal structure independent of the overall temporal structures of the piece of music. The two structures are usually considered independent in music theory. To conform this distinction with our probabilistic design, we define an *Atemporal* object or event as one that posseses a physical space in the score but does not contribute to the physical musical time of the score. Typical examples of atemporal objects are grace notes, internal notes of a trill, or typical ornaments in a baroque style interpretation in western classical notation. While both have physical presence, the individual events do not contribute to the notion of tempo but their relative temporal appearance in the case of the grace note, or their overall *in-time* structure in the case of a trill contribute to the notion of tempo. Other examples of atemporal objects are events with fermatas or free-improvisation boxes seen in various contemporary music repertoire. The distinction between temporal and atemporal events in musical composition is usually attributed to the composer Iannis Xenakis (Xenakis, 1971).

Striated-time vs. Smooth-time

Striated time is one that is based on recurring temporal regularities while smooth time is a continuous notion of time as a flow of information. The pulsed-time used in most western classical music notation is a regulated striated time-flow that uses an internal musical clock usually driven by a tempo parameter in beats-per-minute. In our terminology, we distinguish between a striated time-scale where the notion of time is driven relative to a constantly evolving tempo, and smooth time-scale where the information on the microscopic level consists of individual atemporal elements or is defined relative to a clock-time. Typical example of a smooth-time event in western traditional notation are free *glissandis*. It is important to mention that most *available* classical and popular music pertain to striated time. The distinction of striated and smooth time in music scores, despite its evidence, is attributed to the composer Boulez in describing

foundations of musical time (Boulez, 1964).

Stockhausen (1957) has shown that in electronic music practices, striated time and smooth time are snapshots of the same process but with different frequencies underlying the compositional foundations of time. In a practical and computational view, we can still distinguish the two by considering one as a discrete and other as a continuous process over time.

7.2.4 Probabilistic Models of Time

As of the extreme temporal nature of music, the ability to represent and decode temporal events constitutes the core of any score following system. In general, a live synchronization system evaluates live audio inputs versus timed-models of symbolic score in its memory. Since such systems are promised to operate in uncertain situations probabilistic models have become a trend in modeling since late 1990s. Within this framework, the probabilistic model's goal is to decode temporal dynamics of an outside process. Therefore the performance of such model highly depends on its ability to represent such dynamics within its internal topology. In these problems, any state of a given process occupies some duration that can be deterministic or not. In such tasks, we are interested in a probabilistic model of the macro-state duration and sojourn time. In a musical context, a macro-state can refer to a musical event (note, chord, silence, trills etc.) given an expected duration. A common way to model time series data in the literature is by the use of *state-space models*. A state-space model of a sequence is a time-indexed sequence of graphs (nodes and edges) where each node refers to a state of the system over time. Therefore each state has an explicit time-occupancy that can be used to probabilistically model sojourn time and duration of the events under consideration. In this section, we limit our study to two wide classes of state-space models and their duration models that cover most existing approaches: Markov and Semi-Markov processes.

Markov Time Occupancy

In a parametric Markov time model, the expected duration of a marco-state j (events such as notes, chords etc. that occupy time) is modeled through a set of Markov chains with random variables attached to transition probabilities that parameterize an occupancy distribution function describing the duration $d_j(u)$ where u is the number of times spent in the marco-state markov chains. Figure 7.1 shows a parametric macro-state Markov chain topology commonly used for duration modeling. This way, the macro-state consists of r Markov states and two free parameters p and q corresponding respectively to the exit probability and to the next-state transition probability. The macro-state occupancy distribution associated to this general topology is the compound distribution:

$$\begin{aligned} P(U=u) &= \sum_{n=1}^{r-1} \binom{u-1}{r-1} (1-p-q)^{u-n} q^{n-1} p \\ &\quad + \binom{u-1}{r-1} (1-p-q)^{u-r} q^{r-1} (p+q) \end{aligned}$$

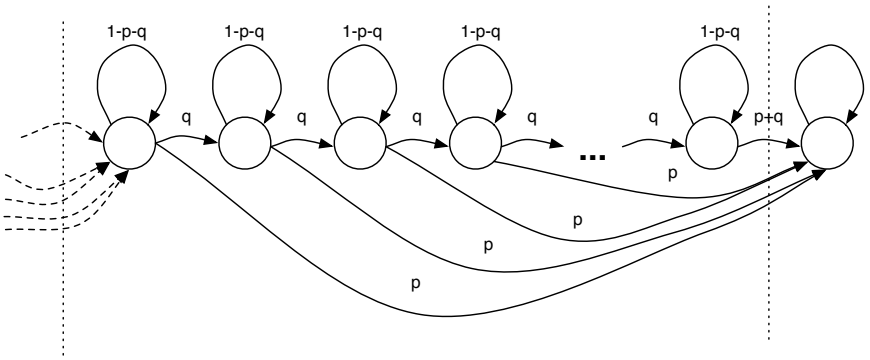


Figure 7.1: Parametric Markov Topology

If $p = 0$, this macro-state occupancy distribution is the negative binomial distribution:

$$P(U=u) = \binom{u-1}{r-1} q^r (1-q)^{u-r}$$

which corresponds to a series of r states with no jumps to the exit state with the shortcoming that the minimum time spent in the macro-state is r. This simplified version has

been widely explored in various score following systems where the two parameters r and q are derived by optimization over macro-state's time duration provided by the music score. For example, Raphael in (Raphael, 1999a, 2004) uses the method of moments to solve for optimized values whereas Orio, Lemouton, Schwarz, and Schnell (2003) optimize the parameters on the mean and variance of the duration by assuming a fixed (usually 50%) variation around the score duration.

As a reminder, a Markov process has the underlying assumption that the evolution of states in a process are *memoryless* or conditional on the *preset* where *future* and *past* processes are considered *independent*. As we saw in previous chapters, such an assumption is not necessarily true for music signals however researchers have mostly adopted Markov processes with approximations.

In the above scheme, the time occupancy of Markov chains are modeled by discrete parameters (p, q, and r). It is also possible to parameterize these time occupancies by explicit distributions or beliefs over time-models as a function $d_j(u)$ for each mico-state. In general, there is no gain in complexity neither design by using non-parametric forms such as this within a Markovian framework. However the idea of using non-parametric time occupancies would become attractive when considered over the macro-states instead of micro-states, leading to semi-Markov topologies.

Semi-Markov Time Occupancy

In a Semi-Markov model, a macro-state can be modelled by a *single* state (instead of a fixed number of mico-states) and using an explicit time occupancy or sojourn probability distribution $d_j(u)$ for each state j and occupancy u. Assuming that S_i is the discrete random variable denoting the marco-states at time i from a state space $\mathcal{S} \subset \mathbb{N}$, and T_m the time spent at each state m, then $S_t = m$ whenever

$$\sum_{k=1}^{m} T_k \leq t < \sum_{k=1}^{m+1} T_k.$$

Or in simple words, we are at state m at time t when the duration models for all states up to m and $m+1$ comply with this timing. In this configuration, the overall process is *not* a

Markov process *within* marco-states but it is a Markov process in-between marco-states. Hence the name *Semi-Markov*.

The explicit occupancy distribution can then be defined as follows:

$$d_j(u) = P(S_{t+u+1} \neq j, S_{t+u-v} = j, v = 0, \ldots, u-2 | S_{t+1} = j, S_t \neq j) \qquad (7.1)$$

where $u = 1, \ldots, M_j$ with M_j the upper bound to the time spent in the macro-state. Associated with the sojourn time distribution, we define a survivor function of the sojourn time as $D_j(u) = \sum_{v \geq u} d_j(v)$.

Semi-Markov models were first introduced by Ferguson (1980) for speech recognition and gained attention because of their intuitive access to temporal structures of models. Semi-Markov topologies are usually much sparser in computations and more controllable than their Markovian counterparts. Moreover, they provide explicit access to time models expressed as occupancy distributions. However, their disadvantage is in finding suitable forms for $d_j(u)$ and integrating them within the probabilistic framework where sojourn probabilities are usually not stationary functions over time themselves.

7.3 General Framework

The general problem of score following is devised as follows: The system disposes a representation of the music score in advance, that is fed into the system off-line. The goal of our system is to map the incoming real-time audio stream onto this representation and decode the current *score position*, *real-time tempo* and undertake *score actions* (if any). The music score is represented through a probabilistic graphical model constructed directly from a symbolic music score inspired by observation in section 7.2.3. Given the probabilistic score representation, the real-time system extracts instantaneous beliefs or probabilities of the audio stream with regards to states of this probability model. This is achieved through *observers* of the system. The goal of the system is then to integrate this instantaneous belief with *past* and *future* beliefs in order to decode the position and tempo in real-time. Figure 7.2 shows a general diagram of

our system.

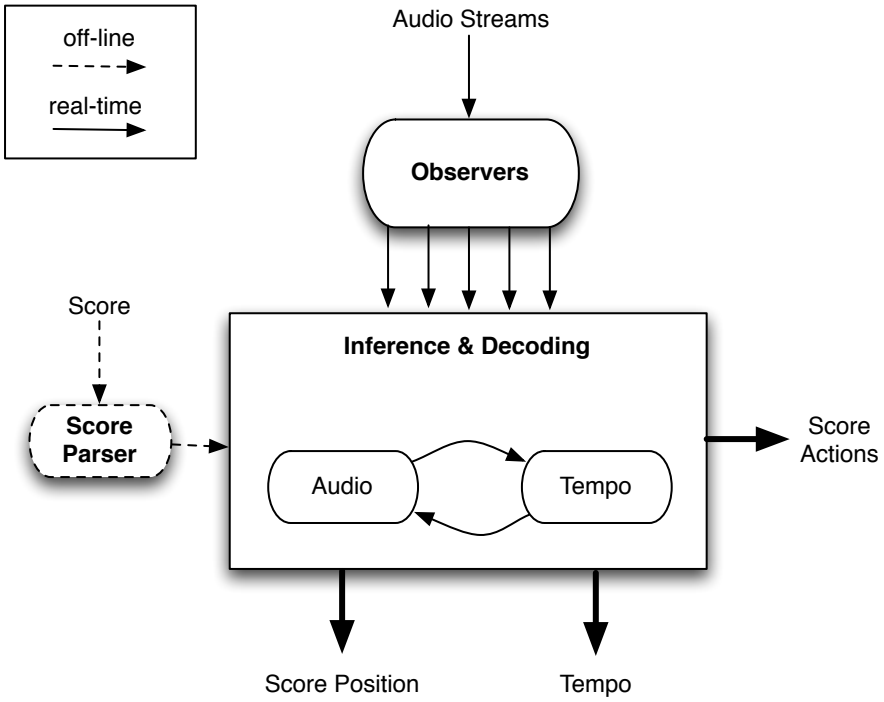


Figure 7.2: General System Diagram

This model has the underlying hypothesis that the audio stream can be totally generated by the underlying state-space score model. Hence the problem of score following is the inverse of this hypothesis, or to find the most likely state-sequence associated with observed real-time audio sequence. Due to the nature of this inverse problem, the underlying state-sequence that generates the audio is not directly observable by the system and thus *hidden*. This process of finding the most likely state-sequence in a hidden process up to the present is referred to as the *inference* problem.

7.3.1 Anticipatory Multimodal Inference

The inference framework proposed for this architecture is multimodal, taking care of both continuous audio and tempo parameters and is also anticipatory. It is based on coupled audio and tempo agents. The two audio and tempo agents collaborate at

all times to map the real-time audio input to the most likely state sequence in the score model. The tempo agent computes on the *event* time-scale and is based on a cognitive model of musical metric structure introduced in section 7.2.2 and provides continuous tempo predictions based on live audio input and the given music score. The inference scheme computes on the *continuous* audio time-scale and assigns probabilistic values to relevant states in the score state-space by combining tempo predictions and continuous audio observations. The proposed model is a *sensorial anticipatory system* (see definition in section 3.3.3, page 47) where the state likelihoods are influenced dynamically by the predicted tempo, and in return, the tempo agent is directly affected by the decisions obtained instantly by the system.

7.3.2 Hybrid Models of Time

The probabilistic (and generative) state-space model of the score describes event types as well as time models of events in the score used during decoding and inference. For the state-space model of our framework, we propose to use the best of both probabilistic time models presented previously in section 7.2.4 and motivated by observations on compositional foundations of time in section 7.2.3. Within this framework, we would like to take advantage of explicit time-models of Semi-Markov chains for *Temporal* and *Striated-time* events, and employ parametric Markov models for *Atemporal* and *Smooth-time* elements in the score. These considerations lead to a probabilistic model based on *Hybrid Markov/Semi-Markov Chains* as first proposed by Guédon (2005). Within our inference framework, this model is *hidden* as noted above. Below, we provide the definitions and basics of hidden hybrid Markov/Semi-Markov chains which constitutes the core of our system's representation.

To formalize the problem, we assume that the audio stream through time τ or x_0^τ (as short for $x_0, \ldots, x_\tau$) is a stochastic process represented by the random variable $\{X_t\}$, which is generated by a sequence of states s_0^τ through the random variable $\{S_t\}$ corresponding to (hidden) states in a hybrid markov/semi-Markov chain constructed from

the score. A discrete hidden hybrid Markov/semi-Markov chain can then be viewed as a pair of stochastic processes (S_t, X_t) where the discrete output $\{X_t\}$ is related to the state process $\{S_t\}$ by a probabilistic function or mapping denoted by f where $X_t = f(S_t)$. Since this mapping f is such that $f(s_j) = f(s_k)$ may be satisfied for different j and k, or in other words, a given output may be observed in different states, the state process S_t is not observable directly but only indirectly through the output process X_t. Beyond this point, we use $P(S_t = j)$ as a short for $P(S_t = s_j)$ denoting the probability that state j is emitted at time t.

Let S_t be a J-state hybrid Markov/semi-Markov chain. It can be then defined by:

- Initial probabilities $\pi_j = P(S_0 = j)$ with $\sum_j \pi_j = 1$.
- Transition Probabilities
  - semi-Markovian state j:

    $$\forall j, k \in \mathbb{N},\ k \neq j: \qquad p_{jk} = P(S_{t+1} = k | S_{t+1} \neq j, S_t = j)$$

    where $\sum_{k \neq j} p_{jk} = 1$ and $p_{jj} = 0$.
  - Markovian state j:

    $$\tilde{p_{jk}} = P(S_{t+1} = k | S_t = j)$$

    with $\sum_k \tilde{p_{jk}} = 1$.
- An *explicit* occupancy (or sojourn time) distribution attached to each semi-Markovian state as in equation 7.1:

  $$d_j(u) = P(S_{t+u+1} \neq j, S_{t+u-v} = j, v = 0, \ldots, u-2 | S_{t+1} = j, S_t \neq j) \quad (7.2)$$

  Hence, we assume that the state occupancy distributions are concentrated on finite sets of time points.

- An *implicit* sojourn distribution attached to each Markovian state j where

$$P(S_{t+1} = k | S_{t+1} \neq j, S_t = j) = \frac{\tilde{p_{jk}}}{1 - \tilde{p_{jk}}}$$

defines an implicit state occupancy distribution as the geometric distribution with parameter $1 - \tilde{p_{jk}}$:

$$d_j(u) = (1 - \tilde{p_{jk}})\tilde{p_{jk}}^{u-1} \tag{7.3}$$

The output (audio) process X_t is related to the hybrid Markov/semi-Markov chain S_t by the observation or emission probabilities

$$b_j(y) = P(X_t = y | S_t = j) \quad \text{where} \quad \sum_y b_j(y) = 1.$$

This definition of the observation probabilities expresses the assumption that the output process at time t depends only on the underlying hybrid Markov / semi-Markov chain at time t.

The original formulations of the hybrid network defined above in (Guédon, 2005) are not aimed for real-time decoding, neither anticipatory, and nor multimodal processing. In the following sections, we extend this framework to our multimodal anticipatory framework.

7.4 Inference Formulation

The solution to the inference problem determines the most-likely state-sequence S_0^τ that would generate X_0^τ and in return the score position and real-time decoded tempi. In a non-realtime context, an exact inference can be obtained using a Viterbi type algorithm (Murphy, 2002) that for each time t uses both beliefs from time 0 through τ (referred to as *forward* or α variable) and future knowledge from present (τ) to a terminal state at time T (referred to as *backward* or β variable). In a score following system that necessitates on-the-fly synchronization of audio with the music score, using the β or backward variable of the Viterbi algorithm is either impossible or would introduce considerable delays in the system. In the proposed system, we hope to compensate for this

absence of future beliefs through our anticipatory model of audio/tempo coupled agents and an adaptive *forward* variable calculation procedure. Here, we formulate a dynamic programming approach for an adaptive α calculation for a hidden hybrid Markov/semi-Markov process.

For a semi-Markovian state j, the Viterbi recursion of the forward variable is provided by the following dynamic programming formulation (see Appendix B.1 for proof and derivation):

$$\begin{aligned} \alpha_j(t) &= \max_{s_0,\ldots,s_{t-1}} P(S_{t+1} \neq j, S_t = j, S_0^{t-1} = s_0^{t-1}, X_0^t = x_0^t) \\ &= b_j(x_t) \times \max \left[\max_{1 \leq u \leq t} \left(\left\{ \prod_{v=1}^{u-1} b_j(x_{t-v}) \right\} d_j(u) \max_{i \neq j} (p_{ij} \alpha_i(t-u)) \right) \right] \end{aligned} \tag{7.4}$$

For a Markovian state j, the same objective amounts to the regular dynamic programming forward calculation for Hidden Markov Models (Rabiner, 1989):

$$\begin{aligned} \tilde{\alpha}_j(t) &= \max_{s_0,\ldots,s_{t-1}} P(S_t = j, S_0^{t-1} = s_0^{t-1}, X_0^t = x_0^t) \\ &= b_j(x_t) \max_i (\tilde{p_{ij}} \tilde{\alpha}_i(t-1)) \end{aligned} \tag{7.5}$$

Within this formulation, the probability of the observed sequence $x_0^{\tau-1}$ jointly with the most probable state sequence is $\underset{j}{\operatorname{argmax}} [\alpha_j(\tau - 1)]$.

In order to compute equations 7.5 and 7.4 in real-time, we need the following parameters:

- State types and topologies determine the type of decoding and transition probabilities p_{ij}. This probabilistic topology is constructed directly from the music score and is discussed in section 7.6.

- Observations $b_j(x_t)$ are calculated in from real-time audio (x_t) and are discussed in details in section 7.7.

- The sojourn distribution $d_j(u)$ that also decodes and model musical *tempo* in real-time, and the upper bound u of the product in eq. 7.4, which are discussed in section 7.5.

- A prior belief (or belief at time zero) as $\alpha_j(0)$, which is usually assigned to the corresponding starting point on the score during a performance.

During real-time decoding, the spatial complexity of the inference algorithm for this mixture of macro-state model has an upperbound of $O(\tau \sum_j M_j)$. Since we are dealing with musical performances and a left-right overall structure in time, this complexity usually amounts to the use of short homogeneous zones in the topology during filtering as a window with a fixed span of states positioned around the latest decoded event.

7.5 Stochastic model of time in music performance

In a western notated music score, time is usually written by values relative to a musical clock referred to as tempo. Tempo is indicated by number of musical beats that are expected to occur in minute (or *BPM*) and accordingly the temporality of events in the score are indicated by the number of beats that they span in time which can be fractions of a unit beat. Depicting the beat duration of an event k in the score by ℓ_k, the absolute score location in clock-time T_k given a (global) tempo Ψ can be directly obtained by the following recursive relationship:

$$T_k = T_{k-1} + \Psi \times \ell_k \tag{7.6}$$

However, in reality and even if an entire piece is depicted with a fixed tempo Ψ, the tempo variable undergoes various dynamics and changes, responsible mostly for the expressivity of a musical performance. Therefore our goal here is to infer the dynamics of the tempo variable through time.

7.5.1 Attentional Model of Tempo

The model we propose here for decoding of the continuous tempo random variable is highly inspired by Large and Jones (1999). Internal tempo is represented through

a random variable Ψ_k revealing how fast the music is flowing with regards to the physical time. After Large, we model the behavior of such random variable as an internal oscillator entraining to the musician's performance. Such internal oscillation can be represented and modeled easily using sine circle maps. These models have been well-studied in the literature and can be considered as non-linear models of oscillations that entrain to a periodic signal and using discrete-time formalism. In this framework, phase of the sine circle map would be an abstraction of time and the time to pass one circular period or the local tempo. Using this framework, we represent the tempo random variable as ψ_k in $seconds/beat$ and note onset positions as phase values ϕ_k on the sine circle. This way, given a local tempo ψ_i, the score onset time t_n can be represented as $\phi_n = \frac{t_n}{\psi_i} + 2k\pi$ where k is the number of tempo cycles to reach t_n. For our model, a phase advance would be the portion of the oscillator's period corresponding to (temporal) event *Inter-Onset-Intervals* (IOI). Saying this, if the tempo is assumed fix (ψ_k) throughout a piece, then

$$\phi_{n+1} = \phi_n + \frac{t_{n+1} - t_n}{\psi_k} \qquad \mod {}^{+0.5}_{-0.5} \tag{7.7}$$

would indicate relative phase position of all events in the music score.

In order to compensate for temporal fluctuations during live music performance, we would need a function of ϕ that would *correct* the phase during live synchronization and at the same time model the *attentional* effect discussed previously, thus enabling perceptual targeting, and coordination of action with musical event. The attentional pulse can be modeled using a periodic probability density function, the *von Mises distribution* which is the circle map version of the Gaussian distribution, as depicted below, where I_0 is a modified Bessel function of first kind and order zero.

$$f(\phi, \phi_\mu, \kappa) = \frac{1}{I_0} e^{\kappa \cos(2\pi(\phi - \phi_\mu))} \tag{7.8}$$

where ϕ_μ and κ are mean and variance equivalents of the von Mises distribution. Figure 7.3 demonstrates a realization of this function on the sine-circle map.

Large and Jones (1999) show that the corresponding phase coupling function (tempo correction factor) for this attentional pulse is the derivative of a unit amplitude

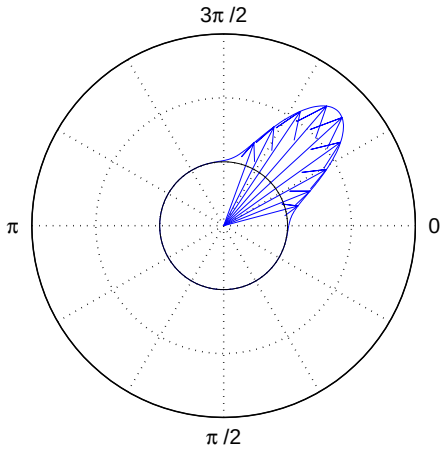


Figure 7.3: Sample Von Mises distribution on a clock-wise sine-circle map, with mean $7\pi/4$ or $-\pi/4$ and variance $\kappa = 15$.

version of the attentional function, depicted in equation 7.9. Figure 7.4 shows this function for different values of κ and $\phi_\mu = 0$.

$$F(\phi, \phi_\mu, \kappa) = \frac{1}{2\pi \exp \kappa} e^{\kappa \cos(2\pi(\phi - \phi_\mu))} \sin 2\pi(\phi - \phi_\mu) \tag{7.9}$$

With the above introduction equation 7.7 can be rewritten as,

$$\phi_{n+1} = \phi_n + \frac{t_{n+1} - t_n}{\psi_k} + \eta_\phi F(\phi_n, \phi_{\mu_n}, \kappa) \qquad \text{mod}_{-0.5}^{+0.5} \tag{7.10}$$

where η_ϕ is the coupling strength of the *phase coupling* equation and ϕ_{μ_n} are the *expected* phase position of the n^{th} event in the score according to previous justifications.

Phase coupling alone is not sufficient to model phase synchrony in the presence of the complex temporal fluctuation. To maintain synchrony, the period (or tempo) must also adapt in response to changes in sequence rate as follows:

$$\psi_{n+1} = \psi_n(1 + \eta_s F(\phi_n, \kappa)) \tag{7.11}$$

Equations 7.10 and 7.11 can recursively update tempo and expected onset positions upon onset arrivals of *temporal* events from the inference engine. However, note

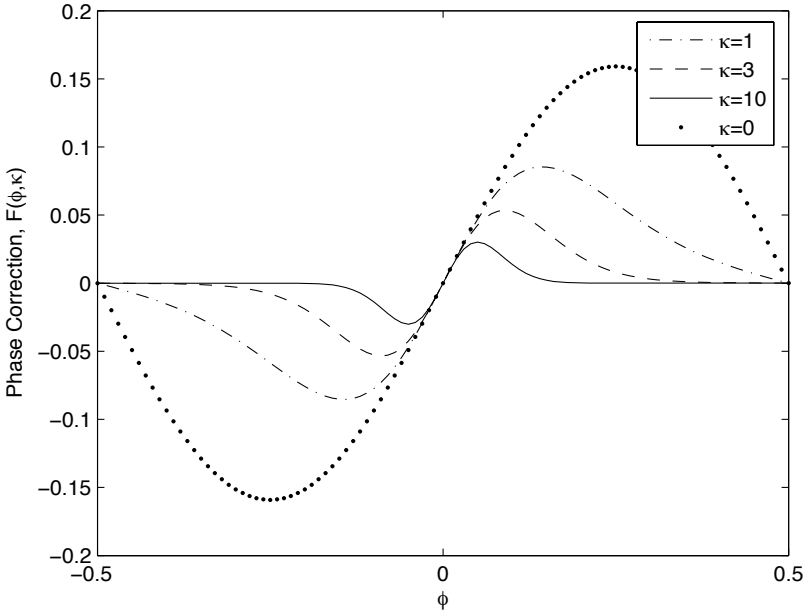


Figure 7.4: Phase Correction function in Equation 7.9

that the phase-time regions where the phase adjustment is most efficient in figure 7.4 are identical to the region around the mean of the attentional distribution (eq. 7.8) spanned by its variance κ. Smaller values of κ spread the correction all over the phase domain, amounting to a wider *variance* in the attentional function meaning that expectancy is dispersed throughout the oscillator. For this reason, the parameter κ is usually referred to as *attentional focus*. This observation suggests that the values of κ should be adjusted at each update to obtain the best possible performance. To this end, before each tempo update, we solve for $\hat{\kappa}$ using a maximum-likelihood formulation on the dispersion about the mean of a sampled population of previously occurred ϕ_ns. This dispersion is given by the following equation on the circular map:

$$r = \frac{1}{n}\sum_{i=1}^{n} \cos 2\pi(\phi_i - \phi_{\mu_i}) \tag{7.12}$$

which can be easily calculated recursively in real-time. Having this, the solution for $\hat{\kappa}$

is shown to be (See Mardia and Jupp, 2000, Section 10.3.1):

$$\hat{\kappa} = A_2^{-1}(r) \qquad \text{where} \qquad A_p(\lambda) = \frac{I_{p/2}(\lambda)}{I_{p/2-1}(\lambda)} \tag{7.13}$$

where $I_\nu(\lambda)$ is the modified Bessel function of the first kind and order ν. The solution to $\hat{\kappa}$ in eq. 7.13 is obtained by a table look-up of $A_2(\lambda)$ and using accumulated dispersions from eq. 7.12 in real-time.

7.5.2 Tempo Agent and Decoding

With the above formulations, we can propose a unified model of temporal fluctuations in a music performance. The tempo agent demonstrated here works in parallel with the audio decoding and observation. As mentioned earlier, the final outcome of the system is a result of collaboration and competition between the two. While the audio agent works on continuous-time level, the tempo agent works on event-time level or in other words, on inter-onset time scale.

The tempo decoding algorithm we present here is a recursive algorithm based on above observations and resembles an *Extended Kalman Filtering* approach (Maybeck, 1979). The Kalman filter estimates a process by using a form of feedback control: the filter estimates the process state at some time and then obtains feedback in the form of environmental measurements. The general Kalman filter algorithm then fall within two steps: *Prediction* and *Correction*. The prediction equations are responsible for projecting forward (in time) the current state and error estimates to obtain the *a priori* estimates for the next time step. The correction equations are responsible for the feedback or incorporating the new measurement into the a priori estimate to obtain an improved a posteriori estimate. While general Kalman filters use linear estimators, Extended Kalman Filters (EKF) assume non-linear estimators (as is our case with the Mises-Von correction factors).

Algorithm 7.1 shows the two *correction* and *prediction* steps for the tempo agent. The correction procedures make use of the *true* arrival time of event n or t_n and within

two steps: In the first, we update the true variance κ needed during updates by accumulating the circular dispersion as in eq. 7.14 which is a real-time approximation of eq. 7.12 by using an accumulation factor η_s which is set to a fixed value. Having corrected the κ value using table lookup, the algorithm then updates the relative phase position of event n, by using previous estimations, current measurements and the score phase position. The prediction step then simply uses the newly corrected phase position of event n or ϕ_n, the score phase position $\hat{\phi}_n$ and the correction factors to obtain the new tempo prediction for event $n+1$. This algorithm is called recursively and upon each arrival of a measurement from the audio agent.

Algorithm 7.1 Real-time Tempo decoding algorithm

Require: Upon decoding of event n at time t_n by the audio agent (measurement), given score IOI phase positions $\hat{\phi}_n$, initial or previously decoded tempo ψ_n

1: *Correction (1):* Update κ (variance)

$$r = r - \eta_s \left[r - \cos\left(2\pi \left(\frac{t_n - t_{n-1}}{\psi_k} - \hat{\phi}_n \right) \right) \right] \tag{7.14}$$

$$\kappa = A_2^{-1}(r) \quad \text{(Table lookup)}$$

2: *Correction (2):* Update ϕ_n

$$\phi_n = \phi_{n-1} + \frac{t_n - t_{n-1}}{\psi_{n-1}} + \eta_\phi F(\phi_{n-1}, \hat{\phi}_{n-1}, \kappa) \qquad \text{mod}_{-0.5}^{+0.5}$$

3: *Prediction:*

$$\psi_{n+1} = \psi_n \left[1 + \eta_s F(\phi_n, \hat{\phi}_n, \kappa) \right]$$

4: **return** ψ_{n+1}

As to the nature of the proposed model, the newly obtained tempo at each step ψ_n is a *predictive* tempo flow that can be used to anticipate future note locations in time. We use this feature of our model in the next section to obtain the survival function needed for inference module.

7.5.3 Survival Distribution Model

In section 7.4 we introduced the global inference method used for a Hybrid Hidden Markov/Semi-Markov model. We also introduced a state-space topology with explicit time-models with the use of explicit sojourn occupancy distributions $d_j(u)$ which are required to calculate the inference formulation in section 7.4. Here we introduce a stochastic time process that can derive the survival function needed for inference.

We consider the process underlying the arrival rate of events over a time-period of musical performance as an Spatial Poisson process with distribution $P(N(t) = k)$ where $N(t)$ is the number of events that have occurred up to time t characterized as:

$$P[(N(t+\tau) - N(t)) = k] = \frac{e^{-\lambda(t)\tau}(\lambda(x,t)\tau)^k}{k!}$$

where $\lambda(x,t)$ is the expected number of events or arrivals that occur at score location x and time t. What we are now interested is a process that can underly the arrival time of the k^{th} event or T_k and from which we can derive the *survival function* needed for eq. 7.4 and defined in eq. 7.2. Depicting the real-time as t and t_{n-1} as the previously decoded event, the survival distribution is

$$\begin{aligned} d_j(t - t_{n-1}) &= P(T_n > t | T_{n-1} = t_{n-1}, t_{n-1} < t) \\ &= P[(N(t_n) - N(t_{n-1})) = 0] \\ &= \exp\left[-\lambda(n,t)(t - t_{n-1})\right] \end{aligned} \tag{7.15}$$

Now that we have a direct formulation of the survival distribution, it only remains to specify $\lambda(n,t)$. Note that the expected value of this distribution is $1/\lambda$ which, for event n, is equivalent to its expected duration according to the score and the latest tempo decoding as demonstrated in section 7.5.1. Therefore,

$$\lambda(n,t) = \frac{1}{\psi_{n-1}\ell_n} \tag{7.16}$$

noting that s_n or the (real-time) decoded local tempo is a function of both time t and score location n. Combining both equations 7.16 and 7.15 would provide us for the

survival distribution to be used along with eq. 7.4 during inference:

$$d_j(t - t_{n-1}) = \exp\left[-\frac{t - t_{n-1}}{\psi_{n-1}\ell_n}\right] \tag{7.17}$$

Note that the upper limit of the product u in eq. 7.4 would also be equal to the expected duration of the corresponding state or $\psi_j \ell_j$.

Overall, the tempo agent described in this section provides the sojourn function $d_j(u)$ as well as upper limits of eq. 7.4 adaptively during a real-time performance, as well as decoding a continuous tempo parameter pertaining to the tempo of the performance under consideration.

7.6 Music Score Model

Using the inference formulation above, each audio observation is mapped to a state-space representation of the music score where each event in the score is modeled as one or more states s_j with appropriate characteristics. The state-space in question would be a hidden hybrid markov/semi-markov model constructed out of a given music score during parsing. The type of the state (Markov or Semi-Markov), its topology and associated symbols are decided based on the musical construct taken from the music score. In this section we describe a set of topologies that were designed to address most temporal structures in western music notation outlined in section 7.2.3. As a convention, in the figures that follow, Markov states are demonstrated by regular nodes whereas Semi-Markov states by double-lines nodes.

7.6.1 Basic Events

A single event can be a single pitch, chord (or set of pitches occurring all at once), silence or grace note. These events can be either temporal or atemporal (see section 7.2.3). A timed event is mapped to semi-Markov state whereas an atemporal event (such as a grace note) is mapped to a Markov state. A semi-Markov state s_i is

described by a set $\{i, \ell_i, obs_i\}$ where i is the event number or discrete location since the beginning of the score, and ℓ_i is its duration expressed as the number beats relative to the initial score tempo, and obs_i are observations or pitch numbers in this case. Figure 7.5 shows a sample graphical score and its equivalent Markov topology after parsing. If the duration associated with a single event is set to 0.0, it is a sign that the associated event is *atemporal* therefore Markovian and are described by $\{i, obs_i\}$. In the example of figure 7.5, grace notes are encoded as Markovian states (circles) where timed pitches are parsed into semi-Markovian (dashed circle) states. In this example, pitches are encoded using MIDI pitch numbers and a left-right Markov topology is created that is in one-to-one correspondence with the score. Note that in this example, a *dummy* atemporal silence is created in the middle. The parser automatically puts dummy silences between events where appropriate to better model the incoming audio.

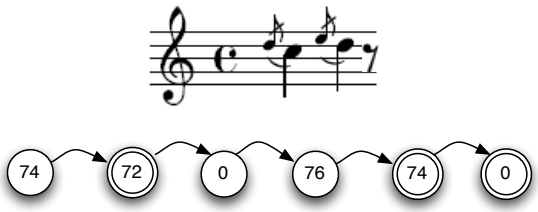


Figure 7.5: Sample state-space topology for basic events

7.6.2 Special timed events

Many score models for alignment purposes stop at this point. However, music notation span a large vocabulary where sometimes events are spread differently over time and interpretations are either varied from performance to performance or are free at large. This is the case with almost every written music piece that contain events such as *trills*, *glissando*s etc. While studying some of these common irregularities we figured out that the particularity of such events are in how they are spreaded over time and how their observations are handled during real-time decoding. We came out with two simple state topologies that address several classical cases as well as more general ones

which are described hereafter. Another motivation for this part of our work, is the use of our system by contemporary music composers who always seek to expand traditional notions of music writing.

`TRILL` Class

As the name suggests, the `TRILL` class is a way to imitate classical music trill notation. In terms of modeling, a trill is one *in-time* event that encompasses several *out-of-time* events. Moreover, time-order, time-span and the number of repetitions of these sub-states are of no importance. For example, a whole-tone trill on a middle *C* with a duration of one beat ($\ell = 1$), can consist of 4 *quarters*, or 8 *eighths*, or 16 *semiquavers* etc. of sequences of *C* and *D*, depending on the musician, music style or dynamics of the performance. To compensate for these effects, we consider the `TRILL` class as one semi-Markov state s_i with a given duration, whose observation obs_i is shared by two or more atemporal states. During real-time decoding, the observation of the general `TRILL` state is the maximum observation among all possibilities for the incoming audio frame or: $b_j(x_t) = \max_{p_j}\{p_j = p(x_t|obs_i^j)\}$. Figure 7.6 shows two musical examples that can be described using the `TRILL` class, where the second[2] demonstrates a *free* glissandi which can also be successfully encoded using the this model.

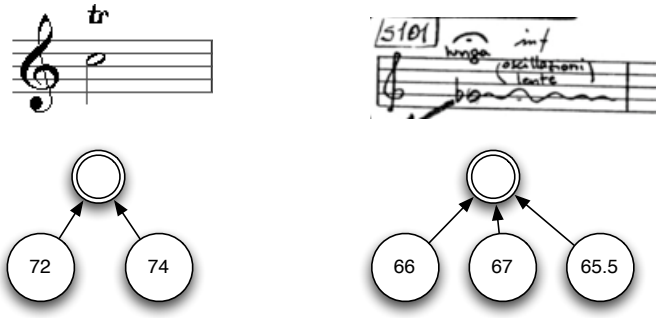


Figure 7.6: State-space topology for the `TRILL` class

[2]The handwritten music samples are taken out of piece *Little-I* for flute and electronics by Marco Stroppa, with kind permission from the composer.

`MULTI` Class

Another situation in music notation, less common than trills but of special interest to us, is continuous-time or smooth time events where the time-span of a single event undergoes change in the observation. An example of this in western classical notation is the continuous *Glissando* or *Portamento*, described as continuously variable pitch, where the musical instrument allows such notations (such as Violin, Trombone, human voice etc.). Moreover, such class of objects would allow matching for continuous data such as audio and gesture, along with symbolic score notations. To this end, we add one more class to allow more complex object and temporal encoding. The `MULTI` class is similar to the `TRILL` class with the exception that the symbols defined within its context are atemporal Markov states that are *ordered in time*.

In this new topology, a high-level semi-markov state represents the overall temporal structure of the whole object that is mapped to a series of sequential left-right Markov chains. Figure 7.7 shows a `MULTI` example for two consecutive notated glissandis.

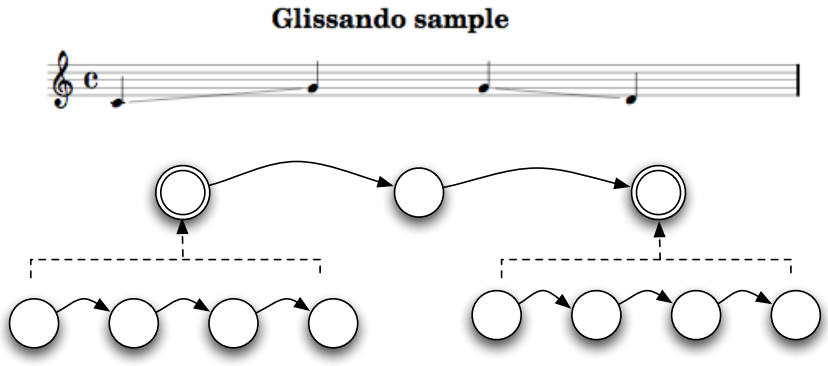


Figure 7.7: State-space topology for the `MULTI` class

7.7 Observation Model

The inference formulation of section 7.4 attempts to map audio signals as discrete frames x_t in time to their corresponding state s_t in a music score. As mentioned earlier, in our problem the states are not directly observable by the system and thus are hidden. The observation probabilities $b_j(x_t)$ in the inference formulation are thus the *eye* of the system towards the outside world and provide probabilities that the observation vector x_t is emitted from state j. In other words, they are the likelihood probabilities $p(x_t|s_j)$ which after entering the inference formulation above become posterior beliefs $p(s_j|x_1, x_2, \ldots, x_t)$.

In this section we show the model that provides the system with observation probabilities $b_j(x_t)$ during real-time inference. In general, the real-time audio entering system are represented as overlapping windows of fixed length over time. So each observation vector x_t corresponds to a vector of fixed time-span where t refers to the center of window. In the experiments shown in this chapter, the time window has a length of $92ms$ with an overlap factor of 4 as a compromise between frequency and time resolution of the input.

In a polyphonic music setting, the observation probabilities should reflect instantaneous pitches that are simultaneously present in an analysis window entering the system in real-time. While polyphonic pitch detection is a hard problem by itself, in our setting we do not need to push the problem that far since the music score provides a priori information regarding expected pitches during the performance. So the goal here is to compute the conditional probabilities $p(x_t|s_j)$ where each state s_j provides the expected pitches in the score.

For this aim, we choose to represent analysis frames x_t in the frequency domain using a simple FFT algorithm and compare the frequency distribution to frequency templates constructed directly out of pitch information of each event in the score decoded in s_j. This choice of observation model is natural since musical pitches tend to preserve quasi-stationary frequency distributions during their life-time which corresponds

to their fundamental frequencies along with several harmonics. Since we are dealing with x_t and s_t here as probability distributions over the frequency domain, it is natural to choose a comparison scheme based on probability density distance, for which we choose the Kullback-Leibler divergence as shown below:

$$D(S_j||X_t) = \sum_i S_j(i) \log \frac{S_j(i)}{X_t(i)} \tag{7.18}$$

where X_t is the frequency domain representation of x_t or $\mathcal{FFT}(x_t)$ and S_j is the frequency probability template corresponding to pitches in s_j.

Note that this framework resembles our Music Information Geometry framework of chapter 4. Here again, we are inherently considering FFT vectors and templates as Multinomial distributions or normalized histograms of frequency distribution. We saw previously in chapter 4 that the Bregman divergence (or similarity metric) for this statistical manifold is just the Kullback-Leibler divergence that is restated above.

The formulation above has a direct probabilistic interpretation that favors its use as the likelihood observation function: If S_j is considered as the "true" frequency distribution of pitches in s_j and X_t as an approximation candidate for S_j, then $D(S_j||X_t)$ gives a measure up to which X_t can describe S_j and is between 0 and $+\infty$ with $D(S_j||X_t) = 0$ iff $S_j = X_t$. To convert eq. 7.18 to probabilities, we pass it through an exponential function that maps $[0, +\infty] \rightarrow [1, 0]$:

$$p(x_t|s_j) = \exp\left[-\beta D(S_j||X_t)\right] \tag{7.19}$$

where β is the scaling factor that controls how fast an increase in distance translates to decrease in probability.

In order to construct the "true" frequency distributions of pitches in s_j, we assume that a pitch consist of a fundamental and several harmonics representing themselves as peaks in the frequency domain. Each peak is modeled as Gaussian centered on the fundamental and harmonics and their variance relative to their centers on a logarithmic musical scale. For this experiment, we use 10 harmonics over each fundamental

with a variance of a half-tone in the tempered musical system which can both be adjusted if needed by the user.

Note that the likelihood in eq. 7.19 requires normalization of X_t and S_j such that their would sum to 1. This normalization process undermine the robustness of the system to low-energy noise. Moreover, there is no single way to model silence or non-events using a template. For this reason, we influence eq. 7.19 by the standard deviation of X_t which reflects energy and also noisiness, to obtain $b_j(x_t)$. A similar method is also reported in (Raphael, 2006).

7.8 Evaluation

In this section, we provide results of real-time alignment and evaluate them in various situations. Evaluation of score following systems with regards to alignment was a topic in *Music Information Retrieval Evaluation eXchange (MIREX)* campaign in 2006 (ScofoMIREX, 2006; Cont et al., 2007c). In that contest, organizers with the help of the research community, prepared references over more than 45 minutes of concert situation acoustic music and defined certain evaluation criteria which we will reuse in this paper. However, no clear methodology is yet proposed for evaluation of tempo synchronization which is basically a different topic than the score alignment.

In order to address both tempo and alignment evaluation, we conduct three different experiments. In the first experiment in section 7.8.1 we evaluate the *tempo predictive model* of section 7.5.2 in a symbolic setting taken out of real performance recordings. We then move to the audio world and in section 7.8.2, evaluate the system against synthesized scores which will allow us to have detailed control over timing and evaluate both tempo and alignment up to milli-second order. In the section 7.8.3, we test the system against real music performances that has been previously checked and referenced in (Cont et al., 2007c) with some extensions.

7.8.1 Evaluation of Tempo Prediction

In section 7.5.2 we presented a simple recursive and online algorithm that provides tempo predictions of the next event using information up to the real-time. In this section we evaluate the performance of this predictive tempo model. The only synchronization system that also decodes and represents tempo along score position besides our proposed method is the on in (Raphael, 2006). We therefore evaluate our system along with a real-time adoption of tempo decoding of Raphael. Details of Raphael's model is presented in Appendix B.2.

This experiment is a basic one-step time-series prediction setup. Each step of the process is the arrival of an onset from live performance, thus simulating a realtime situation. The evaluation is done on a database of aligned music performance to score previously constructed by the author for MIREX Score Following Contest of 2006 (ScofoMIREX, 2006). In this experiment, we only use the original MIDI score and the corresponding aligned MIDI. Table 7.1 describes the contents of the database used in this experiment. The two data sets in the database correspond to a classical piece, Bach's Violin Sonatas performed by Menuhin and Kremer, and a contemporary piece by Pierre Boulez, ...*Explosante-Fixe*..., flute solo sections from *Transition V* and *VII* performed by Ensemble InterContemporain.

Table 7.1: Dataset Description used for Tempo Prediction Evaluation.

	Composer	Piece	No. of Files	No. of Events
Set 1	Pierre Boulez	*Explosante-Fixe*	27	1173
Set 2	J.S.Bach	Violin Sonatas	12	7422

For the actual experiment, we simulate both systems in a realtime situation where they are supposed to predict arrival time and tempo of the next event to come. Both systems start on each file in the database using the initial tempo of the given scores. After each performance, we compute the difference between predictions and actual arrival time (in seconds) as error rates.

Table 7.2 shows the evaluation result summary as statistics of the total error on

both data sets and for both systems. Statistics of the total error is demonstrated in terms of mean and standard deviation (in seconds). Results better predictions for the proposed model than Raphael's while both undergo acceptable performances.

Table 7.2: Tempo Prediction Evaluation Results: Error statistics (in seconds)

	Proposed		Raphael	
	mean	std	mean	std
Set 1	1.61	4.1	-11.8	17.9
Set 2	2.3	7.2	-13.8	11.2

For subjective observation, figure 7.8 demonstrates the results obtained on file#1 of data set 1 (*section 01, Transition V, ...Explosante-Fixe...*) as a rather fast and short piece with lots of temporal fluctuations. The prediction error of our model for this short performance is to the order of 0.045 seconds and thus coincides with the live performance whereas the original score timing are clearly deviated. This is a result of on-line adaptation of our system's parameters (κ). While Raphael's predictions follow the temporal flow of tempo, they deviate more and more from the actual performance due to error propagation during inference and lack of adaptive parameterization.

In practice, Raphael's variance parameters for the generative tempo and time models or $\{\sigma_{s_k}, \sigma_{t_k}\}$ (see appendix B.2) are adapted beforehand to the performance either by off-line learning or hand-tweaking. For the experiments above, we initialized them to half-life duration of the local tempi and IOI in each score. Therefore, Raphael's performance is usually better than what depicted above through pre-adaptation to a concert situation. However, the experiment above showcases the adaptive behavior of our proposed method. We will look at the adaptive characteristics of our tempo model in the following section.

7.8.2 Evaluation over synthesized audio from score

In this section, we evaluate the performance of the system against synthesized audio from a given score. The main reason for separating this procedure from real

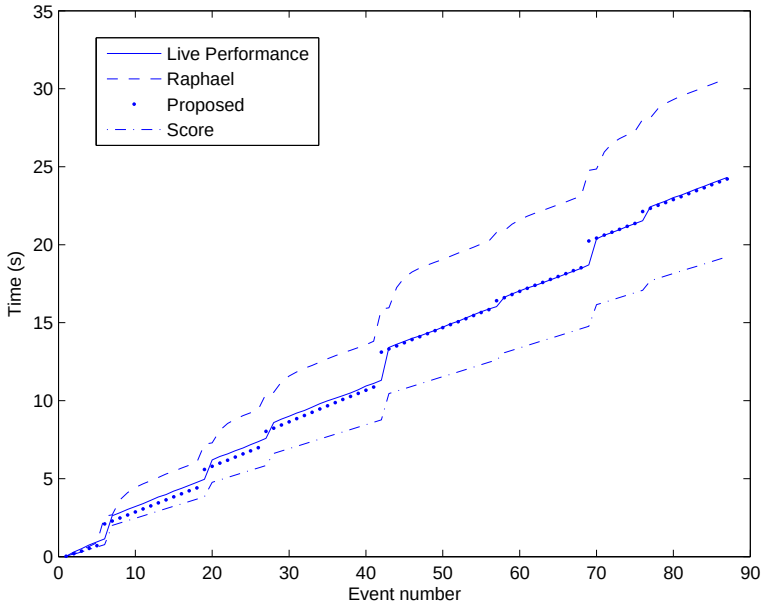


Figure 7.8: Subjective Results for Tempo Prediction Evaluation: Temporal alignment results of two systems with the original

acoustic performances is for reassessment of the tempo synchronization. While evaluating alignment result is easily imaginable using real and acoustic data, evaluation of tempo is a hard task. It is generally impossible to ask a musician to perform a given score using a temporal progression curve up to milli-second precision to be used as a reference. On the contrary, this is quite imaginable using synthesized audio by arranging temporal progressions of score events during the synthesis process.

Before defining the experimental setup and showing results, it is important to undermine several characteristics of the tempo agent described in section 7.5 in terms of its performance. First, the oscillator model has the underlying hypothesis that tempo progresses continuously and tempo process adapts or locks into the new tempo progressively. This means that when an abrupt or discontinuous jump occurs in the tempo, the κ or *attentional focus* should undergo abrupt change with the tempo random variable reaching an optimum equilibrium within a few steps. At the same time, when the tempo

changes continuously (for example in the case of an acceleration or deceleration), the agent should be capable of locking itself to the new tempo even if its best performance is when several equilibrium points exist for the best phase locked result. We study each case separately. In both experiments, we consider a simple score depicted in figure 7.9 containing 30 notes with a score tempo (or prior) of $60bpm$ or $1\ \frac{second}{beat}$. By synthesizing this score to audio, we enforce a different tempo curve than the fixed tempo of the score and feed both the score and synthesized audio into the system and study the results.

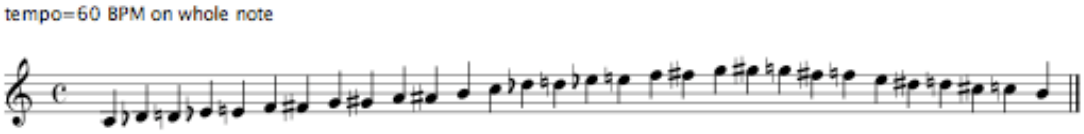


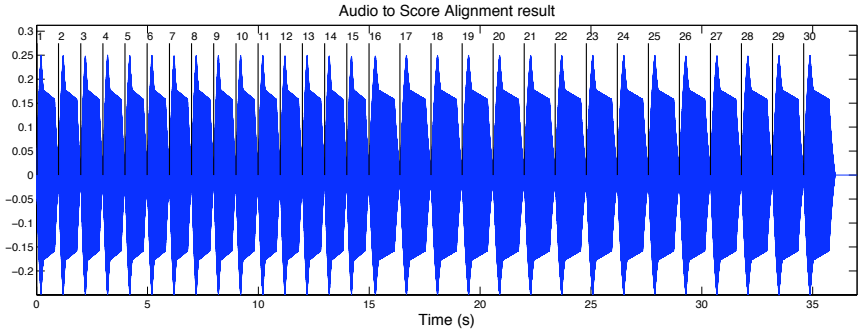
Figure 7.9: Sample score 1 for tempo experiment

The synthesis method used here is based on a simple FM synthesis method described in Moore (1990). We did not experience any significant difference by changing the synthesis method regarding the aims and results for this section. For evaluation on more complex signals (and concretely, real acoustic signals) we refer the reader to section 7.8.3.

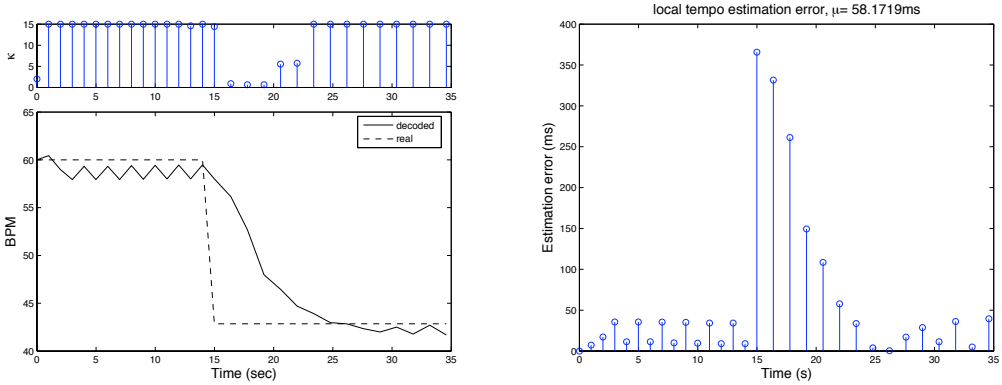
Discrete tempo jumps

We first study the result of one abrupt tempo change in the middle of the score, jumping from the original down to less than $2/3$rd of the original and on one note. The results are demonstrated in figure 7.10 where figure 7.10a shows the synthesized waveform with the alignment results where each number tag refers to one of the 30 notes in the score of figure 7.9. Comparing the left and right portion of this waveform clearly shows the difference in duration length of each event corresponding to the abrupt tempo jump. Figures 7.10b shows the tempo synchronization result along with the the real tempo curve as dashed line on the main left figure and the corresponding κ parameter at

each time step on the top, and local estimation error on the right figure. The estimation error is computed as the difference in milli-second between the real tempo and decoded tempo both expressed in $\frac{\text{milli-seconds}}{\text{beat}}$.



(a) Waveform and alignment result



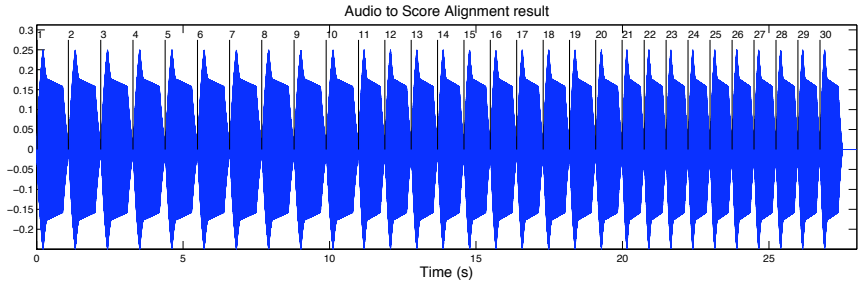
(b) Estimated and real tempi for acceleration and deceleration in BPM

Figure 7.10: Tempo Decoding Evaluation using synthesized score and controlled tempo

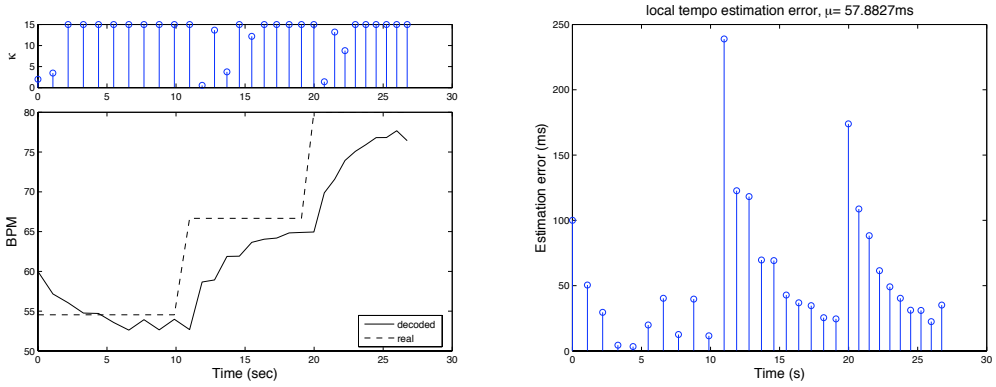
Looking closely at figure 7.10 we can interpret the online tempo adaptation as follows: On the first event after time $15s$, the tempo agent goes into a sudden instability leading to the biggest estimation error as depicted in fig. 7.10b on the right. This instability leads to a sudden change in the κ parameter that controls attentional focus. Therefore, κ at the onset following $t = 15s$ is quite low meaning that important tempo correction is necessary. This process continues for almost 5 consecutive events until the agent finally locks itself around the correct tempo which can be observed by looking at direct results converging to the real tempo, or by observing the decrease in the estima-

tion error, as well as by observing the increase in the adaptive κ parameter reaching its upper bound (here set to 15). The mean tempo estimation error is $58ms$.

We now take a look at another sample, this time by introducing two tempo jumps during the life of the synthesized score of figure 7.9. Results are demonstrated in the same format as above, in figure 7.11. Here, the audio starts with a different tempo than the one indicated by the score prior, so the κ parameter starts low in the beginning until stability and undergoes change every time the system enters inequilibrium as shown in figures 7.11b. The mean estimation error in the course of this session is $57ms$.



(a) Waveform and alignment result



(b) Estimated and real tempi for acceleration and deceleration in BPM

Figure 7.11: Tempo Decoding Evaluation using synthesized score and discretely controlled tempo

The onset estimation error in both examples above vary between $10ms$ and $30ms$ with no missed note (as is clear from alignment results in figures 7.10a and 7.11a). This high precision is not a surprise since here we are dealing with simple synthetic sounds

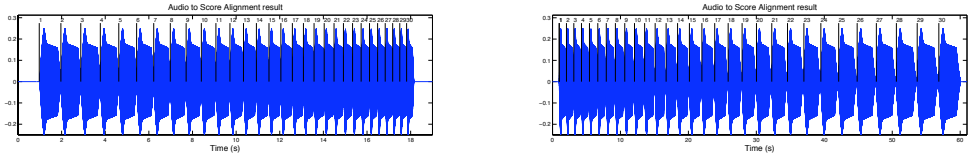
with rather stationary spectrums.

Continuous tempo change

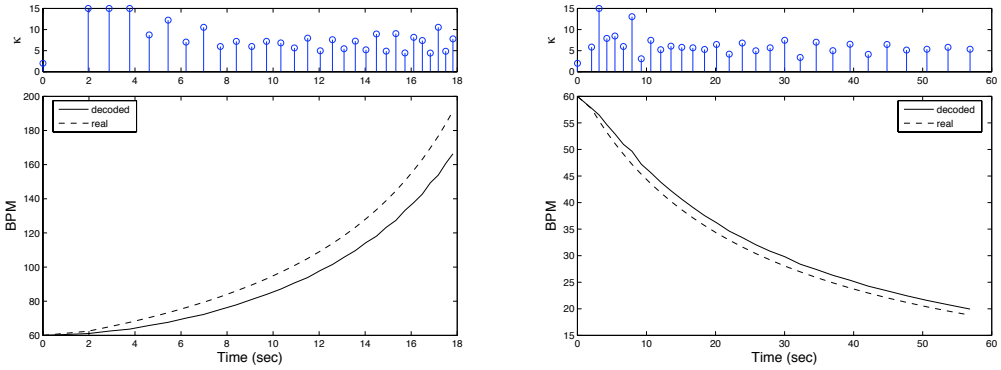
In this section we follow the same procedure as before, using the score of figure 7.9 for synthesis and alignment, but with the difference that the tempo parameter is continuously changing on each note event instead of abrupt or discrete change. This experiment is aimed at simulating acceleration and deceleration common in music performance practice. The control function for tempo during the synthesis is set to an exponential function $e^{\gamma(n-1)}$ where n is the note event number in the score and γ controls the slope of the change with $\gamma < 0$ indicating acceleration and $\gamma > 0$ deceleration over performance time. A partial goal here is to demonstrate the performance of the system despite the lack of time to reach an equilibrium state which was the case in the previous experiment.

Before doing a mass evaluation, we visually demonstrate two result sets to get a better feeling of the performance of our system. Figure 7.12 shows the performance of the system using acceleration (left) and deceleration (right) schemes with $\gamma = \mp 0.04$ resulting to a tempo difference of $131bpm$ and $-41bpm$ respectively. As before, we are demonstrating the resulting synthesis waveforms and alignment tags in fig. 7.12a, the real and estimated tempi along with adaptive κ parameters in fig. 7.12b, as well as tempo estimation error on each event in fig. 7.12c.

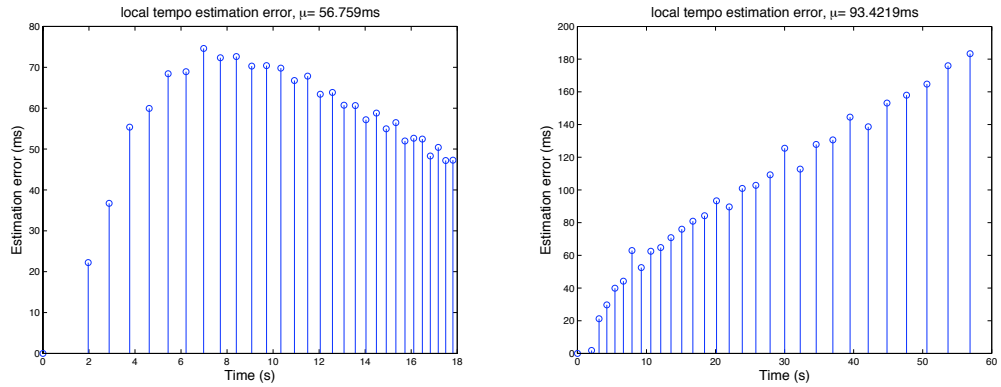
Figure 7.12 leads to the following important observations: First, the κ parameter is constrantly changing over the course of both processes in figures 7.12b. This is normal since the reference tempo is continuously evolving in both cases. Second, note that while γ only changes signs in the two cases, the estimation results and the mean errors are quite different. This phenomena is easy to explain: In the deceleration case (right portion of fig. 7.12, the difference between the two tempo extremes is about $-40bpm$ but the time steps between each event (and their respective tempo-phase) is exponentially increasing, so the system needs more time and steps to reach a better stability point;



(a) Waveforms and alignments for accelerating (left) and decelerating (right) tempi



(b) Estimated and real tempi for acceleration and deceleration in BPM



(c) Estimation error for acceleration (left) and deceleration (right)

Figure 7.12: Tempo Decoding Evaluation using synthesized score and continuously controlled tempo

despite that it follows the original curve correctly. This leads to a bigger estimation error than the acceleration case, where the phase-steps become smaller and smaller at each step. This observation is further enhanced by noticing that the estimation error for the acceleration curve (left of fig. 7.12c) start decreasing after a while which is not the case for the deceleration case.

The observations above are further enhanced by enlarging the evaluation set by vayring γ values during synthesis. Table 7.3 shows the same evaluation procedure above for various values of γ, where the first three columns characterize the synthesized audio from score in figure 7.9, and the last two columns show tempo and onset estimation errors in milli-seconds. Here again, we can observe that accelerating equivalences of $|\gamma|$ (or $\gamma > 0$) have better estimation rates than their decelerating equivalence. The estimation errors here are the mean over all the events in the score (total of 30 in each case). The reader might argue that an estimated error of $158ms$ (reported in the last row of table 7.3) is not acceptable for a tempo synchronization application. In response, we argue that the tempo difference associated with this process ($281.8bpm$) is almost never experienced in a musical performance setting *unless* it is stated explicitly in the music score by a discrete tempo change (combined or not with an acceleration) which would resolve the case.

Table 7.3: Tempo Decoding Evaluation: Batch results over exponential tempo curves

γ	Length (s)	ΔS (bpm)	**Tempo Err (ms)**	**Onset Err (ms)**
-0.06	16.0	-49.5	68.22	9.50
-0.05	17.0	-46.0	62.51	9.35
-0.04	19.0	-41.0	56.32	9.73
-0.03	22.0	-34.9	44.02	9.27
-0.02	24.0	-26.4	37.87	9.26
0.02	43.0	47.1	8.13	10.82
0.03	51.0	83.2	44.44	10.34
0.04	61.0	131.4	93.46	9.59
0.05	73.0	195.7	104.68	9.50
0.06	88.0	281.8	158.78	8.69

7.8.3 Evaluation of real-time Alignment

In table 7.3, we are also reporting the mean onset error which is the elapsed time between the detected time of each event and the synthesis reference. These error rates are extremely low (around 10 milli-seconds) for each fragment. While these results are highly encouraging, in a real acoustic music situation, the audio signals are much less stationary than synthesized signals used in the previous section. In this section, we evaluate the real-time alignment results in the context of acoustic music performances.

In 2006, an international evaluation campaign was organized by the research community for the evaluation of real-time audio to score alignment algorithm for Music Information Retrieval Evaluation eXchange (MIREX) and was reported during the ISMIR conference in Victoria, Canada on August 2006. During this campaign a general consensus was obtained for evaluation metrics and procedures applicable to most available systems. The agreed procedures as well as documentation of all details and dicussions are available through the MIREX web-portal (ScofoMIREX, 2006) and in (Cont et al., 2007c). For the sake of completeness, we briefly describe the evaluation procedure and metrics used as well as our own addition to the database before showing results.

Evaluation consists of running the system on a database of real audio performances with their music scores where an alignment reference exists for each audio/score couple. This procedure aims at simulating a real-time performance situation, thus audio frames are required to enter incrementally into the system. More details as well as calling conventions are accesible in (ScofoMIREX, 2006).

Table 7.4 describes the database used for this evaluation which is a partial copy of the one in (ScofoMIREX, 2006) plus some additions. Items 1 and 2 are strictly monophonic, item 3 is lightly polyphonic with the appearances of music chords of the violin from time to time in the piece, while item 4 is strictly polyphonic with up to 4 different voices happening at the same time. This database contains more than 30 minutes of referenced audio/score pairs and has been chosen to demonstrate the performance of the

system on different musical instruments, and styles (item 1 is in contemporary music style with unconventional timings) and degree of polyphony. Items 1 to 3 are used in (ScofoMIREX, 2006) whereas item 4 is aligned using a heavy offline algorithm reported in (Yeh, 2008) and further enhanced as described in (Yeh et al., 2007).

Table 7.4: Evaluation Database Description

#	**Piece name**	**Composer**	**Instr.**	**Files**	**Prefix**	**Events**
1	Explosante-Fixe	P. Boulez	Flute	7	t*x*-sy	615
2	K. 370	Mozart	Clarinet	2	k370	1816
3	Violin Sonata 1	J.S. Bach	Violin	2	vs1-	2019
4	Fugue BWV.847	J.S. Bach	Piano	1	RA	225

Once every piece is ran through the system, we obtain a set of event tags i with their detection times t_i^d in milli-seconds. The process of evaluation then, is to compare the results with the previously prepared references for each piece with the same tags i and alignment times t_i^r. Evaluation metrics are then the number of misses, and corresponding statistics on the offset time $o_i = t_i^d - t_i^r$ between detected time tags and the associated ones in the reference database. Table 7.5 shows the results of evaluation on each file in the described database, starting from monophonic scores and going gradually towards the polyphonic ones. Here *FP* refers to *false positive* which are misaligned events and are parts of the *missed* events. The *Average Offset* error is the mean over the absolute offset values or $\sum_i |o_i|$ where *Mean Offset* is the regular mean without taking the absolute value. Given these statistics, the *Overall Precision* is calculated as the percentage of total number of events to detect minus the total number of missed notes whereas the *piecewise precision* is the mean of the same rate but over individual files. In (ScofoMIREX, 2006) another metric is proposed pertaining to *latency* and defined as the interval between the detection time and the time the event is reported. This metric was specifically designed for systems that are *real-time* but are not necessarily *on-line*; hence, allowing a delay in reporting the correct alignment. This is the case for example in (Raphael, 2006). We omit this measure from table 7.5 since our system is strictly on-line and real-time and thus, this measure is always zero.

Table 7.5: Real-time Alignment Evaluation Results

Source Info		Offset (ms)			Percentage	
Filename	Events	Average	Mean	STD	Missed	FP
K370.030	908	188.4	188.4	255.3	7.49%	0.22%
K370.032	908	166.1	166.1	208.9	5.95%	0.22%
s01	88	85.7	85.7	24.8	2.27%	0.00%
s04	76	81.7	81.7	29.0	5.26%	0.00%
s06	108	75.1	75.1	34.6	4.63%	0.00%
s11	63	109.4	109.4	217.4	17.46%	0.00%
t7-s03	90	115.3	115.3	63.9	6.67%	0.00%
t7-s16	98	113.0	113.0	26.2	5.10%	0.00%
t7-s21	92	106.0	106.0	25.4	3.26%	0.00%
vs1-4prs	1604	240.9	240.9	165.0	10.41%	0.00%
vs1-1ada	415	130.1	130.1	106.6	12.53%	1.45%
RA-C025D	225	99.8	99.8	75.3	9.33%	0.00%
Total Precision:		$\mathbf{91.49}\%$				
Piecewise Precision:		$\mathbf{92.47}\%$				

Note that since we are in a live performance simulation, meaning that data is fed incrementally into the system, the system can *get lost* or get stuck during performance. The overall high precision rate for each piece shows that this is not the case and the system has successfully terminated each audio through its end. In overall, the reported performance in table 7.5 is comparable to other systems on strictly monophonic pieces. However, we outperform the two other systems in (ScofoMIREX, 2006) on Bach's Violin Sonata files which have light polyphonic and here we report a strictly polyphonic version (for file *RA-C025D*). The fact that *average offsets* and *mean offsets* are always equal indicate that our system is never *early* in detection or $\forall i : o_i \geq 0$. Also note that in our system, no off-line learning is necessary since all system parameters are constantly being updated online.

7.9 Discussions

In this chapter we presented a new approach to live synchronization of audio with music scores. The novelty of our approach is its explicit anticipatory model, coupling audio and tempo into one unique inference framework, and also the fact that it provides users and interactive systems with decoded real-time tempo of the performance. We evaluated the system in three different setups. The proposed system is computationally cheap and easily controllable. It has been implemented within the mainstream interactive computer music systems (*MaxMSP* and *PureData* environments) and has had successful performances worldwide including its premiere in Japan[3], as well as a performance with Los Angeles Philharmonic[4], and more.

Despite its robustness, the design introduced in this chapter provides musicians and composers with necessary access to temporal processes both within their scores and enables ways to think of a *writing of interaction and time* for interactive computer music. This will be our next topic.

[3]Performance of ... *of Silence* of composer Marco Stroppa, for Saxophone and Chamber Electronics, Shizuoka, Japan, November 2007.

[4]Performance of *Explosante-Fixe* of composer Pierre Boulez, LA Philharmonic, Disney Hall, Los Angeles, 13 and 14 January, 2008.

Chapter 8

Towards Writing of Time and Interaction in Computer Music

Computer music, through its relative short history compared to our musical heritage, has spread itself through various mediums that it is harder than ever to define it in a coherent context. Through its short history, computer music trends have always been driven by culmination of mutual research and artistic endeavors. Computer music as an art medium takes a wide variety of forms from fixed "tape" electronics, to interactive compositions, mixed media (gesture, audio, video etc.) as well as written (scored) music for electronics and live musicians to improvised performances. Consequently the research literature within, encompasses a wide range of activities including sound synthesis, signal processing, physical modeling, information retrieval and interactive systems.

Despite this variety, all approaches to computer music whether for pure performance, transcribed compositions, or pure generation, have strong considerations for *musical representations* of ideas in one way or another. With this respect, different practices of computer music have led to various paradigms of composition evolved from different computer programming paradigms addressing different views of musical representations. In return, each specific domain has its own literature concerning representations

and writings of events and processes. For example, the research in sound synthesis has given rise to many softwares and semantics for representing and modeling compositional processes (See Bresson, 2007a), interactive compositional systems to a different set of considerations (e.g. Rowe, 1992), spatial diffusion to distinct considerations for the writing of spatial events (e.g. Nouno and Agon, 2002), and much more; where each topic can be a subject of a PhD thesis itself.

In this chapter we are interested in exploring the idea of *writing* interactive computer music parameters in time. Therefore, we are not directly concerned with specific techniques in computer music. We simply assume that there exist a set of computer music processes with their respective control parameters that can handle desired computer music actions previously conceived for a compositional or performance setup. Writing is then to give form to these parameters in time, in terms of composition, and transcribe them in one way or another, in analogy to the functionality of western music notation for instrumental music. Moreover, we are interested in the writing of computer music parameters in interactive settings, with its most obvious form as interaction between a musician and a machine where the second aims at interpreting its own (pre-composed) "computer music score" along a human performer in a live performance. Thus, this chapter is mostly concerned with computer music practices within the realm of mixed instrumental and electronic music composition where transcription of ideas (in terms of composition) and their realization in a performance are central to the design. Based on observations in previous chapters, we introduce a preliminary framework for the writing of time and interaction in live computer music called *Antescofo*. In our propositions we have no claim of universality of methods; but rather provide a practical solution to the problem of abstracting time and interaction in both writing and performance of scored live electronic pieces. The focus of this chapter, therefore, is on *real time* and *interactive* computer music systems.

To motivate the subject, we survey the computer music literature and practice from a computer programming standpoint centered on the notion of interaction, as well as the practice of scoring interactive parameter and briefly review the stance of the re-

search literature on the issue.

8.1 Background

8.1.1 Computer Music Language Paradigms

The practice of computer music today is highly tied with software platforms that provide environments for expressing computer music processes as computer programs either in a text-based or visual language. A computer language in general presents an abstract model of computation that allows one to write a program without worrying about details that are not relevant to the problem domain of the program. The power of abstraction of a language is tightly related to the computational paradigm the language lies upon. The computational paradigm of a language determines the extent to which abstractions can or can not express desired phenomena. The computational capabilities of computer languages can be comparatively studied as *programming paradigms* or fundamental style of computer programming (Van Roy and Haridi, 2004). Below we look at some of the mainstream programming paradigms in computer music and review their power of expressivity with regard to interactivity[1].

In our survey of computer music programming paradigms we often emphasize on the capability (or absence of) "real time" and "interactive" computability. Both terms need to be defined in a coherent and computational context. A programming language is considered *real time* or *interactive* if in its computational paradigm, evaluation of procedures can be affected at runtime using external elements to the program. These external elements can be human-computer interfaces, indeterministic processes conditioned on an analysis, recognition, or some affectation of data stream into the system, or the like. Therefore, *real time* and *interactive* capability of a programming language in our context has not much to do with fast computation, but rather explicit considerations in the

[1]The list of available softwares in computer music is exhaustive. Here we just provide some names that underlie the majority of users in the field.

platform's design for real time scheduling, garbage collections, etc. that enables a minimum degree of interaction between the computational paradigm and elements outside the system. Many of the programming paradigms introduced below and claimed non real time are actually capable of very basic interactions with an outside element through various forms of procedural evaluations, but are not inherently designed for *real time* or *interactive* programming or their use in those contexts are limited by the paradigm under consideration.

Computer Assisted Composition Languages

The dominant programming paradigm in most *Computer Assisted Composition (CAC)* packages is *functional programming*. In this programming paradigm computation is treated as evaluation of mathematical functions. In a computer music setting, functions can describe musical processes where a chain of functions, hierarchically set up in the program, can describe a complex musical process once evaluated. *OpenMusic (OM)*, *Patchworks* (Assayag et al., 1999) and *Common Music* (Taube, 1997) are examples of this kind of approach with the first two providing a visual environment for musical compositions and all three based on Common Lisp programming language. Among these three packages, *OM* enables object-oriented programming paradigm into its graphical environment by allowing notions of class, inheritance and methods for classes; even though the majority of its users concentrate on the functional programming aspects. In this paradigm, there is no notion of real time computation for the expense of providing rich data-structures and focus on formal structures of music as opposed to processing of audio. This world is close to ideal from the composer's point of view whose creation of musical score is essentially an out-of-time activity. Central to the success of both *OM* and *Patchwork* is the presence of flexible integrated music notation display packages with a transparent connection between the displayed score and the data underneath, enabling easy transitions between the two media (Puckette, 2004). Despite their general lack of direct audio manipulations, these platforms have proven to be ideal

for *controlling* sound synthesis engines in off-line mode. In this case, the basic idea is not to implement the synthesis engine but to handle musical structures visually and then generate low-level parameters to send to an available outside audio engine (Agon et al., 2000). Another facility of this paradigm is the notion of *evaluation on-demand* during computation where the chain of functionals can be evaluated anywhere in the program, stopped and reissued; leaving the user with the ability to interact with the computational process at the expense of loosing the notion of real time computation. The ability to operate within hierarchies constructed out of users' abstractions has attracted many composers to these languages; however, the lack of strong temporal considerations in the language (in terms of semantics and control) still remains unaddressed.

Music-N languages

Most existing synthesis packages have their roots in the *Music* language series introduced early on in the history of computer music by Max Mathews. These languages are commonly referred to as the Music-N paradigm (Mathews et al., 1969). The central concept in Music-N languages is the *unit generator* or the minimal functional entity in the language. Traditional unit generators receive input control signals and produce sound at their outputs and include functionalities such as simple oscillators or envelope generators. Unit generators can be combined into composite structures called instruments or patches. A Music-N program consist of two main parts: the *instrument* or *orchestra* definition and the *score* or *note list*. In the synthesis process the composer uses a group of sound objects (or instruments) known as the orchestra. This orchestra is controlled from a score and is defined using a programming language with specific functions. These functions or modules can be organized and combined in multiple ways in order to define instruments. *CSound*[2] (Vercoe, 1993) is the most successful and used reminiscence of the Music-N paradigm today. The abstractions in *CSound* make writing signal processing algorithms relatively easy. However there is no facility for intercom-

[2]http://www.csounds.com/ (free software)

munication between the pre-scheduled synthesis nodes and real time control input. They simply affect different variables in the "orchestra" which by itself is controlled by the union of the two.

Real time languages

SuperCollider[3] (McCartney, 1996) is a language for sound and image processing that can be considered as a Music-N paradigm language with extensions to real time computations. It is a Music-N style language because of its use of unit generators and other common semantics of Music-N paradigm such as instrument, orchestra and events. It can run in real time and can process live audio and MIDI inputs and outputs, with a strong object-oriented programming style. Like Music-N languages it is based on textual (non-visual) programming but features strong considerations for dynamic memory allocations and periodic garbage collection that allow *live coding* or real time interaction with the code under execution. Particularly, the compositional and synthesis engines are separated in *SuperCollider* to allow simultaneous instances of the synthesis engine through the compositional language via network protocols (Mccartney, 2002). In late versions of the software, graphical user interfaces can be created for visual controls but the programming paradigm itself is inherently text-based. With this respect, *SuperCollider* goes in the direction favored for performance or real time control of composed material through live coding.

MaxMSP[4] (Puckette, 1991; Zicarelli, 2002) and *PureData (Pd)*[5] (Puckette, 1997), particular instances of the same model, are probably the most widely used computer music environments, and among the richest in the community in terms of integrated modules for computer music. They are both real time visual programming environments. Despite other languages which can be safely categorized within a common programming paradigm, *Max* and *Pd* evade easy categorization and whether they can be consid-

[3]http://www.audiosynth.com/ (free software)
[4]http://www.cycling74.com/ (commercial)
[5]http://crca.ucsd.edu/~msp/software.html (free software)

ered a programming language has been subject to debates (Desain and Honing, 1993). However, in their programming practice they can both be considered as *Dataflow* languages. In a dataflow program, a process ready for calculation on an unbound variable waits until the arrival of appropriate stream (instead of quitting with an error). Within this simple framework, real time calculation is possible by carefully scheduling audio, control and graphical processes to synchronize calculations laid out in visual patcher programs. Within this framework, human-computer interaction is possible by connecting live audio or control inputs to live outputs through patcher programs. Despite programming flexibility in both environments (which is the key to both softwares' success), memory allocation is rather static, and data structures remain simple compared to the *CAC* paradigm. *Pd* employs various workarounds to this problem that makes *live scripting* and more complex data structures available to the platform. Relative early arrivals of both *Max* and *Pd* and their flexible and interactive programming environments have made them the most widely used platforms within the live electronics communities.

8.1.2 Practical Status

Puckette (the author of both Max and Pd) points out the division in the community between the *compositional* and *performative* aspects of computer music (Puckette, 2004). This division is partly due to the fact that, as seen above, *real time* environments favor computational transactions that go in the direction of performance whereas *CAC* environments go in the direction favored by the *composers*[6]. In a historical remarks on the development of both softwares, Puckette (2002a) notes that *Max* and *Pd* architectures were inspired from the interaction model of a musical instrument with its performer rather than musical interactions between musicians or a musician with a composed score. Commenting on this community divide, he further assesses that:

> "In its most succinct form, the problem is that, while we have good paradigms for describing processes (such as in the *Max* or *Pd* programs

[6]We use the term *composer* here loosely as the musician who *writes* or *transcribes* ideas down, whose product has an independent life than that of the composer and in a written form.

> as they stand today), and while much work has been done on representations of musical data (ranging from searchable databases of sound to *Patchwork* and *OpenMusic*, and including Pd's unfinished "data" editor), we lack a fluid mechanism for the two worlds to interoperate" (Puckette, 2004).

On the users' side, and in spite of composers' more active role in the beginning of the field, the *performers* of computer music have been faster to grab ideas in real time manipulations and adopting them to their needs. Today, with many exceptions, a wide majority of composed mixed instrumental and electronic pieces are based on simplistic interactive setups that hinder the notion of interactivity. This fact does not degrade the artistic value of such works in any sense but underlies the lack of momentum therein for serious considerations of interactivity among the second group.

Part of this frustration is due to the fact that the *writing composer*, faced with the abundance of possibilities in the electronics and in most cases lack of the culture compared to the rich instrumental traditions where she comes from, has extreme difficulties in abstracting and writing electronic event destined for a performance setting compared to their habitual facility with instrumental writing. When it comes to scored interactive pieces two main issues arise: that of scoring electronics and that of performance.

Scoring Interactive Electronics

The common trend to this date for the repertoire of scored mixed pieces is to devise an *electronic score* separate from the *instrumental score* into the computer that runs in parallel with the musician in a live performance and generates (fix or dynamic) electronics as instructed in the electronic score. The ultimate and common solution to this date is the use of tagged sequential data-structures for electronic scores (called `qlists` in Max and Pd). Such structures store variable/message pairs attached to symbolic time indexes and scheduled on a milli-second time-base. The symbolic time indexes in this structure would correspond to synchronization pivots in the instrumental score, destined for live synchronization. The modularity of environments such as *Max* or *Pd* allow co-

existence of multiple sound processes in a single patch that can be controlled through the sequential electronic score. Therefore in practice a `qlists` (or the like) can store and address any existing process defined in the electronic patch. This basically constitutes the *writing* of electronics in today's most scored interactive pieces.

Puckette (2002b) has introduced more complex data structures into the *Pd* environment which should facilitate scoring of events with various time scales (continuous versus discrete). These structures appear as modular graphical objects whose numerical and graphical properties (such as shape, color etc) are defined by the user. The user has the ability to edit, import and export the designed data structures. It can then be "played back" in real time by controlling a tempo parameter. The Pd data structures are similar in goal to *maquettes* in OpenMusic (Bresson, 2007b), with the exception that the scheduler is real time and hierarchies are not as deep. The main idea behind Pd data structure environment is then to provide a graphical scoring environment where realization of electronic pieces within the same tradition of pieces such as Stockhausen's *Studie II*, or Xenakis's *Mycenae-*α would be possible.

Performing Live Electronics with Acoustic Musicians

The issue of electronic performance is a vast topic. As before, we concentrate on the technicalities of the issue for scored mixed instrumental and live electronic pieces. The issue has much to do with the synchronization between the two scores during a performance, and the forms of interaction between a live musician and the electronics.

The consensus for interaction between a live music performance and electronics dates back to early experiments of Maderna, Stockhausen and Davidovsky among others composers through "fixed" or tape pieces where the live performer is responsible for synchronizing with the electronics using click-tracks or active listening. Later in mid 1980s, the movement is joined by experiments and repertoire of the so called "real time electronics", starting from experiments by Manoury and Boulez, where most often a score following application is responsible for synchronizing events of the live instru-

ments to the pre-written score and triggering the appropriate electronic events (whether fixed or through live generation). In this latter group, until today, composers have leant on the idea of a score following application to automate the performance of the electronics score through synchronization with live performer(s), while some others with Manoury as a forerunner, immediately recognized and incorporated the possibilities in writing interactive electronic scores which are realized during the live performance (Manoury, 1990).

Despite all the advantages that a score following system could bring to the performance of interactive electronic scores, its use has been limited to a small group of composers. Besides technical difficulties of such systems, this lack of enthusiasm is also due to the simplistic vocabulary of such systems for music notation (mostly based on notes, silences and hopefully trills) and no consideration for interaction (rather than mere triggering) during both writing and performance of electronic events.

8.1.3 Compositional Status

Real time capabilities of computer music softwares have provided means of interactivity between acoustical instruments and the computer. For this notion to be compositionally useful, the mode of coordinations between the two during both acts of composition and performance need to be determined. The most evident scenario for an interactive coordination is as follows: From time to time, for synchronization or realization needs, a communication window opens between the computer and the voice of the instrument and closes once the goal, as described by an electronic score, is achieved. This way the instrument and the computer are assumed as independent processes but communicating. A simpler scheme would just be the issue of synchronization between chunks of pre-composed electronics (as audio files) with the live performer where synchronization assures the departure of pre-composed electronics at the right time with the musician but without much control over the temporality of the process. The two scenarios described above are probably a summary of the most frequently used interactive

settings in many scored mixed pieces in the repertoire.

In these modes of communication, interaction is reduced to discrete points of instantaneous reactions, and live electronics consist of the short occasions when the communication window acts as gates to static processes. The notion of *time* as well escapes the electronic process (in both compositional and performative aspects) and reduced to simple disjoint temporal pivots. These modes of communications would surely contribute to the liveliness of music performance and add new horizons for the process of composition specially compared to mixed pieces with fixed ("tape") electronics. But they can in no way come close to the temporality, complexity and abstraction of both music performance and compositional process of their acoustical counterparts.

Naturally the possibility of creating interactive music systems attracted composers and artists for new experimentations and also integration of these ideas within their realm of compositions. Within two volumes, Rowe (2004, 1992) demonstrates how these novel paradigms have affected different practices in computer music. Among composers exposed to these new possibilities in the 1980s, Philippe Manoury was one of the earliest who integrated interactive computer music systems into his compositions and as a compositional process both during writing and live performance. Manoury's early formalizations of the problem, in collaboration with Miller Puckette, led to the birth of the *Max* programming environment, and were further developed and integrated by other early composers such as Boulez, Lippe and Settle, and since then is widely referred to as the *real time school* of composition.

For our review of this section, we base ourself on earliest and latest thoughts on practices of real time composition mainly centered around the writings of one of its pioneers, Philippe Manoury. Our interest in his line of thinking comes from the fact that early on he attempted to integrate and capabilities of a computer in the long tradition of composing and performing. This point of departure for computer music does not limit its practices to traditional norms of music composition but on the contrary puts it in a coherent style where non-traditional practices of computer music such as interactive composition (Chadabe, 1984), hyperinstrument composition (Machover and Chung,

1989), composed improvisations (Chabot et al., 1986) and more could take form as transcribed thoughts well suited for composition and performance. Manoury's integration of real time systems in compositional processes, as described in (Manoury, 1990), is very different from that of CAC, and in a way is a direct consideration of writing of time and interaction in a real time interactive setup. Interestingly, many of Manoury's ideas on writing of time and interaction has haunted the research community for years and even today, upon technological advances many of them remain unaddressed. Below we survey some of his main concepts introduced early on in the field and his more recent pleas and criticisms of the field's progress, followed by some constructive criticisms of his *real time* approach to composition that would shape our motivations for this chapter.

Time of Performance vs. Time of Composition

The core of writing of time and interaction for Manoury is *score following*. Despite the fact that he is one of the advocates of the use of score following technologies for performance and composition of real time repertoire, his musical demands of this technology still goes far beyond what the current systems can offer. To integrate real time processes into his process of composition, he distinguishes two networks of relationships for the *time of a score* and *time of performance*. The score follower or synchronizer is then the oracle that bridges the gap between the two during a live performance with the aim of interpreting scored live electronics in coordination with live musicians.

The performance or interpretation of a music score is neither deterministic and nor totally unpredictable. For automatized and "mechanical" processes such as live electronics, this *time* becomes a crucial source of variation and constitutes the musicality of the output. Therefore access to temporal structures during both writing and performance of musical time is crucial. The richness of our musical heritage that has arrived to us through notated scores and performances within centuries, obliges us to consider the passage between the time of the score to that of performance within instrumental prac-

tices and find analogies within the world of computer music. Western music notation is probably one of the most abstract ways of written expression the human kind has achieved. Even an elaborate music score can hardly describe a bit of the complexity of its realization. It consists of both absolute and relative values which find their essence during their realizations. Despite this inherent indeterminacy in music notations, the subsequent results are consistent without being totally unpredictable or deterministic. For Manoury the real quest of real time composition is to find analogies of these temporal abstractions, and passages between the two. To recapitulate in his own words:

> We should come up with means of composing electronic music in which, as analogy to instrumental music, some components are determined beforehand, and others variant under what is meant to be expressed (Manoury, 2007)[7].

This brings us to his next important theme, the *virtual score* for real time composition, and considerations for their realizations.

Virtual Scores and their realizations

A virtual score, in Manoury's terms, is a musical organization in which we know the nature of the parameters that will be processed but not their exact outcome at run time. Some of these abstract values will be fixed and remain fixed during any interpretation of the piece, and others are expressed relative to an outside environment. The latter underline the influential aspect of an outside process (live buffering, gesture processing etc.) for the realization of the defined compositional process, and evaluated during live performance. In this manner, a score is virtual since its existence is dependent on the music performance with its rich diversity of outcome (Manoury, 1990). It is interesting to note that despite the evident nature of this definition, there is no existing framework that addresses necessary variability, modularities and abstractness of virtual scores. Practicalities such as `qlists` seen before are just approximations to temporal abstractions and variability that Manoury's virtual scores would like to achieve. He later

[7] Translations of texts by Manoury from French by this author.

confesses that “The representations of time in current real time systems do not provide the composer with any symbolic notion close to temporal abstractions in western music scores. The real time system only knows of absolute time values and it is still impossible to write with a tempo or relatively assign accelerandos. Surprisingly, all that our real time systems are aware of is milli-seconds!” (Manoury, 2007).

The idea of virtual score is to bring in both the time of performance and composition into considerations during the process of composition. All this, given that the communication channels between the computer and the musician should ideally allow simulation of complex interactions such as between two human musicians. In his recent writings, after almost two decades of experimenting with the idea in practice, Manoury raises a very interesting subject:

> “If it’s possible to construct sound structures endowed by the reactivity of a music performance, we still perceive a huge resistance for unifying acoustical time with that of electronics to achieve a common musical time. An important boundary between the two conceptions of time seems to be opaque which has something to do with the nature of musical time in general: [...] *The possibility of prediction.* [...] A musician takes control of time within a continuity of gestures and motor controls such as breathing. He has a clear conscious of the past and the future, and intervenes memory and most importantly predictions into the process. We should therefore be able to represent, to the best of we can, the time of the machine as an image of that of humans” (Manoury, 2007).

This *possibility of prediction* and access to its behavior and control is one of the central goals of this thesis and the system proposed in this chapter. In our proposal we take one step further by postulating the notion of anticipation as the outcome and feedback of prediction (see definition 2.1 on page 17).

Criticisms

The advent of score following techniques for synchronization of live performance with electronic score and control of interactive parameters created a lot of momentum both in the research and music communities since early 1980s but not without

criticism. Among many criticisms directed towards *real time electronics* school, of particular interest are the ones by composers Jean-Claude Risset and Marco Stroppa. Risset argues that "Not only does the use of real time systems bring limitations and difficulties for the durability of the music, but one may even argue that the concept of real time concerns primarily performance and may be of little relevance to musical composition" (Risset, 1999). A constructional read of Risset's paper would point to an important drawback of existing systems: the lack of compositional interaction during performance. While this issue is in most parts aesthetical, it has also a great deal to do with a lack of explicit designs for such matter.

Stroppa's extensive criticism of real time electronics is accompanied by the composer's detailed proposal for a *Virtual Interpreter (VI)* system (Stroppa, 1999). In this vast proposal, the composer is mostly concerned with *temporality* of musical events and different temporal degrees of interaction in computer music spanning from explicit interaction of fixed or live electronic processes with real time detected tempi to continuous representations of time to allow fine grain tuning of composed electronics to a live performance. It should be noted that Stroppa is among the few composers who advocate the process of composition through transcription and writing of formal ideas and has developed a modular library called *OmChroma*, native to *OpenMusic*, for the *writing* and formalization of sound synthesis processes (Stroppa, Lemouton, and Agon, 2002). In a way, Stroppa's *Virtual Interpreter* proposal comes close to Manoury's plea for explicit *temporal access* in both the score and performance of computer music.

8.1.4 Research Status

Besides the reluctance of most composers to new possibilities of computer music, the abundance of computer music modules, softwares, and platforms today has rather served as a barrier for the composer's mind for abstraction of his ideas in a formal way to *transcribe* his thoughts in the same way he does with instrumental music. A naive reasoning for this barrier would be to blame it upon a lack of *standard vocabulary*

in electronic music (whether synthesis, analysis and methods of sound manipulations) as opposed to instrumental music notation. A brief look at the evolution of musical notation through the last century reveals that such *standardization* is simply a naive realist view of the complexity of artistic creation. However it is not hard to notice that within this abundance of technologies, researchers (with exceptions) have almost set aside the more complex problem of temporal abstraction of the tools and considerations for interactivity in provided programming paradigms. The computer music research communities have mostly focused on the tools themselves, improving their efficiencies both in computation and quality, without much concern for their abstractions and access in a creative process.

Computer music can always benefit from new realizations of ideas as software or hardware. But it would be simplistic to say that new tools can systematically address existing problems or create new ones. In other words, we have focused on forms of developments (in terms of computer programming), and have set aside the problems themselves. A problem such as temporal access in writing and performance of computer music, is not one that can merely be solved by new tools although clever new paradigms can eventually address the issue. Therefore problems such as this is a problem of *research* at the first sight than mere programming.

The research community on the other hand has stayed even today with the enthusiasm of augmenting our possibilities for sound design with less attention to complex issues such as abstractions of such methods through time, music structures and writing of interaction within computer music processes. Moreover, recent movements such as *Music Information Retrieval* research communities have focused on smaller and easier-to-solve problems mostly concerned with popular and less complex musical structures for commercial exploitations. This has led to extreme *naive realisms* when such computational approaches are to be extended to the cultures of the world and our musical heritage through centuries, not to talk about music composition.

The picture of our research community is not as morbid as the one presented above. Various researchers have attempted to address complex musical problems such as

temporal structure in music with different views and considerable achievements. Probably the most thought provoking systems on these lines belong to earlier history of computer music systems. Buxton et al., in the 1970s, implemented a score-editing tool for electronic sounds destined for real time performance and generation of pre-notated events (Buxton et al., 1979). The *Animal* system of Lindemann (1990) is another example where the notion of *writing of time* through complex hierarchical graphical objects is explicitly considered. However, in both systems consideration of *interaction* control is shifted more towards the performer and less to the composer. In a more theoretical literature, Petri Nets (Allombert and Desainte-Catherine, 2005) and temporal logics (Marsden, 2007) (with many predecessors not cited here) are proposed for enabling temporal access to musical processes in the symbolic domains. None of such proposals have found their way into serious computer music productions. We believe that this is mostly due to the fact that such proposals are inherently based on logics and structures external to music (biological systems, language etc.) and do not address complex networks of temporal relations essential to music.

We strongly believe that music along with western musical notations and all its forms of abstractions, are among the most thoughtful inventions of mankind. A research that brings in pleas from composers and artists on the lines presented above could not see the light of day if not undertaken in a collaborative and creative environment where researchers and artists could elaborate the subject together. The work presented in this chapter is a first step towards implementing a system for the writing of time and interaction in computer music.

8.2 *Antescofo*: A preliminary tool for writing of time and interaction

Antescofo is our preliminary attempt to address the writing of time and interaction in a computer music setting. *Antescofo* became possible through direct collabora-

tion with musicians and particularly composer Marco Stroppa. It is an attempt to address the temporal deficiency of existing systems both on the performance and composition levels and creating the bridge between the time of composition and that of performance, for mixed real time computer music.

At its very core, *Antescofo* is an *anticipatory synchronization* system and the one presented in chapter 7. The anticipatory architecture of *Antescofo* brings in all the advantages of anticipatory modeling outlined in section 3.5, which musically come down to modularity of writing in terms of observation handling and time, access to event structures and timing during writing and performance, bridging the gap between the time of composition and time of performance, as well as providing a simple language for scoring both instrumental and electronic events altogether. The anticipatory design of *Antescofo* also explicitly addresses Manoury's plea for *temporal prediction* during interpretation of virtual scores.

In this section we underline the motivations and objectives for the implementation and modeling of *Antescofo* and outline each aspect of its design and utility in the following sections.

8.2.1 Motivations

In section 8.1, we outlined the realm of *real time* composition in computer music and from different perspectives. Despite its attractive new possibilities, the field has failed to attract many composers and musicians, rooted in the divide between the compositional and performative aspects of computer music. We believe this situation is partially due to the following constraints, detailed before, in available interactive computer music systems:

1. While the common *vocabulary* used in instrumental music scores has extensively expanded, existing score following score vocabularies are extremely limited to very basic semantics of western traditional notation (notes, trills etc.).

2. The notion of *interaction* is most often limited to mere triggering of a separate

electronic score.

3. There has been limited or no consideration for different temporalities of musical events, the temporal interaction within and the writing of time involved.

4. The process of composing electronics and the act of performance have been mostly dealt with separately in existing systems, leaving the important issue of interpretation solely to the computer music artist.

In its very basic use, *Antescofo* is a classical score following application that synchronizes real time audio to a music score (such as MIDI). But it has been designed in its core to address the following extensions handling both flexible score scripting and live interactions:

1. to enable concurrent, flexible and user-defined representations of the audio stream in the score,

2. to concurrently represent and handle different time scales both in compositional time and recognition (performance) time and enable a flexible *writing of time*,

3. to provide a score language that handles interaction between the live performer(s) and electronics both time-wise and process-wise during run-time (i.e. music performance).

8.2.2 General Architecture

The general architecture of *Antescofo* is that of our anticipatory synchronization system presented in section 7.3. We replicate the same schema here again in figure 8.1 and recapitulate the concepts within computer music practices. The main difference here is that the real time inputs are *Media Streams* (rather than Audio streams) that reflect the modularity and concurrency of real time observations of *Antescofo*. This simply means that *Antescofo* can synchronize with pure audio, pitch, gesture data, and

any other sequential data or combinations thereof that is referenced in the score and exists for real time processing. This feature of the system is detailed in section 8.3.

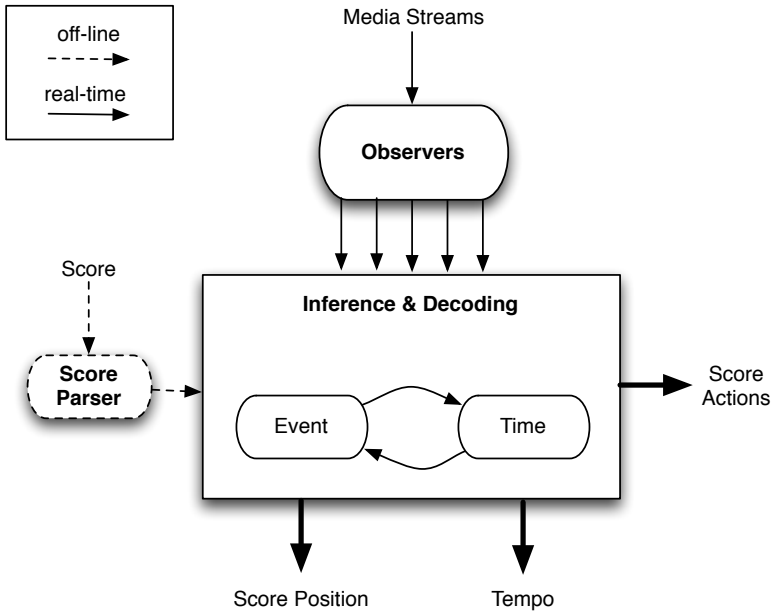


Figure 8.1: *Antescofo*'s general system diagram

During live performance, *Antescofo* provides score location as well as real time tempo. To enable explicit interaction and writing of interactive processes, the *score* given to *Antescofo* can hold *actions* that correspond to electronic processes living outside the system and destined to take place given certain condition and at certain places in the virtual score. The score semantics of *Antescofo* for both instrumental and electronic score are detailed in section 8.4. These semantics have been designed to be minimal and address temporal access to structures at the time of composition as well as time of performance.

Antescofo has been implemented for the two mainstream real time programming environments in computer music, *MaxMSP* and *PureData*, and therefore make use of their expressivity in constructing analysis, synthesis and compositional processes (usually) out-of-time and controls them in-time at the time of the performance and in real

time. Figure 8.2 shows a snapshot of *Antescofo*'s help file in *MaxMSP*[8]. *Antescofo* is basically an implementation of the anticipatory synchronization of chapter 7 where the notion of observation has been extended and time structures made explicitly available for writing of time and interaction. We will detail each feature in the following sections.

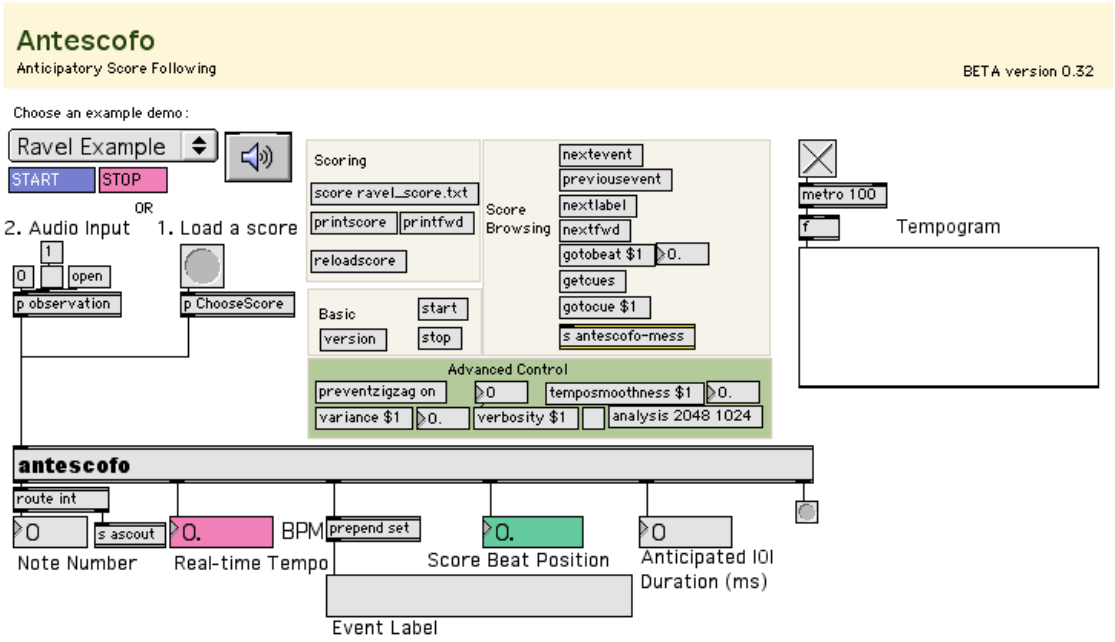


Figure 8.2: *Antescofo*'s Help snapshot in Max/MSP

8.3 *Antescofo*: A modular and concurrent synchronizer

The degrees of control of a musician at the time of performance and towards the score extends that of pitch and sigh-reading of a music score. Pressing (1990) develops the idea of *dimensionality of control* as a way of evaluating the interaction between a player and a performance system. In this article, Pressing summarizes various control aspects of a music performance from control modalities (discrete, continuous or quantized continuous), control monitoring (one-shot or continuous time), dimensionality of

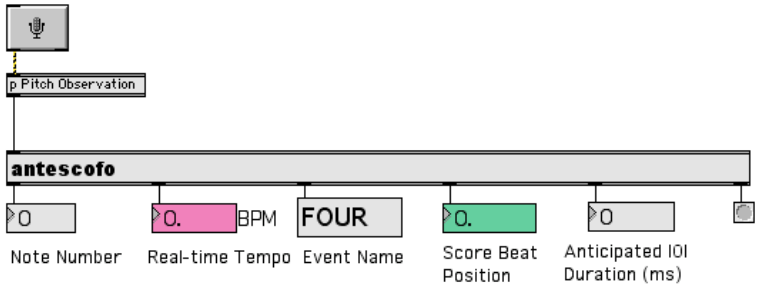
[8]The most recent experimental version is accessible through http://cosmal.ucsd.edu/arshia/antescofo/

control (degrees of freedom), physical variables (from pitch to gesture), to psychological nature of control. In today's practices of computer music, various degrees of control are at use to bridge the gap between the musician and an electronic score extending from pitch detection to live gesture acquisitions of the musician's movements and video analysis (e.g. Camurri et al., 2000; Rasamimanana, 2008). For our purpose, this dimensionality of control should be addressed both at the time of writing and at the time of performance. The former amounts to enabling flexible writing of events and time and discussed in section 8.4, and the latter amounts to the ability of the system in real time to recognize concurrently and modularly different paradigms of writing assigned by the artists and is discussed in this section.

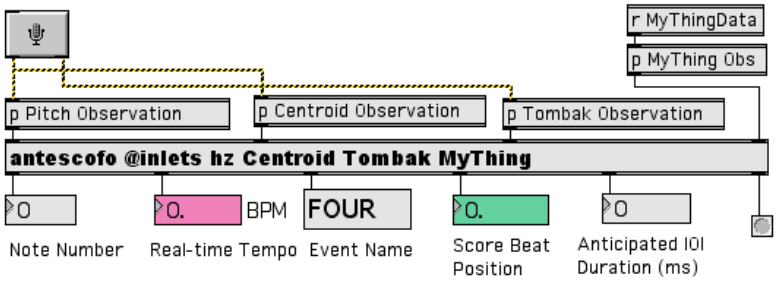
Antescofo has been carefully designed to address this diversity of control in live performances. Following the design presented in chapter 7, *Antescofo*'s window of communication with the outside world is through *observations* made on real time streams. Observations enter the system constantly, as they are available, and continuously in time. Observations also correspond in their nature to the expected events transcribed in the score. For example, if a composer desires to follow *pitches* at some point in the score, pitch observation should be also present among real time observers and if she desires to detect and synchronize with gesture data then the corresponding feature should be made available during real time execution to the system.

Most score following applications come either with their internal observation mechanisms or are tuned towards a specific observation module living outside the score follower itself. For the design of *Antescofo*, we have decided to make it modular by both providing internal observation mechanisms and also enabling user-defined observation inputs. In its basic use, *Antescofo* is a classical score following application that accepts a list of pitches (in Hz or MIDI) or pure audio as input to map it to pitch positions in the score and decode the tempo variable. But for more curious users, *Antescofo* is able to accept concurrent observations of different nature. The number of concurrent observations to the system (which are user-defined and calculated outside *Antescofo*) and their code names are defined by the user during object instantiation in Max or Pd. Figure 8.3

shows the classical and user-defined appearances of *Antescofo* instantiation on a Max window. Here, the creative user of figure 8.3b has attempted to provide four different concurrent observation to the module by overriding and augmenting the default input slots by the `@inlets` command and assigning desired names. In figure 8.3b the user has attempted to create `4` different observations tagged with names `hz`, `Centroid`, `Tombak` and `MyThing`. *Antescofo* comes with reserved code-names for several classical built-in observations which are: `hz`, `midi` and `KL` used respectively for pitch observation using an external pitch tracker, MIDI observation and polyphonic audio observations. User-defined observations can come from different sources as long as they are numerical vectors and need not be synchronous. It is essentially easy to augment this to other types of data types rather than numerical data and it can be done once the need is initiated.



(a) Classical



(b) User-defined

Figure 8.3: Modular Observation in *Antescofo*

By assigning concurrent observations, the user is indicating to the system that

events corresponding to the same code-names exist within the score provided to the system. We cover simultaneous and concurrent event representations within the score semantics in section 8.4.1.

Upon arrival of observation data, *Antescofo*'s goal is to prepare instantaneous beliefs about the real time position in a given score, in the same manner as in chapter 7. The reserved code-name `KL` is the same as the observation module presented in section 7.7. For the rest (including pitch frequency lists in Hz and MIDI pitches), the input vector is fed through Normal distributions centered on expected values in the score and instantaneous probabilities are obtained for inference and decoding. Obviously if the input consists of a vector of values multivariate Normal distributions are employed. The variance of these Normal distributions (or the diagonal covariance matrix for vector inputs) are set to 3% of the expected value (or a semi-tone for pitch related observations), also controllable by the user through the score (section 8.4.1).

8.4 *Antescofo*'s Score Semantics

The ultimate goal of *Antescofo* is the synchronization of a live performance with a given score and execution of electronic actions within the score. Here we provide the score semantics of *Antescofo* for both instrumental and electronic events. When a score is loaded into the system, a parser constructs the equivalent hybrid Markov/semi-Markov chain of the sequential events as described in section 7.6 which will be used by the recognition module. On top of this score model, *Antescofo* integrates instrumental score with that the electronics. The electronic events are *actions* associate to an event or a region in the score and live locally on a state of the score state-space model. In this section we focus on the descriptive levels of *Antescofo*'s score syntax and discuss the handling of score actions in a live performance in section 8.5.

The score language of *Antescofo* has been carefully designed to enable importing of common score formats such as MIDI and to be able to easily describe common classical music repertoire as well as user-defined events coming from different audio

observations and with different temporalities. In this section we describe the basics of *Antescofo*'s score language with their real-world equivalences and show how the same syntaxes can define complex unconventional score events and augment conventional communications between a live performance and live compositional processes. *Antescofo*'s language semantic is designed following the *Kernal Semantic* approach (Van Roy and Haridi, 2004). It is basically young and only *declarative* for the moment. Plans for extending these semantics are discussed in section 8.6.

Given the coexistence of instrumental events and electronic actions, the semantics of *Antescofo* can be studied within three main groups: The first are *Event Declarations* describing instrumental events that are expected to occur during live performance as observations into the system; and the second are *Action Declarations* defining electronic actions associated to each event or region in the score. On another level, users can control the behavior of the synchronizer during the live performance and specific to score locations through various *Control Commands*. We will study each group separately. Some of the event declaration semantics presented here are essentially the sequential models presented before in section 7.6 but are reiterated here for the sake of completeness.

As a convention for syntax description, a plus sign (+) next to each type should be interpreted as "one or more of". A `<float>` indicates a floating number representing the notated observations in the score. For pitched events and as a convention, events would be represented by either MIDI or MIDIcent note numbers and for other user-defined observations they correspond to whatever the user declares. We demonstrate our examples here by assuming a traditional pitch-based score. Other types are represented within < . > signs.

8.4.1 Event Declarations

Events correspond to instrumental actions that are expected to be performed by a live musician. Events in *Antescofo* are defined by their contextual characteristics (for

example pitch) as well as their temporal structures. The temporal structures used for our syntax are essentially based on observations in section 7.2.3 on compositional foundations of time, putting a contrast between *in-time* and *atemporal* events as well as *striated* and *smooth* temporalities. As before, in the figures that follow a Markov state is demonstrated by a regular circle and a Semi-Markov state by a double-lined circle.

`BPM` command

The initial tempo and any tempo change in the score can be encoded by the `BPM` command in Beats-Per-Minute.

Single Event

A single event can be a single pitch, silence or grace note if the observation under consideration is pitch. These events can be either temporal or atemporal (see section 7.2.3). The usual syntax for a single event is as follows:

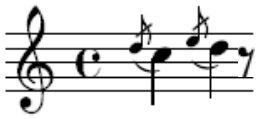
```
<float>          <duration>        <optional name>
```

where the duration is expressed as a floating number indicating the number of beats relative to the initial score tempo. Figure 8.4 shows a sample graphical score, the *Antescofo* equivalent and the state-transition diagram created after parsing. If the duration associated with a single event is set to 0.0, it is a sign that the associated event is *atemporal*. In this example, pitches are encoded using MIDIcent format and a left-right state diagram is created that is in one-to-one correspondence with the score. As a convention, everything that follows a "`;`" on a line is considered as *commentaries*.

When extended to non-pitched observations, *single events* are quasi stationary objects that span an explicit time-range (if they are *in-time*) or implicit duration (if *atemporal*).

The `<duration>` symbol that will appear from now on corresponds to relative beat duration of an event represented by a floating number or a fixed time in milliseconds *if* followed immediately by a `ms` symbol.

```
; This is a sample score
BPM 60
7200    0.0           ; grace note
7200    1.0    note1  ; regular note
7600    0.0
7400    1.0    note2
0       0.5           ; silence
```



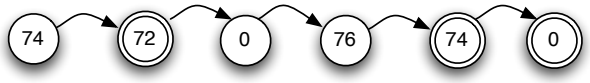


Figure 8.4: *Antescofo*'s single event score sample and state transition diagram

The `<optional name>` column that will appear in all syntaxes, refer to optional labels as *strings* that the user can assign to each event which will be displayed on the *Antescofo* GUI window during live execution and can also be used to retrieve locations in a score using user queries (e.g. during rehearsals).

`TRILL` Class

As the name suggests, the `TRILL` class of *Antescofo* is a way to imitate classical music trill notation. In terms of modeling, *Antescofo*'s `TRILL` is *one in-time* event that has several *out-of-time* events within. Moreover, the order in which these sub-states appear is not important. Figure 8.5 shows two examples for Antesofo's `TRILL` syntax where the second is taken out of the first measure in Marco Stroppa's *Little-I* for flute and electronics and demonstrates a free glissandi which can be successfully encoded using the `TRILL` class in *Antescofo*. The `TRILL` class syntax is as follows:

```
TRILL ( +<float> )  <duration>  <optional name>
```

`CHORD` Class

The chord class denotes a single semi-markov (or markov if duration is set to zero) state that models polyphonic or multidimensional (non-pitched) observations. The

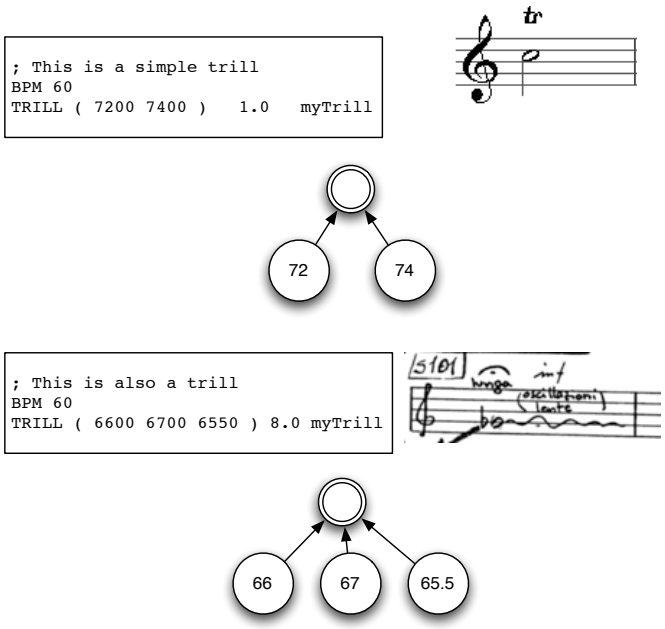


Figure 8.5: *Antescofo*'s `TRILL` class score sample and state transition diagram

regular syntax for the `CHORD` class is similar to the `TRILL` class but translates to only one state

```
CHORD ( +<float> )  <duration>  <optional name>
```

A shortcut for this class (which unifies it with single events) is the representation of events in a list as follows:

```
{ +<float> }  <duration>  <optional name>
```

`MULTI` Class

Using the above commands, any classical music piece can be easily parsed or rewritten in *Antescofo*'s format. We add one more class to allow more complex object and temporal encoding. The `MULTI` class is similar to the `TRILL` class with the exception that the symbols defined within it are *ordered in time*. This new addition to *Antescofo*'s score syntax allows decoding of continuous time events such as glissandis (in western notation). The `MULTI` syntax is as follows:

`MULTI ( +<float> )  <duration>  <optional name>`

In this new topology, a high-level semi-markov state represents the overall temporal structure of the whole object that is mapped to a series of sequential left-right Markov chains. Figure 8.6 shows a `MULTI` example for two consecutive notated glissandis.

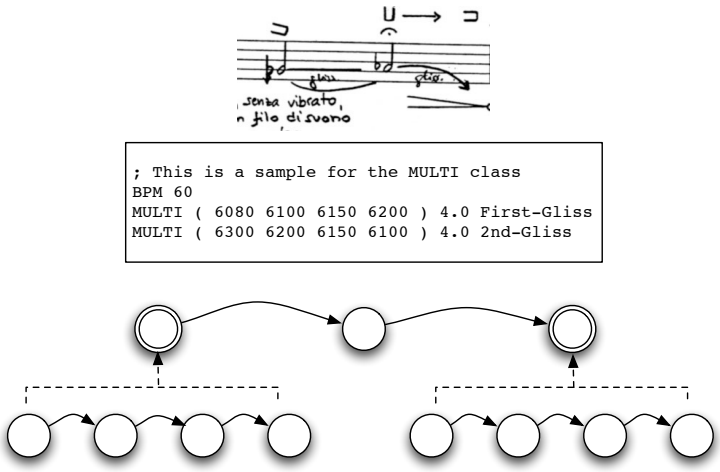


Figure 8.6: *Antescofo*'s `MULTI` class score sample and state transition diagram

The particular power of this class is its ability to encode non-convential and continuous chunks of data such as gesture, image, audio descriptors and more; hence augmenting our synchronization module from a traditional (pitch) score following to gesture following or audio matching. In spite of our systems implementation of this feature, the biggest dilemma is the inability of current data-flow languages in computer music (Max and Pd) to store and transact more complex data structures such as matrices. Due this deficiency, users are obliged to represent multi-dimensional score references either through binary files (such as *SDIF*) or by simply inputting all the data in the score! We are actively searching for solutions to this problem within both environments.

8.4.2 Control Commands

Control commands in *Antescofo* semantic can guide the behavior of the inference and decoding mechanism of the live synchronizer during live performance. Below are some of the most frequently used semantics of this group:

`VARIANCE`

The variance associated with the observers (providing the inference module with instantaneous probabilities) can be controlled in the score and takes as unit, semi-tones (for pitched events) or percentage values of the expected score event (for other observation types) as floating numbers. This is quite useful in several circumstances. For example, when following audio signals from a flute in a live performance, the tuning of high pitches might become different than the expected tuning considered in the score due to warming up of the flute's body. An increase of the observer's variance in such case could save the system and the performance! Also, when dealing with different information sources of audio, one might want to adapt this percentage to the nature of the incoming process.

`TEMPO ON/OFF`

In many cases, the temporal flow of events during live performance is not important in spite of the temporality of events in time of writing. A typical example of this case are *fermatas* in western notation system. In such circumstances, the user can turn off the collaboration between observation and tempo agents of the synchronizer in which case synchronization will be solely based on the detection of events without any explicit timing anticipation and also tempo is not updated. This can be achieved by consecutive use of `TEMPO ON` and `TEMPO OFF` between commands in the score which specifies the performance of the system in real time within the given region.

The @ Operator

As mentioned in section 8.3, *Antescofo* is capable of handling multiple representations of media streams for recognition and interaction. By default, *Antescofo* uses the left-most inlet for all recognition tasks unless specified beforehand by the `@` operator. The string immediately following the `@` operator should conform to the code names created by the user during the object's instantiation otherwise it would be neglected during score parsing and an error message would be displayed. Using this feature, the user can easily switch between various representations and follow the desired aspects simultaneously in a single score.

8.4.3 Action Declarations

Besides the instrumental section, an interactive computer music piece consist of a virtual score indicating electronic actions that should be undertaken under certain circumstances during live performance. As mentioned, traditionally this has been done by a separate sequencer (such as `qlists` in Max or Pd) which are synchronized with a score following system through detected score positions. In *Antescofo*, electronic actions can live simultaneously with instrumental events, thus enjoying more transparency, access to temporal structures, and direct communication with events during both composition and live performance. This feature of *Antescofo*'s score language is constantly growing in accordance with the needs of the *Antescofo*'s users notably at Ircam. Below we describe basic commands and discuss other syntaxes under development and considerations in section 8.6.

`FWD` commands

An interactive (or fixed) electronic music event might be bound to a single event in the score. In this case, a sequence of `FWD` commands with corresponding messages following the event in *Antescofo*'s score would do the work. The simple syntax of `FWD` is as follows:

```
FWD  <symbol>  +<message>
FWD  <delay>    <symbol>  +<message>
```

where <symbol> is the string corresponding to the receiving symbol of an electronic process in Max or Pd, and +<message> correspond to *atom(s)* that would be sent to the symbol at the desired position. The <delay> option if exists, is a floating number indicating the delayed time value in beats which would delay the clock-time for sending the message to outside processes using an internal scheduler coupled to the tempo agent. Optionally, user can employ a ms indicator immediately following this delay value to assign it a fixed rather than relative value.

FWD commands alone provide all that dominates today's mainstream interactive computer music scores within the Max and Pd environments! Nevertheless the simple addition of delay times as beat values (instead of fixed values) greatly enhances the expressivity of the language both during writing and live performance. A sequence of FWD commands with different delays can be viewed as a sequencer with relative timings to a global tempo parameter.

Priority FWD

It might be the case that electronic actions are not necessarily bound to a single event but to a specific region in the score. This is a typical case for example during process initializations or electronic events that cover several measures or longer time-spans (such as audio playback). For this use, FWD commands are particularly dangerous since for many reasons during a performance they can be skipped since they are hooked to a single event and that particular single event can be missed. In such circumstances the PFWD command can be used which shares the same syntax with FWD, but its action handling is relative to its position in the score. Whenever that position is passed during a live performance, no matter if the ensuing even is recognized or not, the PFWD actions are emitted.

Loop `FWD`

The additional command `LFWD` (or loop forward) initiates a periodic temporal process known in the system by its *name* as follows:

```
LFWD   <name>   <period>   <symbol>   +<message>
```

Upon triggering, `LFWD` command sends indicated messages to a symbol in a periodic manner as defined by its syntax. The period is given in beats and is coupled with the decoded tempo inside *Antescofo*. This simply means that the period of the looped message would change appropriately with the tempo of the musician(s).

Optionally, the period factor can be a list of rhythms in beats which will be executed circularly and coupled with the tempo, with the following syntax:

```
LFWD   <name>  { +<period> }  <symbol>   +<message>
```

The `KILLFWD` command can stop a loop forward by calling the process' name anywhere in the score:

```
KILLFWD    <name>
```

8.5 From the Time of Composition to the Time of Performance in *Antescofo*

An *Antescofo* score contains both instrumental events and electronic actions with their respective temporalities that are either fixed (in traditional milli-second units) or in relative beat values. The events of the instrumental score can have flexibility of being in-time, out-of-time, striated or smooth. These simple additions leaves the composer of interactive computer music score with a flexible writing of time, atleast conform to the rich vocabulary of music notation tradition up to our time.

During live performance, the musicians interpret the instrumental score, and *Antescofo* in return interprets electronic scores synchronized to musician's play. At the

onset of each interpretive action of the musician, the inference module of *Antescofo* decodes the ongoing tempo of the musician and couples this tempo with relative time-values of actions to be undertaken. In other words, the relative values in the score are *evaluated* according to the synchronized tempo and on the synchronized event. To this respect, *Antescofo* comprises its own scheduler and timer system for undertaking score actions. This internal scheduler by itself comprises different notions of time that a score can describe and constantly updates the temporality of new actions or actions already scheduled in the timer.

Evaluation of relative notated electronic actions in the score during a live performance and in coordination with the interpretation of the instrumental score by a musician, provides the passage needed from the time of composition to the time of performance. This feature allows abstraction of electronic events during the act of writing, and the ability to imagine them in action at the time of composition in the same way a composer has the habit of abstracting instrumental events while composing. The assured coordination and synchronization at the event level, also allows the composer to imagine and plan the *interaction* between instrumental and electronic events in the same manner she would attempt planning interaction between two human musicians during transcription of her work. In other words, the simple passage from the time of composition to the time of performance in the sense developed here, provides a preliminary mean to the writing of time and interaction in computer music.

In addition to remote actions in the score, *Antescofo*'s interface provide several GUI commands that facilitate live communication between external processes and detected events (see figure 8.2 on page 241). Conventional outputs of the system makes the integration of the system to old repertoire possible. Several score browsing options enable rapid recovery and browsing through the score either using event labels, beat position, or by simply jumping between events, actions or labels. An *Anticipated IOI* parameter provides the user with the predicted time-span of the incoming events in milli-seconds using the internal tempo prediction of the synchronization module. This parameter has been frequently used, for example, to adjust the playback speed of fixed

electronics (audio files) through phase vocoders. This is just to say that even in simple circumstances such as audio playback, the *interaction* is more than just "starting together" but extends itself through the whole temporality of outside processes. Overall, creative use of *Antescofo* is completely left to the user.

8.6 Discussions and Future Directions

The system presented in this chapter is a simple example of how anticipatory computing can enable access and more complex musical behavior by augmenting possibilities in writing and also live situations for computer music. It also demonstrates the first steps towards a wider horizon in interactive computer music composition that is currently being pursued. We started the chapter by providing evidence from the practical and compositional sides of real time computer music realm and discussed deficiencies of current real time systems. *Antescofo*, despite its simplicity, opens up new possibilities that seem to be evident but surprisingly absent in the field. The commonalities between *Antescofo*'s discourse and many of the pleas from composers through their writings as reviewed in section 8.1.3 is a coincidence which also indicates the necessity of augmenting the current score following paradigm. However, this work could not in anyway be possible without the collaborative environment between researchers, composers, and computer music designers in which it was realized. Being young, *Antescofo* has many ways to improve and extend its current stance.

8.6.1 Augmenting the Semantics of Interaction

Antescofo's language semantic is basic and young but can however replicate many computer music pieces that have been realized through the recent years. The birth of this simple language was not the initial aim of the project (which was to become a traditional but more powerful synchronization system as demonstrated in chapter 7). But as more internal variables became explicit, more intuitive and basic musical concepts

became possible. Its semantics contain a minimal set of intuitive concepts, and was designed to be easy for its user to understand and reason in. We defined and designed this language following a *Kernal Semantic* approach (Van Roy and Haridi, 2004). Fortunately, the works and thoughts within computer music demand much more than what our simplistic semantic can offer and therefore this is just a beginning. To augment these semantics, conventional use of real time processes in interactive settings should be studied and unified within a coherent context (without any aim for standardization). An immediate addition to this semantic would be the addition of *conditional* semantics. The most evident scenario to imagine is to condition action behavior on past or present states of the system (either from outside processes or through recognition). This new addition by itself brings in the issue of *concurrency* (in the computer science terms) and shared-state concurrent models. Currently, *Antescofo*'s internal scheduler and timers are threaded concurrently in the system but without sharing states. To enable conditional semantics on actions, their states should be shared. This brings in the idea of directly enabling the declaration of *processes* within *Antescofo*'s semantics, which is naively present within the `LFWD` semantic. This issue by itself brings in the issue of *declarative* and *message-passing concurrencies* (Van Roy and Haridi, 2004), and more importantly the fact that none of the environments *Antescofo* relies on provide this kind of transactional processes that could lead to more expressive programming. These are just subsets of problems that should be considered for any augmentation of *Antescofo*'s current semantical power and will be pursued in near future.

8.6.2 Multimodal Coordination

Modularity of *Antescofo* enables writing and processing of several representations of media streams at the same time. For the moment, the composer has the option of using *only one* representation at each score time for detection and coordination of actions. Therefore, depite the concurrency of representations during matching between state boundaries their final coordination is not concurrent. In other words, this situation

is equal to following an ensemble of instruments but at each measure synchronizing to a specific instrument. Another useful scenario then would be to augment this modularity to *ensemble coordinations*. This intriguing problem has been previously addressed on symbolic settings by Grubb and Dannenberg (1994). An extension of *Antescofo* towards ensemble coordination without central knowledge and using anticipatory mechanisms would be another line of future research.

8.6.3 Intuitive Interfaces

The main goal in the implementation of *Antescofo* was to experiment with the effects of anticipatory modeling within an artistic and collaborative environment. Having this done, and while the system enjoys attention of a small but growing user community, the need for more intuitive score interfaces becomes more apparent. Up to now, we have neglected the issue and made sure the semantics could easily go back and forth between other standards (such as MIDI, MusicXML). There has been several attempts by users to write import and export modules for *Antescofo*'s score language. Currently, the commercial *NoteAbility* graphical score editing environment[9] provides export options to *Antescofo*'s score language. Besides these attempts which address the comfort of the user community, an intuitive way of representing *Antescofo*'s score events and actions would be to make the internal state-space structures (which are quite sparse and corresponding to macro-levels) explicit graphically. This idea will be pursued in a longer future.

8.6.4 Relating to the Community

The *Antescofo* project was born and implemented with the sole idea of augmenting the current vocabulary of real time computer music environments to the demand of composers and artists active in the field. While the underlying engineering architecture and design competes with traditional score following systems (as demonstrated in chap-

[9] http://www.opusonemusic.net/NoteAbility/index.html

ter 7), this project could not have been reached its current status if it was not guided and inspired correctly by the body of computer music artists and researchers. *Antescofo* was first implemented and used for Marco Stroppa's *"... of Silence"* for Saxophone and live electronics[10], and to whom it has much in debt for his rare culmination of musical and scientific knowledge and appreciation for research. The courage of Stroppa in breaking the boundaries of musical expressions and scientific research in a music production situation where things can go smoothly without assuming any risks is an exemplary for the future of computer music research. Clear at the onset of this chapter, the development of ideas for this work has roots in the traditions of realtime electronics literature within which the composer Philippe Manoury is another exemplary artists who has never ceased to be curious and never less demanding than his past with regards to research. The (severe) persistence of Pierre Boulez during test sessions of *"Anthèmes 2"* for Violin and live electronics with *Antescofo* also contributed to the robustness of the system. The relation of *Antescofo* with the computer music community should be continued and it should by its terms accept its shortcomings and naivety and deal with it. A community which hopefully will not cease to question its boundaries and always questions its own view with regards to sciences, not letting *technology* forcefully define her stance.

[10]World premiered in Shizuoka, Japan in November 2007, and by Claude Delange on Saxophone.

Chapter 9

Conclusions

9.1 The story so far

This PhD project was debuted by the simple intuition that musical expectations and anticipations play an important role in many aspects of music from analysis to action-decisions in performance and also artistic creativity. Hence the *title* of this work was created and the aim was set to approach models of musical anticipation that could find use in creative as well as computational aspects of music. While *anticipation* as a word is commonly used by musicians, the term turns out to be troublesome within a scientific perspective. Before anything, the word itself with all its vagueness should have been put into a clear context in order to be considered in a computational framework. Around the same time this work was begun, two important manuscripts emerged in the literature that turned out to be crucial for the preliminary investigations of this work: *Sweet Anticipation: Music and the Psychology of Expectation* by Huron (2006), and the first *ABiALS* book of Butz, Sigaud, and Gérard (2003c). They both provided important insights about the intuitive idea of this work. The former is centered on the psychology of musical expectations as mental representations of many important musical phenomena including emotional responses and more, showing how they elucidate and change many of our beliefs about music. The latter is about designing models that anticipate

without any central aim to provide a universal model for anticipation where *future* is not only the thing we predict but also an important factor in our daily interactions to achieve complex behavior. The second literature commonly refers to this approach of modeling as *Anticipatory Modeling*.

Part I

Part I of this thesis is devoted to the study of both problems above. Chapter 2 reviews the literature on *modeling anticipation* and gathers evidences and modeling implications out of findings in the music cognition literature. At the onset of that chapter, we differentiated anticipation from predictions and expectations. We represented anticipation as a marriage of expectation and action in interaction with the system itself as well as a constantly changing environment. We criticized approaches aiming at providing a syntactic and universal model of anticipation and through our criticisms, specially towards models in music theory and melodic expectancies, we reversed our goal from *modeling anticipation* to that of *models that anticipate* or *anticipatory modeling*. Chapter 3 reviews the literature on *anticipatory modeling*, defines the term, studies different existing approaches and underlies important modeling implications. This thesis, therefore, does not attempt to seek a universal model of anticipation, but aims at providing useful models for sound and music computing that anticipate and/or have the concept of anticipation at the very core of their designs. The models studied throughout the thesis on the other hand are directly inspired by observations from the psychology of musical expectations reviewed in chapter 2, put in a computational framework summarized in chapter 3.

Having set our goal to introduce *anticipatory models* rather than *modeling anticipation*, we devoted the rest of the thesis to three main concerns that an anticipatory system intends to address: *What* to expect, *How* to expect, and *When* to expect. Each question is detailed in the three proceedings parts of the thesis and models and applications pertaining to the appropriate question are proposed.

Part II

The first and most important premise of an anticipatory framework is the *availability of information* both for representational and accessibility purposes. We dedicated part II of this thesis to these two important aspects of music information. In chapter 4 we presented a framework for *Music Information Geometry* as a culmination of different literatures of differential geometry, signal processing, information theory, and machine learning. We showed how this mixture would lead to information theoretic frameworks that not only replicate the state-of-the-art research that addresses the same problem, but also provides easy access, interpretation and control over streams of music information. This part of the thesis, like the preceding part, has its own story. It is the result of continuous review and improvement (or *failure*) of previous works that were presented in (Cont, Dubnov, and Assayag, 2007b) and (Dubnov, Assayag, and Cont, 2007). The goal here was set to expand these models to a general framework where music information could be considered in a similarity metric space. Different experiments and failures led this work to the field of Information Geometry (Amari and Nagaoka, 2000). Through this adventure, we revisited concepts commonly used in music information retrieval such as similarity metrics within a Bregman geometry on exponential distribution manifolds, where each point of the manifold represents an audio frame analysis. Following the work of Dubnov (2008), we separated the *Data* and *Model* aspects of music information and showed how the music information geometry framework provides basic access to those structures. In chapter 5, we introduced two different methods to *access* audio structures based on music information geometry. The first, provides two basic frameworks for online clustering and structure discovery of audio signals by searching structural regularities and without any external intervention. We called this method *Audio Oracle*. We showed how *Audio Oracles* discover and represent structures of audio streams with the premise of easy access to structural information during retrieval schemes. This naturally led us to the consideration of *Audio Oracles* as scalable meta-data in information retrieval schemes. This led to the introduction of the *Guidage* algorithm where

given an audio query, and by navigating *AO* databases, searches the best combinations in a structure to replicate the query. We demonstrated *Guidage* over several application frameworks and discussed how it can also be considered as a front-end for many basic computer music and music information retrieval applications such as unit selection in concatenative synthesis, query-by-example, and automatic assemblage.

Part III

In part III and chapter 6, we aimed at learning anticipatory behavior pictured as learning *planning strategies* for future decision-making in a constantly changing environment. To achieve such reactive and proactive system, learning should then be interactive and adaptive (and as before online). We introduced a framework using the relatively recent paradigm of *Active Learning*. In the proposed framework, the system is in constant interaction with an outside environments through a *reward* system and an internal *memory*. The memory models of the system were chosen as *Audio Oracles* and the rewarding structure as *Guidage*. Learning is then to update the memory models in each interaction and also obtaining *anticipatory values* for each *action* with regards to the present context. The novelty of our representation is in use of multiple, concurrent and competitive active learning agents that would learn to *act* based on beliefs geared towards future. We showed in section 6.7.1 how these interactions can lead to acquiring *knowledge* through blending and accumulations of past and new knowledge of the system through subsequent interactions. The results show that with very little data at each interaction cycle, the system not only discovers and evaluate useful patterns but also validates states whose *future* outcome could lead to valuable outcome. Succession of interactive learning was demonstrated in section 6.7.2 in automatic improvisation and style imitation of a piece of music. Results show evidence of long-term planning and formal long-term structures that are learned implicitly and that correspond to the piece under consideration. These results probably prove the climax of anticipatory design premises, that complex behavior can be achieved through a relatively simple design,

learning can be done with little data, and without any ad-hoc or a priori knowledge.

Part IV

Part IV investigated how *anticipation* can come to the aid of *timing* issues such as live synchronization. Live synchronization by itself is at the heart of every interactive computer music system. Synchronization in this concept is to find the position in the score of a live musician performing the same score, also referred to as *score following*. A major goal of an score following application is to use the computer as a replacement for a human musician accompanying the live performance. Any musician would agree that in a human-human case an important process responsible for synchronization is the predictions that each musician makes with respect to instantaneous actions of others (to which one could add preparatory visual signals). In a Human-Computer Interaction case, no existing system to our knowledge has considered this important anticipatory mechanism. In our proposal in chapter 7, we bring in explicit anticipations of tempo to the help of current decision makings regarding real-time score position. Our sensorial anticipation design consists of two coupled tempo and audio models within a novel inference framework that decodes the score position as well as realtime predicted tempo. We evaluated the performance of both tempo and audio alignment outputs and showed that the performance not only equals the excellence of existing state-of-the-art systems but also extends to real polyphonic cases.

The synchronization system proposed in chapter 7 can easily find its way into standard realtime pieces in the mixed electronic and instrumental repertoire of computer music. At the same time, interactive computer music repertoire that makes use of the score following paradigm strives for tools destined towards a *writing of time and interaction*. In chapter 8 we propose an implementation of our anticipatory synchronization system that makes use of the modeling premises of chapter 3, and in particular that of multimodal interaction and information access, to approach this desired goal in computer music practice. Before doing so, we looked at the background of this still

young field in computer music, its different practices, research status, and compositional stance. Among composers' stance of the problem, we looked at key historical compositional foundations of realtime electronics as drawn by composer Philippe Manoury in the 1980s, also his latest pleas as well as constructive criticisms from other composers notably that of Risset and Stroppa. Many of these criticisms and pleas can be summarized as a call for a solution to the lack of compositional consideration for *realtime* manipulations whose domain is dominated in both practice and research by performers with less considerations for transcription of ideas as scores. Another very important issue common between both Manoury and Stroppa (despite their aesthetical divergences), is the plea for explicit considerations for musical time, access to its structure and to its transcription for both compositional and performance purposes of live computer music. To this end we proposed an implementation of our live synchronization system of chapter 7, called *Antescofo*, as an extension to the score following paradigm in interactive computer music. *Antescofo* extends traditional score following by integrating electronic actions into the instrumental score, enabling concurrent representations of audio (or media) streams rather than pitch, and by allowing explicit control over event temporalities both for compositional and performative aspects of an interactive piece. *Antescofo* is enhanced with a simple language semantic for declaring instrumental events as well as electronic actions. The score semantics of *Antescofo* gives access to various notions of time common in contemporary music writing and can simulate most pieces in the repertoire. *Antescofo* has been implemented in MaxMSP and PureData programming environments and has had several performances worldwide so far including the world premiere of Stroppa's "... *of Silence*" in Japan, Boulez' "...*Explosante-Fixe...*" in Los Angeles with LA Philharmonic, and more.

9.2 Outlook

Some of the projects undertaken during this PhD study mark the beginning of new adventures in both the science of music and also practices of computer music,

some clearly underying their future roadmap and some proposing divergent but unifying perspectives. We overview some important future perspectives for this work:

Music Information Geometry

The new Music Information Geometry framework introduced in chapter 4 has proven to be particularly general with a promising future. In all applications presented in this thesis following its definition, we considered Multinomial statistical manifolds over data. This was quite straightforward in our applicative considerations. An interesting idea would be to extend this framework to sparser statistical representations of the underlying data (such as mixture models, or probabilistically learned models over data with reduced dimensions). There is no theoretical barrier for this extension and in practice it opens the door of its consideration to many music information retrieval problems. It is envisioned that the geometrical intuition of the framework help more rigorous modeling and results in application it is applied to, as was the case in our studied examples.

We showed how our music information geometry framework helps the discovery of structural regularities in realtime audio streams. In our simple clustering framework, we assumed a fixed information gain in model formations over data streams. An immediate improvement would be to make this information radius an adaptive parameter, or in other words, directly model the *jump process* responsible for model segmentations in a data stream. A promising direction for research on this line is the work on financial modeling with jump processes (Cont and Tankov, 2004). Another evident expansion to this framework would then be to devise mathematical constructs that actually quantify the measures of information within found sound structures. This is theoretically immediate but have to be assessed and compared to human measures and compositional aspects of such measures, which requires a well-defined research plan for the future.

Another promising direction for this research is to make the intuitive aspects of the basic geometric tools of this framework explicit in computer music applications. It would be worth to study the actual effect of geometric tools such as *rotation*, *parallel*

transport, *geodesic walks* and more on the statistical manifolds of sound and music, and study their connections to current literatures on music signal processing and computer music and envision extensions to such methods. This endeavor requires a more detailed specification of the mathematical constructs of the manifold and will be pursued in the near future. Our hope is that an information geometric representation of audio signals would help bridge the gap between the signal and symbolic aspects of sounds and further help the notion of writing of time and interaction for sound objects.

Active and Interactive Learning of Music Agents

An interesting extension to the Active Learning framework of chapter 6 would be to integrate it with more realistic anticipatory mechanisms. For example, the *Guidage* algorithm used for the rewarding environmental feedbacks, guides the learning agents to the *most probable* states based on the current environmental *context*. While this is useful during learning, it would be worth to study the inverse or how *rarely* visited states can affect information once generated. Such behavioral mechanisms could lead to generation of surprisal, tension and other affective artifacts.

Applications and integration of the Active Learning framework should also be further studied in computer music paradigms. In chapter 6 we showed several applicative examples mostly as a proof of concept. The experiments on *Knowledge-Based Interactions* in section 6.7.1 showed how an imaginary user can interact and communicate with a learning agent by transfering different forms of knowledge representations. We also showed the full functionality of the system in case of automatic improvisation and style imitation in section 6.7.2. The idea of using computers as generative agents that learn actively through interaction with incoming information is a new trend in the computer music literature. It puts forth the idea of *improvisation* as a musical construct for automatic agents and is commonly referred to as *controlled improvisation* (Assayag, Bloch, and Chemillier, 2006a) and hints towards *open form* computer music compositions. There are basically many questions that need to be addressed once the generative

aspect of the algorithm comes into play. Our proposed framework is actually inscribed in the *OMax* project at Ircam (Assayag, Bloch, Chemillier, Cont, and Dubnov, 2006b) and its implementation and use in computer music will be a subject of future studies and experimentations.

From the time of music to the music of time

The idea of tools that enhance writing of time and interaction is central in interactive computer music, gaining important momentum between computer musicians, and is strangely absent in the discourse of current research. *Antescofo* is just a preliminary step and will be much extended along the lines of artistic needs of its users and by contemplating on existing discourses such as in (Ircam, 2006). Future perspective of *Antescofo* was thoroughly discussed in section 8.6 and compromises various directions such as stepping towards semantics of interaction, multimodal coordination schemes, and enhancing the current development towards more intuitive interfaces.

One of the main promises of this thesis is to make some cognitive aspects of musical expectation explicitly available for artistic or computational purposes around music. To this end, this study considers cognitive aspects of musical time pertaining to expectations and provide models and concepts that makes these abstract processes available and controllable. Some of these concerns were originally brought up by composer Grisey (1987) as the *skin of time* in music, and we attempted to address some of them specifically in different parts of this thesis. It goes without saying that this thesis is a first step for achieving the desired compositional controls over the *skin of time* and in addressing the complexity of music. With this respect, we hope to have shown that music is a self-sufficient domain for the study of complexity and cognition in time. Further research employing the empirical reflections of composers is necessary to approach the fronts of complexity of musical time.

Part V

Appendices

Appendix A

Supplemental Material for part II

A.1 Properties of Multinomial Manifolds

Throughout part II of this thesis, we use Multinomial exponential distribution manifolds for the *Music Information Geometry* framework introduced in chapter 4 and used in chapter 5. Due to its importance, here we summarize their important properties in an information geometric framework and provide the tools that are used throughout chapters 4 and 5.

Recall that a Multinomial distribution is defined as:

$$p(\boldsymbol{x};\boldsymbol{q}) = \frac{N!}{\prod_{j=1}^{d} x_j!} \prod_{j=1}^{d} q_j^{x_j},$$

where $x_j \in \mathbb{Z}_+$ are frequencies of events, $\sum_{j=1}^{d} x_j = N$ and $q_j \geq 0$ are probabilities of events that sum up to 1.

In example 6 on page 71 we derived the Bregman divergence, log-partition function and auxiliary functions for Multinomial manifolds. Due to their importance and use throughout this thesis, we recapitulate those findings in table A.1 below.

In applications of this manifold to music and audio information, we often need to go back and forth between the Multinomial manifold points and the original audio vectors themselves. This is for example the case for calculating symmetrized centroids

Table A.1: Summary of Multinomial Manifold Properties

Natural Parameters:	$\boldsymbol{\theta}$	=	$\left(\log \frac{q_j}{q_d}\right)_{j\in[1,\ldots,d-1]}$
Log-Partition Function:	$F(\boldsymbol{\theta})$	=	$N \log(1 + \sum_{j=1}^{d-1} e^{\theta_j})$
Expectation Parameter:	$\boldsymbol{\mu}$	=	$(Nq_j)_{j\in[1,\ldots,d-1]}$
Legendre Dual:	$F^*(\boldsymbol{\mu})$	=	$N \sum_{j=1}^{d} \left(\frac{\mu_j}{N}\right) \log\left(\frac{\mu_j}{N}\right)$
Bregman Divergence:	$D_{F^*}(\boldsymbol{x}, \boldsymbol{\mu})$	=	$N \sum_{j=1}^{d} \frac{x_j}{N} \log\left(\frac{x_j/N}{\mu_j/N}\right)$
Cumulant Derivative:	$\nabla F(\boldsymbol{\theta})$	=	$\left(\frac{\exp\theta_j}{1+\sum_{i=1}^{d-1}\exp\theta_i}\right)_{j\in[1,\ldots,d-1]}$
Inverse of ∇F*:*	$(\nabla F)^{-1}(\boldsymbol{\eta})$	=	$\left(\log \frac{\eta_j}{1-\sum_{i=1}^{d-1}\eta_i}\right)_{j\in[1,\ldots,d-1]}$

(in case this centroid is needed in the audio domain). This conversion for Multinomial distributions is straightforward since they basically represent the original vector itself by relaxing only one parameter from the original domain. The only constraint for converting a representational front-end to Multinomial manifold points is that they should represent *probability mass functions* or normalized *histograms*. Assuming this is the case, a set of n discrete distributions $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_n$ in the d-simplex where $\boldsymbol{q}_i = (q_i^1, \ldots, q_i^d)$, the conversion to a Multinomial manifold amounts to mapping these probability mass vectors to the *natural parameter* of the Multinomial point, leading to a vector $\boldsymbol{\theta}$ with $d-1$ elements, as shown below:

$$\theta_i^k = \log \frac{q_i^k}{1 - \sum_{j=1}^{d-1} q_i^j} \tag{A.1}$$

For the inverse, that is converting natural parameters to probability mass vectors, the following equations can be easily emplyed:

$$\begin{aligned} q_i^d &= \frac{1}{1 + \sum_{j=1}^{d-1} (1 + \exp \theta_i^j)} \\ q_i^k &= \frac{\exp \theta_i^k}{1 + \sum_{j=1}^{d-1} (1 + \exp \theta_i^j)} \end{aligned} \tag{A.2}$$

Note that this last conversion is possible *only* since we use a Multinomial manifold and is not necessarily possible for other exponential distributions. This is mainly because Multinomial manifolds are useful when the representational front-end exhibits histograms and since the Multinomial conversion does not yield any dimensionality reduction whatsoever.

A.2 Bregman Divergence Symmetrization

In this section we give a full account of the optimization algorithm for symmetrizing Bregman divergences on exponential distribution manifolds as proposed in

(Nielsen and Nock, 2007). We first start by outlining the proof of their algorithm and then provide the algorithm itself pertained to Multinomial manifolds, as used throughout chapters 4 and 5.

Recall that a symmetrized Bregman centroid is defined by the following optimization problem on the set of *points* $\mathcal{P} = \{\boldsymbol{p}_i\}_{i=1}^n \subset \mathcal{X}$:

$$\boldsymbol{c}^F = \underset{\boldsymbol{c}\in\mathcal{X}}{\operatorname{argmin}} \sum_{i=1}^{n} \frac{D_F(\boldsymbol{c}, \boldsymbol{p}_i) + D_F(\boldsymbol{p}_i, \boldsymbol{c})}{2} \tag{A.3}$$

They start their approach by simplifying the minimization problem of the above equation by the following theorem:

Theorem A.1 ((Nielsen and Nock, 2007))**.** *The symmetrized Bregman centroid c^F is unique and obtained by the following simplified minimization:*

$$\boldsymbol{c}^F = \underset{\boldsymbol{c}\in\mathcal{X}}{\operatorname{argmin}} \left[D_F(\boldsymbol{c}_R^F, \boldsymbol{c}) + D_F(\boldsymbol{c}, \boldsymbol{c}_L^F) \right] \tag{A.4}$$

Proof. The right-type average centroid optimization problem by itself can be reformulated using an auxiliary function

$$J_F(\mathcal{P}, \boldsymbol{c}) = \sum_{i=1}^{n} \left(F(\boldsymbol{p}_i) - F(\boldsymbol{c}) - \langle \boldsymbol{p}_i - \boldsymbol{q}, \nabla F(\boldsymbol{c}) \rangle \right) = \sum_{i=1}^{n} D_F(\boldsymbol{p}_i, \boldsymbol{c}).$$

Adding and subtracting a factor $F(\overline{\boldsymbol{p}})$ where $\overline{\boldsymbol{p}} = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{p}_i$to the definition of J would give:

$$\begin{aligned} J_F(\mathcal{P}, \boldsymbol{c}) &= \left(\sum_{i=1}^{n} \frac{1}{n} F(\boldsymbol{p}_i) - F(\overline{\boldsymbol{p}}) \right) \left(F(\overline{\boldsymbol{p}}) - F(\boldsymbol{c}) - \sum_{i=1}^{n} \langle \boldsymbol{p}_i - \boldsymbol{c}, \nabla F(\boldsymbol{c}) \rangle \right) \\ &= \left(\sum_{i=1}^{n} \frac{1}{n} F(\boldsymbol{p}_i) - F(\overline{\boldsymbol{p}}) \right) + D_F(\overline{\boldsymbol{p}}, \boldsymbol{c}) \end{aligned} \tag{A.5}$$

Using Legendre transformation, eq. A.5 can be obtained for the left-type centroid leading to:

$$\begin{aligned} J_F(\boldsymbol{q}, \mathcal{P}) &= J_{F^*}(\mathcal{P}', \boldsymbol{c}') \\ &= \left(\sum_{i=1}^{n} \frac{1}{n} F^*(\boldsymbol{p}_i') - F(\overline{\boldsymbol{p}'}) \right) + D_F(\overline{\boldsymbol{p}'}, \boldsymbol{c}') \end{aligned} \tag{A.6}$$

From Legendre duality, we know that $F^{**} = F$, $\nabla F^* = \nabla F^{-1}$ and $\nabla F^* \circ \nabla F(\boldsymbol{q}) = \boldsymbol{q}$ (see sections 4.2.2 and 4.2.3). Using these properties, it is easy to see that

$$D_{F^*}(\overline{\boldsymbol{p}'}, \boldsymbol{c}') = D_{F^{**}}\left(\nabla F^* \circ \nabla F(\boldsymbol{c}), \nabla F^*(\sum_{i=1}^{n} \nabla F(\boldsymbol{p}_i))\right) = D_F(\boldsymbol{c}, \boldsymbol{c}_L^F).$$

And conversely,

$$D_F(\overline{\boldsymbol{p}}, \boldsymbol{c}) = D_F(\boldsymbol{c}_R^F, \boldsymbol{c}).$$

Now, replacing the two equations above into equations A.6 and A.5, combining them and cancelling equal dual terms would leads to:

$$\begin{aligned}\sum_{i=1}^{n} D_F(\boldsymbol{c}, \boldsymbol{p}_i) + D_F(\boldsymbol{p}_i, \boldsymbol{c}) &= \left(\sum_{i=1}^{n} \frac{1}{n} F(\boldsymbol{p}_i) - F(\overline{\boldsymbol{p}})\right) \\ &+ \left(\sum_{i=1}^{n} \frac{1}{n} F^*(\boldsymbol{p}_i') - F(\overline{\boldsymbol{p}'})\right) \\ &+ D_F(\boldsymbol{c}_R^F, \boldsymbol{c}) + D_F(\boldsymbol{c}, \boldsymbol{c}_L^F)\end{aligned}$$

Note that the first two parts of the equation above *do not* depend on $\boldsymbol{c}$ meaning that the optimization of the left-term boils down to optimizing $D_F(\boldsymbol{c}_R^F, \boldsymbol{c}) + D_F(\boldsymbol{c}, \boldsymbol{c}_L^F)$. Therefore, eq. A.3 if proved.

The uniqueness of $\boldsymbol{c}^F$ is assured since the right side of eq. A.3 can be rewritten using duality as,

$$D_{F^*}(\nabla F(\boldsymbol{c}), \nabla F(\boldsymbol{c}_R^F)) + D_F(\boldsymbol{c}, \boldsymbol{c}_L^F),$$

and $D_F(.,.)$ and ∇F are convex in their first argument by definition, hence admitting a unique solution. □

The solution to the new optimization problem above can be obtained by direct consideration of geometrical properties of the two centroids. Since this proof is quite intuitive, we drop the formal proof (as discussed in Nielsen and Nock, 2007, Theorem 3.2) and provide the main intuition here.

Geometrically, the solution to equation A.3 is a single point $\boldsymbol{c}^F$ which is *equidistant* from the two centroids and which lies on the *geodesic line* connecting the two

centroids and hence its *tangent vector* is *orthogonal* to the geodesic line itself. With this simple intuition, $\boldsymbol{c}^F$ can then be found as the intersection of the two geometric entities proposed above. The *geodesic* connecting $\boldsymbol{c}_R^F$ to $\boldsymbol{c}_L^F$ is defined as

$$\Gamma_F(\boldsymbol{c}_R^F, \boldsymbol{c}_L^F) = \left\{(\nabla F)^{-1}\left[(1-\lambda)\nabla F(\boldsymbol{c}_R^F) + \lambda \nabla F(\boldsymbol{c}_L^F)\right],\ \lambda \in [0,1]\right\}.$$

And the *bisector* (or equidistant hyper-plan from the two points) is defined as

$$M_F(\boldsymbol{c}_R^F, \boldsymbol{c}_L^F) = \left\{\mathbf{x} \in \mathcal{X} \mid D_F(\boldsymbol{c}_R^F, \boldsymbol{x}) = D_F(\boldsymbol{x}, \boldsymbol{c}_L^F)\right\}.$$

Using this fact, Nielsen and Nock provide a simple dichotomic walk over the geodesic $\Gamma_F(\boldsymbol{c}_R^F, \boldsymbol{c}_L^F)$ connecting the two sided centroids (yielding a candidate $\boldsymbol{c}_h$) and evaluate the equidistance property or $D_F(\boldsymbol{c}_R^F, \boldsymbol{q}_h) - D_F(\boldsymbol{q}_h, \boldsymbol{c}_L^F)$, whose sign and amplitude provide hints for the direction and value of the next step to take for the new candidate.

A.2.1 Geodesic-walk Algorithm for Multinomial Manifolds

Algorithm A.1 shows the simple geodesic-walk approximation algorithm for obtaining symmetrized centroids for Multinomial manifolds where all the functions used are introduced in table A.1.

The computational complexity of the algorithm is drastically reduced (compared to similar approaches that use convex optimization schemes) mostly due to our excessive use of duality in calculating required parameters as explained before. This algorithm can be generalized to any other exponential distribution given that functions in table A.1 are provided for the chosen manifold. The conversion to and back from Multinomial points are not necessary and can be omitted but we provide them for the sake of completeness pertained to multiple usages of this algorithm in chapters 4 and 5 of this thesis.

Algorithm A.1 Geodesic-Walk Algorithm For Bregman Divergence Symmetrization on Multinomial Manifolds

Require: n discrete distributions $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_n$ in the d-simplex.

1: Convert to *Natural Parameter* $\{\boldsymbol{\theta}_i\}$ space (eq. A.1)

2: Initialization:

$$\boldsymbol{\theta}_R^F = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{\theta}_i$$

$$\boldsymbol{\theta}_L^F = \nabla F^{-1}(\frac{1}{n}\sum_{i=1}^{n}\nabla F(\boldsymbol{\theta}_i))$$

$$\lambda_m = 0 \quad , \quad \lambda_M = 1$$

3: **while** $\lambda_M - \lambda_m > \epsilon$ **do**

4: $\lambda = (\lambda_M + \lambda_m)/2$

5:

$$\boldsymbol{\theta} = (\nabla F)^{-1}\left((1-\lambda)\nabla F(\boldsymbol{c}_R^F) + \lambda\nabla F(\boldsymbol{c}_L^F)\right)$$

6: **if** $D_F(\boldsymbol{c}_R^F, \boldsymbol{\theta}) > D_F(\boldsymbol{\theta}, \boldsymbol{c}_L^F)$ **then**

7: $\lambda_M = \lambda$

8: **else**

9: $\lambda_m = \lambda$

10: **end if**

11: **end while**

12: Convert the symmetrized centroid in natural space $\boldsymbol{\theta}$ to probability mass function if needed (eq. A.2)

13: **return** Symmetrized centroid $\boldsymbol{\theta}$ or the expected equivalence in the probability mass domain if needed

Appendix B

Supplemental Material for Part IV

B.1 Derivation of Forward Recursion

To solve for an analytical solution of the inference formulation, we are interested in the most-likely state-sequence S_0^τ that would generate the outside process X_0^τ up to time τ and over the entire state-space $\{s_0, \ldots, s_J\}$. Given this, our goal is to deduct a recursive and dynamic programming framework for the *forward* variable or belief of the system up to time τ. By definition and applying the Bayes formula in chains we have:

$$\begin{aligned}
\alpha_j(t) &= \max_{s_0,\ldots,s_{t-1}} P(S_{t+1} \neq j, S_t = j, S_0^{t-1} = s_0^{t-1}, X_0^t = x_0^t) \\
&= \max_{1\leq u\leq t} \max_{i\neq j} P(S_{t+1} \neq j, S_{t-u} = j, v = 0, \ldots, u-1, S_{t-u} = i | X_0^t = x_0^t) \\
&= \max_{1\leq u\leq t} \frac{P(X_{t-u+1}^t = x_{t-u+1}^t | S_{t-v} = j, v = 0, \ldots, u-1)}{P(X_{t-u+1}^t = x_{t-u+1}^t | X_0^{t-u} = x_0^{t-u})} \qquad \text{(B.1)} \\
&\quad \times P(S_{t+1} \neq j, S_{t-v} = j, v = 0, \ldots, u-2 | S_{t-u+1} = j, S_{t-u} \neq j) \qquad \text{(B.2)} \\
&\quad \times \max_{i\neq j} P(S_{t-u+1} = j | S_{t-u+1} \neq i, S_{t-u} = i) \qquad \text{(B.3)} \\
&\quad \times P(S_{t-u+1} \neq i, S_{t-u} = i | X_0^{t-u} = x_0^{t-u}) \qquad \text{(B.4)}
\end{aligned}$$

The nominator in equation B.1 reduces to $\prod_{v=1}^{u-1} b_j(x_{t-v})$ with the assumption that observations b_j are independent. The denominator here is a normalization factor that can be dropped out in our computation. Equation B.2 is the definition of the sojourn function $d_j(u)$ from section 7.3.2. Similarly equation B.4 is the definition of the semi-Markovian transition probabilities p_{ij} and equation B.3 is the definition of α_i at time $t-u$. Replacing these definitions in the equation and factoring indexes, the recursion then becomes:

$$\begin{aligned} \alpha_j(t) &= \max_{s_0,\ldots,s_{t-1}} P(S_{t+1} \neq j, S_t = j, S_0^{t-1} = s_0^{t-1}, X_0^t = x_0^t) \\ &= b_j(x_t) \max \left[\max_{1 \leq u \leq t} \left(\left\{ \prod_{v=1}^{u-1} b_j(x_{t-v}) \right\} d_j(u) \max_{i \neq j} (p_{ij} \alpha_i(t-u)) \right) \right] \end{aligned} \tag{B.5}$$

B.2 Raphael's Tempo Inference Model

Raphael (2006) has proposed a system for offline alignment of music audio with symbolic scores using a hybrid graphical model. The word "hybrid" comes from the fact that he combines latent discrete variables for score position (acoustic model) with a latent continuous tempo process (tempo model). The two modules in Raphael's design are *cascaded* and not *coupled* as is our case. In other places, Raphael uses variants of this model for real-time alignment or score following. Here we deduct the tempo decoding model and formulation of Raphael and adopt its realtime version for our experiment in section 7.8.1.

In this model, tempo is modeled as a time-varying random variable as well as note-by-note deviations from what the local tempo and printed note lengths in the score would predict. Depicting S_k as local tempo at note k and T_k as onset time of note k in the score, this amounts to the following random variables:

$$\begin{aligned} S_k &= S_{k-1} + \sigma_k \\ T_k &= T_{k-1} + \ell_k S_k + \tau_k \end{aligned} \tag{B.6}$$

where ℓ_k is the length of note/chord k in the score in beats, and $\{S_1, T_1\} \cup \{\sigma_k, \tau_k\}$ are independent normal random variables with $\{\sigma_k, \tau_k\}$ having zero mean and controlling

local deviations. Actual onset times t_k and local tempos s_k are assumed as realizations of the random variables described. Letting $s = \{s_1, \cdots, s_k\}$ and $t = \{t_1, \cdots, t_k\}$ depict actual variables, this model leads to the following Bayesian formulation of the joint probability density of $p(s, t)$:

$$p(s,t) = p(s_1)p(t_1)\prod_{k=2}^{k} p(s_k|s_{k-1})p(t_k|t_{k-1}, s_k) \tag{B.7}$$

Being a generative graphical model, Raphael chooses the following for each factor of equation B.7 where $N(.; \mu, \sigma^2)$ denotes the univariate normal density function with mean μ and variance σ:

$$p(s_1) = N(s_1; \mu_{s_1}, \sigma^2_{s_1}) \tag{B.8}$$

$$p(t_1) = N(t_1; \mu_{t_1}, \sigma^2_{t_1}) \tag{B.9}$$

$$p(s_k|s_{k-1}) = N(s_k; s_{k-1}, \sigma^2_{s_k}) \tag{B.10}$$

$$p(t_k|t_{k-1}, s_k) = N(t_1; t_{k-1} + \ell_k s_k, \sigma^2_{t_k}) \tag{B.11}$$

Correct (realtime) alignment of live music performance to score in the above model amounts to finding the optimal local tempo s_k for each event in the score in an incremental manner and using all the information up to event k. This would automatically predict the next event's onset time t_{k+1} using the second equation in eq. B.6. In a more rigorous way, our goal is to find the most likely configuration of the unobserved variable $\hat{s}_k$ given a set of onset times $t_0^k = (t_1, \cdots, t_k)$ and previous observed tempos $s_0^k = (s_1, \cdots, s_k)$ in order to predict t_{k+1}, or

$$\hat{s}_k = \underset{s}{\operatorname{argmax}}\, p(s_0^k, t_0^k) \tag{B.12}$$

If all variables were discrete, equation B.12 would be solvable using traditional dynamic programming techniques, however, the tempo process s is continuous in this framework. To overcome this difficulty, Raphael solves equation B.12 in two stages: one for solving the optimum $\hat{s}_k$ at each time step k and other for solving the discrete variable t_k which determines the alignment. Here we review the first process for solving local tempo

which is of our primary interest. We suffice it to say that the second stage in Raphael's model is a pruning of the search-tree of possible score alignments and using the solution in the first stage to find the optimum solution (Raphael, 2006).

For each partial path $t_0^k = (t_1, \cdots, t_k)$ and s_0^k we solve for

$$\hat{p}_{t_0^k}(s_{k-1}) = \max_{s_1 \cdots s_{k-1}} p(s_1^k, t_0^k) \tag{B.13}$$

For this solution, we consider the fact that all the factors in eq. B.7 were defined as Gaussian distributions, and thus $p(s_1^k, t_0^k)$ is an exponential function of the form,

$$K(s_k; h, m, v) = he^{\frac{-1}{2}(s_k - m)^2/v}$$

Extending this fact into equation B.13, we can solve for

$$\begin{aligned} \hat{p}_{t_0^k}(s_k) &= \max \hat{p}_{t_0^k}(s_{k-1}) p(s_k|s_{k-1}) p(t_k|t_{k-1}, s_k) \\ &= K(s_k; h_k, m_k, v_k) \end{aligned}$$

by incorporating equations B.8 to B.11 into equation B.7 and B.13, we will obtain the following results:

$$\begin{aligned} h_k &= \frac{h_{k-1}}{2\pi\sigma_{s_k}^2\sigma_{t_k}^2} e^{-0.5\frac{(t_k - t_{k-1} - \ell_k m)^2}{\ell_k^2(v_{k-1} + \sigma_{s_k}^2)) + \sigma_{t_k}^2}} \\ m_k &= \frac{m_{k-1}\sigma_{t_k}^2 + \ell_k(t_k - t_{k-1})(v_{k-1} + \sigma_{s_k}^2)}{\ell_k^2(v_{k-1} + \sigma_{s_k}^2) + \sigma_{t_k}^2} \\ v_k &= \frac{(v_{k-1} + \sigma_{s_k}^2)\sigma_{t_k}^2}{\ell_k^2(v_{k-1} + \sigma_{s_k}^2) + \sigma_{t_k}^2} \end{aligned} \tag{B.14}$$

Obviously, using this formulation the maximum of s_k occurs at m_k and we have solved for the optimal continuous local tempo.

Note that equations B.14 are suggestive of some sort of dynamic programming procedure for obtaining s_k. Raphael's original proposal is off-line employs backtracking for obtaining the optimum tempo path. In the experiments presented in section 7.8.1, we adopt an on-line version similar to using the *forward propagation* part of Viterbi for online decoding.

In the procedure described above, parameters $\{\sigma_{s_k}, \sigma_{t_k}\}$ are critical for performance of the system. They model *local tolerance* of the system with temporal fluctuations. These values are set a priori and by hand, although a machine learning technique could find these values easily given some realizations of the given score.

References

Abdallah, S. A., and Plumbley, M. D., 2007: Information dynamics. Technical Report C4DM-TR07-01, Center for Digital Music, Queen Mary University of London.

Adorno, T. W., 1941: On popolar music. In *Studies in Philosophy and Social Science*, volume IX, 17–48. New York: Institute of Social Research.

Agon, C., Stroppa, M., and Assayag, G., 2000: High level musical control of sound synthesis in openmusic. In *International Computer Music Conference*. Berlin, Allemagne.

Allauzen, C., Crochemore, M., and Raffinot, M., 1999: Factor oracle: A new structure for pattern matching. In *Proc. of Conference on Current Trends in Theory and Practice of Informatics*, 295–310. Springer-Verlag, London. ISBN 3-540-66694-X.

Allombert, A., and Desainte-Catherine, M., 2005: Interactive scores: A model for specifying temporal relations between interactive and static events. *Journal of New Music Research*, **34-4**, 361–374.

Amari, S., and Nagaoka, H., 2000: *Methods of information geometry*, volume 191. Oxford University Press. Translations of mathematical monographs.

Assayag, G., and Bloch, G., 2007: Navigating the oracle: A heuristic approach. In *International Computer Music Conference '07*, 405–412. Copenhagen, Denmark.

Assayag, G., Bloch, G., and Chemillier, M., 2006a: Improvisation et réinjection stylistiques. In *Le feed-back dans la création musicale contemporaine - Rencontres musicales pluri-disciplinaires*. Lyon, France.

Assayag, G., Bloch, G., Chemillier, M., Cont, A., and Dubnov, S., 2006b: Omax brothers: A dynamic topology of agents for improvisation learning. In *ACM Multimedia Workshop on Audio and Music Computing for Multimedia*. Santa Barbara.

Assayag, G., and Dubnov, S., 2004: Using factor oracles for machine improvisation. *Soft Computing*, **8-9**, 604–610.

Assayag, G., Rueda, C., Laurson, M., Agon, C., and Delerue, O., 1999: Computer Assisted Composition at Ircam: From PatchWork to OpenMusic. *Computer Music Journal*, **23**(3).

Attali, J., 1985: *Noise: The Political Economy of Music*. University of Minnesota Press. ISBN 0816612870.

Ballard, D. H., 1991: Animate vision. *Artificial Intelligence*, **48**(1), 57–86.

Ballard, D. H., 2002: *Vision and Mind: Selected Readings in the Philosophy of Perception*, chapter On the function of visual representation. MIT Press.

Banerjee, A., Merugu, S., Dhillon, I. S., and Ghosh, J., 2005: Clustering with bregman divergences. *Journal of Machine Learning Research*, **6**, 1705–1749. ISSN 1533-7928.

Bartlett, J. C., 1932: *Remembering*. Cambridge University Press.

Beller, G., Veaux, C., and Rodet, X., 2008: Ircamcorpusexpressivity: Nonverbal words and restructurings. In *LREC workshop on emotions*.

Berger, J., and Gang, D., 2000: A real-time model of formulation and realization of musical expectancy. Unpublished.

Berio, L., 2006: *Remembering the future*. Harvard University Press, the charles eliot norton lectures edition.

Bharucha, J., 1987: Musact: A connectionist model of musical harmony. In *Program of the ninth annual conference of the Cognitive Science Society*, 508–517.

Bharucha, J., 1996: Melodic anchoring. *Music Perception*, **13**, 282–400.

Bharucha, J. J., 1993: Tonality and expectation. In *Musical Perceptions*, editor R. Aiello, 213–239. Oxford University Press, Oxford.

Bharucha, J. J., and Stoeckig, K., 1986: Reaction time and musical expectancy: Priming of chords. *Journal of Experimental Psychology: Human Perception and Performance*, **12**, 403–410.

Biles, J. A., 2003: Genjam in perspective: A tentative taxonomy for genetic algorithm music and art systems. *Leonardo*, **36**(1), 43–45.

Boulez, P., 1964: *Penser la Musique Aujourd'hui*. Gallimard.

Bregman, L. M., 1967: The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics*, **7**, 200–217.

Bresson, J., 2007a: *La synthèse sonore en composition musicale assistée par ordinateur : Modélisation et écriture du son*. Thèse de doctorat, Université de Paris 6, Paris.

Bresson, J., 2007b: Processus compositionnels et opérateurs musicaux dans ml-maquette - les outils de traitement du signal. In *Journées d'Informatique Musicale*. Lyon, France.

Butz, M., and Goldberg, D. E., 2003: *Generalized State Values in an Anticipatory Learning Classifier System.*, chapter 16, 282–301. Number 2684 in LNCS. Springer-Verlag.

Butz, M., Sigaud, O., and Gérard, P., 2003a: Anticipatory behavior: Exploiting knowledge about the future to improve current behavior. In *Anticipatory Behavior in Adaptive Learning Systems*, chapter 1, 1–10. Springer-Verlag.

Butz, M., Sigaud, O., and Gérard, P., 2003b: *Internal Models and Anticipations in Adaptive Learning Systems: Foundations, Theories, and Systems*, chapter 6, 86–109. Number 2684 in LNCS. Springer-Verlag.

Butz, M. V., Sigaud, O., and Gérard, P., editors, 2003c: *Anticipatory Behavior in Adaptive Learning Systems, Foundations, Theories, and SystemsTheories, and Systems*, volume 2684 of *LNCS*. Springer-Verlag. ISBN 3-540-40429-5.

Butz, M. V., Sigaud, O., Pezzulo, G., and Baldassarre, G., editors, 2007: *Anticipatory Behavior in Adaptive Learning Systems, From Brains to Individual and Social Behavior*, volume 4520 of *Lecture Notes in Computer Science*. Springer. ISBN 978-3-540-74261-6.

Buxton, W., Patel, S., Reeves, W., and Baecker, R., 1979: The evolution of the sssp score-editing tools. *Computer Music Journal*, **3**(4), 14–25.

Camurri, A., Hashimoto, S., Ricchetti, M., Ricci, A., Suzuki, K., Trocca, R., and Volpe, G., 2000: Eyesweb: Toward gesture and affect recognition in interactive dance and music systems. *Computer Music Journal*, **24**(1), 57–69. ISSN 0148-9267. doi: http://dx.doi.org/10.1162/014892600559182.

Cano, P., 2007: *Content-Based Audio Search from Fingerprinting to Semantic Audio Retrieval*. Ph.D. thesis, University Pompeu Fabra, Barcelona, Spain.

Carlsen, J. J., 1981: Some factors which influence melodic expectancy. *Psychomusicology*, **1**, 12–29.

Casey, M., 2005: Acoustic lexemes for organizing internet audio. *Contemporary Music Review*, **24**(6), 489–508(20). doi:doi:10.1080/07494460500296169.

Cayton, L., 2008: Fast nearest neighbor retrieval for bregman divergences. In *International Conference on Machine Learning (ICML)*.

Chabot, X., Dannenberg, R., and Bloch, G., 1986: A workstation in live performance: Composed improvisation. In *International Computer Music Conference (ICMC)*, 537–540.

Chadabe, J., 1984: Interactive composing: An overview. *Computer Music Journal*, **8**(1), 22–27. ISSN 01489267.

Chai, W., 2005: *Automated Analysis of Musical Structure*. Ph.D. thesis, Massachusetts Institute of Technology, MA, USA.

Cilibrasi, R., and Vitanyi, P., 2005: Clustering by compression. *IEEE Transactions on Information Theory*, **51**(4), 1523–1545.

Cohn, D. A., Atlas, L., and Ladner, R. E., 1994: Improving generalization with active learning. *Machine Learning*, **15**(2), 201–221.

Conklin, D., 2003: Music generation from statistical models. In *Proceedings of Symposium on AI and Creativity in the Arts and Sciences*, 30–35.

Conklin, D., and Witten, I., 1995: Multiple viewpoint systems for music prediction. In *Journal of New Music Research*, volume 24, 51–73.

Cont, A., 2006: Realtime audio to score alignment for polyphonic music instruments using sparse non-negative constraints and hierarchical hmms. In *IEEE ICASSP*. Toulouse.

Cont, A., Dubnov, S., and Assayag, G., 2007a: Anticipatory model of musical style imitation using collaborative and competitive reinforcement learning. In *Anticipatory Behavior in Adaptive Learning Systems*, editors B. M.V., S. O., P. G., and B. G., volume 4520 of *Lecture Notes in Computer Science / Artificial Intelligence (LNAI)*, 285–306. Springer Verlag, Berlin. ISBN 978-3-540-74261-6.

Cont, A., Dubnov, S., and Assayag, G., 2007b: Guidage: A fast audio query guided assemblage. In *Proceedings of International Computer Music Conference (ICMC)*. Copenhagen.

Cont, A., Schwarz, D., Schnell, N., and Raphael, C., 2007c: Evaluation of real-time audio-to-score alignment. In *International Symposium on Music Information Retrieval (ISMIR)*. Vienna, Austria.

Cont, R., and Tankov, P., 2004: *Financial Modeling with Jump Processes*. Chapman & Hall.

Cope, D., 2001: *Virtual Music*. MIT Press, Cambridge, MA.

Cover, T. M., and Thomas, J. A., 1991: *Elements of Information Theory*. Wiley-Interscience. ISBN 0471062596.

Cuddy, L. L., and Lunny, C. A., 1995: Expectancies generated by melodic intervals: perceptual judgements of continuity. *Perception and Psychophysics*, **57**(4), 451–462.

Dannenberg, R. B., 1984: An on-line algorithm for real-time accompaniment. In *Proceedings of the International Computer Music Conference (ICMC)*, 193–198.

Dannenberg, R. B., 2007: An intelligent multi-track audio editor. In *Proceedings of International Computer Music Conference (ICMC)*, volume 2, 89–94.

de Cheveigné, A., 2002: Scalable metadata for search, sonification and display. In *Proceedings of the 8th International Conference on Auditory Display (ICAD2002)*, editors R. Nakatsu, and H. Kawahara. Advanced Telecommunications Research Institute (ATR), Kyoto, Japan, Kyoto, Japan.

Desain, P., and Honing, H., 1993: The mins of max. *Computer Music Journal*, **17**(2), 3–11.

Dubnov, S., 2006: Spectral anticipations. *Computer Music Journal*, **30**(2), 63–83.

Dubnov, S., 2008: Unified view of prediction and repetition structure in audio signals with application to interest point detection. *IEEE Transactions on Audio, Speech, and Language Processing*, **16**(2), 327–337.

Dubnov, S., Aug. 2004: Generalization of spectral flatness measure for non-gaussian linear processes. *Signal Processing Letters, IEEE*, **11**(8), 698–701. ISSN 1558-2361. doi:10.1109/LSP.2004.831663.

Dubnov, S., and Assayag, G., 2005: Improvisation planning and jam session design using concepts of sequence variation and flow experience. In *Sound and Music Computing (SMC)*. Salerno, Italie.

Dubnov, S., Assayag, G., and Cont, A., 2007: Audio oracle: A new algorithm for fast learning of audio structures. In *Proceedings of International Computer Music Conference (ICMC)*. Copenhagen.

Dubnov, S., Assayag, G., and El-Yaniv, R., 1998: Universal classification applied to musical sequences. In *Proc. of ICMC*, 322–340. Michigan.

Dubnov, S., Assayag, G., Lartillot, O., and Bejerano, G., 2003: Using machine-learning methods for musical style modeling. *IEEE Computer Society*, **36**(10), 73–80.

Dubnov, S., McAdams, S., and Reynolds, R., 2006: Structural and affective aspects of music from statistical audio signal analysis. *Journal of the American Society for Information Science and Technology*, **57**(11), 1526–1536. ISSN 1532-2882. doi:http://dx.doi.org/10.1002/asi.v57:11. Special Topic Section on Computational Analysis of Style.

Edelman, G., 1987: *Neural Darwinism: The Theory of Neuronal Group Selection*. Basic Books.

Eerola, T., 2003: *The Dynamics of Musical Expectancy: Cross-Cultural and Statistical Approaches to Melodic Expectations*. Ph.D. thesis, University of Jyväskylä, Finland.

Eguchi, S., 1983: Second order efficiency of minimum contrast estimators in a curved exponential family. *The Annals of Statistics*, **11**(3), 793–803. ISSN 00905364.

Ellson, J., Gansner, E., Koutsofios, E., North, S., and Woodhull, G., 2003: Graphviz and dynagraph – static and dynamic graph drawing tools. In *Graph Drawing Software*, editors M. Junger, and P. Mutzel, 127–148. Springer-Verlag.

Feder, M., Merhav, N., and Gutman, M., 1992: Universal prediction of individual sequences. *IEEE Trans. Inform. Theory*, **38**(4), 1258–1270.

Ferguson, J. D., 1980: Variable duration models for speech. In *Symposium on the Applications of Hidden Markov Models to text and Speech*, 143–179. Princeton, New Jersey.

Foote, J., 2000: Automatic audio segmentation using a measure of audio novelty. In *IEEE International Conference on Multimedia and Expo (I)*, volume 1, 452–455.

Foote, J., and Cooper, M., 2003: Media segmentation using selfsimilarity decomposition. In *Proceedings of SPIE Storage and Retrieval for Multimedia Databases*, volume 5021, 167–175.

Foote, J. T., 1997: A similarity measure for automatic audio classification. In *Proceedings AAAI 1997 Spring Symposium on Intelligent Integration and Use of Text, Image, Video and Audio Corpora*. American Association for Artificial Intelligence.

Freidman, J. H., Bentley, J. L., and Finkel, R. A., 1977: An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Softw.*, **3**(3), 209–226. ISSN 0098-3500. doi:http://doi.acm.org/10.1145/355744.355745.

gabriel Ganascia, J., Ramalho, G., and yves Roll, P., 1999: An artificially intelligent jazz performer. *Journal of New Music Research*, **28**, 105–129.

Glasersfeld, E. v., 1998: Anticipation in the constructivist theory of cognition. In *Proceedings of the First Computing Anticipatory Systems Conference*, 38–47.

Grisey, G., 1987: Tempus ex machina: A composer's reflections on musical time. *Contemporary Music Review*, **2**(1), 239–275.

Grubb, L., and Dannenberg, R. B., 1994: Automating Ensemble Performance. In *Proceedings of the ICMC*, 63–69.

Grubb, L., and Dannenberg, R. B., 1997: A Stochastic Method of Tracking a Vocal Performer. In *Proceedings of the ICMC*, 301–308.

Guédon, Y., 2005: Hidden hybrid markov/semi-markov chains. *Computational Statistics and Data Analysis*, **49**, 663–688.

Hanslick, E., 1854: *Vom Musikalisch-Schönen*. Leipzig. Translated in 1891 by Gustav Cohen as: The Beautiful in Music. Indianapolis: Bobbs-Merrill Co., 1957.

Held, R., and Hein, A., 1963: Movement-produced stimulation in the development of visually guided behavior. In *Journal of Comp. Physiol. Psych.*, volume 56, 872–6.

Hiller, L. A., and Isaacson, L. M., 1959: *Experimental Music: Composition with an Electronic Computer*. McGraw-Hill Book Company, New York.

Holland, J. H., and Reitman, J. S., 1978: Cognitive systems based on adaptive algorithms. In *Pattern-Directed Inference Systems*, editors D. A. Waterman, and H. F. Roth. Academic Press, New York.

Hoskinson, R., and Pai, D., 2001: Manipulation and resynthesis with natural grains. In *International Computer Music Conference*, 338–341. San Francisco.

Huron, D., 2006: *Sweet Anticipation: Music and the Psychology of Expectation*. MIT Press.

Ircam, 2006: Colloque international écritures du temps et de l'interaction. In *Agora Festival*. Ircam-Centre Pompidou, Paris, France.

Johnson, D., and Sinanović, S., 2001: Symmetrizing the kullback-leibler distance. *IEEE Trans. on Information Theory*.

Johnson-Laird, P. N., 1991: Jazz improvisation: a theory at the computational level. In *Representing Musical Structure*, editors P. Howell, R. West, and I. Cross, 291–325. Academic Press, London.

Kaelbling, L. P., Littman, M. L., and Moore, A. P., 1996: Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, **4**, 237–285.

Large, E. W., and Jones, M. R., 1999: Dynamics of attending: How people track time-varying events. *Psychological Review*, **106**(1), 119–159.

Lazier, A., and Cook, P., 2003: MOSIEVIUS: Feature driven interactive audio mosaicing. In *Proceesing of DAFx*, 312–317. London, UK.

Lefebvre, A., and Lecroq, T., 2000: Computing repeated factors with a factor oracle. In *Proc. of the Australasian Workshop On Combinatorial Algorithms*, 145–158. Hunter Valley, Australia.

Lerdahl, F., 1988: Cognitive constraints on compositional systems. In *Generative Processes in Music: the Psychology of Performance, Improvisation and Composition*, editor J. A. Sloboda, 231–259. Clarendon Press, Oxford, UK. Reprinted in *Contemporary Music Review* 6:97-121.

Lerdahl, F., 2001: *Tonal Pitch Space*. Oxford University Press, New York.

Lerdahl, F., and Jackendoff, R., 1983: *A Generative Theory of Tonal Music*. MIT Press, Cambridge, MA.

Lindemann, E., 1990: Animal : A rapid prototyping environment for computer music systems. In *ICMC: International Computer Music Conference*, editor S. A. et Graham HAIR. Glasgow, Ecosse.

Littman, M. L., 1994: Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the 11th International Conference on Machine Learning*, 157–163. Morgan Kaufmann, New Brunswick, NJ.

Liu, T., Moore, A. W., Gray, A., and Yang, K., 2005: An investigation of practical approximate nearest neighbor algorithms. In *Advances in Neural Information Processing Systems 17*, editors L. K. Saul, Y. Weiss, and L. Bottou, 825–832. MIT Press, Cambridge, MA.

Logan, B., and Chu, S., 2000: Music summarization using key phrases. *icassp*, **2**, II749–II752.

Loui, P., Wessel, D., and Kam, C. H., 2006: Acquiring new musical grammars: a statistical learning approach. In *28th Annual Conference of the Cognitive Science Society*, 1711–1716. Congnitive Science Society, Vancouver, Canada.

Lu, L., Wenyin, L., and Zhang, H.-J., 2004: Audio textures: Theory and applications. *IEEE Transactions on Speech and Audio Processing*, **12**, 156–167.

Machover, T., and Chung, J., 1989: Hyperinstruments: Musically intelligent and interactive performance and creativity systems. In *International Computer Music Conference (ICMC)*, 186–190.

Manoury, P., 1990: *La note et le son*. L'Hamartan.

Manoury, P., 2007: Considérations (toujours actuelles) sur l'état de la musique en temps réel. *Etincelle, le journal de la création à l'Ircam*.

Mardia, K. V., and Jupp, P., 2000: *Directional Statistics*. John Wiley and Sons Ltd., 2nd edition.

Margulis, E., 2005: A model of melodic expectation. *Music Perception*, **22**(4), 663–714.

Marsden, A., 2007: Timing in music and modal temporal logic. *Journal of Mathematics and Music*, **1**(3), 173 – 189.

Martin, A., Seroussi, G., and Weinberger, J., 2004: Linear time universal coding and time reversal of tree sources via fsm closure. *Information Theory, IEEE Transactions on*, **50**(7), 1442–1468.

Mathews, M. V., Miller, J. E., Moore, F. R., Pierce, J. R., and Risset, J. C., 1969: *The Technology of Computer Music*. The MIT Press. ISBN 0262130505.

Maybeck, P. S., 1979: *Stochastic models, estimation and control. Volume I.* Academic Press.

McCartney, J., 1996: Supercollider: a new real time synthesis language. In *Proceedings of the International Computer Music Conference*.

Mccartney, J., 2002: Rethinking the computer music language: Supercollider. *Computer Music Journal*, **26**(4), 61–68. ISSN 0148-9267.

McNab, R. J., Smith, L. A., Witten, I. H., Henderson, C. L., and Cunningham, S. J., 1996: Towards the digital music library: tune retrieval from acoustic input. In *DL '96: Proceedings of the first ACM international conference on Digital libraries*, 11–18. ACM, New York, NY, USA. ISBN 0-89791-830-4. doi:http://doi.acm.org/10.1145/226931.226934.

Meyer, L. B., 1956: *Emotion and Meaning in Music*. Univ. of Chicago Press.

Minsky, M., 2006: *The Emotion Machine: Commonsense Thinking, Artificial Intelligence, and the Future of the Human Mind*. Simon & Schuster. ISBN 0743276639.

Minsky, M., Singh, P., and Sloman, A., 2004: The st. thomas common sense symposium: Designing architectures for human-level intelligence. *AI Magazine*, **25**(2), 113–124.

Moles, A., 1969: *Information Theory and Aesthetic Perception*. University of Illinois Press.

Monteleoni, C., 2006: *Learning with Online Constraints: Shifting Concepts and Active Learning*. Ph.D. thesis, Massachusetts Institute of Technology.

Moore, A., and Atkeson, C., 1993: Prioritized sweeping: Reinforcement learning with less data and less real time. *Machine Learning*, **13**, 103–130.

Moore, F. R., 1990: *Elements of computer music*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA. ISBN 0-13-252552-6.

Murphy, K., 2000: A survey of pomdp solution techniques. Technical report, University of British Columbia.

Murphy, K. P., 2002: *Dynamic Bayesian Networks: Representation, Inference and Learning*. Ph.D. thesis, UC Berkeley, Computer Science Division.

Nadin, M., 2004: *Anticipation: The End is Where We Start From*. Lars Mueller Publishers, Baden, Switzerland.

Narmour, E., 1990: *The analysis and cognition of basic melodic structures: The Implication-Realization Model*. The University of Chicago Press.

Narmour, E., 1992: *The Analysis and Cognition of Melodic Complexity: The Implication-Realization Model*. The University of Chicago Press.

Nielsen, F., Boissonnat, J.-D., and Nock, R., 2007: On bregman voronoi diagrams. In *Proc. 18th ACM-SIAM Sympos. Discrete Algorithms*.

Nielsen, F., and Nock, R., 2007: On the centroids of symmetrized bregman divergences. *Arxiv.org*.

Noë, A., 2004: *Action in Perception*. MIT Press, MA.

Noe, A., October 2005: Against intellectualism. *Analysis*, **65**, 278–290(13). doi:doi: 10.1111/j.1467-8284.2005.00567.x.

Nouno, G., and Agon, C., 2002: Contrôle de la spatialisation comme paramètre musical. In *Actes des Journées d'Informatique Musicale*, 115–119. Marseille, France.

Omohundro, S. M., 1989: Five balltree construction algorithms. Technical report, International Computer Science Institute.

Ong, B. S., 2007: *Structural Analysis and Segmentation of Music Signals*. Ph.D. thesis, University Pompeu Fabra, Barcelona, Spain.

Orio, N., and Déchelle, F., 2001: Score Following Using Spectral Analysis and Hidden Markov Models. In *Proceedings of the ICMC*. Havana, Cuba.

Orio, N., Lemouton, S., Schwarz, D., and Schnell, N., 2003: Score Following: State of the Art and New Developments. In *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*. Montreal, Canada.

Pachet, F., 2002: The continuator: Musical interaction with style. In *Proc. of International Computer Music Conference*. Gotheborg, Sweden.

Pachet, F., 2006: Interactions réflexives: du "c'est marrant" aux machines à flow. In *Actes des recontres musicales pluridisciplinaires*, editor Y. Orlarey. Lyon, Grame.

Pearce, M., Conklin, D., and Wiggins, G., 2004: Methods for combining statistical models of music. In *Computer Music Modelling and Retrieval*, editor U. K. Wiil, 295–312.

Peeters, G., 2004: Deriving musical structures from signal analysis for audio summary generation: “sequence” and “state” approach. In *CMMR*, volume 2771. Montpellier, France.

Peretz, I., and Zatorre, R. J., 2005: Brain organization for music processing. *Annual Review of Psychology*, **56**, 89–114.

Pressing, J., 1987: Improvisation: Methods and models. In *Generative Processes in Music: The Psychology of Performance , Improvisation and Composition*, editor J. Sloboda, 129–178. Oxford University Press.

Pressing, J., 1990: Cybernetic issues in interactive performance systems. *Computer Music Journal*, **14**(1), 12–25.

Pressing, J., 1999: Cognitive complexity and the structure of musical patterns. In *Proceedings of the 4th Conference of the Australasian Cognitive Science Society*. Newcastle.

Puckette, M., 1991: Combining event and signal processing in the max graphical programming environment. *Computer Music Journal*, **15**, 68–77.

Puckette, M., 1997: Pure data. In *Proc. Int. Computer Music Conf.*, 224–227. Thessaloniki, Greece.

Puckette, M., 2002a: Max at seventeen. *Comput. Music J.*, **26**(4), 31–43. ISSN 0148-9267. doi:http://dx.doi.org/10.1162/014892602320991356.

Puckette, M., 2002b: Using pd as a score language. In *Proc. Int. Computer Music Conf.*, 184–187.

Puckette, M., 2004: A divide between ‘compositional’ and ‘performative’ aspects of pd. In *First International Pd Convention*. Graz, Austria.

Puckette, M., and Lippe, C., 1992: Score Following in Practice. In *Proceedings of the ICMC*, 182–185.

Purwins, H., Blankertz, B., and Obermayer, K., 2001: Constant q profiles for tracking modulations in audio data. In *Proceedings of the International Computer Music Conference*, 407–410. Cuba.

Rabiner, L., 1989: A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, **77**(2), 257–285.

Raphael, C., 1999a: A Probabilistic Expert System for Automatic Musical Accompaniment. *Jour. of Comp. and Graph. Stats*, **10**(3), 487–512.

Raphael, C., 1999b: Automatic Segmentation of Acoustic Musical Signals Using Hidden Markov Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **21**(4), 360–370.

Raphael, C., 2004: A hybrid graphical model for aligning polyphonic audio with musical scores. In *ISMIR*.

Raphael, C., 2006: Aligning music audio with symbolic scores using a hybrid graphical model. *Machine Learning*, **65**(2-3), 389–409.

Rasamimanana, N. H., 2008: *Geste instrumental du violoniste en situation de jeu : analyse et modélisation*. Ph.D. thesis, Université Paris 6 - IRCAM UMR STMS.

Rauber, A., Pampalk, E., and Merkl, D., 2002: Using psycho-acoustic models and self-organizing maps to create a hierarchical structuring of music by sound similarities. In *Proceedings of the 3rd International Conference on Music Information Retrieval (ISMIR'02)*.

Risset, J.-C., 1999: Composing in real-time? *Contemporary Music Review*, **18**(3), 31–39.

Rockafellar, R. T., 1970: *Convex analysis*. Princeton Mathematical Series. Princeton University Press, Princeton, N. J.

Rohanimanesh, K., and Mahadevan, S., 2002: Learning to take concurrent actions. In *Proceedings of the Sixteenth Annual Conference on Neural Information Processing Systems*.

Rolland, P.-Y., and Ganascia, J.-G., 2000: Musical pattern extraction and similarity assessment. In *Readings in Music and Artificial Intelligence*, editor E. Miranda, 115–144. Harwood Academic Publishers.

Ron, D., Singer, Y., and Tishby, N., 1996: The power of amnesia: Learning probabilistic automata with variable memory length. *Machine Learning*, **25**(2-3), 117–149.

Rosen, R., 1985: *Anticipatory Systems*, volume 1 of *IFSR International Series on Systems Science and Engineering*. Pergamon Press, Oxford.

Rowe, R., 1992: *Interactive music systems: machine listening and composing*. MIT Press, Cambridge, MA, USA. ISBN 0-262-18149-5.

Rowe, R., 2004: *Machine Musicianship*. MIT Press, Cambridge, MA, USA. ISBN 0262681498.

Sadakata, M., 2006: *Ritme and Rizumu: studies in music cognition*. Ph.D. thesis, University of Nijmegen.

Saffran, J. R., Johnson, E. K., Aslin, R. N., and Newport, E. L., 1999: Statistical learning of tonal sequences by human infants and adults. cognition. In *Cognition*, volume 70, 27–52.

Saul, L. K., and Jordan, M. I., 1999: Mixed memory markov models: Decomposing complex stochastic processes as mixtures of simpler ones. *Machine Learning*, **37**(1), 75–87.

Schellenberg, E., 1997: Simplifying the implication-realization model of musical expectancy. *Music Perception*, **14**(3), 295–318.

Schmuckler, M., 1989: Expectation in music: Investigation of melodic and harmonic processes. *Music Perception*, **7**, 109–150.

Schmuckler, M., 1990: The performance of global expectations. *Psychomusicology*, **9**, 122–147.

Schmuckler, M., 1997: Expectancy effects in memory for melodies. *Canadian Journal of Experimental Psychology*, **51**, 292–306.

Schwarz, D., 2007: Corpus-based concatenative synthesis. *IEEE Signal Processing Magazine*, **24**(1), 92–104.

Schwarz, D., and Wright, M., 2000: Extensions and Applications of the SDIF Sound Description Interchange Format. In *Proceedings of the International Computer Music Conference*. Berlin.

ScofoMIREX, 2006: Score following evaluation proposal. webpage.

Shannon, C., and Weaver, W., 1949: *The Mathematical Theory of Communication*. University of Illinois Press, Urbana, Illinois.

Shannon, C. E., 1948: A mathematical theory of communication. *The Bell System technical journal*, **27**, 379–423.

Simon, H. A., 1969: *The Science of the Artificial*. MIT Press.

Sinanović, S., and Johnson, D. H., 2007: Toward a theory of information processing. *Signal Process.*, **87**(6), 1326–1344. ISSN 0165-1684. doi:http://dx.doi.org/10.1016/j.sigpro.2006.11.005.

Singh, S. P., Jaakkola, T., and Jordan, M. I., 1994: Learning without state-estimation in partially observable markovian decision problems. In *Proceedings of the Eleventh International Conference on Machine Learning*, editors W. W. Cohen, and H. Hirsch, 284–292. Morgan Kaufmann, San Francisco, CA.

Stockhausen, K., 1957: ... how time passes ... In *die Reihe*, volume 3, 10–43. English edition translated by Cornelius Cardew, 1959.

Stroppa, M., 1999: Live electronics or live music? towards a critique of interaction. *Contemporary Music Review*, **18**(3), 41–77.

Stroppa, M., Lemouton, S., and Agon, C., 2002: Omchroma ; vers une formalisation compositionnelle des processus de synthèse sonore. In *JIM 2002*. Marseille.

Sturm, B. L., 2004: MATConcat: An Application for Exploring Concatenative Sound Synthesis Using MATLAB. In *Proceedings of Digital Audio Effects (DAFx)*. Naples, Italy.

Stylianou, Y., and Syrdal, A. K., 2001: Perceptual and objective detection of discontinuities in concatenative speech synthesis. In *ICASSP '01: Proceedings of the Acoustics, Speech, and Signal Processing, 200. on IEEE International Conference*, 837–840. IEEE Computer Society, Washington, DC, USA. ISBN 0-7803-7041-4. doi: http://dx.doi.org/10.1109/ICASSP.2001.941045.

Sutton, R. S., 1991: DYNA, an Integrated Architecture for Learning, Planning and Reacting. In *Working Notes of the AAAI Spring Symposium on Integrated Intelligent Architectures*.

Sutton, R. S., 1997: On the significance of markov decision processes. In *International Conference on Artificial Intelligence and Neural Networks (ICANN)*, 273–282.

Sutton, R. S., and Barto, A. G., 1998: *Reinforcement Learning: An Introduction*. MIT Press. ISBN 0262193981.

Taube, H., 1997: An introduction to common music. *Computer Music Journal*, **21**(1), 29–34. ISSN 01489267.

Teboulle, M., 2007: A unified continuous optimization framework for center-based clustering methods. *Journal of Machine Learning Research*, **8**, 65–102. ISSN 1533-7928.

Tillmann, B., Bharucha, J., and Bigand, E., 2000: Implicite learning of tonality: A self-organizing approach. *Psychological Review*, **107**(4), 885–913.

Tishby, N., Pereira, F., and Bialek, W., 1999: The information bottleneck method. In *Proceedings of the 37-th Annual Allerton Conference on Communication, Control and Computing*, 368–377.

Uchibe, E., and Doya, K., 2004: Competitive-cooperative-concurrent reinforcement learning with importance sampling. In *Proc. of International Conference on Simulation of Adaptive Behavior: From Animals and Animats*, 287–296.

Van Roy, P., and Haridi, S., 2004: *Concepts, Techniques, and Models of Computer Programming*. MIT Press. ISBN 0-262-22069-5.

Varela, F. J., Thompson, E. T., and Rosch, E., 1992: *The Embodied Mind: Cognitive Science and Human Experience*. MIT Press, Cambridge, MA.

Veldhuis, R., and Klabbers, E., Jan 2003: On the computation of the kullback-leibler measure for spectral distances. *Speech and Audio Processing, IEEE Transactions on*, **11**(1), 100–103. ISSN 1063-6676. doi:10.1109/TSA.2002.805641.

Vercoe, B., 1984: The synthetic performer in the context of live performance. In *Proceedings of the ICMC*, 199–200.

Vercoe, B., 1993: *Csound, A Manual for the Audio Processing System and Supporting Programs with Tutorials*. 1993, MIT Media Labs, Ma.

von Hippel, P. T., 2000: Questioning a melodic archetype: do listeners use gap-fill to classify melodies? *Music Perception*, **18**(2), 139–153.

von Hippel, P. T., and Huron, D., 2000: Why do skips precede reversals? the effects of tessitura on melodic structure. *Music Perception*, **18**(1), 59–85.

Wessel, D., 2006: An enactive approach to computer music performance. In *Actes des recontres musicales pluridisciplinaires*, editor Y. Orlarey. Lyon, Grame.

Wittgenstein, L., 1973: *Philosophical Investigations*. Blackwell Publishers. ISBN 0631146709.

Xenakis, I., 1971: *Formalized Music*. University of Indiana Press.

Yeh, C., 2008: *Multiple Fundamental Frequency Estimation of Polyphonic Recordings*. Ph.D. thesis, Université Paris VI.

Yeh, C., Bogaards, N., and Roebel, A., 2007: Synthesized polyphonic music database with verifiable ground truth for multiple f0 estimation. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR'07)*, 393–398. Vienna, Austria.

Zhang, J., 2004: Divergence function, duality, and convex analysis. *Neural Comput.*, **16**(1), 159–195. ISSN 0899-7667. doi:http://dx.doi.org/10.1162/08997660460734047.

Zicarelli, D., 2002: How i learned to love a program that does nothing. *Comput. Music J.*, **26**(4), 44–51. ISSN 0148-9267. doi:http://dx.doi.org/10.1162/014892602320991365.

Zils, A., and Pachet, F., 2001: Musical Mosaicing. In *Proceedings of Digital Audio Effects (DAFx)*. Limerick, Ireland.

Ziv, J., and Lempel, A., 1978: Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, **24**(5), 530–536.